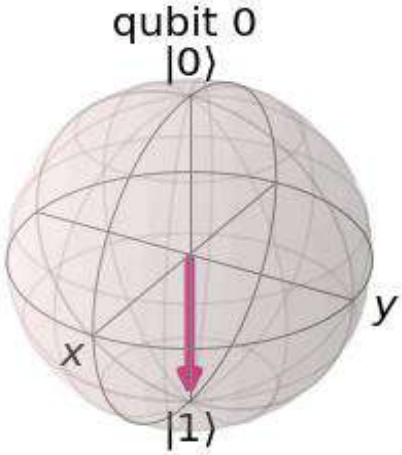
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Π.Μ.Σ. ΜΗΧΑΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΔΙΑΔΙΚΤΥΚΕΣ ΚΑΙ ΦΟΡΗΤΕΣ ΕΦΑΡΜΟΓΕΣ

Κβαντική Υπολογιστική Στη Δευτεροβάθμια Εκπαίδευση



ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ασπασία Οικονόμου (ΑΜ: Μ013121020)

Επιβλέπων: Ηλίας Σάββας, Καθηγητής

Λάρισα, Ιανουάριος 2023

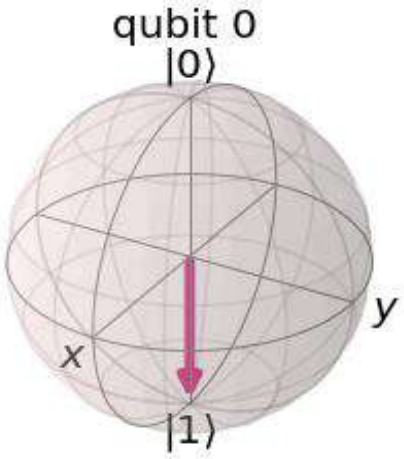
UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

Postgraduate: Software Engineering for Web and Mobile Application

Quantum Computing In Secondary Education



MASTER THESIS DISSERTATION

Aspasia Oikonomou (RN: M013121020)

Supervisor: Ilias Savvas, Professor

Larissa, January 2023

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα μεταπτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

(Υπογραφή)

Ονοματεπώνυμο Φοιτητή

Οικονόμου Ασπασία

Ημερομηνία

16/01/2023

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων/πουσα	**Ονοματεπώνυμο Επιβλέποντα** Βαθμίδα/ιδιότητα επιβλέποντα, Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Ονοματεπώνυμο Μέλους 1** Βαθμίδα/ιδιότητα μέλους 1, Τμήμα/Ίδρυμα μέλους 1
Μέλος	**Ονοματεπώνυμο Μέλους 2** Βαθμίδα/ιδιότητα μέλους 2, Τμήμα/Ίδρυμα μέλους 2

Ημερομηνία έγκρισης: dd-mm-yyyy

Ευχαριστίες

Ένα ευχαριστώ στον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας κ. Σάββα Ηλία, που μου έδωσε την ευκαιρία να γνωρίσω και να έρθω σε επαφή μέσα από το μάθημά του «Πολυπύρηνος και Κβαντικός Προγραμματισμός» και να ασχοληθώ με ένα πολύ ενδιαφέρον και καινοτόμο θέμα με τίτλο «Κβαντική Υπολογιστική στη Δευτεροβάθμια Εκπαίδευση». Θα ήθελα να ευχαριστήσω ολόψυχα αυτόν τον σπουδαίο και δοτικό άνθρωπο κ. Ηλία Σάββα, που είχα την τιμή να είναι ο καθηγητής μου, για το χρόνο που μου διέθεσε, για την πολύ άμεση καθοδήγηση και στήριξή του και που με βοήθησε να αποκτήσω τις βάσεις για τον σχεδιασμό, τη σύνταξη καθώς και για τα πολύτιμα σχόλιά του, στη διπλωματική μου εργασία.

Ένα απλό ευχαριστώ δεν μπορεί να εκφράσει την ευγνωμοσύνη, την εκτίμηση και το θαυμασμό μου στο πρόσωπό του.

Αυτή η διπλωματική μεταπτυχιακή εργασία, είναι αφιερωμένη στα παιδιά μου Γιάννη και Έλενα, από τα οποία αντλώ πάντα δύναμη και φως στη ζωή μου.

Οικονόμου Ασπασία

16/01/2023

Περίληψη

Η συγκεκριμένη διπλωματική εργασία παρουσιάζει ένα εκπαιδευτικό διδακτικό υλικό σχετικά με την κβαντική υπολογιστική για μαθητές της Δευτεροβάθμιας Εκπαίδευσης.

Συγκεκριμένα, οι μαθητές είναι εγγεγραμμένοι σε πειραματικά σχολεία και παρακολουθούν πιλοτικές εφαρμογές. Για αυτή την θεματική ενότητα της κβαντικής υπολογιστικής η ύλη θα καλύπτει δύο εξάμηνα και το μάθημα θα υλοποιείται σε τρεις ώρες την εβδομάδα χωρισμένο σε δύο ημέρες, μετά την λήξη του σχολείου.

Το παρόν διδακτικό υλικό αποτελεί ένα εγχειρίδιο κατανόησης που περιλαμβάνει έξι κεφάλαια, για όλους τους μαθητές του Λυκείου και μόνο για τους μαθητές της Γ' Γυμνασίου που αφορά τα τέσσερα πρώτα κεφάλαια.

Αρχικά γίνεται μια ιστορική ανασκόπηση στην κβαντική μηχανική και τους κβαντικούς υπολογιστές, από το πέρας τον κλασικών στους κβαντικούς υπολογιστές. Δίνονται βασικές ορολογίες, ορισμοί και σχετικές έννοιες καθώς και πώς οι κβαντικοί υπολογιστές αλλάζουν την καθημερινότητά μας.

Σε αυτό το διδακτικό υλικό αναφέρονται όλα τα απαραίτητα στοιχεία της κβαντικής μηχανικής και περιγράφεται η μονάδα της κβαντικής πληροφορίας το qubit. Αναλύονται ορισμοί από τις βασικές ιδιότητες της κβαντομηχανικής, δηλαδή η κβαντική υπέρθεση, η κβαντική διεμπλοκή και η κβαντική υπεροχή και ποιος είναι ο ρόλος τους στην κβαντική υπολογιστική.

Στη συνέχεια αυτή η εργασία έχει στόχο να μυήσει τους μαθητές να αποκτήσουν πολύ βασικές έννοιες και γνώσεις μαθηματικές πάνω στην γραμμική άλγεβρα, τους πίνακες, το τανυστικό γινόμενο πινάκων καθώς και τις βασικές ιδιότητες των μιγαδικών αριθμών, όπου αποτελούν απαραίτητη προϋπόθεση για να κατακτήσουν τα επόμενα κεφάλαια που ακολουθούν στην κβαντική υπολογιστική.

Ακολουθεί ανάπτυξη των απαραίτητων εννοιών και γίνεται εκμάθηση για το πώς συντάσσονται οι βασικές εντολές σε γλώσσα προγραμματισμού Python καθώς και πώς εκτελούνται για τη δημιουργία προγραμμάτων δια μέσω του Jupyter Notebook. Περιγράφονται οι βασικές εντολές και η σύνταξη του Qiskit ώστε να αποκτήσουν οι μαθητές τις κατάλληλες βάσεις για τη δημιουργία απλών κβαντικών προγραμμάτων και της μετέπειτα κβαντικής υπολογιστικής.

Επίσης, δίνεται η διεθνής ορολογία και οι συμβολισμοί στο επιστημονικό πεδίο της κβαντικής υπολογιστικής καθώς και η ανάλυση της δομής των κβαντικών. Ακόμη περιγράφονται οι κβαντικές πύλες και η μέθοδος επίλυσης των κβαντικών κυκλωμάτων.

Παρουσιάζονται κάποια κβαντικά παιχνίδια με τη χρήση του αλγόριθμου των Bernstein-Vazirani καθώς με τα αντίστοιχα κυκλώματά και προγράμματά τους.

Τέλος, για τη μέγιστη δυνατή εμπέδωση, την εποικοδομητική και αποτελεσματική μελέτη του μαθητή όλης της ύλης, δίνονται λυμένα παραδείγματα, εφαρμογές και ασκήσεις για λύση στο τέλος κάθε κεφαλαίου ώστε να είναι σε θέση οι μαθητές να υπολογίσουν και να αναπτύξουν κβαντικά κυκλώματα αλλά και τα δικά τους κβαντικά παιχνίδια.

Η παρακολούθηση μιας τέτοιας θεματικής ενότητας θα αναπτύξει και προετοιμάσει το ενδιαφέρον και το κατάλληλο υπόβαθρο για την μετέπειτα συμμετοχή τους στην τριτοβάθμια εκπαίδευση.

Λέξεις - Κλειδιά: κβαντικοί υπολογιστές, κβαντική μηχανική, κβαντική πληροφορική, Κβαντικά κυκλώματα, κβαντικοί αλγόριθμοι.

Abstract

The current dissertation presents an educational teaching material on quantum computing for secondary education students.

More specifically, the students are enrolled in Experimental Gymnasiums and High Schools and participate in pilot programs. For this module of quantum computing, the curriculum will span two semesters and the students will have to attend 3 hours /2 days a week courses, which will take place after the end of the school program.

The current teaching material constitutes a comprehension manual and includes 6 chapters addressed to all senior high school students, with the first 4 chapters being also appropriate for students who attend the third year of junior high school. Initially, a historical review on quantum mechanics and quantum computers will be presented, covering the transition from classical to quantum computers. Basic terminologies, definitions as well as related concepts are given, along with a description of how quantum computers could change the course of our daily lives.

All the necessary elements of quantum mechanics are mentioned in this teaching material and the notion of a qubit, the basic unit of quantum information, is also described. Furthermore, definitions of the fundamental principles of quantum mechanics, namely quantum superposition, quantum entanglement and quantum supremacy are analyzed and their role in quantum computing is delineated.

Next, the aim of this dissertation study is to enable students to acquire basic math skills and become familiar with the core concepts of linear algebra, matrices, tensor products, as well as the properties of complex numbers which are a prerequisite in order for them to be able to understand the following chapters on quantum computing.

Then, the necessary concepts are analyzed and training on how the basic commands in Python programming language are structured and executed in order to run programs in Jupyter Notebook is provided. The basic commands and syntax of Qiskit are described with a view to helping students set the foundations for the development of simple quantum programs and quantum computing in general.

Moreover, international terminology and annotations in the scientific field of quantum computing as well as an analysis of the quantum structure are provided. The quantum gates and the method of solving quantum circuits are also described.

Certain quantum games based on the use of the Berstein-Varizani algorithm along with their respective circuits and programs are presented as well.

Finally, in order to maximize consolidation and ensure a constructive and effective study of the material on the part of the students, certain solved examples, applications and exercises are provided at the end of each chapter. In this way, students will be able to calculate and create quantum circuits as well as their own games.

Attending such a module will spark students' interest and lay the foundations for their subsequent transition in higher education.

Key - words: quantum computers, quantum mechanics, quantum informatics, quantum circuits, quantum algorithms.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ: 1ο - Ο Κβαντικός κόσμος - Βασικές έννοιες

Κεφάλαιο: 1ο

Ο Κβαντικός Κόσμος – Βασικές Έννοιες

Σύνοψη

Σε αυτό το κεφάλαιο γίνεται μια ιστορική ανασκόπηση στην κβαντική μηχανική και τους κβαντικούς υπολογιστές. Αρχικά γίνεται μια ανάλυση σε βασικές ορολογίες, ορισμούς καθώς και σε σχετικές έννοιες. Συγκεκριμένα, περιγράφονται έννοιες της πληροφορίας στη ζωή του ανθρώπου, της πληροφορίας των υπολογιστών και της κβαντικής πληροφορίας δηλαδή το qubit. Επίσης, αναλύονται ως ένα βαθμό τι είναι ο κβαντικός υπολογιστής και τι είναι κβαντική υπολογιστική. Αναλύονται οι ορισμοί από τις βασικές ιδιότητες κβαντομηχανικής, δηλαδή την κβαντική υπέρθεση, την κβαντική διεμπλοκή και την κβαντική υπεροχή. Στη συνέχεια ακολουθεί μια περιγραφή για την αναγκαιότητα των κβαντικών υπολογιστών στη ζωή μας και αναφέρονται κάποια πλεονεκτήματα και μειονέκτημά τους. Επίσης, γίνεται αναφορά των τριών αλγορίθμων Deutsch, Grover, Shor και συγκεκριμένα στα αποτελέσματα και τα προβλήματα που επιλύουν. Τέλος, κάνουν λόγο για επιστήμονες, ερευνητές και εταιρείες που ασχολούνται με την κβαντική υπολογιστική, την προοπτική των εφαρμογών της καθώς και σε ποιο στάδιο βρισκόμαστε σήμερα. Αυτό το κεφάλαιο απευθύνεται σε μαθητές της Γ’ Γυμνασίου και όλους τους μαθητές Λυκείου.

1.1 Ιστορική Ανασκόπηση της Κβαντομηχανικής

Οι αρχές λειτουργίες ενός κβαντικού υπολογιστή υπαγορεύονται από τους νόμους της φυσικής του χώρου της κβαντικής μηχανικής, όπου άνοιξαν μια νέα προοπτική αντιμετώπισης των πραγμάτων. Η κβαντική μηχανική ή αλλιώς κβαντική θεωρία ή κβαντική φυσική, είναι ο κλάδος της φυσικής που βασίζεται στη θεωρία της μαθηματικής φυσικής με ένα σύνολο κανόνων, που στην ουσία εξετάζει τη μηχανική των μικρών σωματιδίων, των ατόμων, των μορίων με τη βοήθεια μετρήσιμων μεγεθών.

Η λέξη κβαντομηχανική είναι παράγωγο της λέξης κβάντο, που προέρχεται από το λατινικό quantum που σημαίνει «πόσο». Ο όρος κβάντο αποδίδει ως η ελάχιστη διακεκριμένη ποσότητα φυσικού μεγέθους, όπως είναι το ηλεκτρόνιο, το φωτόνιο κλπ. Η κβαντική μηχανική είναι μια θεωρία που δεν έγινε κοινώς αποδεκτή επειδή αμφισβητούνταν από καταξιωμένους φυσικούς όπως ο Αϊνστάιν. Αν και ο Albert Einstein στην αρχή υπήρξε υποστηρικτής των κβαντικών ιδεών, στη συνέχεια άλλαξε στάση ριζικά απορρίπτοντας τη θεωρία τους, γιατί δεν μπορούσε να πιστέψει ότι οι

Εικόνα:1 Δήλωση του Αϊνστάιν – Ο Θεός δεν παίζει ζάρια

Πηγή εικόνας: http://physics4u.gr/blog/2019/03/19/

νόμοι της φυσικής μπορούν να εμπεριέχουν τυχαιότητα, λέγοντας χαρακτηριστικά: «Ο Θεός δεν παίζει ζάρια με το σύμπαν». Ο Niels Bohr αμέσως του απάντησε δηλώνοντας τη φράση: «Albert, don' t tell God what to do».

Αρχικά, η κβαντική θεωρία οριζόταν από υποθέσεις, εμπειρικούς κανόνες, μεθόδους υπολογισμού και θεωρήματα· ωστόσο, έγιναν κάποιες αλλαγές με τις προσπάθειες του Χάιζενμπεργκ και του Σρέντινγκερ.

Μέσα από τη μελέτη του έργου του Μπορν το 1924 με τίτλο «Περί της κβαντομηχανικής», εισάγεται για πρώτη φορά ο όρος «Κβαντική Μηχανική». Ο Μπορν ήταν προσωπικός φίλος με τον Αϊνστάιν, με τον οποίο διαφωνούσαν για την ισχύ της κβαντομηχανικής.

Ο Πολ Ντιράκ(1902-1984) υπήρξε ένας από τους πιο έμπειρους θεωρητικούς της κβαντικής μηχανικής της εποχής του μαζί με τον Σρέντινγκερ και τον Χάιζενμπεργκ, στον οποίο οφείλεται η μορφή του κβαντικού συμβολισμού (το βέλος στα διανύσματα, όπου το μήκος του δίνει το μέτρο του διανύσματος και η αιχμή του βέλους δίνει τη φορά του).

Αξίζει να επισημάνουμε ότι ο Χάιζενμπεργκ αναπτύσσει μια μαθηματική δομή για την κβαντική θεωρία, που στηρίζεται στα μαθηματικά των πινάκων. Σύμφωνα με τον Heisenberg, τα μεγέθη που δεν μπορούν να γίνουν άμεσα αντιληπτά, πρέπει να απορρίπτονται και συνεπώς να ασχολούμαστε με μεγέθη που μπορούμε να παρατηρήσουμε, δηλαδή υπάρχει μόνο αυτό που παρατηρείται.

Από την άλλη μεριά, ο Schrodinger προτείνει το 1926 την περίφημη εξίσωση Schrodinger που θεωρούνταν αναγκαίο εργαλείο για την μελέτη της κίνησης των σωματιδίων μέσω της περιγραφής των κυμάτων de Broglie.

Έτσι λοιπόν, η κβαντική μηχανική θέτει τα θεμέλια για την τεράστια εξέλιξή της επιστήμης και της τεχνολογίας που γνώρισε η ανθρωπότητα.

Συγκεκριμένα, οι τεχνολογικές εφαρμογές της, όπως οι κβαντικοί υπολογιστές, η κβαντική τηλεμεταφορά και η κβαντική κρυπτογραφία έχουν ραγδαία ανάπτυξη. Το πρώτο πείραμα τηλεμεταφοράς πραγματοποιήθηκε το 1993 από έξι αξιόλογους επιστήμονες και το 1998 πρώτος ο Άντον Τσάιλινγκερ και οι συνεργάτες του κατάφεραν να τηλεμεταφέρουν την πρώτη κβαντική πληροφορία με σημαντικά ποσοστά επιτυχίας μεγαλύτερα του 40%.

Με αυτό το τρόπο συνέβαλαν στην υλοποίηση δημιουργίας κρυπτογραφημένων κβαντικών δικτύων τηλεπικοινωνιών και internet, στα οποία δεν υπάρχει κίνδυνος κάποιος ανεπιθύμητος να είναι ικανός να κατανοήσει το περιεχόμενο και να υποκλέψει τις διακινούμενες πληροφορίες, με λίγα λόγια να πέσουν θύμα χάκερ.

Η κβαντική μηχανική μπορεί να παίξει σπουδαίο ρόλο στην επόμενη γενιά των υπολογιστών, λόγω του ότι οι υπολογιστές γίνονται όλο και μικρότεροι.

Πάνω στη θεωρία της κβαντικής μηχανικής δίνεται η δυνατότητα να αντιμετωπιστούν με διαφορετική οπτική οι υπολογιστές, όπου προσφέρει πανίσχυρες μεθόδους για τον χειρισμό της πληροφορίας. Επίσης, εξηγήθηκαν και αναλύθηκαν διάφορα φαινόμενα που συνέβαλλαν αισθητά στην ανάπτυξη και υλοποίηση πολλών εφαρμογών, μια από αυτές είναι και ο κβαντικός υπολογισμός.

Οι αρχές τις κβαντομηχανικής που μας ενδιαφέρουν είναι η κβαντική υπέρθεση, η κβαντική διεμπλοκή και η κβαντική υπεροχή που περιγράφονται παρακάτω.

1.2 Ιστορική Ανασκόπηση των Κβαντικών Υπολογιστών

Οι υπολογιστές στηρίζουν την λειτουργία τους στην θεμελιώδη αρχή που αρχικά οραματίστηκε τον 19ο αιώνα από τον Charlies Babbage, ο οποίος επινόησε τον προγραμματισμό του υπολογιστή και θεωρείται ως «**πατέρας του υπολογιστή**».

Οι κβαντικοί υπολογιστές αν και πειραματικά βρίσκονται στα πρώτα βήματα της ζωής τους, σαν ιδέα δεν είναι καινούργια προσελκύοντας μεγάλο ερευνητικό ενδιαφέρον.

Η ιδέα μίας υπολογιστικής συσκευής, η οποία θα εκμεταλλευόταν τη θεωρία και τις ιδιότητες της κβαντομηχανικής, ξεκίνησε να ερευνάται τη δεκαετία του 1970, από φυσικούς και επιστήμονες του πεδίου της τεχνολογίας των ηλεκτρονικών υπολογιστών.

Ο κβαντικός υπολογισμός ξεκίνησε στις αρχές της δεκαετίας του ΄80 που είχε παρατηρηθεί από μερικούς επιστήμονες, όπως τον φυσικό Paul Benioff και τον μαθηματικό Yuri Manin. Αρχικά, ο Paul Benioff έδειξε με το έργο του ότι ένας υπολογιστής μπορεί να λειτουργήσει με τους νόμους της κβαντικής φυσικής, προτείνοντας ένα κβαντομηχανικό μοντέλο μηχανής Turing. Ο Yuri Manin πρότεινε την ιδέα ενός κβαντικού υπολογιστή με το βιβλίο του «Computable and Uncomputable».

Στη συνέχεια, η έννοια του κβαντικού υπολογισμού χρονολογείται από το 1981, όταν εμπνεύστηκε από τον νομπελίστα φυσικό **Richard Feynman**. Πρόβλεψε την δύναμη των κβαντικών υπολογιστών, αφού παρατήρησε ότι υπάρχουν φυσικά φαινόμενα που εξελίσσονται με τρόπο που ήταν αδύνατο να προσομοιωθούν από τους κλασικούς υπολογιστές, οι οποίοι δεν μπορούσαν να αντιμετωπίσουν αποτελεσματικά και γρήγορα την περίπλοκη δύναμη των κβαντικών συστημάτων. Η ιδέα του Feynman ήταν ότι για την περιγραφή φυσικών φαινομένων, αντί να εκτελούνται πολύπλοκοι και χρονοβόροι υπολογισμοί εξισώσεων, θα ήταν προτιμότερο να εκτελέσει ένα πείραμα, να δημιουργήσει το αντίστοιχο κβαντικό φαινόμενο καταγράφοντας τα αποτελέσματα. Συνεπώς, το αποτέλεσμα θα ήταν και η λύση της εξίσωσης. Ο Richard Feynman το 1999 σε μια δημοσκόπηση από το βρετανικού περιοδικό Physics World, με 130 κορυφαίος φυσικούς παγκοσμίως, κατατάχθηκε ο έβδομος καλύτερος φυσικών όλων των εποχών.

Στη συνέχεια το 1985 ένας Ισραηλινός αξιόλογος επιστήμονας ο David Deutsch, δημοσίευσε μια θεωρητική εργασία στο πανεπιστήμιο της Οξφόρδης και έθεσε

θεωρητικές βάσεις για την ανάπτυξη των κβαντικών υπολογισμών, κάνοντας λόγο για έναν παγκόσμιο κβαντικό υπολογιστή που θα ήταν ικανός να εξομοιώσει ταχύτατα οποιονδήποτε κλασικό υπολογιστή.

Το 1998 ο Isaac Chuang,ο Neil Gershenfeld και ο Mark Kubinec δημιούργησαν τον πρώτο κβαντικό υπολογιστή των δύο qubit που μπορούσε να εκτελέσει υπολογισμούς.

Το μεγάλο άλμα για τους κβαντικούς υπολογισμούς έγινε με την χρήση των τριών κβαντικών αλγορίθμων, τον Shor, τον Grover και τον Deutsch καθώς και την μαθηματική απόδειξη της αποτελεσματικότητάς τους. Η εμφάνιση των κβαντικών αλγορίθμων ξεκίνησε τη δεκαετία του ’90, με σκοπό την αντιμετώπιση των σφαλμάτων κατά την ύπαρξή τους στους κβαντικούς υπολογιστές.

Οι πρώτες συσκευές κατασκευάστηκαν με βάση τις αρχές της κβαντικής μηχανικής. Ένα παράδειγμα που είναι εφαρμοσμένη σε κβαντική τεχνολογία αποτελεί η ατομική θερμοπυρηνική βόμβα και είναι αναμφισβήτητα κάτι διαφορετικό από την κβαντική τεχνολογία στο σύγχρονο κόσμο.

Ξεκινώντας από το 2000 το πεδίο της κβαντικής υπολογιστικής είχε μια τεράστια ανάπτυξη. Οι εταιρείες τεχνολογίας όπως η D-Wave, η Google, η IBM, η Microsoft και η Intel ασχολούνται με την ανάπτυξη κβαντικών συστημάτων, αξιοποιώντας τη δύναμη της κβαντικής φυσικής, με κύριο στόχο τους την επίλυση ενός προβλήματος σε δευτερόλεπτα που ένας κλασικός υπολογιστής θα χρειαζόταν χρόνια. Η εταιρεία D-Wave κατασκεύασε έναν κβαντικό υπολογιστή, η IBM δημιούργησε επεξεργαστή των 50 qubits, η Microsoft ανέπτυξε τη νέα γλώσσα προγραμματισμού και η Google δημιούργησε κβαντικό επεξεργαστή Bristlecone.

Η πρώτη εταιρεία που ασχολήθηκε με την έρευνα για τη δημιουργία των κβαντικών υπολογιστών είναι η D-Ware το 1999. Συγκεκριμένα, η έρευνα αυτή αφορούσε το πώς θα ήταν δυνατό να σχεδιαστεί και να προγραμματιστεί ένας κβαντικός υπολογιστής. Κύριος σκοπός της είναι η διερεύνηση και συνεπώς η εφεύρεση καινούργιων ανακαλύψεων στον τομέα της επιστήμης των υπολογιστών καθώς και στη φυσικής της βιομηχανίας.

Το 2004 η εταιρεία κατασκεύασε δικές της εγκαταστάσεις στο Βανκούβερ του Καναδά για την κατασκευή των κβαντικών υπολογιστών, υποστηρίζοντας ότι δεν είναι αρκετό για τη δημιουργία των κβαντικών υπολογιστών να υπάρχουν μόνο κάποιες ιδέες και γνώσεις. Οι υπολογιστές που δημιούργησε η D-ware, χρησιμοποιούνται από κορυφαίες εταιρείες όπως είναι η Google και η NASA.

Συγκεκριμένα, το Μάιο του 2011 ανακοινώθηκε ο πρώτος κβαντικός υπολογιστής με όνομα D-WAVE ONE, ο οποίος «τρέχει» σε έναν επεξεργαστή 128 qubits, ονόματος Ranier. Για τη συντήρηση και την προστασία του κβαντικού υπολογιστή από οποιοδήποτε θόρυβο καθώς και την απαραίτητη δροσερή θερμοκρασία, περικλείεται από «μονόλιθο». Ο υπολογιστής πουλήθηκε από την εταιρεία έναντι του αστρονομικού ποσού των 10 εκατομμυρίων δολαρίων στην αμερικάνικη Lockheed Martin, οποία δραστηριοποιείται στους τομείς σχετικά με την αεροδιαστημική, την άμυνα και την ασφάλεια δεδομένων.

Εικόνα: 2 Ο πρώτος κβαντικός υπολογιστής στον κόσμο της IBM

Πηγή εικόνας:
https://www.google.com/search?q=%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%BF%CE%B9%20%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CE%B5%CF%82%20%CF%84%CE%B7%CF%82%20IBM&tbm=isch&hl=el&tbs=rimg:CW_1np21jUhVRYa6s2hRg3cJo8AEAsgIMCgIIABAAOgQIABAA&sa=X&ved=0CB0QuIIBahcKEwi488u0zuz5AhUAAAAAHQAAAAAQBg&biw=1903&bih=937#imgrc=zCYmBXjcbm_9kM

Το 2013 κυκλοφόρησε ο δεύτερος κβαντικός υπολογιστής με όνομα D-WAVE TWO, ο οποίος αποτελείται από 512 qubits, δηλαδή με ισχύ τριπλάσια από τον πρώτο υπολογιστή και με την ονομασία Vesuvius. Το 2015 η εταιρεία εμφάνισε τον υπολογιστή με όνομα D-WAVE 2X, που όταν ερευνητές έδωσαν στον υπολογιστή ένα δοκιμαστικό πρόβλημα, το έλυσε 1000.000.000 φορές πιο γρήγορα από ότι θα έλυνε ένας κλασικός υπολογιστής.

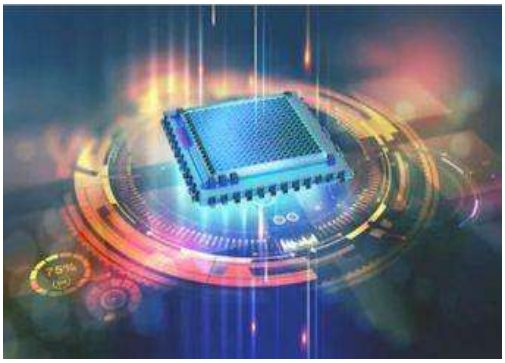
Η εταιρεία IBM πρωτοπόρος στον τομέα των υπολογιστών, από το 2016 έχει διπλασιάσει των κβαντικό όγκο των συστημάτων, ο οποίος όσο μεγαλύτερος είναι τόσο πιο πολύπλοκα προβλήματα μπορούν να αντιμετωπίσουν και αποτελεί βασική προϋπόθεση μαζί με τον αριθμό των qubits, για να χαρακτηρίζει έναν κβαντικό υπολογιστή ισχυρό. Η IBM τον Ιανουάριο το 2019 μας έδωσε μια πρόγευση του κβαντικού μέλλοντος μας, παρουσιάζοντας τον πρώτο κβαντικό υπολογιστή στον κόσμο, με όνομα IBM System One, διαθέτει 20 qubits και έχει μέγεθος ενός μικρού δωματίου, ο οποίος σχεδιάστηκε για εμπορική χρήση. Παρουσιάστηκε στο πλαίσιο της έκθεσης τεχνολογίας CES στο Λας Βέγκας και δεν προορίζεται για πώληση. Θα δοθεί η δυνατότητα online χρήσης του από πελάτες της εταιρείας για εξειδικευμένες ερευνητικές και επιχειρηματικές εφαρμογές, οι οποίοι χρειάζονται μεγάλη υπολογιστική ισχύ, για να πραγματοποιήσουν κβαντικούς υπολογισμούς αυτών των συγκεκριμένων projects. Επίσης, σύστημα για την καλύτερη σταθερότητα, αξιοπιστία και εμπορική χρήση, υποστηρίζεται από τεχνολογία κρυογονικής μηχανικής παρέχοντας έτσι ένα ψυχρό και κατάλληλο κβαντικό περιβάλλον.

Αξίζει να αναφέρουμε ότι η πολυεθνική εταιρεία IBM συνδυάζει έναν νέο περιβάλλον εκτέλεσης προγραμμάτων το Qiskit, ισορροπώντας μεταξύ των κλασικών και κβαντικών υπολογιστικών μεθόδων προκειμένου να πετύχει 100 φορές συντομότερους χρόνους ολοκλήρωσης εργασιών.

Τον Οκτώβρη του 2019, ένα νέο άλμα ανοίγεται στο χώρο των κβαντικών υπολογιστών, δημιουργώντας τον πρώτο κβαντικό επεξεργαστή κατασκευασμένο από μια διεθνής ομάδα ερευνητών από Αυστραλία, Ιαπωνία και ΗΠΑ, επινόησαν τον πρώτο κβαντικό επεξεργαστή, που όλα τα συστατικά του είναι φτιαγμένα από φως λέιζερ.

Η πολυμετοχική εταιρεία IBM συνεχίζει τις προσπάθειές της, προχωρώντας το 2021 στα αποκαλυπτήρια ενός κβαντικού επεξεργαστή των 127 qubits ονόματος Eagle. Για την κατασκευή του IMB Eagle, η εταιρεία ανάπτυξε μια τεχνική τρισδιάστατου «πακεταρίσματος» επιτρέποντάς τη να χωρέσει περισσότερα qubits από ποτέ. Όμως στην παρούσα φάση, αυτή η ισχύς δεν είναι πρακτικά εκμεταλλεύσιμη. Ο Arvind

Krishna ο CEO της εταιρείας, ισχυρίστηκε σε συνέντευξή του στο Axios, πως ο IBM Eagle, πρόκειται για τον μεγαλύτερο επεξεργαστή στον κόσμο, αφού είναι αδύνατον να συγκριθεί ή να προσομοιωθεί με άλλο σύστημα. Η IBM υπόσχεται ότι από το 2023 θα θέσει σε λειτουργία τον κβαντικό υπολογιστή Quantum System Two και θα δουλέψει με επεξεργαστές άνω των 1.000 qubits, ενώ από το 2025 φιλοδοξεί με την εκμετάλλευση των κβαντικών υπολογιστών να επιλύσει τα πρακτικά προβλήματα των πολλών διαφορετικών πεδίων που υπάρχουν.



Εικόνα: 3 Ο πρώτος κβαντικός επεξεργαστής φτιαγμένος από φως λέιζερ

Πηγή εικόνας: https://www.skai.gr/news/technology/i-google-kataskeyase-kvantiko-ypologisti

Οι κβαντικοί υπολογιστές αποτελούν ένα ανώτερο εργαλείο όσον αφορά τη συνολική χρηστικότητα και την τεχνολογική εξέλιξη. Μπορούν να λύσουν τεράστια προβλήματα πολυπλοκότητας γρηγορότερα από τους συμβατικούς υπολογιστές, καθώς στόχος των κβαντικών υπολογιστών είναι η ταχύτητά τους και η αυξημένη χωρητικότητα.

Έτσι, είναι φανερό ότι κορυφαίες τεχνολογικές εταιρείες, ιδιαίτερα την τελευταία δεκαετία, είναι διαθέσιμες για ασταμάτητη έρευνα και πειραματισμό σε αυτό τον τεχνολογικό επενδυτικό ανταγωνισμό, αξιοποιώντας τη δύναμη της κβαντικής μηχανικής για να κατασκευάσουν μια νέα γενιά υπερκβαντικών υπολογιστών.

1.3 Έννοιες του Κβαντικού Κόσμου

Ζούμε στην εποχή της γρήγορης τεχνολογικής ανάπτυξης, με διαθέσιμα μέσα και εργαλεία που πριν μια εικοσαετία δεν μπορούσαμε να φανταστούμε. Έτσι χάρη σε αυτό το τεχνολογικό άλμα, προσεγγίζουμε τη δεύτερη κβαντική επανάσταση. Οι κβαντικοί υπολογιστές συνδέουν δύο τεράστιες έννοιες του 20ου αιώνα, την επιστήμη της πληροφορίας και την κβαντική μηχανική, μοιράζοντας έτσι την ισχύ της και επιτρέποντάς μας να πραγματοποιήσουμε δύσκολες υπολογιστικές εργασίες γρηγορότερα.

1.4 Η Έννοια της Πληροφορίας

Στην σημερινή κοινωνία η πληροφορία παίζει σπουδαίο ρόλο στη ζωή και την εξέλιξη της ανθρωπότητας. Η πληροφορία είναι το αποτέλεσμα της επεξεργασίας κάποιων δεδομένων και είναι απαραίτητη για την ίδια τη ζωή καθώς δίνει αξία στα δεδομένα και παράγει γνώση. Οι πληροφορίες πλέον ανταλλάσσονται με πάρα πολύ γρήγορους

ρυθμούς και είναι αναγκαίες στην ζωή μας καθώς αντιπροσωπεύουν «το βηματισμό» της εξέλιξης.

1.5 Πληροφορία στους Κλασικούς Υπολογιστές

Στην Πληροφορική η πληροφορία σηματοδοτείται από την ποιοτική αξία του bit, βασική πηγή της κλασικής πληροφορίας που μπορεί να εκφραστεί με 0 ή 1. Όλα τα δεδομένα που χρησιμοποιούνται από τους υπολογιστές, όπως τα προγράμματα και οι εφαρμογές, είναι κωδικοποιημένα με το συνδυασμό δύο μόνο ψηφίων το 0 και το 1 (όπου για κάθε χαρακτήρα ή εντολή ο συνδυασμός αυτός είναι μοναδικός), ώστε να είναι κατανοητά από τον υπολογιστή. Για παράδειγμα, πατώντας ένα πλήκτρο στον υπολογιστή δημιουργούνται σειρές μηδενικών και μονάδων. Το bit αποτελεί τη μονάδα πληροφορίας στους κλασικούς υπολογιστές. Η λέξη bit είναι η συντόμευση των λέξεων binary digits (δυαδικό ψηφίο) και αποτελεί την ελάχιστη μονάδα μέτρησης, ενώ μια ομάδα των 8 bit αποτελούν το ένα byte που μπορεί να πάρει 28 = 256 διαφορετικές τιμές. Οι καταστάσεις 0 και 1 των bit αντιστοιχούν σε ηλεκτρικό ρεύμα που περνάει ή δεν περνάει μέσω πολύ μικρών τμημάτων, τα λεγόμενα τρανζίστορ, τα οποία λειτουργούν με διακόπτες. Συγκεκριμένα, όταν περνάει ρεύμα το τρανζίστορ είναι “on” και αντιστοιχεί σε 1 bit, ενώ όταν δεν περνάει ρεύμα είναι “off “ και αντιστοιχεί σε 0 bit. Συνεπώς, με αυτή τη διαδικασία ένας υπολογιστής επεξεργάζεται δεδομένα και παράγει πληροφορίες. Έτσι, ο άνθρωπος προσδίδει νόημα στα επεξεργασμένα δεδομένα μετατρέποντας την πληροφορία σε γνώση.

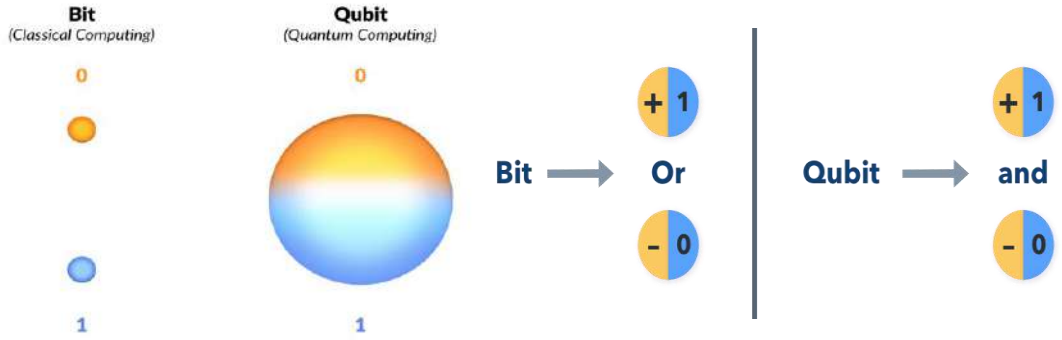
1.6 Κβαντική Πληροφορία - Κβαντικά bit

Στους κβαντικούς υπολογιστές η μονάδα πληροφορίας είναι το κβαντικό bit (quantum bit), το οποίο γράφεται εν συντομία qubit. Το qubit είναι ένα κβαντικό σύστημα δύο καταστάσεων και συμβολίζονται με $|0\rangle$ και $|1\rangle$. Τα qubits μπορεί να υπάρχουν σε υπερθέσεις που περιέχουν συγχρόνως το 0 και το 1 καθώς αποτελούν τις οι δύο βασικές καταστάσεις. Επίσης, μπορούν να διεπλέκονται. Όσο μεγαλύτερος είναι ο αριθμός των qubit σε ένα κβαντικό υπολογιστή, τόσο μεγαλύτερος είναι ο όγκος πληροφοριών που αποθηκεύονται σε αυτά.

1.6.1 Σύγκριση bits και qubits

Η πληροφορία έχει πάντα και ένα φυσικό περιεχόμενο, δεν είναι καθαρά μαθηματική υπόθεση. Σε γενικές γραμμές η πληροφορία είναι αυτή που επηρεάζει με μη τυχαίο τρόπο την εξέλιξη οποιοδήποτε συστήματος.
Ένα qubit δεν είναι ισοδύναμο με ένα κλασικό bit.
Η διαφορά από το «κλασικό» δυαδικό ψηφίο (bit) είναι ότι ενώ το bit μπορεί να πάρει μία μόνο από τις δυνατές τιμές 0 ή 1, ενώ το qubit μπορεί να πάρει ταυτόχρονα και τις δύο τιμές ταυτόχρονα, αυξάνοντας κατακόρυφα τις υπολογιστικές δυνατότητες.

Δηλαδή τα qubits μπορούν να πάρουν την τιμή 0 ή 1 ή συγχρόνως 0 και 1, όπου είναι το άθροισμα των δύο καταστάσεων ταυτόχρονα. Αυτό οφείλεται στο γεγονός ότι στη κβαντομηχανική ένα σωματίδιο είναι δυνατό να βρίσκεται ταυτόχρονα σε πολλά σημεία στο χώρο, αλλά όταν μετρήσουμε το φυσικό του μέγεθος από εκεί και μετά ξεκινά η εξέλιξή του.



Εικόνα: 4 Αναπαράσταση και σύγκριση της πληροφορίας bit και qubit

Πηγή εικόνας: https://www.google.com/search?q=bits%20%CE%BA%CE%B1%CE%B9%20qubits&tbm=isch&tbs=rimg

Αυτό το γεγονός ανοίγει νέους ορίζοντες στην επιστήμη της πληροφορικής.

Συγκριτικά με τις πληροφορίες που είναι αποθηκευμένες στον ίδιο αριθμό bit, οι πληροφορίες σε qubits αυξάνονται εκθετικά.

Ερευνητές του Ινστιτούτου Έρευνας στον Καναδά απέδειξαν κατά την εκτέλεση πειράματος πως η πληροφορία που είναι αποθηκευμένη σε qubit μπορεί να συμπιεστεί εκθετικά χωρίς το περιεχόμενο της να υποστεί την παραμικρή απώλεια. Στο πείραμα χρησιμοποίησαν φωτόνια ως qubit μεταφέροντας την πληροφορία μέσω σπιν τους, μιας κβαντομηχανικής τους ιδιότητας, παίρνοντας διακριτές τιμές. Τελικά, με μετρήσεις ανακάλυψαν ότι 10 qubits είναι αρκετά για να αποθηκεύσουν 1.000 qubits καθώς και 2.000 qubits αρκούν για να περιγράψουν την κατάσταση ενός εκατομμυρίου qubits. Συνεπώς, ουσιαστικά επιβεβαίωσαν πως η πληροφορία που μεταφέρουν τρία qubits μπορεί να αποθηκευτεί πλήρως σε δύο qubits ενώ η διαδικασία συμπίεσης στη συνέχεια πραγματοποιείται εκθετικά.

Αυτή η ανακάλυψη αποτελεί σημαντικό παράγοντα για τη μελλοντική προοπτική των κβαντικών υπολογιστών, καθώς και την αποθήκευση της πληροφορίας.

Τι είναι όμως ένας κβαντικός υπολογιστής και ποιες είναι οι ιδιότητές του; Ποια είναι η βασική διαφορά ενός κβαντικού από έναν κλασικό υπολογιστή; Πώς μπορεί να αλλάξει και να βελτιώσει την καθημερινότητά μας ανατρεπτικά, ποια τα πλεονεκτήματα και τα μειονεκτήματά του; Ποιες είναι οι εφαρμογές του και σε ποιο σημείο βρισκόμαστε σήμερα; Ποιες είναι η προοπτική των κβαντικών υπολογιστών;

Ας δούμε κάποιες απαντήσεις σε αυτά τα ερωτήματα.

1.7 Κβαντικός Υπολογιστής – Πώς Λειτουργεί

Ο κβαντικός υπολογιστής είναι μια συσκευή που εκτελεί κβαντικούς υπολογισμούς, για την επεξεργασία δεδομένων και έχουν την ικανότητα να λύνουν πολύπλοκα προβλήματα με πολύ μεγάλη ταχύτητα από τους παραδοσιακούς υπολογιστές, χρησιμοποιώντας λιγότερη ενέργεια. Βασίζεται σε ένα διαφορετικό μοντέλο από τους κλασικούς υπολογιστές, χρησιμοποιώντας τη θεωρία και κάποιες ιδιότητες της κβαντομηχανικής, ώστε να μπορούν να επεξεργαστούν και να αποθηκεύσουν την πληροφορία. Αυτή η τεχνολογία δεν έχει καμία σχέση με τους κανονικούς υπολογιστές. Το πεδίο των κβαντικών υπολογιστών εμφανίστηκε τη δεκαετία ‘80, όπου στη συνέχεια αποκαλύφθηκε ότι οι κβαντικοί αλγόριθμοι ήταν πιο αποτελεσματικοί από τους ισοδύναμους κατά την επίλυση ενός προβλήματος.

Στους κβαντικούς υπολογιστές η βασική μονάδα εγγραφής είναι ένα κβαντικό σύστημα (δεν είναι όπως ένα κλασικό), που όπως είδαμε παραπάνω, η στοιχειώδες μονάδα πληροφορίας είναι τα qubit, που μπορούν να έχουν την τιμή από 0 και 1 και οτιδήποτε ανάμεσα. Αυτό επιτρέπει ταυτόχρονα πολλαπλάσιους υπολογισμούς τόσο πολύπλοκους, που όπως αναφέρει ένας καθηγητής φυσικής είναι σαν να περνάς από την ασπρόμαυρη τηλεόραση στις σύγχρονες LED TV.

Εικόνα: 5 Κβαντικοί υπολογιστές

Πηγή εικόνας: https://www.newsbeast.gr/weekend/arthro/3306993/ti-ine-o-kvantikos-ipologistis-ke-pos-tha-allaxi-tin-kathimerinotita-mas

Ένας άλλος καθηγητής επισημαίνει ότι η λειτουργία, η ισχύ επεξεργασίας δεδομένων ενός κβαντικού υπολογιστή, φανταστείτε ότι είναι σαν να θέσετε όλα τα δεδομένα ταυτόχρονα σε δύο διαφορετικά μέρη και αυτός ο υπολογιστής μπορεί να επεξεργάζεται ταυτόχρονα, όλα τα δεδομένα.
Ένας κβαντικός υπολογιστής είναι εντελώς καινούργια τεχνολογία και λειτουργεί με τρόπο που καμία άλλη τεχνολογία που δεν μπορεί να συγκριθεί μαζί της και δεν χρησιμοποιείται σήμερα.

Έναν κβαντικό υπολογιστή σίγουρα δεν θα τον δούμε τοποθετημένο επάνω σε ένα γραφείο, αλλά ούτε και μέσα στην τσέπη μας. Καταλαμβάνει μεγάλο όγκο και είναι πολύ ευαίσθητη τεχνολογία. Πρέπει να διατηρείται σε θερμοκρασίες κοντά στο απόλυτο μηδέν αυστηρά, ώστε να μπορεί να χρησιμοποιείται.

Όταν ακούμε ή διαβάζουμε σχετικά για έναν κβαντικό υπολογιστή αναφέρονται με υπερβολικές εκφράσεις όπως «οι κβαντικοί υπολογιστές θα αλλάξουν τον κόσμο» και ενδεχομένως θα το πετύχουν.

Τι μαγικό κάνει όμως ένας κβαντικός υπολογιστής και όλοι ψάχνουν να τον κάνουν εμπορικά βιώσιμο; Και, κυρίως, γιατί πρέπει να μας απασχολεί;

Θα δούμε στη συνέχεια πολλές απαντήσεις από όλα τα παραπάνω ερωτήματα.

1.8 Κβαντική Υπολογιστική

Ο κβαντικός υπολογισμός είναι ένας τύπος υπολογισμού που αξιοποιεί τις ιδιότητες της κβαντομηχανικής για την εκτέλεση υπολογισμών. Τέτοιες ιδιότητες είναι η υπέρθεση, η διεμπλοκή και η υπεροχή.

1.9 Βασικές Ιδιότητες Κβαντομηχανικής

• Κβαντική Υπέρθεση

Η κβαντική υπέρθεση είναι ένα θεμελιώδες φαινόμενο της κβαντομηχανικής. Κατά την υπέρθεση, δύο κβαντικές καταστάσεις προστίθενται μεταξύ τους με τρόπο που τους επιτρέπει να συνυπάρχουν ταυτόχρονα. Ουσιαστικά η κβαντική υπέρθεση είναι η ικανότητα ενός κβαντικού συστήματος να βρίσκεται ταυτόχρονα (κατά την μέτρηση ή παρατήρηση) σε πολλές καταστάσεις. Η κβαντική κατάσταση υπέρθεσης επιτρέπει σε ένα qubit να καταχωρήσει ταυτόχρονα το 0 και το 1. Δύο qubits μπορούν να καταχωρήσουν τους τέσσερις δυαδικούς αριθμούς 00, 01, 10 και 11 συγχρόνως.

Συνεπώς, όταν δύο qubits βρίσκονται σε υπέρθεση, η μέτρηση της κατάστασης του ενός qubit δεν επηρεάζει την κατάσταση του άλλου.

• Κβαντική Διεμπλοκή

Ένα από τα πιο παράξενα φαινόμενα του μικρόκοσμου αποτελεί η κβαντική διεμπλοκή. Η Κβαντική διεμπλοκή ονομάζεται εκείνο το φαινόμενο κατά το οποίο η κατάσταση δύο ή περισσότερων κβαντικών bit, δηλαδή qubit, δεν μπορεί να περιγραφεί σαν συνδυασμός των καταστάσεων του κάθε qubit ξεχωριστά. Κβαντική διεμπλοκή με λίγα λόγια είναι το φαινόμενο που επιτρέπει δύο σωματίδια να

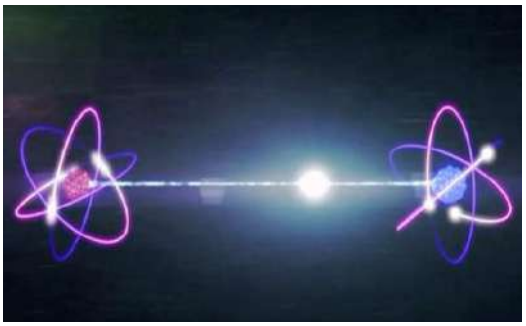
συμπεριφέρονται παρόμοια και ακαριαία, ανεξάρτητα από το πόσο μακριά βρίσκονται. Στη διεμπλοκή όταν δύο ή περισσότερα σωματίδια αλληλοεπιδρούν το ένα με το άλλο, τότε δεν είναι πλέον δυνατό να περιγραφούν χωριστά.
Συμπεριφέρονται δηλαδή ως μια ενιαία οντότητα ακόμα και αν στη συνέχεια χωριστούν από μεγάλες αποστάσεις.
Αν για παράδειγμα σταλεί ένα από τα δύο σωματίδια στο άλλο άκρο του σύμπαντος και σε ένα από τα δύο σωματίδια συμβεί κάτι, τότε το άλλο αντιδρά ακαριαία.

Επομένως, είτε η πληροφορία ταξιδεύει με άπειρη ταχύτητα, είτε στην πραγματικότητα τα δύο σωματίδια βρίσκονται ακόμα σε «επαφή» μεταξύ τους, σε κατάσταση διεμπλοκής.

Δηλαδή όταν δύο qubits βρίσκονται σε διεμπλοκή, η μέτρηση της κατάστασης του ενός qubit καθορίζει την κατάσταση του άλλου.

Συνεπώς, αν γνωρίζουμε ότι δύο qubits βρίσκονται σε διεμπλοκή και μετρήσουμε το ένα, τότε αμέσως γνωρίζουμε και την κατάσταση του άλλου qubit χωρίς να χρειάζεται να το μετρήσουμε.

Η κβαντική διεμπλοκή είναι ένα φαινόμενο υπαρκτό και παρατηρείται σε πειράματα του μικρόκοσμου, αλλά και σε μεγάλης κλίμακας. Το συγκεκριμένο φαινόμενο δεν αποτελεί καινούργια ανακάλυψη, αλλά επινοήθηκε το 1935 από τον Άλμπερτ Αϊνστάιν, στην προσπάθειά να καταρρίψει την κβαντομηχανική. Συγκεκριμένα, ο Αϊνστάιν δεν υποστήριζε ότι δεν ισχύει το συγκεκριμένο φαινόμενο, αλλά κάτι άλλο συνέβαινε στο «παρασκήνιο» που το προκαλεί και πρέπει να ανακαλυφθεί. Την ίδια χρονολογία στο περιοδικό Naturwissenschaften ο Έρβιν Σρέντιγκερ επινοεί τον όρο «εμπλοκή» και αναπτύσσει το περίφημο νοητικό «πείραμα της γάτας», η γάτα που υπάρχει ταυτόχρονα σε μια κατάσταση ζωντανή και νεκρή.



Εικόνα:6 Το φαινόμενο της κβαντικής διεμπλοκής

Πηγή εικόνας:
http://physics4u.gr/blog/2017/02/11/k%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%AE-%CE%B4%CE%B9%CE%B5%CE%BC%CF%80%CE%BB%CE%BF%CE%BA%CE%AE-%CF%84%CE%BF-%CF%80%CE%B9%CE%BF-%CF%80%CE%B1%CF%81%CE%AC%CE%BE%CE%B5%CE%BD%CE%BF-%CF%86%CE%B1%CE%B9/

Αξίζει να αναφέρουμε ότι Κινέζοι επιστήμονες το Φεβρουάριο του 2020 ανακοίνωσαν ότι πέτυχαν το φαινόμενο της κβαντικής διεμπλοκής ανάμεσα σε δύο κβαντικές μνήμες, σε απόσταση ρεκόρ 50 χιλιομέτρων, φέρνοντας πιο κοντά το κβαντικό Διαδίκτυο. Κβαντικές μνήμες σημαίνει όταν δύο νέφη ατόμων αποθηκεύουν κβαντικές πληροφορίες. Το επίτευγμα αυτό μπορεί να ανοίξει δρόμο μελλοντικά, αλλάζοντας τον τρόπο που λειτουργεί το Διαδίκτυο, δηλαδή να δημιουργηθεί ένα νέου είδους δικτύου, χρησιμοποιώντας στοιχεία του κβαντικού κόσμου για τη μετάδοση δεδομένων σε γιγαντιαίες αποστάσεις σε περισσότερους κόμβους, με εντυπωσιακές ταχύτητές και ασφάλεια.
Η κβαντική τηλεμεταφορά μέσω δικτύων οπτικών ινών έχει τη δυνατότητα να βελτιώσει σημαντικά την ασφάλεια και την ταχύτητα των συνδέσεων του Διαδικτύου. Κατά τη μετάδοση και την ανταλλαγή πληροφοριών στην καθημερινή μας ζωή, η

ασφάλεια των δεδομένων είναι εξαιρετικά απαραίτητη. Αξιοποιώντας τα κβαντικά κρυπτογραφημένα συστήματα τα οποία δημιουργούν ένα κλειδί κρυπτογράφησης, τα οποία προστατεύουν εντυπωσιακά το περιεχόμενο των δεδομένων από ανεπιθύμητους χρήστες, εξασφαλίζοντας ένα ασφαλές κανάλι επικοινωνίας μεταξύ των κατόχων. Οι διάφορες μέθοδοι κβαντικής κρυπτογράφησης στην ουσία είναι παραλλαγές από το φαινόμενο της κβαντικής διεμπλοκής και το θεώρημα μη κλωνοποίησης.

1.10 Θεώρημα μη Κλωνοποίησης (no-cloning theorem)

Το θεώρημα της μη κλωνοποίησης δηλώνει την κατάσταση ενός qubit που βρίσκεται σε μια τυχαία κατάσταση και είναι αδύνατο να δημιουργήσουμε ένα αντίγραφό του.

Το θεώρημα αυτό στη φυσική, σημαίνει ότι δεν μπορεί να δημιουργηθεί ένα ανεξάρτητο και πανομοιότυπο αντίγραφο μιας αυθαίρετης κατάστασης που δεν είναι γνωστή. Αυτή η κατάσταση έχει σημαντικές επιδράσεις στον τομέα του κβαντικού υπολογισμού μεταξύ άλλων.

Θα μπορούσαμε να πούμε ότι το πρόβλημα βρίσκεται στο μετασχηματισμό μιας κατάστασης σε μια άλλη. Δηλαδή αν προσπαθήσουμε να αντιγράψουμε το qubit, καταστρέφεται το αρχικό αντίγραφο, ισχυρίστηκε ο Πομπίλι, αναφερόμενος στο θεώρημα μη κλωνοποίησης της Φυσικής.

Αυτό οφείλεται στο εκπληκτικό **θεώρημα της μη κλωνοποίησης**, όπου απαγορεύεται η αντιγραφή.

Ας δούμε ένα παράδειγμα.

Η Αλίκη επιδιώκει να επικοινωνήσει με ακεραιότητα με τον Μπομπ όμως ο Τσάρλι προσπαθεί να κρυφακούσει τη συζήτησή τους. Λαμβάνουμε ως δεδομένο ότι η Αλίκη χρησιμοποιεί qubits αντί για bit. Αν ο Τσάρλι καταφέρει τελικά να υποκλέψει τη συζήτησή τους, δεν μπορεί να την αποθηκεύσει και να δημιουργήσει κάποιο όμοιο αντίγραφό της. Η επέμβαση όμως του Τσάρλι γίνεται αντιληπτή από την Αλίκη και τον Μπομπ.

Στην ουσία η κβαντική πληροφορία, το qubit εξαφανίζεται στην πλευρά του αποστολέα (από την Αλίκη) και εμφανίζεται ξαφνικά αστραπιαία στην πλευρά του παραλήπτη (στον Μπομπ) πετυχαίνοντας το φαινόμενο της τηλεμεταφοράς της πληροφορίας.

Το θεώρημα της μη κλωνοποίησης συντάχθηκε το 1970 από τον Τζέιμς Παρκ και αποτελεί μια εξέλιξη του θεωρήματος «no-go».

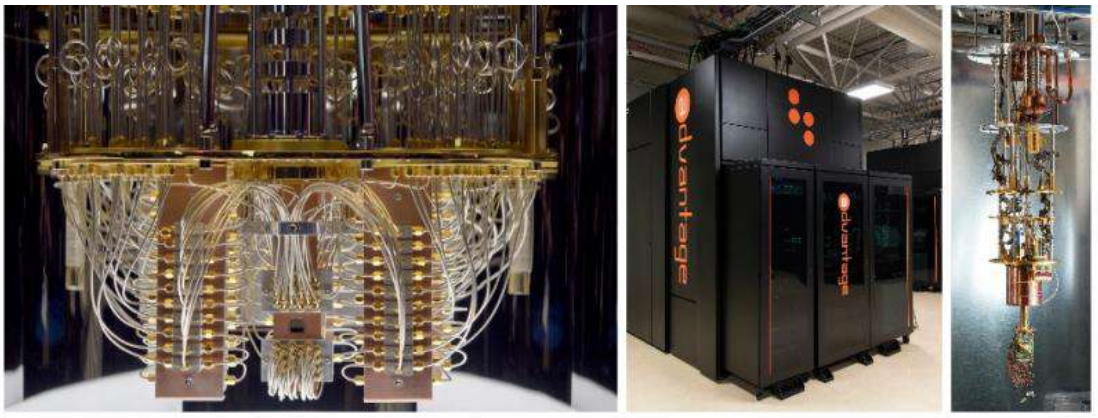
• Κβαντική Υπεροχή

Ο όρος κβαντική «υπεροχή» επινοήθηκε το 2012 από ένα φυσικό καθηγητή του Ινστιτούτου Τεχνολογίας της Καλιφόρνιας, τον Τζον Πρέσκιλ.

Κβαντική υπεροχή είναι η δημιουργία ενός κβαντικού υπολογιστή, που καταφέρνει να πραγματοποιήσει υπολογισμούς ταχύτερα, που ξεπερνούν τις δυνατότητες και του ισχυρότερου υπερ-υπολογιστή που κυκλοφορεί στην αγορά σήμερα, λύνοντας ένα συγκεκριμένο πρόβλημα. Συνεπώς, μπορούμε να πούμε ότι η κβαντική υπεροχή αποτελεί την απόδειξη πως ένας κβαντικός υπολογιστής μπορεί να εκτελέσει υπολογισμούς που ένας κλασικός υπολογιστής θα χρειαζόταν χρόνια για να τα καταφέρει.

Επίσημη δημοσίευση της Google επιβεβαιώνει ότι πέτυχε την κβαντική υπεροχή με την εμφάνιση του κβαντικού επεξεργαστή Sycamore, που τελικά λειτούργησε με 53 qubits, ο οποίος εκτέλεσε σε 3 λεπτά και 20 δευτερόλεπτα ένα υπολογισμό, που ο κορυφαίος παγκοσμίως αμερικάνικος υπερυπολογιστής Summit θα χρειαζόταν 10.000 χρόνια περίπου να λύσει. Ο επεξεργαστής για να πετύχει γρηγορότερες ταχύτητες επεξεργασίας οι οποίες είναι ασύλληπτες για ένα συμβατικό υπολογιστή, πρέπει να αξιοποιήσει τα κβαντικά φαινόμενα της υπέρθεσης και της διεμπλοκής.

Από την άλλη μεριά η IBM, έχει αναπτύξει δικό της κβαντικό επεξεργαστή των 53 qubits πανίσχυρο αλλά και συμβατικό με αυτό που δημιούργησε η Google και θα έλυνε το πρόβλημα σε δυόμισι μέρες και όχι σε 100.000 χρόνια.



Το 2001 η IBM δημιούργησε τον πρώτο κβαντικό επεξεργαστή και, το 2017, παρουσίασε έναν επεξεργαστή 20 qubit (αριστερή φωτογραφία). Ομως, το 2018, τα πρωτεία έκλεψε η Google δημιουργώντας τον «Bristlecone» (72 qubit). Σήμερα, ένας από τους πιο ισχυρούς κβαντικούς επεξεργαστές είναι ο D-Ware-Two (512 qubit) (δεξιά)

Εικόνα: 6

Πηγή εικόνας: https://www.efsyn.gr/epistimi/mihanes-toy-noy/333817_apo-toys-psifiakoys-stoys-kbantikoys-ypologistes

Επίσης, η Κίνα αναφέρει ότι ο κβαντικός υπολογιστής που δημιούργησε με όνομα Jiuzhang έλυσε ένα πρόβλημα μέσα σε 200 δευτερόλεπτα που κατά τον ισχυρισμό της ένας κλασικός υπολογιστής για να το υλοποιήσει αυτό θα χρειαζόταν 2,5 δισεκατομμύρια χρόνια. Δηλαδή ο αλγόριθμός τους αυτός μπορεί να είναι «μαζικά πιο αποτελεσματικός από τις υπάρχουσες μεθόδους» και ισχυρίζονται ότι οι κλασικοί υπολογιστές για συγκεκριμένες εργασίες είναι απίθανο να ανταγωνιστούν τα κβαντικά μηχανήματα, αφού θα εμφανίσουν συντριπτικά πλεονεκτήματα έναντι των κλασικών υπολογιστών.

Έτσι, στο πεδίο των κβαντικών υπολογιστών, υπάρχει παγκόσμιος ανταγωνισμός ανάμεσα στις χώρες ΗΠΑ, Ευρώπη, Κίνα και άλλες χώρες, οι οποίες προσπαθούν να πετύχουν την κβαντική υπεροχή στο έπακρο, χρησιμοποιώντας τα οφέλη του

κβαντικού υπολογιστή σε επίπεδο τεχνολογίας, οικονομίας, επιστήμης και εθνικής ασφάλειας.

1.11 Η Αναγκαιότητα των Κβαντικών Υπολογιστών στη Ζωή μας

Εικόνα: 7 Η κβαντική υπολογιστική στην καθημερινότητά μας

Πηγή εικόνας: https://www.ylaouris.com/%CE%B5%CF%80%CE%B9%CF%84%CF%85%CF%87%CE%AF%CE%B5%CF%82-%CF%83%CF%84%CE%B7%CE%BD-%CF%80%CF%BB%CE%B7%CF%81%CF%BF%CF%86%CF%BF%

Μια τρομερή τεχνολογική και επιστημονική επανάσταση αναμφισβήτητα αποτελεί τον ερχομό του κβαντικού υπολογιστή που βασίζεται στη σχεδίαση μιας εντελώς διαφορετικής αρχιτεκτονικής σε σχέση με τους κλασικούς υπολογιστές (οι οποίοι στηρίζουν την λειτουργία τους στα τρανζίστορ). Η αναγκαιότητα των κβαντικών υπολογιστών στη ζωή μας, αποτελεί έναν βασικό λόγο για τον οποίο το μέγεθος των τρανζίστορ πάνω στα τσιπ, πλησιάζει το μέγεθος όπου η κβαντομηχανική γίνεται κύριος παράγοντας ρύθμισης της μεταφοράς των ηλεκτρονίων στο τσιπ.

Παράγοντας που σχετίζεται με την πρακτική διαπίστωση του νόμου του Moore, επαληθεύοντας ότι ο αριθμός των τρανζίστορ ενός ολοκληρωμένου κυκλώματος διπλασιάζεται κάθε δύο χρόνια. Δηλαδή η χωρητικότητα της μνήμης των κλασικών υπολογιστών κάθε δύο χρόνια διπλασιάζεται. Έτσι, το γεγονός ότι η βασική μονάδα μνήμης δεν είναι επαρκής, είναι αδύνατον να συμπυκνώσουμε περισσότερο τα ηλεκτρονικά μέρη πάνω στα υποστρώματα πυριτίου στους κλασικούς υπολογιστές. Επίσης, η ανάγκη της αύξησης υπολογιστικής ισχύος και η αναγκαιότητα για την επίλυση κάποιων προβλημάτων γρηγορότερα από έναν κλασικό υπολογιστή, οι οποίοι θα χρειαζόταν αιώνες για να τις εκτελέσουν ή θα έσβηναν πριν ολοκληρώσουν την προσπάθειά τους. Επομένως, η αναγκαιότητα για έρευνα και εξέλιξη της τεχνολογίας της πληροφορικής των κβαντικών υπολογιστών, είναι θέμα χρόνου στις επόμενες δεκαετίες να εφαρμοστούν αναπόφευκτα οι κανόνες της κβαντομηχανικής.

Ας δούμε ποια είναι τα πλεονεκτήματα των κβαντικών υπολογιστών.

1.12 Πλεονεκτήματα των Κβαντικών Υπολογιστών

Οι κβαντικοί υπολογιστές όσον αφορά την υπολογισιμότητά τους δεν πλεονεκτούν έναντι των κλασικών υπολογιστών. Δηλαδή ένας κβαντικός υπολογιστής δεν σημαίνει ότι μπορεί να λύνει πιο δύσκολα προβλήματα από έναν κλασικό υπολογιστή, αλλά μπορεί να δώσει αναμφισβήτητα πιο γρήγορη λύση. Έτσι, τα ουσιαστικά πλεονεκτήματα των κβαντικών υπολογιστών από τους κλασικούς υπολογιστές είναι η **μεγάλη ταχύτητα**, η **τεράστια μνήμη** και η **αποτελεσματική δυνατότητα επίλυσης ορισμένων πολύπλοκων προβλημάτων.**

Συνεπώς, όταν αναφερόμαστε στα πλεονεκτήματα των κβαντικών συστημάτων, προφανώς δεν αναφερόμαστε σε αισθητικά κριτήρια, αλλά στις δυνατότητες των κβαντικών, οι οποίες θα υπερβαίνουν κατά πολύ τις αντίστοιχες δυνατότητες των μηχανημάτων που χρησιμοποιούμε σήμερα. Για να αντιληφθούμε καλύτερα την πάρα πολύ γρήγορη ταχύτητα των κβαντικών υπολογιστών, ας σκεφτούμε ότι θα είναι 100 εκατομμύρια φορές ταχύτεροι από το γρηγορότερο laptop που κυκλοφορεί στην αγορά, γι’ αυτό και η λειτουργία τους βασίζεται σε κβαντομηχανικά φαινόμενα.

Έτσι, οι κβαντικοί υπολογιστές έχουν την ικανότητα να λύνουν πολύπλοκα προβλήματα, απίστευτα ταχύτερα από τους συμβατικούς υπολογιστές. Η τεχνολογία αυτή δεν έχει καμία σχέση με τους κανονικούς υπολογιστές. Αν και η σχετική πρακτική τεχνολογία των κβαντικών υπολογιστών, βρίσκεται σε πειραματικό στάδιο ανάπτυξης, τα αποτελέσματα των σχετικών πειραμάτων είναι ενθαρρυντικά που θα τα δούμε στο όχι και τόσο μακρινό μέλλον.

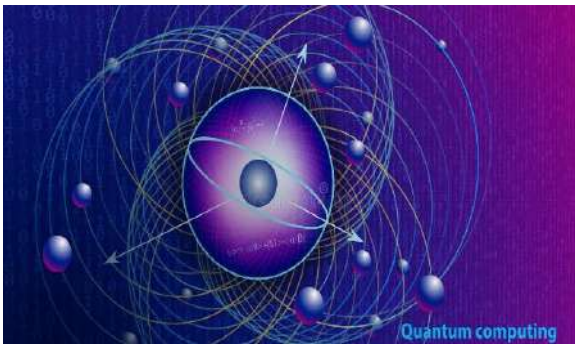
1.13 Προβλήματα των Κβαντικών Υπολογιστών

Υπάρχουν πολλοί παράγοντες που στέκονται εμπόδιο στην ανάπτυξη των κβαντικών υπολογιστών. Το κυριότερο πρόβλημα στη δημιουργία των κβαντικών υπολογιστών είναι η εμφάνιση κάποιων σφαλμάτων και άρα απαιτείται η αντιμετώπιση τους. Τα κβαντικά συστήματα είναι ιδιαίτερα ευαίσθητα στην αλληλεπίδρασή τους με το περιβάλλον τους, καθώς επηρεάζονται οι ιδιότητές τους. Συγκεκριμένα, οι αλληλεπιδράσεις που δημιουργούνται μεταξύ του περιβάλλοντος και των qubits έχει ως αποτέλεσμα τη διακοπή των πληροφοριών που αποθηκεύονται στον κβαντικό υπολογιστή και επομένως λάθη στον υπολογισμό. Έτσι, επειδή μέχρι σήμερα οι κβαντικοί επεξεργαστές για να δουλεύουν επιτυχώς, χωρίς να επηρεάζονται από τους «θορύβους» του φυσικού περιβάλλοντος, πρέπει να τοποθετούνται μέσα σε ειδικούς καταψύκτες και να δουλεύουν σε θερμοκρασίες κοντά στο απόλυτο μηδέν (-273 ^{o} C). Διαφορετικά, αν οι κβαντικοί υπολογιστές επηρεάζονται από τους θορύβους κατά την εκτέλεση των υπολογισμών τους, τότε τα αποτελέσματά τους θα είναι λανθασμένα. Τέλος, ένα επιπλέον εμπόδιο για την χρήση της γιγαντιαίας υπολογιστικής ισχύς των κβαντικών υπολογιστών είναι ότι εργάζονται με πολύπλοκους αλγόριθμους, που κάποιες φορές οι πληροφορίες που κωδικοποιούν τα qubits είναι δυσανάγνωστες.

Οι επιστήμονες ερευνούν και προσπαθούν να χρησιμοποιήσουν συστήματα που να έχουν πολύ μικρή αλληλεπίδραση με το περιβάλλον καθώς και να εφευρίσκουν αλγορίθμους που να ελαττώνουν τα σφάλματα. Έτσι προς το παρόν, οι κβαντικοί υπολογιστές έχουν μπροστά τους πολλή έρευνα, όμως παρά τα τεχνολογικά εμπόδια η είσοδος των κβαντικών υπολογιστών είναι αναπόφευκτη.

1.14 Κβαντικοί Αλγόριθμοι – Shor, Grover και Deutsch

Ένας κβαντικός αλγόριθμος είναι ένας αλγόριθμος που εκτελείται σε έναν μοντέλου κβαντικού υπολογιστή. Είναι δηλαδή μια διαδικασία βημάτων ή εντολών που χρησιμοποιούνται για την επίλυση ενός προβλήματος και μπορούν να εκτελεσθούν σε έναν κβαντικό υπολογιστή. Αυτό που κάνει τους κβαντικούς αλγόριθμους ενδιαφέροντες και να υπερτερούν σημαντικά έναντι των κλασικών αλγορίθμων, είναι ότι επιλύουν ορισμένα προβλήματα γρηγορότερα από τους κλασικούς αλγόριθμους.



Εικόνα: 8 Απεικόνιση μεγάλων αλγορίθμων
Αφηρημένη τεχνική εικόνα
Πηγή εικόνας:
https://www.google.com/search?q=%CF%86%CF%89%CF%84%CE%BF+%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%AE+%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CE%B9%CE%BA%CE%AE&sxsrf=ALiCzsb8voZJ1FVcQB-Sz4VbhzT1n_y1cw

Οι κβαντικοί υπολογιστές χρησιμοποιούν αλγορίθμους λόγω της ύπαρξης κάποιων σφαλμάτων, οι οποίοι είναι κωδικοποιημένοι σε κβαντική λογική και φυσικά δεν είναι εφικτό να τρέξουν σε κλασικούς υπολογιστές, αλλά κάποιες φορές να προσομοιωθούν. Επίσης, χρησιμοποιούνται για να διασφαλίσουν την ακεραιότητα και τη ασφάλεια των πληροφοριών από κακόβουλους χρήστες.

Ας δούμε τρεις κβαντικούς αλγόριθμους, το **Shor, το Grover** και τον **Deutsch**.

Ο αλγόριθμος **Deutsch** αποτελεί έναν από τους πρώτους κβαντικούς αλγορίθμους που προτάθηκε, δηλαδή ένας αλγόριθμος που μπορεί να τρέξει σε έναν κβαντικό υπολογιστή, εκτελώντας υπολογισμούς που είναι αδύνατο να εκτελεσθούν σε έναν κλασικό υπολογιστή.

Ο αλγόριθμος **Shor** έδωσε σημαντική ώθηση στο πεδίο των κβαντικών υπολογισμών καθώς υπήρξε αργή μέχρι το 1994. Ο Peter Shor δημοσιεύοντας μια εργασία αναβάθμισε τον κλάδο της πληροφορικής, αναπτύσσοντας έναν κβαντικό αλγόριθμο, τον αλγόριθμο του Shor. Είναι ένας πολυωνυμικός αλγόριθμος που χρησιμοποιείται για την παραγοντοποίηση μεγάλων αριθμών. Ο αλγόριθμο του Shor είναι πάρα πολύ σημαντικός, γιατί ο αριθμός των βημάτων που χρειάζεται για να παραγοντοποιήσει έναν αριθμό είναι πολύ μικρότερος από τον κλασικό και συνιστάται στην παραγοντοποίηση μεγάλων αριθμών, γρηγορότερα από τους συμβατικούς

υπολογιστές. Ο αλγόριθμος αυτός είναι άμεσα εφαρμόσιμος σε κρυπτογραφικές επιθέσεις.

Δύο χρόνια μετά τη δημοσίευση του αλγορίθμου του Shor, το 1996 εμφανίστηκε ο αλγόριθμος του **Lov Grover**. Αν και ο αλγόριθμος Grover δεν είναι τόσο αποτελεσματικός σε θέματα ευαισθησίας και ταχύτητας όπως ο αλγόριθμος του Shor, παρόλα αυτά αποτελεί έναν σημαντικό αλγόριθμο, γιατί είναι αποδοτικότερος από οποιοδήποτε κλασικό αλγόριθμο. Ο αλγόριθμος Grover μπορεί να ψάξει και να ταξινομήσει μια βάση δεδομένων που είναι μη ταξινομημένη πολύ γρηγορότερα από έναν κλασικό υπολογιστή.

1.15 Η Προοπτική και οι Εφαρμογές των Κβαντικών Υπολογιστών

Η κβαντική υπολογιστική έχει φέρει τεράστια πρόοδο όχι μόνο στη Φυσική, αλλά σε πολλούς κλάδους της επιστήμης και μπορούν τα μεταμορφώσουν πεδία όπως είναι η ιατρική, η ασφάλεια πληροφοριών, η κρυπτανάλυση, η τεχνητή νοημοσύνη, φαρμακευτική και πολλούς άλλους τομείς.

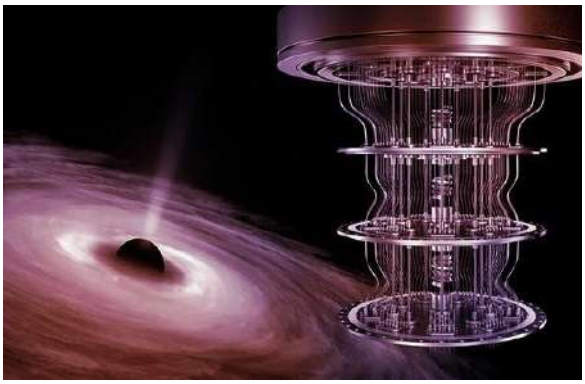
Οι μελλοντικές εφαρμογές των κβαντικών υπολογιστών απασχολούν σίγουρα διαφορετικά επιστημονικά πεδία, από την εντυπωσιακή εξέλιξη των υπολογιστικών διεργασιών στην οικονομία, από την αυτόματη μάθηση, την ιατρική, μέχρι και την τεχνητή νοημοσύνη. Γενικά, οι εφαρμογές της κβαντικών συστημάτων μπορούν να χρησιμοποιηθούν σε επιστημονικές και βιομηχανικές εφαρμογές, καθώς είναι απρόβλεπτες και ραγδαίες, για το λόγο ότι τα τελευταία χρόνια οι εφαρμογές αυτές παρουσιάσανε εντυπωσιακή ανάπτυξη, ανάλογη με αυτή που παρουσίασαν οι πρώτοι μικροεπεξεργαστές τη δεκαετία του 1960.

Ακόμα, στο πεδίο της ιατρικής θα μπορούσε να συμβάλλει αποφασιστικά στην ανάπτυξη νέων υλικών και φαρμάκων, στην ιατρική διαγνωστική θεραπεία, στη μοντελοποίηση των πολύπλοκων οικολογικών, γεωλογικών και πλανητικών φυσικοχημικών συστημάτων.

Αξίζει να επισημάνουμε ότι οι κβαντικοί υπολογιστές δεν είναι κατάλληλοι για όλες τις υπολογιστικές διεργασίες. Για παράδειγμα, δεν είναι κατάλληλες για να επιταχύνουν την επεξεργασία κειμένου ή την πλοήγηση στο διαδίκτυο.

Ένας θεωρητικός φυσικός της εποχής μας, ο Jon Phillip Preskill και καθηγητής του πανεπιστημίου της Καλιφόρνιας καθώς και διευθυντής του Ινστιτούτου Τεχνολογικής Πληροφορίας και Ύλης, αναφέρεται στην δυνατότητα των κβαντικών υπολογιστών να δώσουν απαντήσεις σε μερικά δύσκολα προβλήματα.

Τέτοια είναι στην φυσική, όπως σχετικά με την φύση του χώρου και του χρόνου, αλλά και την επίλυση δύσκολων προβλημάτων σε πληθώρα επιστημονικών πεδίων. Σε συνέντευξή του δήλωσε ότι η ανάπτυξη των κβαντικών υπολογιστών βρισκόμαστε σε βρεφικό στάδιο· παρόλα αυτά διαθέτουμε στις μέρες μας κβαντικούς υπολογιστές που διαθέτουν ασύλληπτες δυνατότητες.



Εικόνα: 9 Το μέλλον των κβαντικών υπολογιστών

Πηγή εικόνας: https://physicsgg.me/2022/03/26/%CE%BF%CE%B9-%CE%BC%CE%B5%CE%BB%CE%BB%CE%BF%CE%BD%CF%84%CE%B9%CE%BA%CE%AD%CF%82-%CE%B5%CF%86%CE%B1%CF%81%CE%BC%CE%BF%CE%B3%CE%AD%CF%82-%CF%84%CF%89%CE%BD-%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA/

Γι’ αυτόν ακριβώς τον λόγο, τα τελευταία χρόνια προσπαθούνε ισχυρές επιχειρήσεις όπως η Google, η IBM, Intel και η Microsoft επενδύοντας δισεκατομμύρια δολάρια στην ανάπτυξή τους, κάνοντας αγώνα δρόμου να αναδειχθούν πρώτες στο πεδίο της κβαντικής πρωτοπορίας.

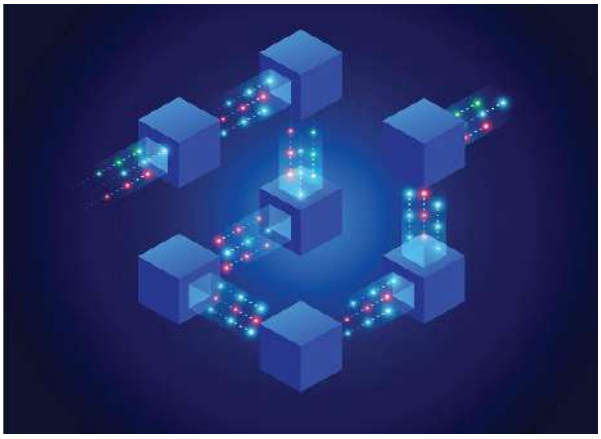
Μια πολύ σημαντική εφαρμογή των κβαντικών υπολογιστών που είναι ήδη πραγματικότητα, είναι η κβαντική κρυπτογραφία, που παρέχει ασφαλή επικοινωνία και με προδιαγραφές απλησίαστες για τους κλασικούς ψηφιακούς υπολογιστές. Οι κβαντικοί υπολογιστές είναι δυνατόν να διαπεράσουν γρήγορα στις τεχνικές κρυπτογράφησης, ενώ ο καταλληλότερος συμβατός υπολογιστής θα χρειαζόταν χρόνια για το πετύχει. Στο μέλλον θα είναι η αξιοποίηση των κβαντικών μηχανημάτων για την προστασία απόρρητων και ευαίσθητων δεδομένων. Αυτό οφείλεται στο γεγονός ότι η κβαντική τεχνολογία παρέχει πολύ μεγάλη ασφάλεια, με αποτέλεσμα να είναι πάρα πολύ δύσκολη η πρόσβαση από κακόβουλους σε τραπεζικούς λογαριασμούς και emails χρηστών διαδικτυακά. Καθώς υπάρχει ένας κβαντικός αλγόριθμος δεδομένων αναζήτηση πληροφορίας στο διαδίκτυο, θα πραγματοποιείται πάρα πολύ γρήγορα.

Θα μπορούσε ένα κβαντικό σύστημα να «σπάσει» από έναν κβαντικό υπολογιστή;

Όχι, τα δεδομένα είναι προστατευμένα, γιατί τα κβαντικά κρυπτογραφημένα συστήματα δεν θα ήταν εφικτό να «σπάσουν» από κβαντική τεχνολογία.

Έτσι, καθώς αναπτύσσονται οι κβαντικοί υπολογιστές και γίνονται όλο και περισσότερο ακαταμάχητοι, θα μπορούσαν να σπάσουν τη σημερινή κρυπτογράφηση (συμβατική κρυπτογράφηση), αποκαλύπτοντας ευαίσθητα δεδομένα. Οι κβαντικοί υπολογιστές θα μπορούσαν να υπονομεύσουν και τα κρυπτονομίσματα χρησιμοποιώντας επίσης τη σημερινή τεχνολογία της κρυπτογράφησης.

Οι ερευνητές προσπαθούν εδώ και καιρό να δημιουργήσουν ένα νέο είδος δικτύου που θα χρησιμοποιεί στοιχεία του κβαντικού κόσμου για τη μετάδοση δεδομένων με εξαιρετικά γρήγορες ταχύτητες και με ασφάλεια. Το κβαντικό δίκτυο λειτουργεί όπως και το παραδοσιακό διαδίκτυο, προσφέροντας τεράστιες δυνατότητες.



Εικόνα: 10 Κβαντικό δίκτυο

Πηγή εικόνας: https://hellas-now.com/kvantiki-diemploki-neo-epiteygma-allazei-ton-tropo-leitoyrgias/?fbclid=IwAR0G7SRrZi5GXm8LjfSkOdecLeoSuNGVQwKYS738SFbwY4V3UwC1ksTVU9A

Η ανταλλαγή πληροφοριών πραγματοποιείται μέσω qubits, μεταξύ των κβαντικών επεξεργαστών, σε τεράστιες ποσότητες και σε εντυπωσιακά μεγάλες αποστάσεις. Η κβαντική τηλεμεταφορά μέσω των οπτικών ινών, συμβάλει αποτελεσματικά στην ακεραιότητα των δεδομένων καθώς και τη γρηγορότερη σύνδεση του Διαδικτύου.

Γνωρίζουμε ότι η τεχνητή νοημοσύνη και οι κβαντικοί υπολογιστές θεωρούνται δύο πολύ σπουδαίοι τομείς της ανθρωπότητας. Χάρη στη σημαντική δύναμη που μπορεί να συμβάλλει αισθητά στη θετική εξέλιξη όλων των πεδίων του ανθρώπινου πολιτισμού οι κυβερνήσεις αποσκοπούν στην χρήση αυτών των πολύτιμων εργαλείων. Ωστόσο αν βρεθεί τρόπος να συνεργαστούν αυτά τα δύο επιστημονικά πεδία, με την υπερβολική τους χρήση, τα αποτελέσματά τους μπορεί να επιφέρει μεγάλους κινδύνους, να είναι καταστρεπτικά καθώς και να αποτελέσουν απειλή για την ανθρωπότητα.

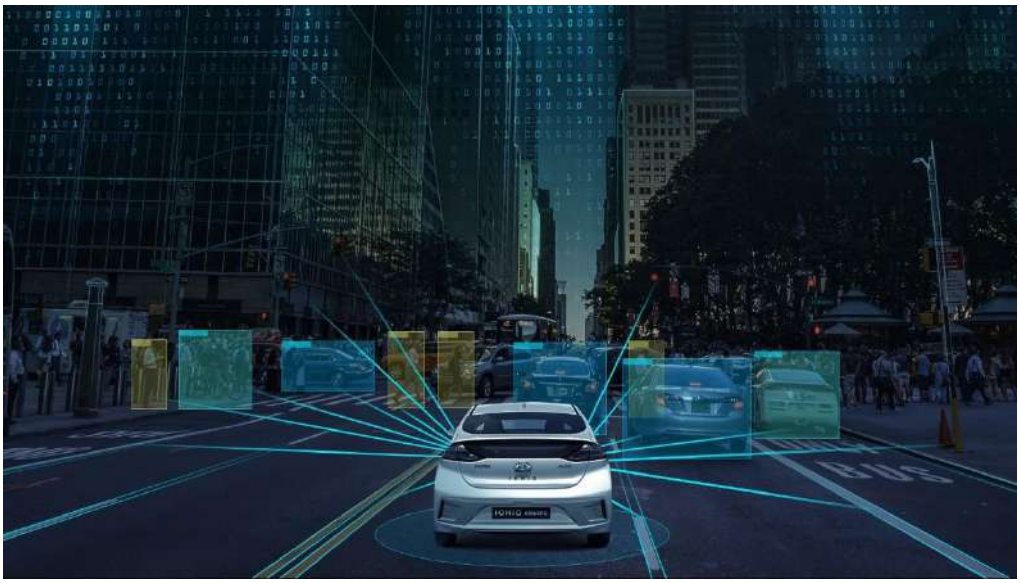
Ακόμη, η βελτίωση της χρήση GPS συστημάτων στην καθημερινή μας ζωή, που χρησιμοποιούνται σε αυτοκίνητα για την ανίχνευση μιας θέσης προς αναζήτηση, τα οποία βασίζονται στις ιδιότητες της κβαντομηχανικής. Μέσω των κβαντικών υπολογιστών θα υπάρξει βελτίωση των ρυθμίσεων και συνεπώς πιο αξιόπιστα αποτελέσματα.

Αξίζει να αναφέρουμε δύο εταιρείες η Hyundai και η IonQ (η οποία έχει προσεγγίσει τα καλύτερα και τεχνικά μυαλά οποιασδήποτε επιχείρησης κβαντικών υπολογιστών μέχρι σήμερα), δημοσίευσαν τη συνεργασία τους για τη δημιουργία ενός νέου προγράμματος, με στόχο την ανάπτυξη και τη βελτίωση της υπολογιστικής λειτουργικότητας, βασιζόμενοι στις αρχές της κβαντομηχανικής σε κβαντικούς υπολογιστές, που αποτελούν τα θεμελιώδη βήματα της μελλοντικής κινητικότητας όπως των αυτονόμων οχημάτων.

Συγκεκριμένα, θα επιδιώξουν την ανάπτυξη κβαντικών τεχνικών σε ταξινόμηση εικόνων, αναγνώριση οδικών πινακίδων και την ανίχνευση τρισδιάστατων αντικειμένων όπως ποδήλατα και πεζούς. Στο πλαίσιο αυτού του προγράμματος η χρήση των κβαντικών επεξεργαστών, έχουν σκοπό την επεξεργασία τεράστιων όγκο πληροφοριών σε γιγαντιαίες ταχύτητες και με πάρα πολύ μεγάλη ακρίβεια από τα κλασικά συστήματα.

Ας φανταστούμε έναν υπολογιστή, ο οποίος θα «σπάσει» κρυπτογραφημένα μηνύματα σε δευτερόλεπτα, κάτι που οι συμβατοί υπολογιστές θα χρειάζονταν αιώνες. Ή ας φανταστούμε ένα υπολογιστή, ο οποίος σε μια στιγμή ακαριαία θα μπορούσε να ανιχνεύσει όλα τα πιθανά φάρμακα για μια ασθένεια και να αναζητήσει το καταλληλότερο φάρμακο.

Η΄ ας φανταστούμε έναν υπολογιστή που μπορεί να ελέγχει την κίνηση των αυτοκινήτων και σε κλάσματα δευτερολέπτου να καταφέρει ρυθμίζοντας τα φανάρια, να μην υπάρχει μποτιλιάρισμα οπουδήποτε. Ας φανταστούμε ένα υπολογιστή, ο οποίος να αναλύει όλες τις λειτουργίες, τις παραλλαγές, τις εκδοχές του ίδιου του εαυτού του, σχεδιάζοντας έναν καινούργιο, αποδοτικότερο υπολογιστή, που να υπερέχει. Και άλλα πολλά ας φανταστούμε. Αυτή είναι η μαγεία και το μεγαλείο των κβαντικών υπολογιστών, η ταχύτητά τους όπου στο μέλλον μπορούν να μας επιφυλάξουν πάρα πολλά.



Εικόνα: 11 Αυτόνομα αυτοκίνητα ανιχνεύοντας αντικείμενα βασισμένα σε κβαντικούς υπολογιστές

Πηγή εικόνας: https://www.4troxoi.gr/epikairotita/kosmos/hyundai-ionq/

Η Amazon Braket, μια πλήρως διαχειριζόμενη υπηρεσία κβαντικής υπολογιστικής σχεδιασμένη για να ενισχύει την επιστημονική έρευνα και ανάπτυξη λογισμικού για κβαντικούς υπολογιστές, ανακοίνωσε πρόσφατα ότι θα παρέχει το δικαίωμα και την πρόσβαση σε οργανισμούς στην υπηρεσία της. Η εταιρεία θα χρεώνει τους πελάτες για τις παροχή υπηρεσιών της που θα έχουν πρόσβαση σε κβαντικά υπολογιστικά συστήματα κατασκευασμένα από τις εταιρείες της Rigetti, IonQ και D-Wave, μόνο για τους υπολογιστικούς πόρους που θα χρησιμοποιούν.

Αν επαληθευτούν από όλα αυτά έστω και ένα μέρος, τότε οι κβαντικοί υπολογιστές και οι νέες κβαντικές τεχνολογίες θα φέρουν τη δεύτερη κβαντική επανάσταση που σίγουρα στο μέλλον δεν θα επηρεάσει την τεχνολογία, την υγεία, την οικονομία, την ασφάλεια πληροφοριών αλλά και την καθημερινότητα των ανθρώπων.

Παρά το πρώιμο ερευνητικό στάδιο ανάπτυξης, οι κβαντικοί υπολογιστές με την εμφάνισή τους, θεωρούνται ένα εποικοδομητικό εργαλείο της επιστήμης της τεχνολογίας καθώς θέτουν τα θεμέλια της εξέλιξης της ανθρωπότητας και της επιστήμης. Σύμφωνα με τις μελέτες που έχουν διεξαχθεί, τα εντυπωσιακά αποτελέσματα αναμένονται αρκετά ενθαρρυντικά και αναμφισβήτητα αποτελούν πρόκληση για συνεχή εξέταση σε άπειρα θεμελιώδη εμπόδια που υπάρχουν. Η

επένδυση των κβαντικών υπολογιστών είναι αποτελεσματική και πάρα πολύ κερδοφόρα, γι’ αυτό και κολοσσοί επιχειρήσεων συμμετέχουν με πολύ μεγάλο ανταγωνισμό μεταξύ τους. Ειδικοί μελετητές υποστηρίζουν ότι δεν θα κυριαρχήσουν έναντι των συμβατών υπολογιστών, αλλά θα δουλεύουν παράλληλα με αυτούς, αφού για κάποιες συγκεκριμένες εργασίες οι κλασικοί υπολογιστές είναι καταλληλότεροι.

Η κβαντική υπολογιστική είναι μια έννοια ασύλληπτη για πολλούς επιστήμονες που χρειάζεται αρκετό χρόνο σε θεωρητικό και πρακτικό επίπεδο καθώς έχουν να προσφέρουν πολλά σχετικά με τις δυνατότητες χειρισμού της πληροφορίας στο μακρινό μέλλον· έως ότου οι κβαντικοί υπολογιστές πετύχουν ευρεία εφαρμογή σε θέματα της καθημερινότητας και να οδηγήσουν σε ένα κυριολεκτικό κβαντικό άλμα, σε μια κβαντική επανάσταση που υπόσχεται να αλλάξει το μέλλον συντριπτικά.

ΚΕΦΑΛΑΙΟ: 2ο - Μαθηματικό υπόβαθρο

ΚΕΦΑΛΑΙΟ 2ο

Μαθηματικό Υπόβαθρο

Σύνοψη

Σε αυτό το κεφάλαιο πραγματεύονται διάφορες βασικές έννοιες στη γραμμική άλγεβρα, τους πίνακες και το τανυστικό γινόμενο πινάκων. Συγκεκριμένα, αναφέρονται έννοιες, ιδιότητες και τύποι για τα διανύσματα πινάκων στήλης και γραμμής, των πιθανοτήτων και των μιγαδικών αριθμών. Δίνονται πράξεις πρόσθεσης, αφαίρεσης και πολλαπλασιασμού μεταξύ πινάκων. Στη συνέχεια ακολουθούν παραδείγματα πειράματος τύχης, ώστε να κατανοήσουν οι μαθητές την έννοια της πιθανότητας και δίνονται οι τύποι για τον υπολογισμό μιας πιθανότητας. Ακόμα, γίνεται αναφορά στην έννοια του μιγαδικού αριθμού, στις βασικές ιδιότητες και την εκτέλεση των πράξεων τους όπως είναι η πρόσθεση, η αφαίρεση, ο πολλαπλασιασμός και η διαίρεση. Τέλος σε αυτό το κεφάλαιο, δίνονται έννοιες και πράξεις του τανυστικού γινομένου πινάκων. Για την καλύτερη εμπέδωση της ύλης του μαθητή, δίνονται λυμένα παραδείγματα και εφαρμογές καθώς και ασκήσεις για λύση για την ουσιαστική και αποδοτική μελέτη του μαθητή. Αυτό το κεφάλαιο απευθύνεται σε όλους τους μαθητές Γυμνασίου και Λυκείου εκτός της ενότητας των μιγαδικών αριθμών για τους μαθητές γυμνασίου.

2.1 Η Έννοια του Πίνακα

Έχουμε κάποια αριθμητικά δεδομένα ορθογώνιας διάταξης, κλεισμένα μέσα σε αγκύλες, όπως φαίνεται παρακάτω:

$$\begin{bmatrix} 4 & 20 & 9 \\ 10 & 5 & 12 \\ 7 & 1 & 33 \end{bmatrix}$$

Τότε λέμε ότι σχηματίζουν έναν πίνακα έναν πίνακα με 3 γραμμές και 3 στήλες, ή αλλιώς, έναν πίνακα με 3 γραμμές και 3 στήλες και πιο σύντομα έναν πίνακα τύπου 3 x 3 ή πιο συντομότερα έναν 3 x 3 πίνακα.

Έστω το σύστημα:

$$\begin{cases} x - 2y + 2z - w = 2 \\ 5x \quad - 2z + w = 1 \\ \quad y - 5z + 2w = 3 \end{cases}$$

Οι συντελεστές των αγνώστων, δηλαδή των μεταβλητών x, y, z και w μπορούν να σχηματίσουν ένα πίνακα 3 x 4 όπως φαίνεται παρακάτω:

$$\begin{bmatrix} 1 & -2 & 2 & -1 \\ 5 & 0 & -2 & 1 \\ 0 & 1 & 5 & 2 \end{bmatrix}$$

Και οι συντελεστές των αγνώστων τιμών μαζί με τους με τους σταθερούς όρους σχηματίζουν τον παρακάτω 3 x 5 πίνακα:

$$\begin{bmatrix} 1 & -2 & 2 & -1 & 2 \\ 5 & 0 & -2 & 1 & 1 \\ 0 & 1 & 5 & 2 & 3 \end{bmatrix}$$

Όλοι οι αριθμοί ενός πίνακα λέμε ότι αποτελούν τα **στοιχεία του πίνακα.**

Έτσι, το στοιχείο 1 του παραπάνω πίνακα, παριστάνει ένα στοιχείο του πίνακα που βρίσκεται στην **1η γραμμή** και στη **2η στήλη** και συμβολίζεται **α_{11}**.

Το στοιχείο 5 του παραπάνω πίνακα, παριστάνει ένα στοιχείο του πίνακα που βρίσκεται στη **2η γραμμή** και στην **1η στήλη** και συμβολίζεται **α_{21}**.

Το στοιχείο 0 του παραπάνω πίνακα, παριστάνει ένα στοιχείο του πίνακα που βρίσκεται στη **3η γραμμή** και στην **1η στήλη** και συμβολίζεται **α_{31}**.

Το στοιχείο -2 του παραπάνω πίνακα, παριστάνει ένα στοιχείο του πίνακα που βρίσκεται στην **1η γραμμή** και τη **2η στήλη** και συμβολίζεται **α_{22}**.
Και ούτω καθεξής.

Έτσι μπορούμε να πούμε ότι ένας πίνακας είναι μια ορθογώνια διάταξη αριθμών, διατεταγμένων σε γραμμές και στήλες.

ΟΡΙΣΜΟΣ

Μια διάταξη κ · ν το πλήθος των αριθμών σε μορφή ορθογωνίου σχήματος με κ γραμμές και ν στήλες, λέγεται **πίνακας τύπου κ x ν** ή πιο απλά **κ x ν πίνακας**.

Οι αριθμοί με τους οποίους δημιουργούμε έναν πίνακα ονομάζονται **στοιχεία του πίνακα**.

Τους πίνακες τους συμβολίζουμε συνήθως με κεφαλαία γράμματα Α, Β κλπ. ενώ τα στοιχεία τους με μικρά γράμματα α, β κλπ.

Ένα στοιχείο κάποιου πίνακα Α που το συμβολίζουμε **$\alpha_{i\,j,}$** σημαίνει ότι το στοιχείο α ανήκει στην **i γραμμή** και στη **j στήλη**.

Έτσι ένας **πίνακας** Α τύπου **μ x ν** γράφεται:

j-στήλη

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1j} & \cdots & \alpha_{1\nu} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2j} & \cdots & \alpha_{2\nu} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \alpha_{i1} & \alpha_{i2} & \cdots & \alpha_{ij} & \cdots & \alpha_{i\nu} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \alpha_{\mu 1} & \alpha_{\mu 2} & \cdots & \alpha_{\mu j} & \cdots & \alpha_{\mu\nu} \end{bmatrix}$$ **i-γραμμή**

Μπορούμε να το γράψουμε εν συντομία:

$[\alpha_{i\,j}]$, $1 \leq i \leq \mu$, $1 \leq j \leq \nu$.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε έναν πίνακα $[\alpha_{i\,j}]$ με $\alpha_{i\,j} = i - j$ έχει στοιχεία $\alpha_{1\,1} = 1 - 1 = 0$, $\alpha_{1\,2} = 1 - 2 = -1$, $\alpha_{2\,1} = 2 - 1$ και $\alpha_{22} = 2 - 2 = 0$.

Έτσι ο πίνακας γράφεται: $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

Επομένως μπορούμε να ορίσουμε ισότητα δύο πινάκων:

ΟΡΙΣΜΟΣ

Δύο πίνακες Α και Β λέμε ότι είναι ίσοι, όταν είναι του ίδιου τύπου, δηλαδή έχουν τον ίδιο αριθμό γραμμών, τον ίδιο αριθμό στηλών και τα αντίστοιχα στοιχεία τους είναι ίσα. Την ισότητα των δύο πινάκων Α και Β τη συμβολίζουμε Α = Β .

Προσοχή

Δύο πίνακες διαφορετικού τύπου δεν μπορεί να είναι ίσοι.
Όταν ένας πίνακας είναι τύπου ν x ν για κάποιο ν ϵ N*, δηλαδή έχει τον ίδιο αριθμό γραμμών και των ίδιο αριθμό στηλών, τότε ο πίνακας ονομάζεται **τριγωνικός πίνακας**.

Έστω ότι έχουμε έναν τετραγωνικό πίνακα Α και τα στοιχεία που σχηματίζουν την **κύρια διαγώνιό** του είναι α_{11}, α_{22}, α_{33}, ……., α_{vv} . Αν τα στοιχεία του πίνακα Α που δεν ανήκουν στην κύρια διαγώνιο είναι όλα μηδέν, τότε ο πίνακας ονομάζεται **διαγώνιος πίνακας**.

Παρακάτω έχουμε τρεις διαφορετικούς διαγώνιους πίνακες:

$$\begin{bmatrix} 8 & 0 \\ 0 & 15 \end{bmatrix}, \quad \begin{bmatrix} 7 & 0 \\ 0 & 32 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

Ένας πίνακας που έχει μόνο μία γραμμή 1 x n και έχει τη μορφή $[\alpha 1 \quad \alpha 2 \quad \dots \quad \alpha n]$, τότε ονομάζεται **πίνακας γραμμή.**

Παρακάτω φαίνονται μερικά παραδείγματα πίνακα γραμμή:

$$[\,2 \quad 1 \quad 3\,],\ [\,-1 \quad 8 \quad 17 \quad 10\,],\ [\,5 \quad 11 \quad -26 \quad 3 \quad 1\,]$$

Επίσης, ένας πίνακας που έχει μόνο μία στήλη m x 1 και έχει τη μορφή $\begin{bmatrix} \alpha 1 \\ \alpha 2 \\ \vdots \\ \alpha m \end{bmatrix}$

τότε ονομάζεται **πίνακας στήλη**.

Παρακάτω φαίνονται μερικά παραδείγματα πίνακα στήλη:

$$\begin{bmatrix} 5 \\ 4 \end{bmatrix},\quad \begin{bmatrix} 6 \\ 3 \\ 1 \end{bmatrix},\quad \begin{bmatrix} 7 \\ 2 \\ -2 \\ 0 \\ 3 \end{bmatrix}$$

Υπάρχει και η περίπτωση όπου ένας πίνακας έχει μόνο ένα στοιχείο δηλαδή 1 x 1 πίνακας, για παράδειγμα [5], [-2] κλπ. τότε ονομάζεται **πίνακας στοιχείο**.

Όταν σε έναν τετραγωνικό πίνακα όλα τα στοιχεία του που βρίσκονται κάτω από την κύρια διαγώνιο είναι μηδέν, τότε ο πίνακας ονομάζεται **τριγωνικός άνω.**
Παρακάτω φαίνονται μερικά παραδείγματα τριγωνικού άνω πίνακα:

$$\begin{bmatrix} 2 & 9 & 4 \\ 0 & 5 & -1 \\ 0 & 0 & 0 \end{bmatrix},\ \begin{bmatrix} 4 & 5 & 1 & 20 \\ 0 & 8 & 2 & 7 \\ 0 & 0 & 3 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Όταν σε έναν τετραγωνικό πίνακα όλα τα στοιχεία του που βρίσκονται πάνω από την κύρια διαγώνιο είναι μηδέν, τότε ο πίνακας ονομάζεται **τριγωνικός κάτω.**
Παρακάτω φαίνονται μερικά παραδείγματα τριγωνικού κάτω πίνακα:

$$\begin{bmatrix} 2 & 0 & 0 \\ 9 & 5 & -1 \\ 4 & 2 & 10 \end{bmatrix},\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 1 & 6 & 4 & 6 \\ 20 & 0 & 0 & 1 \end{bmatrix}$$

Ένας πίνακας που έχει όλα τα στοιχεία της κύριας διαγώνιου του ίσο με ένα και τα στοιχεία που βρίσκονται πάνω και κάτω από την κύρια διαγώνιο είναι όλα μηδέν, τότε ο πίνακας **ονομάζεται μοναδιαίος**. Ο μοναδιαίος πίνακας **συμβολίζεται** με το **γράμμα I**.

Παρακάτω φαίνονται παραδείγματα μοναδιαίου πίνακα.

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix},\ \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix},\ \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Μπορούμε να κατανοήσουμε όλα τα παραπάνω αν τα δούμε μέσα από μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 2

Έστω δύο πίνακες Α και Β, ίδιου τύπου 2 x 3 με Α=$\begin{bmatrix}a & b & c\\ x & y & z\end{bmatrix}$ και Β=$\begin{bmatrix}2 & 4 & 0\\ 8 & \sqrt{2} & -1\end{bmatrix}$.

Παρατηρούμε ότι το α_{11} στοιχείο του πίνακα Α είναι το α, ενώ το α_{22} στοιχείο του είναι το x κλπ.

Ενώ το α_{11} στοιχείο του πίνακα Β είναι το 2, ενώ το α_{22} στοιχείο του είναι το 8 κλπ.

Οι πίνακες Α και Β είναι ίσοι όταν τα αντίστοιχα στοιχεία τους είναι ίσα.
Επομένως από την ισότητα Α=Β των δύο πινάκων παίρνουμε τις έξι εξισώσεις:
α=2, b=4, c=0, x=8, y=$\sqrt{2}$ και z= -1.

ΠΑΡΑΔΕΙΓΜΑ 3

Έστω δύο πίνακες Α και Β, ίδιου τύπου 2x3 με Α=$\begin{bmatrix}a & 5 & \sqrt{3}\\ -2 & y & z\end{bmatrix}$ και Β=$\begin{bmatrix}6 & b & c\\ x & -1 & 0\end{bmatrix}$.

Παρατηρούμε ότι το α_{11} στοιχείο του πίνακα Α είναι το α, ενώ το α_{22} στοιχείο του είναι το -2 κλπ. Το α_{11} στοιχείο του πίνακα Β είναι το 6, ενώ το α_{22} στοιχείο του είναι το -1 κλπ.

Οι **πίνακες Α και Β είναι ίσοι** όταν τα **αντίστοιχα στοιχεία τους είναι ίσα**.
Επομένως από την ισότητα Α=Β των δύο πινάκων παίρνουμε τις έξι εξισώσεις:
α=6, b=5, c=$\sqrt{3}$, x=-2, y=-1 και z= 0.

Ας Θυμηθούμε

Τα βασικά **σύνολα αριθμών** είναι:

- Το σύνολο των **φυσικών αριθμών** $\mathbb{N} = \{0,1,2,3, \dots\}$.
- Το σύνολο των **ακεραίων αριθμών** $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$.
- Το σύνολο των **ρητών αριθμών** $\mathbb{Q} = \{\frac{\alpha}{\beta} / \alpha \in Z \text{ και } \beta \in N^*\}$.
- Το σύνολο των **πραγματικών αριθμών** $\mathbb{R}$.

Για τα παραπάνω σύνολα ισχύουν τα εξής:

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$$

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Δίνεται ο πίνακας $A=\begin{bmatrix}-1 & 7 & 4\\ 0 & 1 & -2\\ -2 & 4 & 0\end{bmatrix}$.

   i. Να βρείτε τον τύπο του.

   ii. Ποιος θα είναι ο τύπος του πίνακα Α αν αυξήσουμε τις γραμμές του κατά δύο και μειώσουμε τις στήλες του κατά τρεις. Πώς λέγεται ο πίνακας που προκύπτει;

   iii. Ποια είναι η θέση των στοιχείων στον πίνακα, που είναι ίσα με -2;

   iv. Ποια είναι τα στοιχεία α_{11}, α_{34}, α_{31}, α_{23};

2) Δίνονται οι παρακάτω δύο πίνακες οι οποίοι είναι ίσοι. Να βρείτε τα x, y, z ϵ R, για τα οποία ισχύει:

$$\begin{bmatrix} x & 5 & y+1 \\ z+2 & y & -2x+1 \end{bmatrix} = \begin{bmatrix} 1 & 5 & -3 \\ 4 & y & -1 \end{bmatrix}.$$

3) Να βρείτε τα x, z, w έτσι ώστε ο πίνακας $A=\begin{bmatrix} 8 & -5x-x+12 \\ 4y+x & 1 \end{bmatrix}$ να είναι διαγώνιος.

4) Να βρείτε για ποια τιμή του x και του y ϵ R, οι πίνακες $A=\begin{bmatrix} 15 & 5x-x-3 \\ 2y-1 & -3 \end{bmatrix}$ και $B=\begin{bmatrix} 15 & 1 \\ 2y+2x-9 & -3 \end{bmatrix}$ είναι ίσοι.

5) Βρείτε τον πίνακα $A=\begin{bmatrix} x & 3y & y+2 \\ x+2y+9 & 1 & y \\ x-1 & 3y+20x-5 & 2x \end{bmatrix}$, αν γνωρίζεται ότι είναι τριγωνικός πάνω.

6) Να βρείτε τις τιμές των x, y, z, w ϵ R για τις οποίες οι πίνακες $A=\begin{bmatrix} x+1 & y-2 \\ w+3 & z-4 \end{bmatrix}$ και $B=\begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}$ να είναι ίσοι.

2.2 ΑΠΛΕΣ ΠΡΑΞΕΙΣ ΠΙΝΑΚΩΝ

2.2.1 Πρόσθεση Πινάκων

Ένας μαθητής Γ' Λυκείου για τρία μαθήματα Μαθηματικά, Πληροφορική και Έκθεση, για τα δύο εξάμηνα οι βαθμολογίες του είναι:

Μαθήματα	**1ο Εξάμηνο**	**Bonus**	**2ο Εξάμηνο**	**Bonus**
Μαθηματικά	17	1	18	1
Πληροφορική	17	2	19	1
Έκθεση	16	1	17	2

Επομένως για τα δύο εξάμηνα ο συγκεκριμένος μαθητής, η βαθμολογία του για τα τρία μαθήματα μαζί με τους bonus και για τα δύο εξάμηνα είναι:

Μαθήματα	**1ο Εξάμηνο**	**2ο Εξάμηνο**
Μαθηματικά	17+1	18+1
Πληροφορική	17+2	19+1
Έκθεση	16+1	17+2

Αν γράψουμε με τη μορφή πίνακα τις παραπάνω βαθμολογίες του μαθητή για τα τρία μαθήματα, τότε θα είναι:

1ο εξάμηνο $A=\begin{bmatrix}17 & 1\\ 17 & 2\\ 16 & 1\end{bmatrix}$ και 2ο εξάμηνο $B=\begin{bmatrix}18 & 1\\ 19 & 1\\ 17 & 2\end{bmatrix}$.

Και για τα δύο εξάμηνα συνολικά θα έχουμε:

$$\Gamma=\begin{bmatrix}17+1 & 18+1\\ 17+2 & 19+1\\ 16+1 & 17+2\end{bmatrix}=\begin{bmatrix}18 & 19\\ 19 & 20\\ 17 & 19\end{bmatrix}.$$

Ο πίνακας Γ λέγεται άθροισμα των πινάκων Α και Β και συμβολίζεται με Α + Β, δηλαδή Γ = Α + Β.

Επομένως μπορούμε να ορίσουμε πρόσθεση δύο πινάκων:

ΟΡΙΣΜΟΣ

Άθροισμα δύο πινάκων $A=[\boldsymbol{\alpha}_{ij}]$ και $B=[\boldsymbol{\beta}_{ij}]$ λέγεται ένας πίνακας τύπου μ x ν του οποίου κάθε στοιχείο του είναι άθροισμα των αντίστοιχων στοιχείων του Α και Β. Ο πίνακας αυτός συμβολίζεται με Α + Β.
Δηλαδή,

$$A+B=[\boldsymbol{\alpha}_{ij}+\boldsymbol{\beta}_{ij}]$$

Η πράξη με την οποία βρίσκουμε το άθροισμα δύο πινάκων λέγεται **πρόσθεση πινάκων.**

Προσοχή

Δεν ορίζεται το άθροισμα πινάκων όταν είναι διαφορετικού τύπου. Για να **προσθέσουμε** δύο ή περισσότερους πίνακες θα πρέπει να είναι **του ίδιου τύπου μ x ν.** Δηλαδή οι πίνακες θα πρέπει να έχουν οπωσδήποτε τον ίδιο αριθμό γραμμών μ και τον ίδιο αριθμό στηλών ν.

Ας Θυμηθούμε

- 4 + 2 = 6
- 10 - 2 = 8
- - 4 - 2 = - 6
- - 4 + 2 = - 2

ΕΦΑΡΜΟΓΗ 1

Έχουμε δύο πίνακες ίδιου τύπου 2 x 2 $A = \begin{bmatrix} -1 & 5 \\ 7 & 2 \end{bmatrix}$ και $B = \begin{bmatrix} 2 & 4 \\ 0 & 6 \end{bmatrix}$ τότε το άθροισμά των δύο πινάκων Α και Β θα είναι:

$$A + B = \begin{bmatrix} -1 & 5 \\ 7 & 2 \end{bmatrix} + \begin{bmatrix} 2 & 4 \\ 0 & 6 \end{bmatrix} = \begin{bmatrix} -1+2 & 5+4 \\ 7+0 & 2+6 \end{bmatrix} = \begin{bmatrix} 1 & 9 \\ 7 & 8 \end{bmatrix}.$$

ΕΦΑΡΜΟΓΗ 2

Έχουμε δύο πίνακες Α και Β τύπου 3 x 3 με $A = \begin{bmatrix} 2 & -1 & 0 \\ 3 & 4 & 5 \\ -2 & 0 & 1 \end{bmatrix}$ και

$B = \begin{bmatrix} 7 & 6 & -1 \\ 2 & 10 & 3 \\ 0 & -4 & 6 \end{bmatrix}$.

Τότε το άθροισμά τους θα είναι το αποτέλεσμα της πρόσθεσης των στοιχείων τους, όπως φαίνεται παρακάτω:

$$A + B = \begin{bmatrix} 2+7 & -1+6 & 0+(-1) \\ 3+2 & 4+10 & 5+3 \\ -2+0 & 0+(-4) & 1+6 \end{bmatrix} = \begin{bmatrix} 9 & 5 & -1 \\ 5 & 14 & 8 \\ -2 & -4 & 7 \end{bmatrix}.$$

Όμως όταν έχουμε να προσθέσουμε δύο πίνακες Δ και Ε με $\Delta = \begin{bmatrix} 2 & 10 \\ 7 & -2 \\ 0 & 5 \end{bmatrix}$ και $E = \begin{bmatrix} 2 & 7 & 1 \\ -3 & 10 & 0 \\ 8 & 6 & 5 \end{bmatrix}$ παρατηρούμε ότι είναι διαφορετικού τύπου. Ο πίνακας Δ είναι τύπου 3 x 2 και ο πίνακας Ε είναι τύπου 3 x 3 και έτσι δεν μπορούμε να τους προσθέσουμε.

2.2.1.1 Ιδιότητες της πρόσθεσης πινάκων

Στις πράξεις μεταξύ των πινάκων, ισχύουν γνωστές ιδιότητες των αντίστοιχων πράξεων στο R δουλεύοντας σα να γίνονται οι πράξεις μεταξύ πραγματικών αριθμών.

Συγκεκριμένα για την πρόσθεση πινάκων και σύμφωνα με τον ορισμό προκύπτουν οι παρακάτω ιδιότητες.

Αν Α, Β, Γ είναι μ x ν πίνακες τότε:

- **Αντιμεταθετή Ιδιότητα**

$A + B = B + A$

- **Προσεταιριστική Ιδιότητα**

$A + (B + Γ) = (A + B) + Γ$

Ουδέτερο Στοιχείο

Ο μηδενικός πίνακας Ο που όλα τα στοιχεία του είναι μηδέν, αποτελεί το ουδέτερο στοιχείο στην πρόσθεση των πινάκων.
Για κάθε πίνακα Α ισχύει:

$$A + O = O + A = A$$

Αντίθετος Πίνακας

Για κάθε μ x ν πίνακα Α υπάρχει μοναδικός μ x ν πίνακας του οποίου όλα τα στοιχεία είναι αντίθετα των στοιχείων του πίνακα Α και ισχύει:

$$A + (-A) = (-A) + A = 0$$

Ο πίνακας **-Α** λέγεται **αντίθετος** του πίνακα **Α**.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε τον πίνακα $A=\begin{bmatrix} 5 & -6 & 0 \\ 1 & 7 & -3 \end{bmatrix}$ τότε ο αντίθετος του πίνακα Α είναι ο (-Α) και περιέχει όλα τα στοιχεία του αντίθετα των στοιχείων του πίνακα Α.
Δηλαδή $[-A] = \begin{bmatrix} -5 & 6 & 0 \\ -1 & -7 & 3 \end{bmatrix}$.

Προσοχή

Οι παραπάνω **ιδιότητες ισχύουν** εφόσον **ορίζονται** τα αντίστοιχα **αθροίσματα**. Οι πίνακες πρέπει να είναι του ίδιου τύπου.

ΕΦΑΡΜΟΓΗ 3

Έστω $A=\begin{bmatrix} 4 & 3 \\ -2 & 1 \end{bmatrix}$ $B=\begin{bmatrix} 3 & 1 \\ -5 & 8 \end{bmatrix}$ και $\Gamma=\begin{bmatrix} 1 & 0 \\ -2 & 3 \end{bmatrix}$

α) $A + B = \begin{bmatrix} 4 & 3 \\ -2 & 1 \end{bmatrix} + \begin{bmatrix} 3 & 1 \\ -5 & 8 \end{bmatrix} = \begin{bmatrix} 4+3 & 3+1 \\ -2+(-5) & 1+8 \end{bmatrix} = \begin{bmatrix} 7 & 4 \\ -7 & 9 \end{bmatrix}$

β) $B + A = \begin{bmatrix} 3 & 1 \\ -5 & 8 \end{bmatrix} + \begin{bmatrix} 4 & 3 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} 3+4 & 1+3 \\ -5+(-2) & 8+1 \end{bmatrix} = \begin{bmatrix} 7 & 4 \\ -7 & 9 \end{bmatrix}$

Παρατηρούμε ότι το αποτέλεσμα της πρόσθεσης των Α , Β πινάκων, Α + Β αλλά και Β + Α είναι το ίδιο.

γ) $B + \Gamma = \begin{bmatrix} 3 & 1 \\ -5 & 8 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -2 & 3 \end{bmatrix} = \begin{bmatrix} 3+1 & 1+0 \\ -5+(-2) & 8+3 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ -7 & 11 \end{bmatrix}$

δ) $A + (B+\Gamma) = \begin{bmatrix} 4 & 3 \\ -2 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 1 \\ -7 & 11 \end{bmatrix} = \begin{bmatrix} 4+4 & 3+1 \\ -2+(-11) & 1+11 \end{bmatrix} = \begin{bmatrix} 8 & 4 \\ -13 & 12 \end{bmatrix}$

ΕΦΑΡΜΟΓΗ 4

Να γράψετε τον αντίθετο των παραπάνω πινάκων [Α], [Β], [Γ], [Α + Β], [Β+Γ],

[Α + (Β + Γ)].

Έχουμε:

- $[-A] = \begin{bmatrix} -4 & -3 \\ 2 & -1 \end{bmatrix}$
- $[-B]=\begin{bmatrix} -3 & -1 \\ 5 & -8 \end{bmatrix}$
- $[-\Gamma]=\begin{bmatrix} -1 & 0 \\ 2 & -3 \end{bmatrix}$
- $-[A + B] = -[B + A]= \begin{bmatrix} 7 & 4 \\ -7 & 9 \end{bmatrix}$
- $-[B + \Gamma]= \begin{bmatrix} -4 & -1 \\ 7 & -11 \end{bmatrix}$
- $-[A + (B+\Gamma)]= \begin{bmatrix} -8 & -4 \\ 13 & -12 \end{bmatrix}$

2.2.2 Αφαίρεση Πινάκων

Η **αφαίρεση** ή αλλιώς διαφορά μεταξύ πινάκων γίνεται ομοίως όπως και η πρόσθεση πινάκων. Η αφαίρεση πινάκων ορίζεται με τη βοήθεια της πρόσθεσης, όπως και στις πράξεις των πραγματικών αριθμών. Αν Α, Β δύο πίνακες τύπου μ x ν, τότε η διαφορά Α - Β ορίζεται με την αφαίρεση των στοιχείων του πίνακα Β από τα αντίστοιχα στοιχεία του πίνακα Α.
Η αφαίρεση ορίζεται ως εξής:

Α – Β = Α + (-Β)

Προσοχή

Δεν ορίζεται η διαφορά πινάκων όταν είναι διαφορετικού τύπου, όπως ισχύει και για την πρόσθεση δύο πινάκων. Για να υπολογίσουμε τη διαφορά από δύο ή περισσότερους πίνακες θα πρέπει να είναι **του ίδιου τύπου μ x ν.** Δηλαδή οι πίνακες θα πρέπει να έχουν οπωσδήποτε τον ίδιο αριθμό γραμμών μ και τον ίδιο αριθμό στηλών ν.

Ας δούμε ένα παράδειγμα.

Έχουμε τον πίνακα $A=\begin{bmatrix} 2 & -3 \\ 4 & 0 \\ 3 & 1 \end{bmatrix}$ και $B=\begin{bmatrix} 1 & 2 \\ -3 & 2 \\ 1 & 0 \end{bmatrix}$ τότε:

$$A - B = \begin{bmatrix} 2 & -3 \\ 4 & 0 \\ 3 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ -3 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2-1 & -3-2 \\ 4-(-3) & 0-2 \\ 3-1 & 1-0 \end{bmatrix} = \begin{bmatrix} 1 & -5 \\ 7 & -2 \\ 2 & 1 \end{bmatrix}.$$

ΕΦΑΡΜΟΓΗ 1

Δίνονται οι πίνακες:

$A=\begin{bmatrix} 2 & -3 & 0 \\ 1 & -2 & 4 \end{bmatrix}$, $B=\begin{bmatrix} 3 & -2 \\ -4 & 3 \\ -4 & 0 \end{bmatrix}$, $\Gamma=\begin{bmatrix} 0 & 1 \\ -3 & 2 \\ -4 & 4 \end{bmatrix}$, $\Delta=\begin{bmatrix} -2 & 0 & 1 \\ 4 & -2 & -1 \end{bmatrix}$.

Να βρείτε αν ορίζονται οι πίνακες:

i) Α - Δ ii) Α - Β - Ε και iii) Β - Γ

Έχουμε:

i) Οι πίνακες Α, Δ είναι του ίδιου τύπου 2 x 3, άρα ορίζεται το άθροισμα και η διαφορά τους.

$A - \Delta = \begin{bmatrix} 2 & -3 & 0 \\ 1 & -2 & 4 \end{bmatrix} - \begin{bmatrix} -2 & 0 & 1 \\ 4 & -2 & -1 \end{bmatrix} =$

$= \begin{bmatrix} 2-(-2) & -3-0 & 0-1 \\ 1-4 & -2-2 & 4-(-1) \end{bmatrix} = \begin{bmatrix} 4 & -3 & -1 \\ -3 & -4 & 5 \end{bmatrix}.$

ii) Οι πίνακες Α, Β δεν είναι του ίδιου τύπου. Επομένως δεν ορίζεται η διαφορά Α - Β με αποτέλεσμα να μην ορίζεται ούτε η διαφορά Α – Β – Ε.

iii) Οι πίνακες Β, Γ είναι του ίδιου τύπου 3 x 2, άρα ορίζεται το άθροισμα και η διαφορά τους. Είναι:

$B - \Gamma = \begin{bmatrix} 3 & -2 \\ -4 & 3 \\ -4 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -3 & 2 \\ -4 & 4 \end{bmatrix} = \begin{bmatrix} 3-0 & -2-1 \\ -4-(-3) & 3-2 \\ -4-(-4) & 0-4 \end{bmatrix} = \begin{bmatrix} 3 & -3 \\ -1 & 1 \\ 0 & -4 \end{bmatrix}.$

2.2.3 Διάνυσμα Θέσης

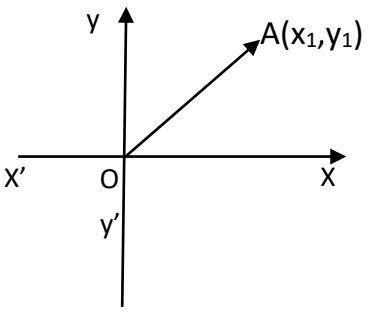
Διάνυσμα θέσης ή **διανυσματική ακτίνα** ενός σημείου Α, ονομάζουμε το διάνυσμα $\overrightarrow{OA}$, όπου Ο είναι η αρχή των αξόνων.

ΠΑΡΑΔΕΙΓΜΑ 1

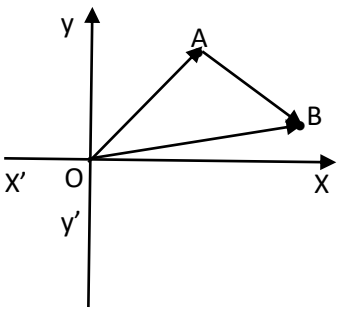
Αν $A(x_1,y_1)$, τότε το διάνυσμα θέσης του σημείου Α το γράφουμε ως εξής:

$\overrightarrow{OA} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ ή $\overrightarrow{OA} = [x_1 \quad y_1]$.

Τα x_1, y_1 τα λέμε συντεταγμένες του διανύσματος $\overrightarrow{OA}$ (σχ-1).



(σχ-1)



(σχ-2)

Αν είναι γνωστές οι συντεταγμένες των διανυσμάτων θέσης των σημείων **Α(x_1,y_1)** και **Β(x_2,y_2)**, μπορούμε να βρούμε τις **συντεταγμένες** του διανύσματος $\overrightarrow{AB}$, που προκύπτει αν από τις συντεταγμένες του **τέλος Β αφαιρέσουμε** τις συντεταγμένες **της αρχής Α** (σχ-2).

ΠΑΡΑΔΕΙΓΜΑ 2

Αν οι συντεταγμένες των διανυσμάτων θέσης των σημείων είναι **Α(x_1,y_1)** και **Β(x_2,y_2)**, τότε είναι $\overrightarrow{OA} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ και $\overrightarrow{OB} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$.

Οπότε:

$$\overrightarrow{OA} + \overrightarrow{AB} = \overrightarrow{OB} \Leftrightarrow \overrightarrow{OA} - \overrightarrow{OB} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$$

Για να προσθέσουμε ή να αφαιρέσουμε διανύσματα, προσθέτουμε ή αφαιρούμε τις αντίστοιχες συνταραγμένες τους.

Έτσι, αν $\vec{\alpha} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ και $\vec{\beta} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$, τότε θα είναι:

$$\vec{\alpha} + \vec{\beta} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix} \quad \text{και} \quad \vec{\alpha} - \vec{\beta} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \end{bmatrix}$$

ΕΦΑΡΜΟΓΗ 1

Έχουμε τα σημεία Α, Β, Γ με συντεταγμένες Α(2,-1), Β(-1,3) και Γ(1,0), να βρεθούν οι συντεταγμένες των διανυσμάτων:

i. $\overrightarrow{AB}$, ii. $\overrightarrow{A\Gamma}$ και iii. $\overrightarrow{B\Gamma}$

Τα διανύσματα θέσης των Α(2,-1), Β(-1,3) και Γ(1,0) είναι αντίστοιχα:

$\overrightarrow{OA} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $\overrightarrow{OB} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$ και $\overrightarrow{OA} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Έχουμε:

i. $\overrightarrow{AB} = \overrightarrow{OB} - \overrightarrow{OA} = \begin{bmatrix} -1 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} -1-2 \\ 3+1 \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$.

ii. $\overrightarrow{A\Gamma} = \overrightarrow{O\Gamma} - \overrightarrow{OA} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1-2 \\ 0-1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

iii. $\overrightarrow{B\Gamma} = \overrightarrow{O\Gamma} - \overrightarrow{OB} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 1+1 \\ 0-3 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$.

2.2.4 Εξίσωση πινάκων

Εξίσωση πινάκων ονομάζουμε μια ισότητα πινάκων στην οποία ζητείται να υπολογιστεί κάποιος πίνακας.

Από τους ορισμούς της πρόσθεσης και της αφαίρεσης προκύπτει ότι:

$$X + B = A \Leftrightarrow X = A - B$$

Όντως,

- Αν $X + B = A$ τότε $X + B - B = A - B$

άρα $X = A - B$ ενώ

- Αν $X = A - B$ τότε $X + B = A - B + B$

άρα $X + B = A$.

ΠΑΡΑΔΕΙΓΜΑ 1

Δίνονται οι πίνακες $A= \begin{bmatrix} 3 & 1 \\ -1 & 0 \end{bmatrix}$ και $B= \begin{bmatrix} 0 & 1 \\ -4 & 2 \end{bmatrix}$. Να βρεθεί ο πίνακας Χ που ικανοποιεί την εξίσωση $X - A = B$.

Είναι:

Η εξίσωση $X - A = B$, γίνεται: $X = A + B$.

Αντικαθιστώντας τους πίνακες Α και Β προκύπτει:

$$X = \begin{bmatrix} 3 & 1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ -4 & 2 \end{bmatrix} = \begin{bmatrix} 3+0 & 1+1 \\ -1+(-4) & 0+2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ -5 & 2 \end{bmatrix}.$$

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Δίνονται οι πίνακες:

$A=\begin{bmatrix} 2 & -3 & 0 \\ 2 & 4 & 1 \end{bmatrix}$, $B=\begin{bmatrix} 1 & 3 \\ -1 & 2 \\ 0 & -4 \end{bmatrix}$, $\Gamma=\begin{bmatrix} 1 & 1 \\ -3 & 4 \\ 0 & -2 \end{bmatrix}$, $\Delta=\begin{bmatrix} -2 & 1 & 0 \\ 1 & -3 & 4 \end{bmatrix}$ και

$E=\begin{bmatrix} 0 & -1 & 0 \\ -2 & -1 & 1 \end{bmatrix}$.

Να βρείτε αν ορίζονται τα αθροίσματα και οι διαφορές των πινάκων:

i. A + B
ii. B - Γ
iii. (Δ + E) - A
iv. A - Δ
v. (A - Δ) + E

2) Δίνονται οι πίνακες: $A=\begin{bmatrix} 3 & 0 \\ 1 & -2 \end{bmatrix}$, $B=\begin{bmatrix} 5 & -1 \\ -2 & 3 \end{bmatrix}$ και $\Gamma=\begin{bmatrix} 4 & 2 \\ 1 & 0 \end{bmatrix}$.

Να υπολογίσετε το άθροισμα και τη διαφορά των παρακάτω πινάκων:

i. (A + B) - Γ
ii. A - Γ
iii. (A - B) + Γ
iv. (B - Γ) + A

3) Να βρείτε τον πίνακα X, που ικανοποιεί τις σχέσεις:

i. A – X = B + Γ
ii. (A + B) + X = Γ

4) Να βρείτε τους πίνακες X, για τους οποίους ισχύει:

i. $\begin{bmatrix} 2 & -3 & 0 \\ 2 & 4 & 1 \end{bmatrix} + X = \begin{bmatrix} 4 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}$.

ii. $X - \begin{bmatrix} 1 & 2 \\ 6 & 4 \end{bmatrix} = \begin{bmatrix} 3 & -2 \\ -7 & 0 \end{bmatrix}$.

iii. $\begin{bmatrix} -1 & 0 \\ 4 & -2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} 0 & 6 \\ -1 & 2 \\ 1 & -1 \end{bmatrix} - X = \begin{bmatrix} 8 & 0 \\ -4 & -1 \\ 1 & 0 \end{bmatrix}$.

5) Να βρείτε τα x, y, w για τα οποία ισχύει η ισότητα:

$$\begin{bmatrix} 5 & 2 \\ -1 & 2-x \\ 2 & 6 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 4 & -1 \\ y-2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & w-6 \\ 0 & 1 \\ 5 & 10 \end{bmatrix}.$$

6) Να βρείτε τις τιμές των x, y ϵ R για τις οποίες ο πίνακας

$A = \begin{bmatrix} 2x-y & 0 \\ x-3 & 4y-8x \\ 0 & y-6 \end{bmatrix}$ είναι μηδενικός πίνακας.

2.3 Πολλαπλασιασμός Αριθμού με Πίνακα

ΟΡΙΣΜΟΣ

Γινόμενο ενός πραγματικού αριθμού λ με έναν πίνακα $A=[\boldsymbol{\alpha}_{ij}]$ λέγεται ο πίνακας που προκύπτει αν πολλαπλασιάσουμε όλα τα στοιχεία του πίνακα Α με τον αριθμό λ. Ο πίνακας αυτός συμβολίζεται με $\lambda \cdot A$ ή Λα. Δηλαδή
$\lambda \cdot A = [\boldsymbol{\lambda}\, \boldsymbol{\alpha}_{ij}] = [\boldsymbol{\lambda} \cdot \boldsymbol{\alpha}_{ij}]$ για κάθε ζεύγος (i, j).

Ισχύει προφανώς:

$$\boldsymbol{\lambda}\,[\boldsymbol{\alpha}_{ij}] = [\boldsymbol{\lambda}\cdot\boldsymbol{\alpha}_{ij}] = [\boldsymbol{\alpha}_{ij}\cdot\boldsymbol{\lambda}] = [\boldsymbol{\alpha}_{ij}]\cdot\boldsymbol{\lambda}$$

Η πράξη με την οποία βρίσκουμε το **γινόμενο ενός αριθμού με έναν πίνακα** λέγεται **πολλαπλασιασμός αριθμού με πίνακα**.

Αν $A = \begin{bmatrix} \boldsymbol{\alpha_1} & \boldsymbol{\beta_1} & \boldsymbol{\gamma_1} \\ \boldsymbol{\alpha_2} & \boldsymbol{\beta_2} & \boldsymbol{\gamma_2} \\ \boldsymbol{\alpha_3} & \boldsymbol{\beta_3} & \boldsymbol{\gamma_3} \end{bmatrix}$, τότε:

$$\lambda \cdot A = \lambda \cdot \begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{bmatrix} = \begin{bmatrix} \lambda\cdot\alpha_1 & \lambda\cdot\beta_1 & \lambda\cdot\gamma_1 \\ \lambda\cdot\alpha_2 & \lambda\cdot\beta_2 & \lambda\cdot\gamma_2 \\ \lambda\cdot\alpha_3 & \lambda\cdot\beta_3 & \lambda\cdot\gamma_3 \end{bmatrix}.$$

2.3.1 Ιδιότητες του πολλαπλασιασμού με πίνακα

Και στον πολλαπλασιασμό αριθμού με πίνακα ισχύουν οι γνωστές πράξεις στο R και έτσι μπορούμε να δουλέψουμε όπως και στους πραγματικούς αριθμούς. Έχουμε Α, Β είναι πίνακες τύπου μ x ν και κ, λ πραγματικοί αριθμοί, τότε σύμφωνα με τον ορισμό, πίνακα επί έναν αριθμό προκύπτουν οι παρακάτω ιδιότητες :

$$(\kappa + \lambda)\cdot A = \kappa\cdot A + \lambda\cdot A$$
$$\lambda\cdot(A + B) = \lambda\cdot A + \lambda\cdot B$$
$$\kappa\cdot(\lambda\cdot A) = (\kappa\cdot\lambda)\cdot A$$

Επιπλέον ισχύουν οι ισοδυναμίες:

$$1\cdot A = A \quad \text{και} \quad \lambda\cdot A = O \Leftrightarrow \lambda = 0 \quad \text{ή} \quad A = O$$

Ας Θυμηθούμε

• $4 \cdot 2 = 8$	• $0 \cdot (-2) = -8$
• $(-4) \cdot (-2) = 8$	• $(-4) \cdot (+2) = -8$

ΠΑΡΑΔΕΙΓΜΑ 1

Θέλουμε να πολλαπλασιάσουμε τον αριθμό λ = -2 με τον πίνακα $A=\begin{bmatrix} 1 & -2 & 1 \\ 0 & 3 & 5 \end{bmatrix}$.

Τότε θα είναι:

$$\lambda \cdot A = -2 \cdot \begin{bmatrix} 1 & -2 & 1 \\ 0 & 3 & 5 \end{bmatrix} = \begin{bmatrix} -2 \cdot 1 & -2(-2) & -2 \cdot 1 \\ -2 \cdot 0 & -2 \cdot 3 & -2 \cdot 5 \end{bmatrix} = \begin{bmatrix} -2 & 4 & -2 \\ 0 & -6 & -10 \end{bmatrix}.$$

ΠΑΡΑΔΕΙΓΜΑ 2

Να βρεθεί ο πίνακας Χ για τον οποίο ισχύει:

$$2 \cdot \begin{bmatrix} 4 & 0 \\ -2 & 1 \end{bmatrix} + X = 4 \cdot \begin{bmatrix} 0 & -2 \\ 5 & 1 \end{bmatrix}.$$

Παρατηρούμε ότι έχουμε εξίσωση πινάκων. Ακολουθούμε τα βήματα όπως ακριβώς και στις εξισώσεις πραγματικών αριθμών. Πρώτα κάνουμε τους πολλαπλασιασμούς αριθμού με πίνακα και έχουμε:

$$\begin{bmatrix} 2 \cdot 4 & 2 \cdot 0 \\ 2(-2) & 2 \cdot 1 \end{bmatrix} + X = \begin{bmatrix} 4 \cdot 0 & 4(-2) \\ 4 \cdot 5 & 4 \cdot 1 \end{bmatrix} \Leftrightarrow$$

Χωρίζουμε γνωστούς από αγνώστους:

$$X = \begin{bmatrix} 0 & -8 \\ 20 & 4 \end{bmatrix} - \begin{bmatrix} 8 & 0 \\ -4 & 2 \end{bmatrix} \Leftrightarrow X = \begin{bmatrix} 0-8 & -8-0 \\ 20-(-4) & 4-2 \end{bmatrix} \Leftrightarrow$$

$$X = \begin{bmatrix} -8 & -8 \\ 24 & 2 \end{bmatrix}.$$

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Έστω οι πίνακες Α = $\begin{bmatrix} 1 & 2 & -3 \\ 6 & -4 & 5 \end{bmatrix}$ και Β= $\begin{bmatrix} 1 & 0 & 4 \\ 2 & -1 & 7 \end{bmatrix}$, τότε να υπολογίσετε:

   $3 \cdot A + 2 \cdot B$

2) Αν $\begin{bmatrix} 1 & 4 & -2 \\ 2 & 4 & 0 \end{bmatrix}$, Β=$\begin{bmatrix} 2 & 0 & 2 \\ 3 & -1 & 4 \end{bmatrix}$ και Γ=$\begin{bmatrix} 7 & -3 & 5 \\ 18 & 2 & 4 \end{bmatrix}$ να υπολογίσετε:

   i. $4 \cdot A + 3 \cdot B - 2 \cdot Γ$ και ii. $2 \cdot A - B + 4 \cdot Γ$

3) Αν Α=$\begin{bmatrix} 5 & 3 \\ 2 & -1 \end{bmatrix}$ και Β=$\begin{bmatrix} 0 & 1 \\ 2 & 6 \end{bmatrix}$ να βρείτε τους πίνακες:

   i. $2 \cdot A$ ii. $2 \cdot (-3 \cdot A)$ iii. $3 \cdot B - 2 \cdot A$

4) Αν Α=$\begin{bmatrix} 1 & -3 \\ 0 & 2 \end{bmatrix}$ και Β=$\begin{bmatrix} 3 & 4 \\ -2 & 5 \end{bmatrix}$ να βρείτε τον πίνακα Γ=$\begin{bmatrix} x & y \\ z & w \end{bmatrix}$ έτσι ώστε να ισχύει: $3 \cdot Γ = 2 \cdot A - 2 \cdot B$

5) Να βρεθούν τα x, y, z αν είναι:

   i. $\begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 0 \end{bmatrix}$ και ii. $-2 \begin{bmatrix} -1 \\ 2 \\ 5 \end{bmatrix} + 4 \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$.

6) Να βρεθεί ο πίνακας Χ για τον οποίο ισχύει:

   $2 \cdot \begin{bmatrix} -2 & 0 \\ -3 & 5 \\ 1 & 1 \end{bmatrix} - 3 \cdot X = 2 \cdot \begin{bmatrix} 5 & 0 \\ -2 & -1 \\ 3 & 0 \end{bmatrix}$.

7) Να βρείτε τους πίνακες Χ, Υ για τους οποίους ισχύει:

   $2 \cdot X + Y = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ και $3 \cdot X + 2 \cdot Y = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$.

8) Να βρείτε τους πραγματικούς αριθμούς x, y, z, w έτσι ώστε να ισχύει η παρακάτω ισότητα:

   $2 \cdot \begin{bmatrix} x & 2 \\ z & w \end{bmatrix} - \begin{bmatrix} 1 & y \\ 4 & w \end{bmatrix} = \begin{bmatrix} 1 & y+1 \\ -2z & 4 \end{bmatrix}$.

2.4 Πολλαπλασιασμός Πινάκων

ΟΡΙΣΜΟΣ

Αν Α=[α_{ik}] είναι ένας πίνακας μ x ν και Β=[β_{kj}] είναι ένας ν x ρ πίνακας, τότε ορίζουμε ως γινόμενο του πίνακα Α με τον πίνακα Β και το συμβολίζουμε Α · Β ή με ΑΒ του μ x ρ πίνακα, του οποίου κάθε στοιχείο γ_{ij} είναι το άθροισμα των γινομένων των ν στοιχείων της i-γραμμής του Α με τα αντίστοιχα στοιχεία της j-στήλης του Β. Δηλαδή,

$$\gamma_{ij} = \alpha_{i1} \cdot \beta_{1j} + \alpha_{i2} \cdot \beta_{2j} + \ldots + \alpha_{iv} \cdot \beta_{vj}$$

Σχηματικά,

$$[A] \cdot [B] = [\Gamma]$$

j-στήλη

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1j} & \cdots & \alpha_{1v} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2j} & \cdots & \alpha\ 2v \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \alpha_{i1} & \alpha_{i2} & \cdots & \alpha_{ij} & \cdots & \alpha_{iv} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \alpha_{\mu 1} & \alpha_{\mu 2} & \cdots & \alpha_{\mu j} & \cdots & \alpha_{\mu v} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1j} & \cdots & \beta_{1v} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2j} & \cdots & \beta\ 2v \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \beta_{i1} & \beta_{i2} & \cdots & \beta_{ij} & \cdots & \beta_{iv} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \beta_{\mu 1} & \beta_{\mu 2} & \cdots & \beta_{\mu j} & \cdots & \beta_{\mu v} \end{bmatrix} =$$

$$= \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1j} & \cdots & \gamma_{1v} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2j} & \cdots & \gamma\ 2v \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ij} & \cdots & \gamma_{iv} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \gamma_{\mu 1} & \gamma_{\mu 2} & \cdots & \gamma_{\mu j} & \cdots & \gamma_{\mu v} \end{bmatrix}$$

i-γραμμή

Δύο πίνακες Α=[α_{ij}] και Β=[β_{ij}] μπορούν να πολλαπλασιαστούν κατά την τάξη Α-Β, τότε και μόνο, αν το πλήθος των στηλών του πίνακα Α είναι ίσο με το πλήθος των γραμμών του πίνακα Β. Δηλαδή, αν ο πίνακας Α είναι τάξεως μ x ν και ο πίνακας Β κ x ν.

Τότε το γινόμενο Αμ x ν · Βκ x ν είναι ένας πίνακας Γμ x ν=[γ_{ij}] τάξεως μ x ν, του οποίου κάθε στοιχείο προκύπτει από τον πολλαπλασιασμό της i-γραμμής του πίνακα Α επί την j-στήλη του πίνακα Β.
Όπως είδαμε παραπάνω δηλαδή,

$$\gamma_{ij} = \alpha_{i1} \cdot \beta_{1j} + \alpha_{i2} \cdot \beta_{2j} + \ldots + \alpha_{iv} \cdot \beta_{vj} =$$

$$\sum_{\kappa=1}^{\mu} \alpha_{i\kappa} \cdot \beta_{\kappa j}$$

Σχηματικά ο πολλαπλασιασμός δύο πινάκων δίνεται από τη σχέση:

$$\mathbf{A}_{\mu x \nu} \cdot \mathbf{B}_{\kappa x \nu} = \mathbf{\Gamma}_{\mu x \rho}$$

Προσοχή

Το γινόμενο $A \cdot B$ των πινάκων Α και Β, ορίζεται μόνο όταν ο αριθμός των στηλών του πίνακα Α είναι ίσος με τον αριθμό των γραμμών του πίνακα Β. Δηλαδή,

- Αν δύο πίνακες Α, Β είναι τετραγωνικοί πίνακες 2 x 2 και ορίζονται οι πολλαπλασιασμοί $A \cdot B$, $B \cdot A$ τότε το γινόμενό τους θα είναι σε κάθε περίπτωση πίνακας τύπου 2 x 2.
- Αν ο πίνακας Α είναι τύπου 1 x 3, ενώ ο πίνακας Β είναι τύπου 2 x 3, τότε δεν ορίζεται ο πολλαπλασιασμός $A \cdot B$ αλλά ούτε ο πολλαπλασιασμός $B \cdot A$.

Ακόμη,

- Αν ένας πίνακας Α τύπου 2 x 3 και ο Β πίνακας τύπου 3 x 4 τότε το γινόμενό τους ΑΒ θα είναι τύπου 2 x 4.
- Αν ένας πίνακας Α τύπου 2 x 5 και ο Β πίνακας τύπου 3 x 6 τότε το γινόμενό τους ΑΒ δεν ορίζεται.

Ισχύει:

$$\mathbf{A}_{\mu x \nu} \cdot \mathbf{B}_{\nu x \rho} = \mathbf{\Gamma}_{\mu x \rho}$$

2.5 Πολλαπλασιασμός γραμμής πίνακα επί στήλη

Έστω δύο πίνακες Α=[α_{ij}] όπου i=,1,2,…,n και Β=[β_{ij}] όπου j=1,2,…,n. Ο πίνακας Α είναι πίνακας γραμμή τύπου 1 x n και ο Β είναι πίνακας στήλη τύπου n x 1.

ΟΡΙΣΜΟΣ

Ορίζουμε ως γινόμενο $A \cdot B$ της διανυσματικής γραμμής Α επί τη διανυσματική στήλη Β τον πίνακα Γ τύπου 1 x 1 ο οποίος δίνεται από την ισότητα:

$$\Gamma = A \cdot B = [\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_n] \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$

Δηλαδή,

$$[\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_n] \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} = \sum_{\kappa=1}^{n} \alpha_\kappa \cdot \beta_\kappa$$

Έτσι στον πολλαπλασιασμό γραμμή επί στήλη, κάθε στοιχείο της γραμμής πολλαπλασιάζεται με το αντίστοιχο στοιχείο της στήλης και προστίθενται τα αποτελέσματα των γινομένων τους.

Προσοχή

- Στον πολλαπλασιασμό των πινάκων **δεν ισχύει η αντιμεταθετική ιδιότητα**. Δηλαδή για δύο πίνακες Α, Β γενικά είναι $A \cdot B \neq B \cdot A$ ακόμη και αν ορίζεται το γινόμενο $B \cdot A$.
- Το γινόμενο $A \cdot B$ ορίζεται όταν ο **αριθμός των στηλών** του πίνακα Α είναι **ίσος με τον αριθμό των γραμμών** του πίνακα Β.

Αν για δύο πίνακες Α, Β ισχύει $A \cdot B = B \cdot A$ τότε οι πίνακες ονομάζονται **αντιμεταθετικοί**.

2.5.1 Ιδιότητες του πολλαπλασιασμού πινάκων

Αν Α, Β είναι πίνακες με λ, ν πραγματικοί αριθμοί, τότε ισχύουν οι παρακάτω ισότητες:

- **Προσεταιριστική Ιδιότητα**

$A \cdot (B \cdot \Gamma) = (A \cdot B) \cdot \Gamma$

- **Επιμεριστική Ιδιότητα**

$A \cdot (B+\Gamma) = A \cdot B + A \cdot \Gamma$ και $(B+\Gamma) \cdot A = B \cdot A + \Gamma \cdot A$

$(\lambda \cdot A) \cdot (\mu \cdot B) = (\lambda \cdot \mu) \cdot (A \cdot B)$

$[(A \cdot B) \cdot \Gamma] \cdot \Delta = (A \cdot B) \cdot (\Gamma \cdot \Delta) = A \cdot [(B \cdot \Gamma) \cdot \Delta)] = [A \cdot (B \cdot \Gamma)] \cdot \Delta$

• $A \cdot I_\nu = I_\nu \cdot A = A$,
όπου $\mathbf{I_\nu}$ είναι ο **μοναδιαίος πίνακας** τύπου ν x ν του οποίου κάθε **στοιχείο** της κύριας **διαγώνιου** είναι ίσο με **ένα**.

• Αν έχουμε έναν πίνακα Α τύπου μ x ν , τότε ισχύει:

$$\boxed{A \cdot I_\nu = A} \quad \text{και} \quad \boxed{I_\mu \cdot A = A}$$

• Αν $A \cdot B = 0$ δεν ισχύει πάντα ότι Α=Ο ή Β=Ο.

Δηλαδή, αν $A=\begin{bmatrix} 2 & 0 \\ -3 & 0 \end{bmatrix}$ και $B=\begin{bmatrix} 0 & 0 \\ 1 & 4 \end{bmatrix}$, τότε:

$$A \cdot B = \begin{bmatrix} 2 & 0 \\ -3 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} 2 \cdot 0 + 0 \cdot 1 & 2 \cdot 0 + 0 \cdot 4 \\ -3 \cdot 0 + 0 \cdot 1 & -3 \cdot 0 + 0 \cdot 4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = O.$$

$A \cdot B = O$ χωρίς κανένας από τους δύο πίνακες Α και Β να είναι μηδενικός.

• Αν $A \cdot B = A \cdot \Gamma$ ή $B \cdot A = \Gamma \cdot A$ δεν σημαίνει πάντα ότι Β=Γ ή ακόμη και αν ο πίνακας είναι διαφορετικός από το μηδενικό πίνακα.

• Στους **τετραγωνικούς** πίνακες του τύπου ν x ν **ορίζεται** πάντα το **γινόμενο**.

Έχουμε έναν πίνακα Α με εκθέτη το θετικό ακέραιο κ, τότε ισχύει:

$$A^2 = A \cdot A \quad \text{και} \quad A^0 = I$$

$$A^3 = A^2 \cdot A$$

$$A^\nu = A^{\nu-1} \cdot A$$

• Δεν ισχύει επίσης η ιδιότητα της διαγραφής. Δηλαδή με την υπόθεση $A \cdot B = A \cdot \Gamma$ δεν μπορώ να συμπεράνω ότι και Β=Γ, άσχετα αν $A \neq O$.

• Δεν ισχύει η σχέση $(AB)^\nu = A^\nu \cdot B^\nu$, εκτός αν οι πίνακες Α,Β είναι αντιμεταθετικοί.

• Στους πίνακες δεν ισχύουν πάντα οι γνωστές ταυτότητες των αριθμών, αφού δεν ισχύει η αντιμεταθετική ιδιότητα.

ΕΦΑΡΜΟΓΗ 1

Αν ένας πίνακας $A=\begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 0\end{bmatrix}$ και έχουμε $I_2=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, $I_3=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$, τότε:

• $I_2 \cdot A = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 0\end{bmatrix} = \begin{bmatrix}1\cdot 1+0\cdot 2 & 1\cdot 2+0\cdot 1 & 1\cdot 3+0\cdot 0\\ 0\cdot 1+1\cdot 2 & 0\cdot 2+1\cdot 1 & 0\cdot 3+1\cdot 0\end{bmatrix}$

$= \begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 0\end{bmatrix} = A.$

• $A\cdot I_3 = \begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 0\end{bmatrix}\cdot\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 2 & 3\\ 2 & 1 & 0\end{bmatrix} = A,$ κ.λπ.

Ενώ $\boxed{A\cdot B \neq B\cdot A}$

ΕΦΑΡΜΟΓΗ 2

Αν $A=\begin{bmatrix}2 & 0\\ 1 & 1\end{bmatrix}$ και $B=\begin{bmatrix}1 & 0\\ -2 & 1\end{bmatrix}$, τότε:

• $A\cdot B=\begin{bmatrix}2 & 0\\ 1 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 0\\ -2 & 1\end{bmatrix} = \begin{bmatrix}2\cdot 1+0(-2) & 2\cdot 0+0\cdot 1\\ 1\cdot 1+1(-2) & 1\cdot 0+1\cdot 1\end{bmatrix} = \begin{bmatrix}2 & 0\\ -1 & 1\end{bmatrix}$

• $B\cdot A=\begin{bmatrix}1 & 0\\ -2 & 1\end{bmatrix}\cdot\begin{bmatrix}2 & 0\\ 1 & 1\end{bmatrix} = \begin{bmatrix}1\cdot 2+0\cdot 1 & 1\cdot 0+1\cdot 0\\ -2\cdot 2+1\cdot 1 & -2\cdot 0+1\cdot 1\end{bmatrix} = \begin{bmatrix}2 & 0\\ -5 & 1\end{bmatrix}$

Παρατηρούμε ότι το γινόμενό $A\cdot B$ δεν είναι ίδιο με το $B\cdot A$.

ΕΦΑΡΜΟΓΗ 3

Δίνονται οι πίνακες $A=\begin{bmatrix}0 & 3 & 4\\ 6 & 5 & 2\\ 1 & 0 & 1\end{bmatrix}$, $B=\begin{bmatrix}0 & 3\\ 6 & 5\\ 1 & 0\end{bmatrix}$ και $\Gamma=\begin{bmatrix}-2\\ 2\\ 0\end{bmatrix}$.

Να υπολογίσετε τα παρακάτω γινόμενα:

i. $A \cdot B$ ii. $A \cdot \Gamma$ iii. $\Gamma \cdot A$

Ο πίνακας Α είναι τύπου 3 x 3, ο πίνακας Β είναι τύπου 3 x 2 και ο πίνακας Γ είναι τύπο 3 x 1, επομένως το γινόμενο $A \cdot B$ θα είναι τύπου 3 x 2, το γινόμενο $A \cdot \Gamma$ θα είναι τύπου 3 x 1 και το γινόμενο $\Gamma \cdot A$ θα είναι τύπου 3 x 1.

Έχουμε,

i. $A \cdot B = \begin{bmatrix} 0 & 3 & 4 \\ 6 & 5 & 2 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 3 \\ 6 & 5 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0\cdot 0 + 3\cdot 6 + 4\cdot 1 & 0\cdot 3 + 3\cdot 5 + 4\cdot 0 \\ 6\cdot 0 + 5\cdot 6 + 2\cdot 1 & 6\cdot 3 + 5\cdot 5 + 2\cdot 1 \\ 1\cdot 0 + 0\cdot 6 + 1\cdot 1 & 1\cdot 3 + 0\cdot 5 + 1\cdot 0 \end{bmatrix} =$

$= \begin{bmatrix} 22 & 15 \\ 32 & 45 \\ 1 & 3 \end{bmatrix}.$

ii. $A \cdot \Gamma = \begin{bmatrix} 0 & 3 & 4 \\ 6 & 5 & 2 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0(-2) + 3\cdot 2 + 4\cdot 0 \\ 6(-2) + 5\cdot 2 + 2\cdot 0 \\ 1(-2) + 0\cdot 2 + 1\cdot 0 \end{bmatrix} = \begin{bmatrix} 6 \\ -22 \\ -2 \end{bmatrix}.$

iii. $\Gamma \cdot A = \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 3 & 4 \\ 6 & 5 & 2 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -2\cdot 0 + 2\cdot 6 + 0\cdot 1 \\ -2\cdot 3 + 2\cdot 5 + 0\cdot 0 \\ -2\cdot 4 + 2\cdot 2 + 0\cdot 1 \end{bmatrix} = \begin{bmatrix} 12 \\ 4 \\ -4 \end{bmatrix}.$

ΕΦΑΡΜΟΓΗ 4

Να βρείτε τα γινόμενα των παρακάτω πινάκων:

i. $\begin{bmatrix} 1 & 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix}$ ii. $\begin{bmatrix} 2 & -3 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ -4 \end{bmatrix}$

ii. $\begin{bmatrix} 3 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 3 & 4 \\ 6 & 5 & 2 \\ 1 & 0 & 1 \end{bmatrix}$ iv. $\begin{bmatrix} 0 & 1 & 2 \\ 3 & 2 & 3 \\ 4 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix}$

Είναι:

i. $\begin{bmatrix} 1 & 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix} = [1 \cdot (-1) + 1 \cdot 3 + 3 \cdot 1] = -1 + 3 + 3 = -1 + 6 = 5.$

ii. $\begin{bmatrix} 2 & -3 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ -4 \end{bmatrix} = [2 \cdot 1 - 3 \cdot 3 + 6 \cdot (-4)] = 2 - 9 - 24 = 2 - 33 = -31.$

iii. $[3 \quad 1 \quad 0]\cdot\begin{bmatrix}0 & 3 & 4\\ 6 & 5 & 2\\ 1 & 0 & 1\end{bmatrix}=$

1 x 3 3 x 3

$[3\cdot 0+1\cdot 6+0\cdot 6 \quad 3\cdot 3+1\cdot 5+0\cdot 0 \quad 3\cdot 4+1\cdot 2+0\cdot 1]=$

$=[6 \quad 14 \quad 14].$

iv. $\begin{bmatrix}3 & 3 & 0\\ 18 & 5 & 0\\ 3 & 0 & 0\end{bmatrix}.\begin{bmatrix}2\\ 4\\ 3\end{bmatrix}=\begin{bmatrix}3\cdot 2+3\cdot 4+0\cdot 3\\ 18\cdot 2+5\cdot 4+0\cdot 3\\ 3\cdot 2+0\cdot 4+0\cdot 3\end{bmatrix}=\begin{bmatrix}18\\ 56\\ 6\end{bmatrix}.$

3 x 3 3 x 1 1 x 3 3 x 1

ΕΦΑΡΜΟΓΗ 5

Αν $A=\begin{bmatrix}3 & 7\\ -1 & -3\end{bmatrix}$ και $B=\begin{bmatrix}1 & 3\\ -1 & -1\end{bmatrix}$, τότε να υπολογίσετε τα παρακάτω γινόμενο:

i. $A\cdot B$, ii. $A^2\cdot B$ και iii. $A\cdot B^2$

Είναι:

• $A^2= A\cdot A =\begin{bmatrix}3 & 7\\ -1 & -3\end{bmatrix}\cdot\begin{bmatrix}3 & 7\\ -1 & -3\end{bmatrix}=$

$\begin{bmatrix}3\cdot 3+7(-1) & 3\cdot 7+7(-3)\\ -1\cdot 3+(-3)(-1) & -1\cdot 7+(-3)(-3)\end{bmatrix}=\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}.$

• $B^2 = B\cdot B =\begin{bmatrix}1 & 3\\ -1 & -1\end{bmatrix}\cdot\begin{bmatrix}1 & 3\\ -1 & -1\end{bmatrix}=$

$=\begin{bmatrix}1\cdot 1+3(-1) & 1\cdot 3+3(-1)\\ -1\cdot 1+(-1)(-1) & -1\cdot 3+(-1)(-1)\end{bmatrix}=\begin{bmatrix}-2 & 0\\ 0 & -2\end{bmatrix}.$

i. $A\cdot B =\begin{bmatrix}3 & 7\\ -1 & -3\end{bmatrix}\cdot\begin{bmatrix}1 & 3\\ -1 & -1\end{bmatrix}=$

$=\begin{bmatrix}3\cdot 1+7(-1) & 3\cdot 3+7(-1)\\ -1\cdot 1+(-3)(-1) & -1\cdot 3+(-3)(-1)\end{bmatrix}=\begin{bmatrix}-4 & 2\\ 2 & 0\end{bmatrix}.$

ii. $A^2 \cdot B = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 \\ -1 & -1 \end{bmatrix} =$

$= \begin{bmatrix} 2\cdot 1 + 0(-1) & 0\cdot 3 + 2(-1) \\ 0\cdot 1 + 2(-1) & 0\cdot 3 + 2(-1) \end{bmatrix} = \begin{bmatrix} 2 & -2 \\ -2 & -2 \end{bmatrix}.$

iii. $A \cdot B^2 = \begin{bmatrix} 3 & 7 \\ -1 & -3 \end{bmatrix} \cdot \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix} =$

$\begin{bmatrix} 3(-2) + 7\cdot 0 & 3\cdot 0 + 7(-2) \\ -1(-2) + (-3)\cdot 0 & -1\cdot 0 + (-3)(-2) \end{bmatrix} = \begin{bmatrix} -6 & -14 \\ 2 & 6 \end{bmatrix}.$

ΕΦΑΡΜΟΓΗ 6

Να βρεθούν οι τιμές των x, y, z και w $\epsilon \mathbb{R}$ για τις οποίες ισχύει:

$\begin{bmatrix} 1 & 0 \\ -3 & x \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & y \end{bmatrix} = \begin{bmatrix} w & z \\ -1 & 1 \end{bmatrix}.$

Είναι:

$\begin{bmatrix} 1 & 0 \\ -3 & x \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 1 & y \end{bmatrix} = \begin{bmatrix} w & z \\ -1 & w \end{bmatrix} \Leftrightarrow$

$\begin{bmatrix} 1\cdot 0 + 0\cdot 1 & 1(-1) + 0\cdot y \\ -3\cdot 0 + x\cdot 1 & -3(-1) + x\cdot y \end{bmatrix} = \begin{bmatrix} w & z \\ -1 & w \end{bmatrix} \Leftrightarrow$

$\begin{bmatrix} 0 & -1 \\ x & 3 + x\cdot y \end{bmatrix} = \begin{bmatrix} w & z \\ -1 & w \end{bmatrix} \Leftrightarrow$

Από την ισότητα πινάκων έχουμε:

$$\Leftrightarrow \begin{cases} w = 0 \\ z = -1 \\ x = -1 \\ 3 + x\cdot y = w \end{cases} \Leftrightarrow$$

$\Leftrightarrow 3 + x\cdot y = w \Leftrightarrow o = 3 + (-1)\cdot y \Leftrightarrow y = -3.$

Άρα: x= -1, y = -3, z = -1 και w = 0.

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Να βρείτε αν ορίζονται τα γινόμενα $A \cdot B$ και $B \cdot A$ σε κάθε μια από τις παρακάτω περιπτώσεις:

   i. $A=\begin{bmatrix} -2 & 1 \\ 0 & 3 \end{bmatrix}$, $B=\begin{bmatrix} 2 & 0 \\ -1 & 4 \end{bmatrix}$.

   ii. $A=[3 \quad 0 \quad -1]$, $B=\begin{bmatrix} 0 & 0 & 2 \\ 2 & 3 & 1 \\ -1 & -2 & 3 \end{bmatrix}$.

   iii. $A=\begin{bmatrix} 2 & 1 \\ 0 & 3 \\ 4 & -2 \end{bmatrix}$, $B=\begin{bmatrix} 3 \\ 1 \end{bmatrix}$.

2) Δίνονται οι πίνακες $A=\begin{bmatrix} 1 & 4 & -3 \\ 2 & 5 & 6 \end{bmatrix}$, $B=\begin{bmatrix} 2 & 0 & 3 \\ 6 & -1 & 2 \end{bmatrix}$ και $\Gamma=\begin{bmatrix} 6 & -6 & 1 \\ 1 & 4 & 8 \end{bmatrix}$. Να βρείτε το $4 \cdot A + 5 \cdot B - 2 \cdot \Gamma$.

3) Αν $A=[1 \quad 2 \quad 3]$, $B=\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ και $\Gamma=\begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}$, να βρείτε τα γινόμενα των πινάκων $A \cdot B$, $A \cdot \Gamma$ και $B \cdot \Gamma$.

4) Αν $A=\begin{bmatrix} 2 & 0 & 1 \\ 3 & 1 & 4 \end{bmatrix}$ και $B=\begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}$ να υπολογίσετε τους πίνακες:

   i. $A \cdot B$ και ii. $A^2 \cdot B^2$.

5) Να βρεθούν οι τιμές των x, y, z $\epsilon \mathbb{R}$ για τις οποίες ισχύει:

   i. $\begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ -4 \\ 0 \end{bmatrix}$ και ii. $\begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix} + 2 \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \\ 5 \end{bmatrix}$.

6) Δίνονται οι πίνακες $A=\begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix}$, $B=[1 \quad -2 \quad 2]$, $\Gamma=\begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}$, $\Delta=\begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}$ και $E=[0 \quad 4 \quad -1]$. Να υπολογίσετε τα παρακάτω γινόμενα πινάκων:

   i. $A \cdot B$ ii. $A \cdot \Gamma$ iii. $B \cdot \Delta$ iv. $B \cdot E$ και v. $E \cdot A$

7) Να βρείτε τους πραγματικούς αριθμούς x, y, z και w ώστε να ισχύει η ισότητα:

$$2 \cdot \begin{bmatrix} x & 3 \\ z & w \end{bmatrix} - \begin{bmatrix} 1 & y \\ 4 & w \end{bmatrix} = \begin{bmatrix} 1 & y+2 \\ -2z & 3 \end{bmatrix}.$$

2.6 Τανυστικό Γινόμενο Πινάκων

Οι κβαντικοί υπολογιστές εκτελούν πολύ γρήγορους υπολογισμούς, για αυτό θα ασχοληθούμε με το τανυστικό γινόμενο σε μορφή πινάκων. Το τανυστικό γινόμενο ή αλλιώς γινόμενο kronecker έχει εφαρμογή στην μελέτη και ανάλυση πινάκων καθώς και την επίλυση γραμμικών εξισώσεων πινάκων. Το γινόμενο kronecker πήρε το όνομά του από το Γερμανό μαθηματικό Leopold kronecker (1823-1891) και ήταν ο πρώτος που το όρισε και το χρησιμοποίησε.

ΟΡΙΣΜΟΣ

Έχουμε δύο πίνακες $A=[\alpha_{ij}]$ τύπου μxν και $B=[\beta_{ij}]$ τύπου ρxκ ϵ $M_{\mu,\nu}$, τότε το **τανυστικό γινόμενο** είναι ένας σύνθετος πίνακας, ο οποίος προκύπτει από τον **πολλαπλασιασμό του κάθε στοιχείου του πίνακα Α με όλο τον πίνακα Β** και γράφεται **Α⊗Β τύπου ρμ x νκ**.

Δεν ισχύει όπως τον πολλαπλασιασμό των κοινών πινάκων, όπου ο αριθμός των στηλών του πρώτου πίνακα πρέπει να είναι ίσος με τον αριθμό των γραμμών του δεύτερου πίνακα.

Το τανυστικό γινόμενο **ορίζεται** από δύο πίνακες **οποιοδήποτε διαστάσεων**.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε δύο πίνακες, $A = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1\nu} \\ \vdots & \ddots & \vdots \\ \alpha_{\mu 1} & \cdots & \alpha_{\mu\nu} \end{bmatrix}$ και $B = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1\nu} \\ \vdots & \ddots & \vdots \\ \beta_{\mu 1} & \cdots & \beta_{\mu\nu} \end{bmatrix}$ και το τανυστικό τους γινόμενο είναι:

$$\mathbf{A\otimes B} = \begin{bmatrix} \boldsymbol{\alpha_{11}\cdot B} & \cdots & \boldsymbol{\alpha_{1\nu}\cdot B} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\alpha_{\mu 1}\cdot B} & \cdots & \boldsymbol{\alpha_{\mu\nu}\cdot B} \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} \alpha_{11}.\begin{bmatrix} \beta_{11} & \cdots & \beta_{1\nu} \\ \vdots & \ddots & \vdots \\ \beta_{\mu 1} & \cdots & \beta_{\mu\nu} \end{bmatrix} & \cdots & \alpha_{1\nu}.\begin{bmatrix} \beta_{11} & \cdots & \beta_{1\nu} \\ \vdots & \ddots & \vdots \\ \beta_{\mu 1} & \cdots & \beta_{\mu\nu} \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \alpha_{\mu 1}.\begin{bmatrix} \beta_{11} & \cdots & \beta_{1\nu} \\ \vdots & \ddots & \vdots \\ \beta_{\mu 1} & \cdots & \beta_{\mu\nu} \end{bmatrix} & \cdots & \alpha_{\mu\nu}.\begin{bmatrix} \beta_{11} & \cdots & \beta_{1\nu} \\ \vdots & \ddots & \vdots \\ \beta_{\mu 1} & \cdots & \beta_{\mu\nu} \end{bmatrix} \end{bmatrix} = \Gamma.$$

Παρατηρούμε το τανυστικό γινόμενο Α⊗Β είναι ένας μεγαλύτερος πίνακας Γ και τα στοιχεία του προκύπτουν:

- $\gamma_{11} \rightarrow$ είναι το γινόμενο του στοιχείου α_{11} του πίνακα Α με τον πίνακα Β, κλπ.
- $\gamma_{1\nu} \rightarrow$ είναι το γινόμενο του στοιχείου $\alpha_{1\nu}$ του πίνακα Α με τον πίνακα Β, κλπ.
- $\gamma_{\mu 1} \rightarrow$ είναι το γινόμενο του στοιχείου $\alpha_{\mu 1}$ του πίνακα Α με τον πίνακα Β, κλπ.
- $\gamma_{\mu\nu} \rightarrow$ είναι το γινόμενο του στοιχείου $\alpha_{\mu\nu}$ του πίνακα Α με τον πίνακα Β, κλπ.

Προσοχή

- Με το τανυστικό γινόμενο δύο πινάκων Α, Β ο πίνακας που προκύπτει είναι μεγαλύτερος διαστάσεων από τους αρχικούς πίνακες Α, Β. Δηλαδή δημιουργούμε μεγαλύτερους πίνακες.
- Δεν υπάρχουν κάποιοι περιορισμοί στο τανυστικό γινόμενο πινάκων. Μπορούμε να έχουμε πίνακες οποιοδήποτε διαστάσεων.
- Δεν ισχύει γενικότερα ότι το τανυστικό γινόμενο Α⊗ Β δύο πινάκων Α, Β ϵ $M_{\mu,\nu}$, είναι ίσο με το τανυστικό γινόμενο Β ⊗Α. Είναι διαφορετικοί πίνακες.

Δηλαδή:

$$A \otimes B \neq B \otimes A$$

ΠΑΡΑΔΕΙΓΜΑ 2

Έχουμε δύο πίνακες στήλη $A = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ και $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, να βρείτε το τανυστικό τους γινόμενο: Α ⊗ Β και Β ⊗ Α.

Έχουμε:

- $A \otimes B = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.\begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 2.\begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \end{bmatrix}.$

- $B \otimes A = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1.\begin{bmatrix} 1 \\ 2 \end{bmatrix} \\ 0.\begin{bmatrix} 1 \\ 2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix}.$

2.6.1 Βασικές Ιδιότητες Τανυστικού Γινομένου

Κάποιες πολύ βασικές ιδιότητες για το τανυστικό γινόμενο, είναι:

- $(\alpha \cdot A) \otimes B = A \otimes (\alpha \cdot B)$
- $(A \otimes B)^T = A^T \cdot B^T$
- $(A \otimes B) \otimes \Gamma = A \otimes (B \otimes \Gamma)$
- $(A + B) \otimes \Gamma = (A \otimes \Gamma) + (B \otimes \Gamma)$
- $A \otimes (B + \Gamma) = A \otimes B + A \otimes \Gamma$
- $(A \otimes B) \otimes (\Gamma \otimes \Delta) = (A \cdot \Gamma) \otimes (B \cdot \Delta)$
- $A \otimes 0 = 0 \otimes A = 0$

ΕΦΑΡΜΟΓΗ 1

Έχουμε δύο πίνακες $A = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix}$ και $B = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix}$. Να υπολογιστεί το τανυστικό γινόμενο $A \otimes B$.

Έχουμε:

$$A \otimes B = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2.\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} & 3.\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \\ 1.\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} & 1.\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix} \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} 2\cdot 2 & 2(-1) & 3\cdot 2 & 3(-1) \\ 2\cdot 0 & 2\cdot 1 & 3\cdot 0 & 3\cdot 1 \\ 1\cdot 2 & 1(-1) & 1\cdot 2 & 1(-1) \\ 1\cdot 0 & 1\cdot 1 & 1\cdot 0 & 1\cdot 1 \end{bmatrix} = \begin{bmatrix} 4 & -2 & 6 & -3 \\ 0 & 2 & 0 & 3 \\ 2 & -1 & 2 & -1 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

ΕΦΑΡΜΟΓΗ 2

Έχουμε δύο πίνακες στήλη $A = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ και $B = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$. Να υπολογιστεί το τανυστικό γινόμενο $A \otimes B$ και $B \otimes A$.

Έχουμε:

• $A \otimes B = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} -1\begin{bmatrix} 3 \\ 0 \end{bmatrix} \\ 1\begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 3 \\ 0 \end{bmatrix}$.

• $B \otimes A = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3\begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ 0\begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} -3 \\ 3 \\ 0 \\ 0 \end{bmatrix}$.

ΕΦΑΡΜΟΓΗ 3

Έχουμε δύο πίνακες στήλη $A = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ και $B = \begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix}$. Να υπολογιστεί το τανυστικό γινόμενο $A \otimes B$ και $B \otimes A$.

Έχουμε:

• $A \otimes B = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix} \\ 0\begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix} \\ -1\begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ -2 \\ 0 \\ 0 \\ 0 \\ -3 \\ 2 \end{bmatrix}$.

• $B \otimes A = \begin{bmatrix} 0 \\ 3 \\ -2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \\ 3\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \\ -2\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ -3 \\ -2 \\ 0 \\ 2 \end{bmatrix}$.

ΕΦΑΡΜΟΓΗ 4

Έχουμε τους πίνακες $A = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$, $B = \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix}$, $\Gamma = [2 \quad 0 \quad -1]$ και $\Delta = [-1 \quad 0 \quad 1]$. Να υπολογίσετε το τανυστικό γινόμενο των:

i. $A \otimes B$, ii. $A \otimes \Gamma$ και iii. $B \otimes \Delta$

Έχουμε:

i. $A \otimes B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0.\begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} & 0.\begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} \\ 1.\begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} & -1.\begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} \end{bmatrix} =$

$$= \begin{bmatrix} 0(-1) & 0(-1) & 0(-1) & 0(-1) \\ 0(-1) & 0\cdot 1 & 0(-1) & 0\cdot 1 \\ 1(-1) & 1(-1) & -1(-1) & -1(-1) \\ 1(-1) & 1\cdot 1 & -1(-1) & -1\cdot 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}.$$

ii. $A \otimes \Gamma = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \otimes [2 \quad 0 \quad -1] = \begin{bmatrix} 0.[2 & 0 & -1] & 0.[2 & 0 & -1] \\ 1.[2 & 0 & -1] & -1.[2 & 0 & -1] \end{bmatrix} =$

$$= \begin{bmatrix} 0\cdot 2 & 0\cdot 0 & 0(-1) & 0\cdot 2 & 0\cdot 0 & 0(-1) \\ 1\cdot 2 & 1\cdot 0 & 1(-1) & -1\cdot 2 & -1\cdot 0 & -1(-1) \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -1 & -2 & 0 & 1 \end{bmatrix}.$$

iii. $B \otimes \Delta = \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} \otimes [-1 \quad 0 \quad 1] = \begin{bmatrix} -1[-1 & 0 & 1] & -1[-1 & 0 & 1] \\ -1[-1 & 0 & 1] & 1.[-1 & 0 & 1] \end{bmatrix} =$

$$= \begin{bmatrix} -1(-1) & -1\cdot 0 & -1\cdot 1 & -1(-1) & -1\cdot 0 & -1\cdot 1 \\ -1(-1) & -1\cdot 0 & -1\cdot 1 & 1(-1) & 1\cdot 0 & 1\cdot 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & -1 & 1 & 0 & -1 \\ 1 & 0 & -1 & -1 & 0 & 1 \end{bmatrix}.$$

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Να υπολογίσετε το τανυστικό γινόμενο Α ⊗ Β, Β ⊗ Γ και Α ⊗ Γ των πινάκων Α $= [0 \quad 1 \quad -1]$, Β $= \begin{bmatrix} -1 & 0 & -1 \\ 4 & -2 & 2 \\ 1 & 3 & 0 \end{bmatrix}$ και Γ $= \begin{bmatrix} 3 & 0 \\ 1 & -1 \end{bmatrix}$.

2) Δίνονται οι πίνακες Α $= \begin{bmatrix} -2 & 1 \\ 0 & 3 \end{bmatrix}$, Β$= \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix}$ και Γ$= [1 \quad 0 \quad -1]$.

Να υπολογίσετε το τανυστικό γινόμενο των πινάκων:

ii. Α ⊗ Β, ii. Α ⊗ Γ, iii. Β ⊗ Γ και iv. Γ ⊗ Α

3) Έχουμε τους πίνακες Α $= \begin{bmatrix} -1 \\ 2 \\ 0 \\ 1 \end{bmatrix}$, Β $= [-1 \quad 3 \quad 0 \quad 2]$, Γ $= \begin{bmatrix} 0 \\ -3 \\ 1 \\ 0 \end{bmatrix}$,

Δ $= [0 \quad -2 \quad 1 \quad 0]$. Να υπολογίσετε το τανυστικό γινόμενο των πινάκων:

iii. Α ⊗ Β, ii. Α ⊗ Δ, iii. Β ⊗ Γ και v. Δ ⊗ Γ

4) Αν λ = -1 να υπολογίσετε την παρακάτω παράσταση:

$\lambda \cdot \begin{bmatrix} 0 & 4 \\ 0 & 2 \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 0 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 & 0 \\ 2 & -1 \end{bmatrix} \otimes [-1 \quad -1 \quad -1] \cdot \lambda$

5) Δίνονται οι πίνακες Α $= \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}$, Β $= \begin{bmatrix} 3 & 0 \\ 2 & -1 \end{bmatrix}$, Γ $= [-2 \quad 1 \quad 0]$ και

Δ $= [4 \quad -2 \quad 0]$. Να υπολογίσετε το τανυστικό γινόμενο των πινάκων:

i. Α ⊗ Β, ii. Β ⊗ Γ, iii. Α ⊗ Δ και iv. Δ ⊗ Β

6) Δίνονται οι πίνακες Α $= \begin{bmatrix} 3 & -1 \\ 0 & 0 \end{bmatrix}$, Β $= [0 \quad 0 \quad 1]$, Γ $= \begin{bmatrix} 0 & 0 \\ -2 & 1 \end{bmatrix}$ και

Δ $= [0 \quad -2 \quad -1]$. Αν α = 2, να βρείτε αν ισχύουν παρακάτω παραστάσεις:

i. $(\alpha \cdot A) \otimes B = A \otimes (\alpha \cdot B)$
ii. $A \otimes (B \otimes \Gamma) = \alpha \cdot [A \otimes B]$
iii. $(A \otimes B) \otimes (\Gamma \otimes \Delta) = \alpha \cdot (A \otimes \Delta)$

2.7 Πιθανότητες

2.7.1 Πείραμα τύχης – Δειγματικός χώρος - Ενδεχόμενα

Η θεωρία των πιθανοτήτων ως κύρια εφαρμογή έχει την μοντελοποίηση παιχνιδιών. Στην Πληροφορική, τις Τηλεπικοινωνίες, τη Φυσική και σε άλλες επιστήμες οι εφαρμογές της είναι πολύ περισσότερες .

Υπάρχουν κάποια παιχνίδια όπως το στρίψιμο ενός κέρματος, το παιχνίδι «πέτρα ψαλίδι χαρτί», η ρίψη ενός ζαριού κ.λπ., που μπορούν να θεωρηθούν χαρακτηριστικά παραδείγματα πειραμάτων τύχης που το κύριο χαρακτηριστικό τους είναι ότι **δεν μπορούμε εκ των προτέρων να γνωρίζουμε το αποτέλεσμά τους**. Τα πειράματα τύχης μελετώνται από τη θεωρία των πιθανοτήτων.

ΟΡΙΣΜΟΣ

Πείραμα τύχης λέγεται κάθε πείραμα του οποίου κύριο χαρακτηριστικό τους είναι πως **δεν είναι μπορούμε να προβλέψουμε το αποτέλεσμά τους**, όσες φορές και αν επαναλάβουμε το πείραμα αυτό κάτω από τις **ίδιες ακριβώς συνθήκες**.
Το **σύνολο όλων των δυνατών αποτελεσμάτων** ενός πειράματος τύχης λέγεται **Δειγματικός χώρος (Δ.Χ.)** του πειράματος και **συμβολίζεται με Ω**.

Ας δούμε το παράδειγμα ρίψης ενός κέρματος.

ΠΑΡΑΔΕΙΓΜΑΤΑ

Έχουμε ένα κλασικό κέρμα με δύο όψεις γράμματα (Γ) και κορόνα (Κ) όπως όλα τα κέρματα και το στρίβουμε στον αέρα. Όταν το κέρμα πέσει στο τραπέζι, η πιθανότητα να είναι γράμματα (Γ) ή κορόνα (Κ) είναι ισοπίθανη. Δηλαδή υπάρχει η ίδια αβεβαιότητα και για τις δύο περιπτώσεις, όταν στρίψουμε το κέρμα να εμφανίσει γράμματα (Γ) ή κορόνα (Κ).
Στο συγκεκριμένο παράδειγμά μας, μπορούμε να θεωρήσουμε ότι ο δειγματικός χώρος αποτελείται από τις επιλογές των λέξεων «γράμματα» (Γ) και «κορόνα» (Κ).

Δηλαδή είναι:

$$\Omega = \{\Gamma, \text{Κ}\}$$

Αν θεωρήσουμε ως πείραμα τύχης τη ρίψη ενός ζαριού και μας ενδιαφέρει η ένδειξη της άνω έδρας, τότε μπορούμε να θεωρήσουμε ότι ο δειγματικός χώρος είναι :

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

ΟΡΙΣΜΟΣ

Οποιοδήποτε **σύνολο δυνατών αποτελεσμάτων** ενός πειράματος τύχης ονομάζεται **ενδεχόμενο (event) ή γεγονός**, δηλαδή αποτελούν ένα υποσύνολο του δειγματικού χώρου.

• Έτσι για το στρίψιμο του κέρματος μπορούμε να πούμε ότι κάποια ενδεχόμενα είναι :

Α: «Το κέρμα να εμφανίσει γράμματα (Γ)»
Β: Το κέρμα να εμφανίσει κορόνα (Κ)»

• Στη ρίψη ενός ζαριού μπορούμε να πούμε ότι κάποια ενδεχόμενα είναι:

Α: «Εμφανίζεται περιττός αριθμός»
Β: «Εμφανίζεται αριθμός μεγαλύτερος του 2»
Γ: «Εμφανίζεται ο αριθμός 6»

Κάθε **ενδεχόμενο** αντιστοιχεί σε ένα **υποσύνολο** του **δειγματικού χώρου** και τα έτσι τα ενδεχόμενα γράφονται:

$$A = \{1, 3, 5\},\ B = \{3, 4, 5, 6\} \text{ και } \Gamma = \{6\}$$

Αν για μια ρίψη ζαριού το αποτέλεσμα είναι 4, παρατηρούμε ότι το Β περιέχει τον αριθμό 4, λέμε ότι είναι ένα **ευνοϊκό αποτέλεσμά** του και τότε το Β πραγματοποιείται. Ο αριθμός 4 δεν περιέχεται στο Α και το Γ και έτσι λέμε ότι δεν πραγματοποιούνται το Α και το Γ. Άλλα ευνοϊκά αποτελέσματα του Β είναι το 4, 5 και το 6, ενώ για το Α είναι το 1, 3 και 5. Για το Γ ευνοϊκό αποτέλεσμα είναι μόνο το 6.

Είναι:

- Ο ίδιος ο δειγματικός χώρος Ω ενός πειράματος θεωρείται ότι είναι ενδεχόμενο, το οποίο πραγματοποιείται πάντα, αφού όποιο και να είναι το αποτελέσματα του πειράματος θα ανήκει στο Ω.

- Ένα ενδεχόμενο που περιέχει **ένα μόνο ενδεχόμενο**, λέγεται **απλό**. Για το παράδειγμα στη ρίψη ενός ζαριού που είδαμε παραπάνω, τα ενδεχόμενα $A = \{1, 3, 5\}$, $B = \{3, 4, 5, 6\}$ είναι σύνθετα ενώ το $\Gamma = \{6\}$ είναι απλό.

- Όταν τα αποτελέσματα ενός πειράματος, σε μια συγκεκριμένη εκτέλεσή του είναι στοιχείο ενός ενδεχομένου, τότε λέμε ότι το ενδεχόμενο αυτό **πραγματοποιείται** ή **συμβαίνει**. Τα στοιχεία ενός ενδεχομένου λέγονται **ευνοϊκές περιπτώσεις** για την πραγματοποίησή του.

- Ένα ενδεχόμενο Α που περιέχει **ένα μόνο ενδεχόμενο**, λέγεται **βέβαιο** (Α=Ω).

- Ένα ενδεχόμενο Α που **δεν πραγματοποιείται** ποτέ, σε καμία εκτέλεση του πειράματος τύχης λέγεται **αδύνατο** (Α=Ø). Γι' αυτό λέμε ότι το Ø είναι το αδύνατο ενδεχόμενο.

- Δύο ενδεχόμενα Α και Β λέγονται **ασυμβίβαστα** όταν η πραγματοποίηση του ενός αποκλείει την πραγματοποίηση του άλλου, δηλαδή όταν $A \cap B = \emptyset$.

- Δύο ενδεχόμενα θα λέγονται **αντίθετα**, όταν η μη πραγματοποίηση του ενός συνεπάγεται την πραγματοποίηση του άλλου.

Στοιχεία από τα σύνολα

- ✓ Ένα σύνολο Α λέγεται **υποσύνολο** του Β και συμβολίζεται $A \subseteq B$, αν και μόνο αν, όλα τα στοιχεία του Α περιέχονται στο Β.
- ✓ **Τομή** δύο ενδεχομένων Α και Β, συμβολίζεται με $A \cap B$, ονομάζεται το σύνολο που αποτελείται από τα **κοινά στοιχεία** των δύο συνόλων. Πραγματοποιείται όταν το Α και το Β πραγματοποιούνται.
- ✓ **Ένωση** δύο ενδεχομένων Α και Β, συμβολίζεται με $A \cup B$ ονομάζεται το σύνολο που αποτελείται από **όλα τα στοιχεία** των δύο συνόλων. Πραγματοποιείται όταν ένα τουλάχιστον από τα Α ή Β πραγματοποιείται.
- ✓ Αν $A \subseteq B$, ονομάζουμε **συμπληρωματικό ή αντίθετο του ενδεχομένου** Α ως προς το Ω και **συμβολίζεται** με **Α'**, το σύνολο που αποτελείται από όλα τα **στοιχεία** του Ω που **δεν ανήκουν στο Α**. Το Α΄ πραγματοποιείται αν το Α δεν πραγματοποιείται.

Για να βρούμε το δειγματικό χώρο ενός πειράματος τύχης, ακολουθούμε έναν από τους παρακάτω τρόπους.

ΕΦΑΡΜΟΓΗ 1

Όταν το πείραμα επαναλαμβάνεται μια μονάχα φορά, τότε γράφουμε τον δειγματικό χώρο με αναγραφή των στοιχείων του.

Είδαμε προηγούμενος ότι ο δειγματικός χώρος είναι στο παράδειγμα με το κέρμα είναι:

$$\Omega = \{\Gamma, \text{Κ}\}$$

Ενώ, στο παράδειγμα με τη ρίψη ζαριού ο δειγματικός χώρος είναι:

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

ΕΦΑΡΜΟΓΗ 2

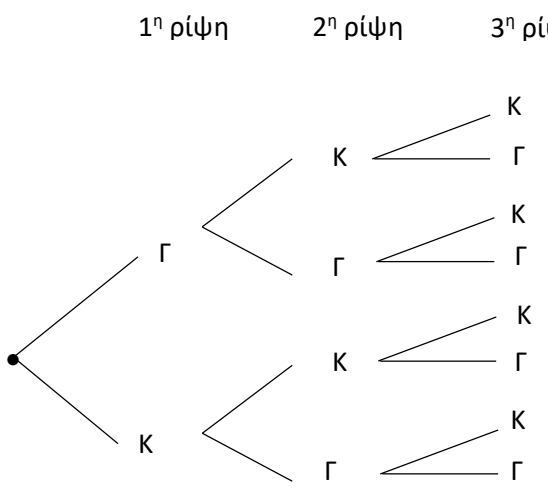
Όταν το πείραμα αποτελείται από διαδοχικά βήματα, σχηματίζουμε το δεντροδιάγραμμα.

Ας πάρουμε το παράδειγμα με το κέρμα.

Στρίβουμε το κέρμα 3 φορές και τα αποτελέσματα που καταγράψαμε είναι τα εξής:

- Την 1η φορά το κέρμα εμφανίζεται γράμματα (Γ).
- Την 2η φορά το κέρμα εμφανίζεται κορόνα (Κ).
- Την 3η φορά το κέρμα εμφανίζεται κορόνα (Κ).

Το δεντροδιάγραμμα θα είναι:



Άρα ο δειγματικός χώρος θα είναι:

Ω = { ΓΚΚ,ΓΚΓ,ΓΓΚ,ΓΓΓ,ΚΚΚ,ΚΚΓ,ΚΓΚ,ΚΓΓ}.

ΕΦΑΡΜΟΓΗ 3

Όταν το πείραμα επαναλαμβάνεται δύο φορές, τότε μπορούμε να φτιάξουμε ένα πίνακα διπλής εισόδου.

Ας πάρουμε το παράδειγμα με το ζάρι και το ρίχνουμε 2 φορές.

Έχουμε:

1η \ 2η	1	2	3	4	5	6
1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)
3	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)
4	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)
5	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)
6	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)

Επομένως ο δειγματικός χώρος θα είναι:

Ω = {(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6), (3,1), (3,2), (3,3), (3,4), (3,5), (3,6), (4,1), (4,2), (4,3), (4,4), (4,5), (4,6), (5,1), (5,2), (5,3), (5,4), (5,5), (5,6), (6,1), (6,2), (6,3), (6,4), (6,5), (6,6)}.

2.7.2 Η Έννοια της Πιθανότητας

Είναι δύο παίκτες Α και Β και συμμετέχουν σε ένα παιχνίδι στριψίματος κέρματος. Ο καθένας στρίβει το κέρμα 2 φορές και καταγράφουν τα αποτελέσματα.

- Ο παίκτης Α, αν και για τις 2 ρίψεις του κέρματος εμφανίσει γράμματα, κερδίζει.
- Ο παίκτης Β, αν εμφανίσει σε κάθε ρίψη διαφορετικό αποτέλεσμα, κερδίζει.

Είναι άραγε αυτά το παιχνίδι δίκαιο και για τους δύο παίκτες;
Αν στρίβανε το κέρμα 10 φορές αντί για δύο ή στρίβανε δύο κέρματα αντί για ένα θα επηρέαζε το αποτέλεσμα;

Πώς θα μπορούσαμε να το εξηγήσουμε;

Τι θα μπορούσε να αλλάξει ώστε να αποτελέσουν δίκαια τα παιχνίδια και για τους δύο παίκτες;

Το παιχνίδι αυτό θεωρείται πείραμα τύχης όπως είδαμε, αφού δεν μπορούμε να προβλέψουμε το αποτέλεσμά του.
Αν θεωρήσουμε το ενδεχόμενο «κερδίζει ο Α παίκτης», είναι ίδια η βεβαιότητα για την πραγματοποίησή του στο παιχνίδι;
Πώς μπορούμε να «μετρήσουμε» τη βεβαιότητα αυτή;

Η πιθανότητα ενός ενδεχομένου είναι ένας αριθμός που εκφράζει το μέτρο της βεβαιότητας που αποδίδουμε στο να πραγματοποιηθεί το ενδεχόμενο αυτό.

Αν έχουμε ένα **αμερόληπτο κέρμα**, που τα δυνατά αποτελέσματά του κατά το στρίψιμο του κέρματος **είναι ισοπίθανα**, δηλαδή αν στρίψουμε το κέρμα 10 φορές η ένδειξη θα είναι 5 φορές γράμματα και 5 φορές κορόνα, τότε ισχύει ο παρακάτω ορισμός.

ΟΡΙΣΜΟΣ

Σε ένα πείραμα τύχης με ν **ισοπίθανα** αποτελέσματα, η **πιθανότητα ενός ενδεχομένου Α** που περιέχει κ τέτοια αποτελέσματα είναι:

$$P(A) = \frac{\text{Πλήθος ευνοϊκών αποτελεσμάτων για το Α}}{\text{Πλήθος όλων των δυνατών αποτελεσμάτων}} = \frac{N(A)}{N(\Omega)} = \frac{\kappa}{\nu}$$

Σύμφωνα με τον ορισμό προκύπτει επίσης:

$$P(\Omega) = \frac{N(\Omega)}{N(\Omega)} = 1 \quad \text{και} \quad P(0) = \frac{N(0)}{N(0)} = 0$$

$$P(A) = 1 \text{ και } P(\emptyset) = 0$$

Για κάθε ενδεχόμενο Α του δειγματικού χώρου ισχύει:

$$0 \leq P(A) \leq 1$$

Δηλαδή, αφού το πλήθος των στοιχείων ενός ενδεχομένου είναι ίσο ή μικρότερο από το πλήθος των στοιχείων του δειγματικού χώρου.

- Αν $A = \emptyset$, τότε $P(A) = 0$ και
- Αν $A = \Omega$, τότε $P(A) = 1$

Ο ορισμός που είδαμε παραπάνω, ισχύει για ισοπίθανα ενδεχόμενα.

Στις περιπτώσεις των πειραμάτων τύχης που ο δειγματικός χώρος δεν αποτελείται από ισοπίθανα ενδεχόμενα, χρησιμοποιούμε τον **αξιωματικό ορισμό της πιθανότητας**.
Έτσι αν $\Omega = \{w_1, w_2, w_3, \dots, w_v\}$ είναι ένας δειγματικός χώρος με πεπερασμένο πλήθος στοιχείων, τότε για κάθε απλό ενδεχόμενο $\{w_i\}$ αντιστοιχίζουμε ένα πραγματικό αριθμό και συμβολίζεται $P(w_i)$ και πρέπει να ισχύει:

$$0 \leq P(A) \leq 1$$

$$P(w_1) + P(w_2) + \dots + P(w_v) = 1$$

Ο αριθμός $P(w_i)$ λέγεται **πιθανότητα του ενδεχομένου** $\{w_i\}$, ενώ ως πιθανότητα ενός ενδεχομένου $A = \{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_\kappa\} \neq \emptyset$, ορίζουμε το άθροισμα:

$P(\alpha_1) + P(\alpha_2) + \dots + P(\alpha_\kappa)$. Ως πιθανότητα του αδύνατου ενδεχομένου $\emptyset$, ορίζεται ο αριθμός $P(\emptyset) = 0$ και ισχύει $P(\Omega) = 1$.

Έτσι, η πιθανότητα ενός διαφόρων συμβάντων έχει πάντα σύνολο ένα.

2.7.2.1 Πράξεις – Κανόνες με Πιθανότητες

Για τον υπολογισμό των πιθανοτήτων, ισχύουν οι παρακάτω προτάσεις.

1) Για οποιαδήποτε **ασυμβίβαστα** μεταξύ τους ενδεχόμενα Α και Β ισχύει:

$$\text{Αν } A \cap B = \emptyset, \text{ τότε } P(A \cup B) = P(A) + P(B)$$

2) Για δύο συμπληρωματικά ενδεχόμενα Α και Α΄ ισχύει:

$$P(A') = 1 - P(A) \Leftrightarrow P(A) = 1 - P(A')$$

3) Για δύο ενδεχόμενα Α και Β ενός δειγματικού χώρο με $A \cap B = \emptyset$, ισχύει:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

4) Αν $A \subseteq B$, τότε:

$$P(A) \leq P(B)$$

5) Για δύο ενδεχόμενα ενός δειγματικού χώρου Ω ισχύει:

$$P(A - B) = P(A) - P(A \cap B)$$

ΕΦΑΡΜΟΓΗ 1

Έχουμε ένα αμερόληπτο κέρμα και κατά το στρίψιμό του έχουμε το δειγματικό χώρο Ω = {1, 2, 3, 4, 5, 6} και έτσι N(Ω) = 6, με ενδεχόμενο Α να είναι:

Α: «Το αποτέλεσμα της ρίψης είναι μικρότερο του 5», δηλαδή:

$$A = \{1, 2, 3, 4\} \text{ και } N(A) = 4.$$

Η πιθανότητα να πραγματοποιηθεί το ενδεχόμενο Α είναι:

$$P(A) = \frac{N(A)}{N(\Omega)} = \frac{4}{6} = \frac{2}{3}.$$

Με λίγα λόγια, έτσι μπορούμε να πούμε πόσο πιθανά θεωρούμε να συμβούν αυτά που μπορούν να συμβούν.

ΕΦΑΡΜΟΓΗ 2

Είχαμε δει στην προηγούμενη παράγραφο το παράδειγμα που στρίβουμε ένα κέρμα 3 φορές και είχαμε βρει το δειγματικό χώρο του πειράματος αυτού με Ω = { ΓΚΚ,ΓΚΓ,ΓΓΚ,ΓΓΓ,ΚΚΚ,ΚΚΓ,ΚΓΚ,ΚΓΓ}. Ποια είναι η πιθανότητα να φέρουμε δύο φορές κορόνα;

Αφού ο δειγματικός χώρος είναι Ω = { ΓΚΚ,ΓΚΓ,ΓΓΚ,ΓΓΓ,ΚΚΚ,ΚΚΓ,ΚΓΚ,ΚΓΓ}, τότε Ν(Ω) = 8.

Έστω το ενδεχόμενο Α: «Να φέρουμε δύο φορές κορόνα».

Τότε θα είναι:

Α = {ΚΚΓ, ΚΓΚ, ΓΚΚ} και Ν(Α) = 3.

Επομένως:

$$P(A) = = \frac{N(A)}{N(\Omega)} = \frac{3}{8}.$$

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Έστω ο δειγματικός χώρος Ω = {1,2,3,…,10}, Α = {1,2,3}, Β = {3,4,5}, Γ {4,5,9,10}. Να βρείτε τα παρακάτω σύνολα:

   i. $A \cap B$, ii. $A \cup B$ και iii. $A \cap \Gamma$

2) Τα αποτελέσματα τριών αγώνων μπάσκετ είναι τα εξής:

1: αν κερδίσει η γηπεδούχος και

2: αν κερδίσει η φιλοξενούμενη ομάδα.

Να γράψετε το δειγματικό χώρο του πειράματος.

3) Ρίχνουμε ένα ζάρι δύο φορές.

α) Να γράψετε το δειγματικό χώρο του πειράματος

β) Να βρείτε τα ενδεχόμενα, όταν:

Α: «Και οι δύο ενδείξεις να είναι μεγαλύτερες του 3»

Β: «Το άθροισμα των ενδείξεων να είναι 10»

Γ: «Και οι δύο ενδείξεις να είναι άρτιες»

γ) Να γράψετε τα ενδεχόμενα: $A \cap \Gamma$ και $B \cap \Gamma$

4) Μια κάλπη περιέχει 3 κίτρινες και 2 μαύρες σφαίρες αριθμημένες με 1, 2, 3 και 4, 5 αντίστοιχα. Επιλέγουμε μία από αυτές με τους εξής τρόπους:

Α: Ταυτόχρονα

Β: Μία μία

Γ: με επανατοποθέτηση .

Να γράψτε τον δειγματικό χώρο του πειράματος.

5) Αν Α και Β είναι δύο ενδεχόμενα ενός δειγματικού χώρου, να εκφράσετε με την βοήθεια των συνόλων τα παρακάτω ενδεχόμενα:

Γ: «Πραγματοποιείται ένα τουλάχιστον από τα Α και Β»

Δ: «Πραγματοποιείται και το Α και το Β»

Ε: Πραγματοποιείται το πολύ ένα από τα Α και Β»

Ζ: «Πραγματοποιείται ακριβώς ένα από τα Α και Β.

6) Ασανσέρ μεταφέρει δύο άτομα και σταματά σε 3 ορόφους. Να βρείτε τις πιθανότητες των παρακάτω ενδεχομένων:

Α: «Να κατέβουν και οι δύο στον ίδιο όροφο».

Β: «Ο ένας από τους δύο να κατέβει στον πρώτο όροφο».

7) Μια κάλπη περιέχει 4 κόκκινες σφαίρες, 3 κίτρινες και 3 μαύρες. Επιλέγουμε μία από αυτές. Να βρείτε την πιθανότητα να είναι:

i. κίτρινη, ii. Κίτρινη ή μαύρη.

8) Στρίβουμε ένα κέρμα 3 φορές. Να βρείτε τις πιθανότητες των ενδεχομένων:

Α: «Δύο τουλάχιστον ενδείξεις να είναι γράμματα».

Β: «Δύο το πολύ ενδείξεις να είναι γράμματα».

9) Ρίχνουμε δύο ζάρια. Ποια είναι η πιθανότητα να φέρουμε ένα τουλάχιστον άσσο;

10) Επιλέγουμε τυχαία ένα μονοψήφιο αριθμό. Θεωρούμε τα ενδεχόμενα:

$A = \{0, 1, 2, 3\}$, $B = \{x \in \Omega: \text{άρτιος}\}$, $\Gamma = \{ x \in \Omega: x \geq 7\}$. Να βρείτε τις πιθανότητες των παρακάτω ενδεχομένων:

i. A, ii. $A \cup B$ iii. $B \cap \Gamma$ iii. Γ' και iv. $B' \cap A$.

2.8 Μιγαδικοί Αριθμοί

2.8.1 Η Έννοια του Μιγαδικού Αριθμού

Οι μιγαδικοί αριθμοί στα μαθηματικά αποτελούν μια επέκταση του συνόλου των πραγματικών αριθμών $\mathbb{R}$**.** Το σύνολο των μιγαδικών αριθμών συμβολίζεται με το γράμμα $\mathbb{C}$. Τα στοιχεία του $\mathbb{C}$ **λέγονται μιγαδικοί αριθμοί**.

Οι μιγαδικοί αριθμοί εφευρέθηκαν έναν μαθηματικό Ιταλό τον Τζερόλαμο Καρντάνο, στην προσπάθειά του να αναζητήσει αναλυτικές λύσεις σε κυβικές εξισώσεις και έτσι τους ονόμασε φανταστικές.

Γνωρίζουμε ότι μια εξίσωση 2ου βαθμού με αρνητική διακρίνουσα, δεν έχει λύση στο σύνολο των πραγματικών αριθμών .

ΠΑΡΑΔΕΙΓΜΑ 1

Η εξίσωση $x^2+1 = 0 \Leftrightarrow x^2 = -1$ δεν έχει λύση στο σύνολο των πραγματικών αριθμών $\mathbb{R}$. Παρατηρούμε ότι το τετράγωνο ενός πραγματικού x είναι ίσο με έναν αρνητικό αριθμό -1, που αυτό είναι αδύνατο και επομένως δεν έχει λύση.

Έτσι το σύνολο των μιγαδικών αριθμών $\mathbb{C}$, μας διευκολύνει και ισχύει στο σύνολο $\mathbb{C}$, ότι υπάρχει μια τουλάχιστον ρίζα της εξίσωσης $x^2 = -1$.

Επομένως, **υπάρχει ένα στοιχείο i, ώστε να ισχύει $i^2 = -1$**, όπου το i **ονομάζεται φανταστική μονάδα.**

Οι **μιγαδικοί αριθμοί αποτελούνται από δύο μέρη**, έναν **πραγματικό αριθμό α** και έναν **φανταστικό αριθμό βi** και **σχηματίζονται από το άθροισμά** τους. Δηλαδή οι μιγαδικοί αριθμοί, έχουν τη μορφή **z = α + βi.**

Το σύνολο των φανταστικών αριθμών συμβολίζεται με I.

Συμβολικά, για έναν μιγαδικό αριθμό z = α + βi, γράφουμε:

α = Re(z), για το **πραγματικό** μέρος και

β = Im(z), για το **φανταστικό** μέρος.

Επομένως,

Η έκφραση **α + βi** , όπου α, β ϵ $\mathbb{R}$ **παριστάνει έναν μιγαδικό αριθμό** $\mathbb{C}$, όπου **α** είναι ένας **πραγματικός αριθμός** και **βi** λέμε έναν **φανταστικό αριθμό**.

ΠΑΡΑΔΕΙΓΜΑ 2

Να βρείτε το πραγματικό και το φανταστικό μέρος των παρακάτω μιγαδικών αριθμών:

i. z = 2 -8i ii. w = 10 iii. r = -2i

Έχουμε:
Το πραγματικό μέρος ενός μιγαδικού αριθμού z, συμβολίζεται με Re(z) και το φανταστικό του μέρος συμβολίζεται με Im(z).
Έτσι έχουμε:

i. Για τον μιγαδικό αριθμό z = 2 -8i, είναι:
Re(z) = 2 και Im(z) = - 8i

ii. Για τον μιγαδικό αριθμό w = 10, είναι:
Re(z) = 10 και Im(z) = 0

iii. Για τον μιγαδικό αριθμό r = -2i, είναι:
Re(z) = 0 και Im(z) = - 2i

Ας Θυμηθούμε

Τα βασικά **σύνολα αριθμών** είναι:

- Το σύνολο των **φυσικών αριθμών** $\mathbb{N} = \{0,1,2,3, \dots\}$.
- Το σύνολο των **ακεραίων αριθμών** $\mathbb{Z} = \{\dots, -2, -1,0,1,2, \dots\}$.
- Το σύνολο των **ρητών αριθμών** $\mathbb{Q} = \{\frac{\alpha}{\beta} / \alpha \in Z \text{ και } \beta \in N^*\}$.
- Το σύνολο των **πραγματικών αριθμών** $\mathbb{R}$.
- Το σύνολο των **φανταστικών αριθμών** $\mathbb{I}$.
- Το σύνολο των **μιγαδικών αριθμών** $\mathbb{C}$.

Για τα παραπάνω σύνολα ισχύουν τα εξής:

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C} \quad \text{και} \quad \mathbb{I} \subset \mathbb{C}$$

Στο σύνολο $\mathbb{C}$ επεκτείνονται όλες οι πράξεις των πραγματικών αριθμών ακριβώς ίδιες όπως με τις αλγεβρικές πράξεις και ιδιότητες των πράξεων αυτών. Συγκεκριμένα, ορίζονται οι πράξεις της πρόσθεσης, της αφαίρεσης, του πολλαπλασιασμού και της διαίρεσης όπως και στο $\mathbb{R}$, με το μηδέν να είναι το ουδέτερο στοιχείο της πρόσθεσης και το ένα να είναι το ουδέτερο στοιχείο του πολλαπλασιασμού.

Στους μιγαδικούς αριθμούς δεν ορίζεται η διάταξη, δηλαδή δεν έχει σημασία να συγκρίνουμε δύο μιγαδικούς αριθμούς για να εξετάσουμε αν ένας μιγαδικός αριθμός είναι μεγαλύτερος ή μικρότερος από τον άλλον.

Έτσι το σύνολο $\mathbb{C}$ έχει ως στοιχεία τα εξής:

- Όλους τους πραγματικούς αριθμούς
- Όλα τα στοιχεία της μορφής βi, που είναι γινόμενα των στοιχείων του συνόλου $\mathbb{R}$ με το i
- Όλα τα αθροίσματα της μορφής α + βi, με α και β πραγματικοί αριθμοί.
- Κάθε στοιχείο z του $\mathbb{C}$ γράφεται με μοναδικό τρόπο με τη μορφή

$$z = \alpha + \beta i \qquad \text{όπου } \alpha, \beta \in \mathbb{R}.$$

Οι μιγαδικοί αριθμοί z = α + βi, συμβολίζονται και ως διατεταγμένα ζεύγη (α, β), όπου α είναι το πραγματικό μέρος και β το φανταστικό μέρος του μιγαδικού αριθμού. Συμβολικά γράφουμε z = (α, β).
Το σύνολο των μιγαδικών αριθμών, περιέχει στοιχεία της μορφής z = (α, β).

Ο μιγαδικός αριθμός **z = 0 + 0i**, αποτελεί το **μηδενικό** στοιχείο του συνόλου των μιγαδικών αριθμών.

Για κάθε μιγαδικό αριθμό z = α + βi, ορίζεται ο **αντίθετός** του που είναι ο - z = - (α + βi).

Δηλαδή,

$$-z = -\alpha - \beta i \qquad \text{όπου } \alpha, \beta \in \mathbb{R}.$$

2.8.2 Ισότητα Μιγαδικών Αριθμών

Έχουμε δύο μιγαδικούς αριθμούς $z_1 = \alpha_1 + \beta_1 i$ και $z_2 = \alpha_2 + \beta_2 i$.

Οι **δύο μιγαδικοί** αριθμοί z_1, z_2 είναι **ίσοι** $\mathbf{z_1 = z_2}$ αν κα μόνο αν είναι **α = γ** και **β = δ.** Δηλαδή αν τα **πραγματικά τους μέρη** και τα **φανταστικά τους μέρη** είναι μεταξύ τους **ίσα**.
Συμβολικά γράφουμε:

$$\alpha_1 + \beta_1 i = \alpha_2 + \beta_2 i \Leftrightarrow \alpha_1 = \alpha_2 \text{ και } \beta_1 = \beta_2$$

Ακόμα, επειδή 0 = 0 + 0i, έχουμε:

$$\alpha + \beta i = 0 \Leftrightarrow \alpha_1 = 0 \text{ και } \beta = 0$$

Προσοχή

Στην επέκταση από το $\mathbb{R}$ στο $\mathbb{C}$, παρόλο που εξακολουθούν να ισχύουν και στο $\mathbb{C}$ οι πράξεις και οι ιδιότητες αυτών, η διάταξη και οι ιδιότητές της δεν μεταφέρονται.

ΕΦΑΡΜΟΓΗ 1

Να βρείτε για ποιες τιμές των x, y ϵ $\mathbb{R}$ είναι ίσιοι οι μιγαδικοί αριθμοί:
$z = 2x + y + (x - y)i$ και $w = y – 4 + (3x + y - 2)i$.

Έχουμε:

$z = w \Leftrightarrow 2x + y + (x - y)i = y – 4 + (3x + y - 2)i$
Από τον ορισμό της ισότητα δύο μιγαδικών αριθμών είναι:
$2x + y = y - 4 \Leftrightarrow$ και $2x + y + xi – yi - y + 4 - 3xi + 2yi = 0 \Leftrightarrow 2x – 2xi + yi + 4 = 0 \Leftrightarrow 2x – (2x – y)i + 4 = 0$.

2.8.3 Πράξεις στο σύνολο των Μιγαδικών Αριθμών

Στο σύνολο $\mathbb{C}$ των μιγαδικών αριθμών ορίζονται οι πράξεις της πρόσθεσης και του πολλαπλασιασμού.

Πρόσθεση

Έχουμε δύο μιγαδικούς αριθμούς $z_1 = \alpha_1 + \beta_1 i$ και $z_2 = \alpha_2 + \beta_2 i$.
Το άθροισμά τους $z_1 + z_2$ τους ορίζεται από τη σχέση:
$z_1 + z_2 = (\alpha_1 + \beta_1 i) + (\alpha_2 + \beta_2 i) \Leftrightarrow$

$$z_1 + z_2 = (\alpha_1 + \alpha_2) + (\beta_1 + \beta_2) i$$

Αφαίρεση

Η διαφορά $z_1 - z_2$ δύο μιγαδικών αριθμών $z_1 = \alpha_1 + \beta_1 i$ και $z_2 = \alpha_2 + \beta_2 i$, ορίζεται από τη σχέση:

$$z_1 - z_2 = z_1 + (- z_2)$$

Ο αντίθετος του z_2 $= - \alpha_2 - \beta_2 i$ και είναι:
$z_1 + (- z_2) = (\alpha_1 + \beta_1 i) + [-(\alpha_2 + \beta_2 i)] = \alpha_1 + \beta_1 i - \alpha_2 - \beta_2 i$

$$z_1 - z_2 = (\alpha_1 - \alpha_2) + (\beta_1 - \beta_2) i$$

2.8.3.1 Οι ιδιότητες της πρόσθεσης

Στην πρόσθεση των μιγαδικών αριθμών ισχύουν οι παρακάτω ιδιότητες:

- **Αντιμεταθετική**

$z_1 + z_2 = z_2 + z_1$

- **Προσεταιριστική**

$(z_1 + z_2) + z_3 = z_1 + (z_2 + z_3)$

- **Αντίθετο στοιχείο**

$z + (- z) = - z + z = 0$

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε τους μιγαδικούς αριθμούς $z_1 = 4 + 6i$ και $z_2 = 2 + 3i$, τότε το άθροισμα και η διαφορά τους είναι:

- $z_1 + z_2 = (4 + 6i) + (2 + 3i) = (4 + 2) + (6 + 3)i = 6 + 5i.$
- $z_1 - z_2 = (4 + 6i) - (2 + 3i) = (4 - 2) + (6 - 3)i = 2 + 3i.$

2.8.3.2 Πολλαπλασιασμός

Για τους μιγαδικούς αριθμούς $z_1 = \alpha_1 + \beta_1 i$ και $z_2 = \alpha_2 + \beta_2 i$, το γινόμενό τους ορίζεται από τη σχέση:

$$z_1 \cdot z_2 = (\alpha_1 + \beta_1 i) \cdot (\alpha_2 + \beta_2 i)$$

$$z_1 \cdot z_2 = (\alpha_1 \cdot \alpha_2 - \beta_1 \cdot \beta_2) + (\alpha_1 \cdot \beta_2 + \alpha_2 \cdot \beta_1) i$$

2.8.3.3 Οι ιδιότητες του πολλαπλασιασμό

Στον πολλαπλασιασμό των μιγαδικών αριθμών ισχύουν οι παρακάτω ιδιότητες:

- **Αντιμεταθετική**

$z_1 \cdot z_2 = z_2 \cdot z_1$

- **Προσεταιριστική**

$(z_1 \cdot z_2) \cdot z_3 = z_1 \cdot (z_2 \cdot z_3)$

- **Επιμεριστική**

$z_1 \cdot (z_2 + z_3) = z_1 \cdot z_2 + z_1 \cdot z_3$

- **Ουδέτερο στοιχείο**

1 = 1 + 0 i
$1 \cdot z = z \cdot 1 = z$

Προσοχή
Για κάθε $z \in \mathbb{C}$, με $z \neq 0$ υπάρχει ένα και μόνο $z^* \in \mathbb{C}$, τέτοιο ώστε $z \cdot z^* = 1$.

Ο **αντίστροφος** του **μιγαδικού αριθμού** $z = \alpha + \beta i \neq 0$ ονομάζεται το πηλίκο $\frac{1}{z}$ και συμβολίζεται με z-1.

Έτσι για $z \neq 0$, είναι:

$$z^{-1} = \frac{1}{z} = \frac{1}{\alpha + \beta i} = \frac{a}{\alpha^2 + \beta^2} + \frac{-\beta}{\alpha^2 + \beta^2}$$

2.8.3.4 Πηλίκο

Το πηλίκο $\frac{z_1}{z_2}$ δύο μιγαδικών αριθμών $z_1 = \alpha_1 + \beta_1 i$ και $z_2 = \alpha_2 + \beta_2 i$, με $z_2 \neq 0$ ορίζεται από τη σχέση:

$$\frac{z_1}{z_2} = z_1 \cdot z_2^{-1}$$

Προσοχή
Οι δυνάμεις $(\alpha + \beta i)^\kappa$, με $\kappa \in \mathbb{Z}$ ορίζονται όπως το $\mathbb{R}$.

2.8.3.5 Δυνάμεις του i

Οι δυνάμεις ενός μιγαδικού αριθμούz με εκθέτη ακέραιο αριθμό ορίζονται όπως ακριβώς και στους πραγματικούς αριθμούς.
Είναι:

- $z^1 = z$
- $z^2 = z \cdot z$ και γενικά
- $z^\nu = z^{\nu-1} \cdot z$ για κάθε θετικό ακέραιο , με $\nu \geq 0$

Ακόμα αν $z \neq 0$, ορίζεται :

- $z^0 = 1$
- $z^{-\nu} = \frac{1}{z^\nu}$, για κάθε $\nu > 0$

Οι ιδιότητες για τις δυνάμεις των μιγαδικών αριθμών είναι ίδιες όπως των πραγματικών αριθμών και για τις δυνάμεις του ισχύει:

- $i^0 = 1$
- $i^1 = i$
- $i^2 = -1$
- $i^3 = i^2 \cdot i = (-1) \cdot i = -i$ ενώ για
- $i^4 = i^2 \cdot i^2 \cdot i = (-1) \cdot (-1) = 1$
- $i^5 = i^4 \cdot i = 1 \cdot i = i$
- $i^6 = i^4 \cdot i^2 = 1 \cdot (-1) = -1$
- $i^7 = i^4 \cdot i^3 = 1 \cdot (-i) = -i$

Παρατηρούμε ότι οι δυνάμεις του i με εκθέτη 4 και μετά i^4, οι τιμές του i^v επαναλαμβάνονται. Δηλαδή για να υπολογίσουμε τις δυνάμεις του i, μπορούμε να γράφουμε τον εκθέτη στη μορφή:

$$v = 4\rho + \upsilon$$

όπου ρ: το πηλίκο και ν: το υπόλοιπο ευκλείδειας διαίρεσης του ν με το 4.

Οπότε έχουμε:

$$i^v = i^{4\rho+\upsilon} = i^{4\rho} \cdot i^\upsilon = (i^4)^\rho \cdot i^\upsilon = 1^\rho \cdot i^\upsilon = i^\upsilon = \begin{cases} 1, & \alpha\nu\ \upsilon = 0 \\ i, & \alpha\nu\ \upsilon = 1 \\ -1, & \alpha\nu\ \upsilon = 2 \\ -i, & \alpha\nu\ \upsilon = 3 \end{cases}$$

Γενικά, για κάθε $\nu \in \mathbb{N}$ είναι:

- $i^{4v+1} = i$
- $i^{4v+2} = -1$,
- $i^{4v+3} = -i$
- $i^{4v} = 1$,

Οπότε κάθε δύναμη του i^κ του i, με κ, $\upsilon \in \mathbb{N}$ και $0 \leq \upsilon < 4$ γράφεται:

$$i^\kappa = i^{4v+\upsilon}$$

ΠΑΡΑΔΕΙΓΜΑ 2

Είναι:

- $i^9 = = i^{4\cdot 2+1} = i^3 = -i$
- $i^{12} = i^{2\cdot 6} =$

- $i^{14} = i^{3\cdot 4+2} = i^2 = i^2 = -1$
- $i^{15} = = i^{4\cdot 3+3} = i^3 = -i$
- $i^{16} = = i^{4\cdot 4+0} = i^0 = 1$

Ας Θυμηθούμε

Οι **βασικές ταυτότητες** που ισχύουν και στους μιγαδικούς αριθμούς είναι:

- $(\alpha + \beta)^2 = \alpha^2 + \beta^2 + 2\alpha\cdot\beta$
- $(\alpha - \beta)^2 = \alpha^2 + \beta^2 - 2\alpha\cdot\beta$
- $\alpha^2 - \beta^2 = (\alpha - \beta)\cdot(\alpha + \beta)$
- $(\alpha + \beta)^3 = \alpha^3 + \beta^3 + 3\alpha^2\cdot\beta + 3\alpha\cdot\beta^2$
- $\alpha^3 + \beta^3 = (\alpha + \beta)\cdot(\alpha^2 + \beta^2 - \alpha\cdot\beta)$
- $\alpha^3 - \beta^3 = (\alpha - \beta)\cdot(\alpha^2 + \beta^2 - \alpha\cdot\beta)$

ΕΦΑΡΜΟΓΗ 1

Έχουμε τους μιγαδικούς αριθμούς $z_1 = 4 - 5i$ και $z_2 = 3 + 2i$, τότε το γινόμενό και το πηλίκο τους είναι:

- $z_1\cdot z_2 = (4 - 5i) + (3 + 2i) = [4\cdot 3 - (-5)\cdot 2] + [(4\cdot 2) + (-5)\cdot 3]I = (12+10) + (8-15)i =$
  $= 22 - 7i.$
- $\frac{z_1}{z_2} = \frac{4-5i}{3+2i} = \frac{(4-5i)(3+2i)}{(3+2i)(3+2i)} = \frac{(12+8i-15-10i)}{3^2-2i^2} = \frac{-3-2i}{9-2^2(i^2)} = \frac{-3-2i}{9+4(-1)}$

  $= \frac{-3-2i}{13} = -\frac{3}{13} - \frac{2}{13}i$.

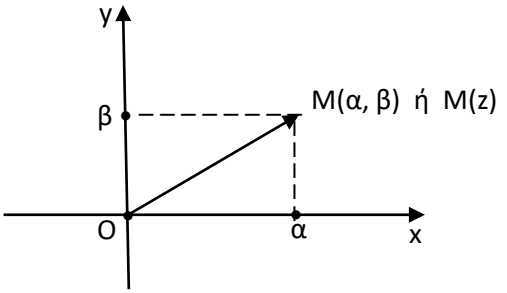
2.8.4 Γεωμετρική Παράσταση Μιγαδικών Αριθμών

Για κάθε μιγαδικό αριθμό $z = \alpha + \beta i$, μπορούμε να τον απεικονίσουμε σε ένα καρτεσιανό επίπεδο στο σημείο Μ(α, β) και αντιστρόφως. Δηλαδή κάθε σημείο Μ(α, β) του καρτεσιανού επιπέδου μπορούμε να το αντιστοιχίσουμε στο μιγαδικό αριθμό $z = \alpha + \beta i$.

Το σημείο αυτό Μ(α, β) λέγεται **εικόνα** του μιγαδικού αριθμού $z = \alpha + \beta i$.

Το σημείο Μ(α, β) το συμβολίζουμε και με Μ(z).

Ένα καρτεσιανό επίπεδο του οποίου τα σημεία είναι εικόνες των μιγαδικών αριθμών θα αναφέρεται ως **μιγαδικό επίπεδο**.



• Στον άξονα x' x ανήκουν τα σημεία Μ(α, 0) που είναι εικόνες των πραγματικών αριθμών z = α + β i και ο άξονα x' x λέγεται **πραγματικός άξονας**.

• Στον άξονα y' y ανήκουν τα σημεία Μ(0, β) που είναι εικόνες των πραγματικών αριθμών z = α + β i και ο άξονα y' y λέγεται **φανταστικός άξονας**.

• Ένας μιγαδικός αριθμός z = α + β i μπορεί να αναπαρασταθεί στο μιγαδικό επίπεδο με τη διανυσματική ακτίνα $\overrightarrow{OM}$, που έχει αρχή το κέντρο Ο των αξόνων και τέλος το σημείο Μ(α, β).

2.8.5 Μέτρο Μιγαδικού Αριθμού

Το μέτρο του ενός μιγαδικού αριθμού, ορίζεται ως το μέτρο του διανύσματος $\overrightarrow{OM}$, ή ως η απόσταση του σημείου Μ από το κέντρο Ο του μιγαδικού επιπέδου.
Είναι $\overrightarrow{OM}$ (α, β).

ΟΡΙΣΜΟΣ

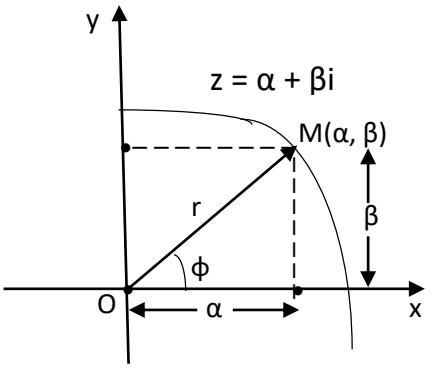
Ονομάζουμε **μέτρο** ενός μιγαδικού αριθμού z = α + β i τον μη αρνητικό αριθμό αριθμού $\sqrt{\alpha^2 + \beta^2}$ και το συμβολίζουμε |z|.

Έτσι,

$$|z| = |\alpha + \beta i| = \sqrt{\alpha^2 + \beta^2} \geq 0$$

Είναι $\overrightarrow{OM}$ είναι η διανυσματική ακτίνα του μιγαδικού αριθμού z = α + βi με

$\overrightarrow{OM} = \sqrt{\alpha^2 + \beta^2} = r$.



Παρατηρούμε ότι όλοι οι μιγαδικοί αριθμοί, που έχουν το ίδιο μέτρο με τον z έχουν τις εικόνες τους στον κύκλο (0, r). Η εικόνα ενός συγκεκριμένου μιγαδικού αριθμού z στο μιγαδικό επίπεδο προσδιορίζεται ακριβώς αν γνωρίζουμε το μέτρο του και τη γωνία ϕ που σχηματίζει η διανυσματική του ακτίνα $\overrightarrow{OM}$, με τον θετικό ημιάξονα Ox.

Το ζεύγος (r, ϕ) προσδιορίζει πλήρως την εικόνα Μ(α, β) του μιγαδικού αριθμού z, όπου r το μέτρο του και ϕ ϵ [0, 2kπ).

Οι συντεταγμένες (r, ϕ) και οι καρτεσιανές συνταραγμένες συνδέονται με τις παρακάτω σχέσεις:

$r = \sqrt{\alpha^2 + \beta^2}$, με $0 \leq \varphi \leq 2\pi$

$$\cos\varphi = \frac{\alpha}{r} = \frac{\alpha}{\sqrt{\alpha^2+\beta^2}}, \quad \sin\varphi = \frac{\beta}{r} = \frac{\beta}{\sqrt{\alpha^2+\beta^2}}$$

Από τις σχέσεις προκύπτει, α = r · cos ϕ και β = r · sin ϕ .

Ο μιγαδικός αριθμός z = α + β i, σύμφωνα με τις παραπάνω σχέσεις γίνεται:

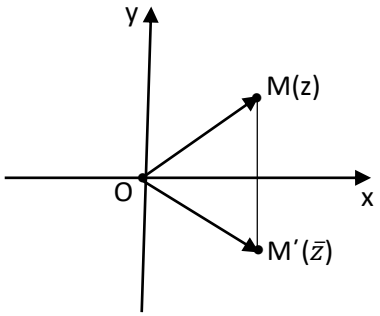
z = r · cos ϕ + i · sin ϕ ⇒ **z = r · (cos ϕ + i · sin ϕ)**

όπου $r = \sqrt{\alpha^2 + \beta^2}$, με $0 \leq \varphi \leq 2\pi$.

2.8.6 Συζυγείς Μιγαδικοί Αριθμοί

Για κάθε μιγαδικό αριθμό z = α + β i, ορίζεται ο μιγαδικός αριθμός $\bar{z} = \alpha - \beta i$ ο οποίος λέγεται **συζυγής** του z.

Στο μιγαδικό επίπεδο οι εικόνες Μ(α, β) και Μ(α, -β) δύο συζυγών μιγαδικών αριθμών z = α + β i, όπου ο z είναι συμμετρικά ως προς τον πραγματικό αριθμό.



Είναι:

$$(\bar{z}) = \alpha + \beta i = z$$

Οι μιγαδικοί αριθμοί z και $\bar{z}$ ονομάζονται **συζυγείς μιγαδικοί αριθμοί**.

Για δύο συζυγείς μιγαδικούς αριθμούς z = α + β i και $\bar{z} = \alpha - \beta i$ ισχύει:

$$z + \bar{z} = 2\alpha \quad \text{και} \quad z\bar{z} = 2 \cdot \alpha$$

2.8.6.1 Ιδιότητες Συζυγών Μιγαδικών Αριθμών

Για τους συζυγείς μιγαδικούς αριθμούς εύκολα αποδεικνύονται οι παρακάτω ιδιότητες:

- $\overline{z_1 + z_2} = \bar{z_1} + \bar{z_2}$ και γενικά $\overline{z_1 + z_2 + \cdots + z_\nu} = \bar{z_1} + \bar{z_2} + \cdots + \bar{z_\nu}$, $\nu \in \mathbb{N}$
- $\overline{z_1 - z_2} = \bar{z_1} - \bar{z_2}$
- $\overline{z_1 \cdot z_2} = \bar{z_1}.\bar{z_2}$ και γενικά $\overline{z_1 \cdot z_2 \ldots z_\nu} = \bar{z_1}\bar{z_2} \ldots \bar{z_\nu}$, $\nu \in \mathbb{N}$
- $(\overline{z^\nu}) = (\bar{z})^\nu$, $\nu \in \mathbb{N}$
- $\left(\overline{z^{-1}}\right) = (\bar{z})^{-1}$, $z \neq 0$
- $\left(\frac{\bar{z_1}}{z_2}\right) = \frac{\overline{z_1}}{\overline{z_2}}$, $z_2 \neq 0$

• $\overline{(\kappa \cdot z)} = \kappa \cdot \bar{z}$, $\kappa \in \mathbb{R}$

Επειδή είναι $z\bar{z} = \alpha^2 + \beta^2$, έχουμε:

$$|z| = \sqrt{z \cdot \bar{z}}$$

Αν ο z είναι πραγματικός αριθμός το μέτρο του συμπίπτει με την απόλυτη τιμή του:

$$|z| = \sqrt{\alpha^2} = |\alpha|$$

Για κάθε μιγαδικό αριθμό z = α + β i , με β≠0 ισχύει:

$$|z|^2 \neq z^2$$

2.8.6.2 Ιδιότητες του μέτρου

• $|z| = |-z| = |\bar{z}|$

• $|z_1 z_2| = |z_1|.|z_2|$ και γενικά $|z_1 z_2 \dots z_\nu| = |z_1|.|z_2| \dots |z_\nu|$, $\nu \in \mathbb{N}$

• $|z^\nu| = |z|^\nu$, $\nu \in \mathbb{N}$

• $|z^{-1}| = |z|^{-1}$, $z \neq 0$

• $\left|\frac{z_1}{z_2}\right| = \frac{|z_1|}{|z_2|}$, $z_2 \neq 0$

• $|z_1 + z_2| \leq |z_1| + |z_2|$ (τριγωνομετρική ισότητα)

ΕΦΑΡΜΟΓΗ 1

Να βρείτε τους συζυγείς αριθμούς των παρακάτω μιγαδικών αριθμών:

i. $z_1 = 5 + 2i$ ii. $z_2 = 3 - i$ και iii. $z_3 = -4i$

Οι συζυγείς αριθμοί των z_1 , z_2 και z_3 είναι αντίστοιχα $\bar{z_1}$, $\bar{z_2}$ και $\bar{z_3}$.

Έχουμε:

i. $\bar{z_1} = \overline{5 + 2i} = 5 - 2i$

ii. $\bar{z_2} = \overline{3 - i} = 3 + i$

iii. $\bar{z_3} = \overline{-4i} = \overline{0 - 4i} = 0 + 4i = 4i$

ΕΦΑΡΜΟΓΗ 2

Να βρείτε τα μέτρα των παρακάτω μιγαδικών αριθμών:

ii. $z_1 = -3 + 4i$ ii. $z_2 = -5$ και iii. $z_3 = 2i$

Τα μέτρα από τους συζυγείς αριθμούς z_1, z_2 και z_3 είναι :

i. $|\mathbf{z_1}| = |-3 + 4i| = \sqrt{(-3)^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$

ii. $|z_2| = |-5 + 0i| = \sqrt{(-5)^2 + 0^2} = \sqrt{25} = 5$

iii. $|z_3| = |4i| = \sqrt{0^2 + 2} = \sqrt{4} = 16$

ΕΦΑΡΜΟΓΗ 3

Αν για τον μιγαδικό αριθμό z ισχύει ότι $|-z| = 3$, να βρείτε το μέτρο του μιγαδικού αριθμού $w = \frac{z^{12}}{\bar{z}^{10}}$.

Αφού $|-z|=3$, ισχύει και $|z|=|\bar{z}|=3$.

Επομένως έχουμε:

$|w| = \left|\frac{z^{12}}{\bar{z}^{10}}\right| = \frac{|z^{12}|}{|\bar{z}^{10}|} = \frac{|z|^{12}}{|z|^{10}} = \frac{|3|^{12}}{|3|^{10}} = \frac{3^{12}}{3^{10}} = 3^{12-10} = 3^2 = 9.$

2.9 Εκθετική Μορφή Μιγαδικών Αριθμών

Ένας μιγαδικός αριθμός γράφεται και σε πολική ή τριγωνομετρική μορφή.
Αυτές οι συντεταγμένες ενός μιγαδικού αριθμού είναι το ζεύγος (r, φ) όπου r=|z| είναι το μέτρο του μιγαδικού αριθμού φ του ορίσματος z.

Έτσι κάθε μιγαδικός αριθμός γράφεται στην εκθετική μορφή:

$$z = x + i\,y = r \cdot (\cos\phi + i \cdot \sin\phi) = r \cdot e^{i\varphi}$$

όπου: $r = |z| = \sqrt{x^2 + y^2} \geq 0$ και ϕ το όρισμα

Χρησιμοποιώντας την **εξίσωση του Όιλερ** η τριγωνομετρική μορφή μετατρέπεται στην **εκθετική μορφή**:

$$z = |z|^{i\varphi}$$

Αντικαθιστούμε στην $e^{i\varphi} = \cos\phi + i \cdot \sin\phi$ το i με το –i, για να βρούμε τη συζυγή μορφή $e^{-i\varphi} = \cos\phi - i \cdot \sin\phi$. Προσθέτοντας και αφαιρώντας τις δύο τελευταίες σχέσεις βρίσκουμε:

$$\cos\phi = \frac{1}{2}(e^{i\varphi} + e^{-i\varphi}) \qquad \text{και} \qquad \sin\phi = \frac{1}{2i}(e^{i\varphi} - e^{-i\varphi})$$

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Να προσδιοριστούν τα x, y ϵ $\mathbb{R}$, ώστε οι παρακάτω μιγαδικοί αριθμοί:

i.(3x+11y)-(2x-5y)i και 1-2i να είναι ίσοι.

ii. z_1=x-5y-3i και z_2=-3+i(2x-y) να είναι συζυγείς.

2) Να υπολογιστούν τα παρακάτω:

i. $\frac{5+2i}{2+3i}$ ii. $\frac{7-2i}{4+3i}$ iv. $2i + i^3 - i^9 + i^{12}$ και v. $-i^3 + 3i^2 - 5i^7$

3) Να γίνουν οι παρακάτω πράξεις των μιγαδικών αριθμών:

i. $(3 + 2i)\cdot(2 + i)$

ii. $(4 - i)\cdot(1 + 2i)\cdot i$

iii. $(5 + i)\cdot(2 - i) - (2 - 3i)\cdot(4 + i)$

iv. $\frac{9-8i}{5+2i}$ και v. $\frac{i(8-i)}{(3-2i)(2+i)}$

4) Να επιλυθούν οι εξισώσεις:

i. $\bar{z} = -4z$, ii. $\bar{z} = 8z$, iii. $2 - 3z + (\overline{-z}) = 0$ και iv. $3z + (\bar{z}) - 1 = 0$

5) Να βρείτε τα μέτρα των παρακάτω μιγαδικών αριθμών:

i. $\frac{2+3i}{4-i}$ ii. $\frac{5-i}{1-2i}$ και iii. $\frac{(\sqrt{2}+i)^2}{(1-2i\sqrt{3})^2}$

6) Αν z = x + y i, να βρεθούν τα x και y ώστε:

$|z - 1| = |z - 2| = |z - i|$

7) Να βρεθεί ο μιγαδικός αριθμός z που να επαληθεύει την εξίσωση:

i. $\bar{z} = 2 - z$ ii. $|z| + z^2 = 0$ και iii. $|z + 1|^2 + |z - 1|^2 - 4 = 0$

8) Να βρείτε τους συζυγείς αριθμούς των παρακάτω μιγαδικών αριθμών:

i. z_1 = 2 - 2i, ii. z_2 = - 4 – i, iii. z_3 = - 8i, iv. z_5 = - 1 + 5i

vi. z_5 = -2 και vi. z_6 = -8i

9) Να βρείτε τα μέτρα των παρακάτω μιγαδικών αριθμών:

i. z_1 = 6 + 2i, ii. z_2 = - 4, iii. z_3 = 3i, iv. z_4 = - 8 + 6i

v. z_5 = - 8 + 6i και vi. z_6 = -13i

ΚΕΦΑΛΑΙΟ: 3ο - Γλώσσα προγραμματισμού Python

ΚΕΦΑΛΑΙΟ 3ο

Python - Jupyter Notebook

Σύνοψη

Σε αυτό το κεφάλαιο αναπτύσσονται διάφορες βασικές έννοιες καθώς και η δημιουργία προγραμμάτων χρησιμοποιώντας τη γλώσσα προγραμματισμού Python. Αρχικά, γίνεται μια μικρή αναφορά στην Python, σχετικά με το τι είναι, την ιστορία της, τι μπορεί να κάνει, καθώς και γιατί να χρησιμοποιήσουμε Python. Ακολουθεί η διαδικασία εγκατάστασης της Python και του Jupyter Notebook σε υπολογιστή με Windows. Πολύ σημαντικό να τονιστεί ότι επειδή δεν είναι εφικτό να «τρέξουμε» τα κβαντικά προγράμματα, χρησιμοποιούμε βοηθητικά το Jupyter Notebook. Δηλαδή, χρησιμοποιούμε Python γράφοντας κείμενο, όχι όμως στο περιβάλλον της Python αλλά δια μέσω Jupyter Notebook. Στη συνέχεια γίνεται εκμάθηση για το πώς συντάσσονται οι βασικές εντολές στην Python και πώς εκτελούνται. Στο κεφάλαιο αυτό θα ασχοληθούμε με τις Βιβλιοθήκες numpy και radom. Επίσης, θα δούμε σε Python, απλές βασικές μεταβλητές και συναρτήσεις, δομές ελέγχου και επανάληψης, διαχείριση πινάκων, βασικές ενσωματωμένες μεθόδους που μπορούμε να χρησιμοποιήσουμε σε λίστες/πίνακες καθώς και πράξεις πινάκων και μιγαδικών αριθμών. Συγκεκριμένα, θα δούμε τελεστές, τύποι δεδομένων, τι νόημα έχουν οι μεταβλητές και οι συναρτήσεις, πώς συντάσσονται καθώς και τις μαθηματικές τους πράξεις. Θα ασχοληθούμε με τις εντολές ελέγχου και επανάληψης της for, της if και της while. Επιπλέον, δημιουργούνται προγράμματα των πράξεων της πρόσθεσης, αφαίρεσης και πολλαπλασιασμού μεταξύ πινάκων και μιγαδικών αριθμών. Τέλος, σε αυτό το κεφάλαιο για την καλύτερη εμπέδωση της ύλης του μαθητή, δίνονται λυμένα παραδείγματα και εφαρμογές καθώς και άλυτες ασκήσεις για λύση για την εποικοδομητική και αποτελεσματική μελέτη του μαθητή. Αυτό το κεφάλαιο απευθύνεται σε όλους τους μαθητές Γυμνασίου και Λυκείου εκτός της ενότητας των μιγαδικών αριθμών για τους μαθητές γυμνασίου.

3.1 Τι είναι η Python?

Εικόνα:1 - Python
Πηγή εικόνας: https://elearn-aegean.gr/product/python_programming/

Η Python είναι μια δυναμική γλώσσα προγραμματισμού υψηλού επιπέδου, γενικής χρήσης, φορητότητα (portability) και τα μοντέρνα χαρακτηριστικά της την κάνουν κατάλληλη ως πρώτη γλώσσα προγραμματισμού. Κατατάσσεται σταθερά ως μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού. Σχεδιάστηκε κυρίως για αναγνωσιμότητα, την κομψότητα, την ευχρηστία της καθώς και την απλότητα του συντακτικού της καθώς επιτρέπει στους προγραμματιστές να δημιουργήσουν μικρότερο κώδικα σε σχέση με άλλες γλώσσες προγραμματισμού, όπως είναι η java και η C++.

Ανήκει στις γλώσσες προστακτικού προγραμματισμού, δηλαδή στηρίζεται στην ιδέα ότι χρησιμοποιούμε εντολές που στηρίζονται σε κοινές μεταβλητές. Αυτή η γλώσσα προγραμματισμού είναι αντικειμενοστραφής, δηλαδή ο τρόπος και οι τεχνικές προγράμματος, μέσω μια δομής δεδομένων του είναι αυτόνομη με δικά της χαρακτηριστικά.

Η Python υποστηρίζει συλλογή απορριμμάτων, η οποία είναι μια μορφή διαχείρισης μνήμης και λειτουργεί στο υπόβαθρο, κατά την εκτέλεση ενός προγράμματος.

Η Python διαθέτει μεταγλωττιστές (compilers) και διερμηνευτές (interpreters) για να επεξεργάζεται τη μετατροπή του κώδικα σε «γλώσσα μηχανής» ώστε να εκτελεστεί από τον Η/Υ (στην περίπτωση της Python η επεξεργασία πραγματοποιείται από διερμηνευτή). Θεωρείται μια γλώσσα προγραμματισμού απλή, ισχυρή και μεσαίου επιπέδου.

Η Python είναι επεκτάσιμη και ενσωματωμένη. Αν χρειαζόμαστε ένα σημαντικό κομμάτι κώδικα να «τρέχει» πολύ γρήγορα ή αν πρέπει να έχουμε ένα κομμάτι ενός αλγορίθμου που να μην είναι ανοικτό, τότε μπορούμε να προγραμματίσουμε εκείνο το κομμάτι σε C ή C++ και στη συνέχεια να το χρησιμοποιήσουμε από το Python πρόγραμμά μας.

Αξίζει να σημειωθεί ότι αναγνωσιμότητα στον προγραμματισμό σημαίνει η ευκολία με την οποία ένας χρήστης μπορεί να κατανοήσει το σκοπό, τον έλεγχο της ροής και τη λειτουργία του πηγαίου κώδικα. Είναι πολύ σημαντική λόγω του γεγονός ότι οι προγραμματιστές αντί να συντάσσουν καινούργιο κώδικα, αφιερώνουν πολύ χρόνο προσπαθώντας να κατανοήσουν, να επαναχρησιμοποιήσουν και τροποποιήσουν τον κώδικα, που πολλές φορές μπορεί να τους οδηγήσει σε σφάλματα.

3.2 Τι κάνει η Python

Πώς μπορεί να χρησιμοποιηθεί αυτή η γλώσσα προγραμματισμού;

- ✓ Η Python χρησιμοποιηθεί (σε διακομιστή) για τη δημιουργία διαδικτυακών εφαρμογών (web applications) καθώς και την ανάπτυξη λογισμικού για τη δημιουργία διάφορων εργασιών.

- ✓ Η Python είναι εφικτό να συνδεθεί με συστήματα βάσεων δεδομένων(database systems), να διαβάσει ή και να τροποποιήσει αρχεία.

- ✓ Η Python μπορεί να διαχειριστεί αρχεία και αρχεία μεγάλου όγκου δεδομένων.

- ✓ Η Python είναι δυνατόν να εκτελέσει πολύπλοκες μαθηματικές πράξεις.

- ✓ Η Python μπορεί να χρησιμοποιηθεί για ταχύτερη δημιουργία πρωτοτύπων ή σε συνδυασμό για ανάπτυξη λογισμικού έτοιμου για παραγωγή.

3.3 Γιατί Python?

- ✓ Ουσιαστικά μόνο η Python μέχρι σήμερα διαθέτει βιβλιοθήκες που επιτρέπουν τον κβαντικό προγραμματισμό , οι οποίες αναπτύχθηκαν από πολύ μεγάλες εταιρίες όπως η IBM, Google κλπ.

- ✓ Η Python διαθέτει διερμηνέα (interpreter) αλλά και μεταγλωττιστή (compiler) ταυτόχρονα. Μπορεί να χρησιμοποιηθεί σε διαφορετικές πλατφόρμες όπως Windows, Linux, Mac, Raspberry, κ.λπ., γι’ αυτό και αποτελεί την πιο χρησιμοποιούμενη γλώσσα προγραμματισμού σήμερα.

- ✓ Είναι ελεύθερου λογισμικού ανοικτού κώδικα, δηλαδή μπορούμε να διανείμουμε ελεύθερα αντίγραφα αυτού του λογισμικού, να διαβάσουμε τον πηγαίο κώδικα, να πραγματοποιήσουμε αλλαγές σε αυτό καθώς και να χρησιμοποιήσουμε «κομμάτια» του κώδικα σε νέα δωρεάν προγράμματα. Βασίζεται στην ιδέα μιας κοινότητας που μοιράζεται τη γνώση και έτσι βελτιώνεται συνεχώς.

- ✓ Η σύνταξή της είναι απλή, είναι κατάλληλη για αρχάριους και έμπειρους προγραμματιστές, μοιάζει με την αγγλική γλώσσα με επιρροή από τα μαθηματικά, γι’ αυτό και η εκμάθησή της είναι κατανοητή και γρήγορη.

- ✓ Η Python προτείνετε ως μια πολύ καλή σύγχρονη γλώσσα με στοιχεία Διαδικτύου για μαθητές Γυμνασίου και Λυκείου, που αρχίζουν να μαθαίνουν προγραμματισμό Η/Υ.

- ✓ Παρέχει τη δυνατότητα να γράφονται προγράμματα σε λιγότερες γραμμές συγκριτικά με άλλες γλώσσες προγραμματισμού.

- ✓ Διαθέτει τεράστια γκάμα από βιβλιοθήκες που διευκολύνουν ιδιαίτερα πολλές εργασίες.
- ✓ Η Python εκτελείται σε ένα σύστημα διερμηνέα γρήγορα, συνεπώς μόλις συνταχθεί ο κώδικας εκτελείται αμέσως.

- ✓ Είναι ταυτόχρονα διαδικαστική, αντικειμενοστραφής αλλά και συναρτησιακή.

- ✓ Η Python για να ολοκληρώσει μια εντολή χρησιμοποιεί νέες γραμμές, ενώ άλλες γλώσσες προγραμματισμού χρησιμοποιούν συνήθως ερωτηματικά ή παρενθέσεις.

- ✓ Η Python για να ορίσει ένα εύρος (όπως το εύρος των βρόγχων, των συναρτήσεων και των κλάσεων) βασίζεται στη εσοχή, χρησιμοποιώντας κενό διάστημα, σε αντίθεση με άλλες γλώσσες προγραμματισμού που χρησιμοποιούν συνήθως αγκύλες.

3.4 Γνωστές Εμπορικές Εφαρμογές Γραμμένες σε Python

- YouTube
- Facebook
- Dropbox
- Google (Μεγάλη χρήση της σε βασικές εφαρμογές)
- Instagram
- Bit Torrent (Προηγούμενη Έκδοση)
- Second Life

Το μειονέκτημά της είναι πιο αργή από τις μεταγλωττιζόμενες (compiled) γλώσσες προγραμματισμού όπως είναι η C και η C^{++}, έτσι είναι ακατάλληλη για τη δημιουργία λειτουργικών συστημάτων.

3.5 Λίγη Ιστορία για την Python

Εικόνα:2 - Guido van Rossum
Πηγή εικόνας:
https://en.wikipedia.org/wiki/Guido_van_Rossum

Η Python δημιουργήθηκε το 1991 από τον Ολλανδό προγραμματιστή Γκίντο βαν Ρόσσουμ (Guido van Rossum) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI).

Το όνομά της η Python δεν το πήρε από το ερπετό (φίδι πύθωνα), ούτε αποτελεί ακρωνύμιο, αλλά οφείλεται στη χιουμοριστική σειρά του BBC «Monty Python's Flying Circus», της δεκαετίας του 1970 (καθώς ο δημιουργός της ήταν θαυμαστής). Απώτερος σκοπός του ονόματος ήταν να είναι σύντομο, μοναδικό και μυστηριώδες, γι' αυτό και την ονόμασε Python.

Αρχικά η Python ήταν γλώσσα σεναρίων ή αλλιώς γλώσσα επέκτασης, δηλαδή μια γλώσσα προγραμματισμού που επιτρέπει τον έλεγχο μιας ή περισσοτέρων εφαρμογών. Η γλώσσα προγραμματισμού στην οποία πέτυχε η Python είναι η γλώσσα προγραμματισμού ABC, η οποία είχε διασύνδεση με το λειτουργικό κατανεμημένο σύστημα Amoeba (το λογισμικό του υπολογιστή που είναι υπεύθυνο για τη διαχείριση και τον συντονισμό των εργασιών και των διαθέσιμων πόρων) και θεωρείται διάδοχος της γλώσσας προγραμματισμού ABC, αφού αποτέλεσε βασική πηγή έμπνευσης για τον Guido van Rossum.

Οι δύο από τις πιο χρησιμοποιούμενες εκδόσεις είναι η Python 2.0 και η Python 3.0, όπου υπάρχει μεγάλος ανταγωνισμός μεταξύ τους, έχοντας ποικίλους θαυμαστές.

Η Python 2.0 κυκλοφόρησε τον Οκτώβρη του 2000, όπου διακόπηκε με την έκδοση 2.7.28 του 2020. Ωστόσο, η Python 2, εκτός από ενημερώσεις ασφαλείας δεν ενημερώνεται άλλο, εξακολουθεί να είναι αρκετά δημοφιλής.

Το Δεκέμβρη του 2008 κυκλοφόρησε η έκδοση 3.0, γνωστή και Python 3000. Πολλά από τα καινούργια χαρακτηριστικά της έκδοσης αυτής, μεταφέρθηκαν στις εκδόσεις 2.6 και 2.7 που είναι συμβατές. Ιστορικά η Python 3.0 είναι η πρώτη γλώσσα προγραμματισμού που «σπάει» την συμβατότητά με προηγούμενες εκδόσεις , με στόχο τη βελτίωση κάποιον παλιών λαθών και να εξελιχθεί ακόμη περισσότερο σχετικά με την απλότητα και την ευχρηστία της. Η Python 3.0 συνοδεύεται με πρόγραμμα 2to3.py, επιτρέποντας την μετατροπή κώδικα της έκδοσης 2.0 σε 3.0.

Από τις πρόσφατες εκδόσεις είναι η Python 3.9.2 που δημοσιεύτηκε το Φεβρουάριο του 2021, ακολουθεί η σταθερή έκδοση είναι η Python 3.10.4.

Η πιο πρόσφατα και σημαντική έκδοση της γλώσσας προγραμματισμού αυτής είναι η έκτη έκδοση συντήρησης της Python η έκδοση 3.10.6 που κυκλοφόρησε τον Αύγουστο του 2022 και περιέχει πολλές δυνατότητες και βελτιστοποιήσεις.

Αυτή η γλώσσα προγραμματισμού μόλις έκλεισε τα 30 χρόνια περίπου και έχει ακόμα δρόμο για να «φέρει» τον τίτλο της πιο δημοφιλούς γλώσσας κωδικοποίησης στον κόσμο. Πρόσφατα στο pycon22 (σύσκεψη python) κυκλοφόρησε μια νέα δυνατότητα από το ίδρυμα Anaconda, η οποία είναι γνωστή και ως pyscript, όπου μπορεί να γραφτεί και να εκτελεστεί στο πρόγραμμα περιήγησης όπως η JavaScript, κάτι το οποίο παλιά ήταν αδύνατο.

Η Python αναμφισβήτητα αποτελεί μια από τις δημοφιλείς γλώσσες προγραμματισμού και λόγω της κομψότητας, της απλότητας και της χρηστικότητάς της, την χρησιμοποιούν κορυφαίοι τεχνολογικοί οργανισμοί όπως οι Dropbox, Google, Mozilla, Quora, Hewlett-Packard, Qualcomm, IBM και Cisco. Αποτελεί έμπνευση για πολλές άλλες γλώσσες κωδικοποίησης όπως Rudy, Cobra, Boo, Groovy, Swift Go, Julia κ.λπ.

Είναι εφικτό να γράψουμε Python σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης, όπως το Thonny, το Pycharm, το NetBeans ή το Eclipse που είναι κατάλληλα για τη διαχείριση πολύ μεγάλων συλλογών αρχείων Python.

3.6 Ξεκινώντας Python

Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε την πιο πρόσφατη κύρια έκδοση της Python 3, όπου θα γραφτεί σε ένα πρόγραμμα επεξεργασίας κειμένου μέσω του λογισμικού Jupyter.

Προσοχή!

Στο κεφάλαιο αυτό θα αναφερθούμε γράφοντας τον κώδικα σε κείμενο χρησιμοποιώντας την Python αλλά δια μέσω του ελεύθερου λογισμικού Jupyter Notebook, αφού δεν μπορούμε να «τρέξουμε» αλλιώς κβαντικά προγράμματα.

Δηλαδή, γράφουμε κείμενο στο Jupyter Notebook και όχι στο περιβάλλον της Python.

Το Jupyter Notebook είναι μία πρωτότυπη διαδικτυακή εφαρμογή η οποία επιτρέπει την επεξεργασία και την εκτέλεση αρχείων Notebook μέσω ενός web browser και τη δημιουργία και την κοινή χρήση υπολογιστικών εγγράφων. Τα Notebook είναι αρχεία που δημιουργούνται από την εφαρμογή Jupyter Notebook, τα οποία περιλαμβάνουν κώδικα (π.χ. Python, Julia). Αποτελεί ένα ελεύθερο λογισμικό, ανοιχτού πρότυπου, διαδραστικού περιβάλλοντος ανάπτυξης υπολογιστών, σε όλες τις γλώσσες προγραμματισμού, με υπηρεσίες που βασίζεται στο web για σημειωματάρια, κώδικα και δεδομένα, προσφέροντας μια απλή, εξελιγμένη εμπειρία με επίκεντρο τα έγγραφα.

3.7 Εγκατάσταση Python και Jupyter Notebook

Η Python αναπτύσσεται ως «ανοικτού λογισμικού» (open source) καθώς η διαχείριση και ο κώδικας πραγματοποιείται και διανέμεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, δίνοντας τη δυνατότητα στην Python την εκτέλεση κώδικα σε μεγάλη ποικιλία συστημάτων. Ακόμα, χρησιμοποιώντας κάποια εργαλεία τρίτων, ο κώδικας της Python είναι δυνατόν να "πακεταριστεί" σε αυτόνομα εκτελέσιμα προγράμματα σε κάποια από τα γνωστά λειτουργικά συστήματα, επιτρέποντας τη χρήση αυτού του λογισμικού σε αυτά τα περιβάλλοντα χωρίς να απαιτείται η εγκατάσταση του διερμηνευτή της Python.

Αρχικά πρέπει να εγκαταστήσουμε την Python στον υπολογιστή μας, που διατίθενται ελεύθερα για «κατέβασμα», εκτελώντας download την εντελώς δωρεάν από τον ακόλουθο ιστότοπο: https://www.python.org ⟶

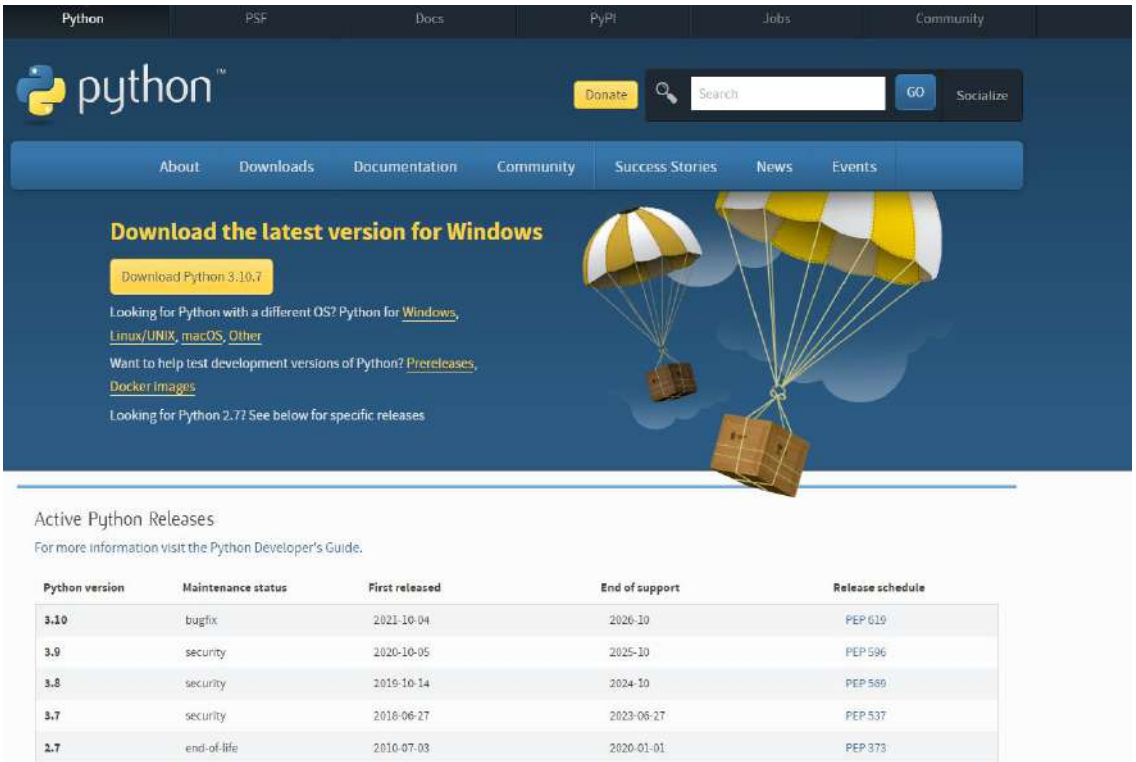
https://www.python.org/downloads/windows/.

Για Microsoft Windows υπάρχουν εκδόσεις των 32 ή 64 bits. Επισκεπτόμαστε την επίσημη σελίδα της Python που προαναφέραμε όπου βλέπουμε την τελευταία έκδοση της, την οποία συνήθως και εγκαθιστούμε.

Εγκαθιστούμε την Python 3.10.1 για Microsoft Windows (64 bits), από το επίσημο site της Python, ακολουθώντας τα παρακάτω βήματα:

Επισκεπτόμαστε το επίσημο site της Python:

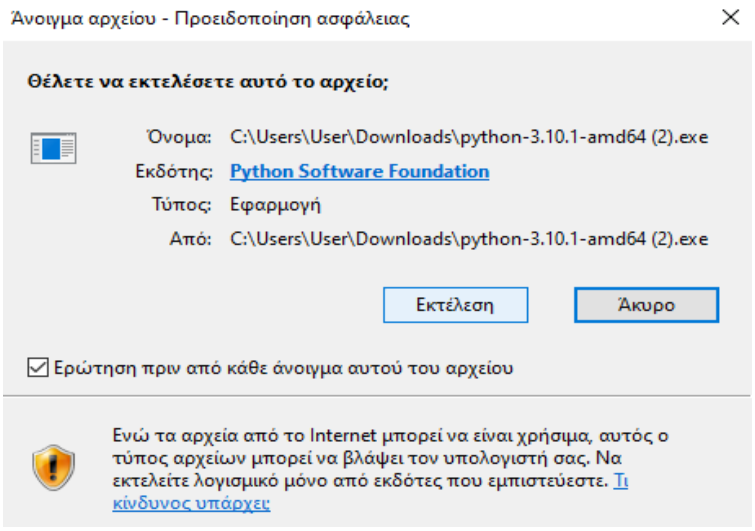
https://www.python.org/downloads/windows/

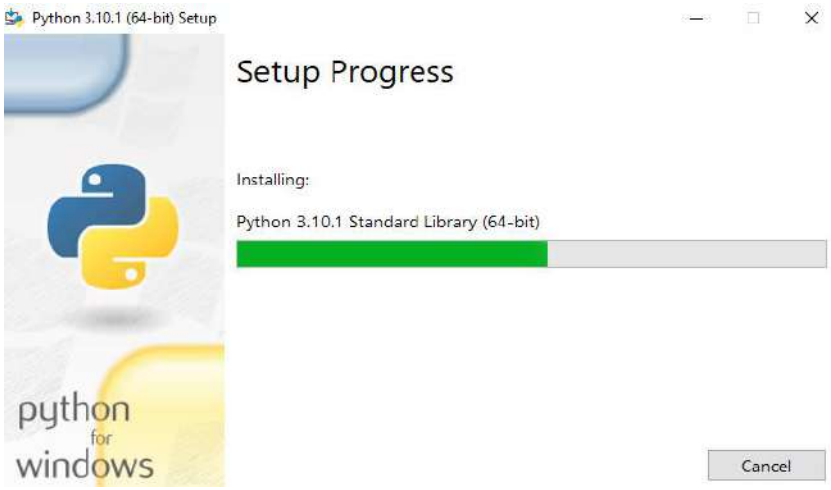


Επιλέγουμε το αρχείο για download (Download Windows installer 64 – bit) και το εκτελούμε:

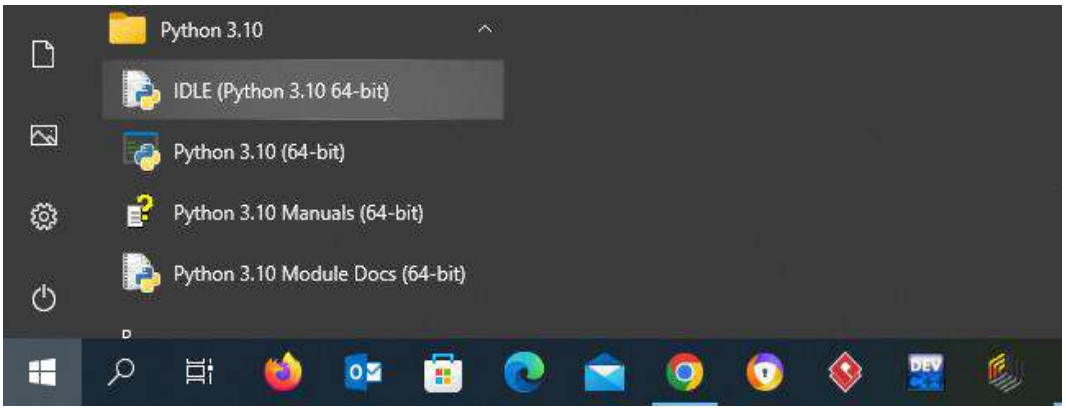
Κάνουμε κλικ στο κουτάκι "Add Python 3.10 to PATH και στη συνέχεια κλικ πάνω στο Install Now:

Πραγματοποιείται η εγκατάσταση της Python 3.10.1:





Από το μενού μπορούμε να δούμε την εγκαταστημένη Python 3.10.1:



Η επιλογή ενός επεξεργαστή είναι πάρα πολύ σημαντική, ώστε να μας μας βοηθήσει να γράφουμε κώδικα Python με έναν εύκολο και ασφαλή τρόπο.

Επίσης, είναι σημαντικό να τονίσουμε ότι μία από τις βασικές απαιτήσεις σύνταξης είναι η χρωματική επισήμανση σύνταξης, δηλαδή τα διαφορετικά τμήματα του προγράμματος Python χρωματίζονται κατάλληλα για να μπορούμε να παρακολουθούμε το πρόγραμμα ευκολότερα και έτσι να έχουμε καλύτερη εικόνα του. Η επισήμανση μας βοηθάει να εντοπίσουμε τυχόν συντακτικά λάθη που ίσως κάναμε στον κώδικα (δεν χρωματίζεται σωστά μια εντολή ή μια λέξη σε περίπτωση λάθους).

Ακόμα, το IDLE (το IDE - Integrated Development Environment- Ολοκληρωμένο Περιβάλλον Ανάπτυξης), όπου εγκαθίσταται εξ ορισμού από το πρόγραμμα της εγκατάστασης της Python σε Windows και πραγματοποιεί συντακτική επισήμανση του κώδικα καθώς και μας επιτρέπει να «τρέχουμε» τα προγράμματά μας μέσα από το IDLE ανάμεσα σε άλλα πράγματα. Καλό είναι να μην χρησιμοποιούμε το Notepad, γιατί δεν αποτελεί καλή επιλογή, αφού δεν πραγματοποιεί συντακτική επισήμανση καθώς και δεν υποστηρίζει τη στοίχιση κειμένου που είναι απαραίτητη για τη συγγραφή κώδικα σε Python.

Στη συνέχεια πραγματοποιούμε την εγκατάσταση του Jupyter Notebook (Anaconda3) στον υπολογιστή μας, κατεβάζοντάς την εντελώς δωρεάν από τον ακόλουθο ιστότοπο: https://ciksiti.com/el/chapters/3137-install-and-configure-jupyter-notebook-on-centos-8--linux .

3.8 Εισαγωγή Προγραμματισμός – Δομή Προγράμματος

Μια γλώσσα προγραμματισμού δίνει τη δυνατότητα στον προγραμματιστή γενικά να εκφράσει τις σκέψεις του, με όσο το δυνατό καλύτερο τρόπο, αξιοποιώντας το υλικό του υπολογιστή.
Προγραμματισμός είναι η διαδικασία εκτέλεσης ενός συγκεκριμένου υπολογισμού, που χρησιμοποιούμε συνήθως με το σχεδιασμό και τη δημιουργία ενός εκτελέσιμου προγράμματος στον υπολογιστή και περιλαμβάνει εργασίες όπως ανάλυση και εφαρμογή αλγορίθμων, επιλέγοντας μια γλώσσα προγραμματισμού.

Κάθε πρόγραμμα έχει την εξής δομή:

- Αποθηκεύει δεδομένα
- Τα επεξεργάζεται
- Μας δίνει το αποτέλεσμα

Όλες οι πληροφορίες που χρησιμοποιούν τα προγράμματα αποθηκεύονται στην μνήμη του υπολογιστή. Μνήμη είναι ο χώρος στον οποίο αποθηκεύονται όλα τα δεδομένα που χρησιμοποιούνε τα προγράμματα τα οποία τρέχουν στον υπολογιστή.

Συνεπώς, όταν «τρέχουμε» ένα πρόγραμμα στον υπολογιστή μας, στην ουσία τα προγράμματα δεσμεύει ένα μέρος της μνήμης, έχοντας το δικαίωμα να γράφουμε τα κατάλληλα δεδομένα που χρειαζόμαστε για την επεξεργασία του προγράμματος.

Ο χρήστης μπορεί να δώσει εντολή στην Python να βρει και να δεσμεύσει ένα χώρο στη μνήμη (όπου θα αποθηκεύει μια συμβολοσειρά, έναν αριθμό κλπ.) δίνοντας μια τιμή και ονομάζοντας αυτό το χώρο με ένα συγκεκριμένο όνομα που θα δίνει ο χρήστης.

3.9 Σύνταξη – Αριθμητική και Πράξεις με Python

3.9.1 Εκκίνηση – Αποθήκευση - Εκτέλεση σε Python

Η Python είναι μια ερμηνευμένη γλώσσα προγραμματισμού όπως προαναφέραμε και ο χρήστης γράφει τα αρχεία Python, αποθηκεύοντάς τα με όνομα αρχείου και κατάληψη (.py), σε ένα πρόγραμμα επεξεργασίας κειμένου και στη συνέχεια τα αρχεία αυτά για να εκτελεστούν τοποθετούνται στο διερμηνέα python. Ο διερμηνέας

πραγματοποιεί τη μετατροπή της εκτέλεσης του προγράμματος, ο οποίος ουσιαστικά δρα σαν ένα απλό κομπιουτεράκι.

ΠΑΡΑΔΕΙΓΜΑ 1

Ένας τρόπος εκτέλεσης ενός αρχείου Python, αποθηκεύοντάς το στο σκληρό δίσκο C, στο φάκελο με όνομα Users, σε ένα φάκελό του με όνομα Your Name και με όνομα αρχείου helloworld, βλέπουμε παρακάτω πώς συντάσσονται οι εντολές:

```
C:\Users\Your Name>python helloworld.py
```

Όπου helloworld.py, είναι το όνομα του αρχείου Python, το οποίο μπορεί να πραγματοποιηθεί σε οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου.

Δηλαδή για να συντάξουμε ένα αρχείο Python, δημιουργήσαμε ένα αρχείο Python στον διακομιστή, αποθηκεύοντας το αρχείο χρησιμοποιώντας την επέκταση αρχείου .py και εκτελώντας το στη Γραμμή εντολών:

```
C:\Users\Your Name>python myfile.py
```

Και για το παράδειγμά μας συγκεκριμένα:

```
C:\Users\Your Name>python helloworld.py
```

Και αυτό που πρέπει να εμφανίζεται, είναι:

```
Hello, World!
```

Ένας άλλος τρόπος σύνταξης της Python που μπορεί να εκτελεστεί είναι να γράφοντας απευθείας στη γραμμή εντολών:

```
>>> print(“Hello, World!”)
```

Και εμφανίζεται στην οθόνη μας πάλι :

```
Hello, World!
```

Και έτσι γράψαμε και εκτελέσαμε το πρώτο πρόγραμμα Python.

Αν χρησιμοποιήσουμε το IDLE, μπορούμε να επιλέξουμε από το μενού Run ⟶ Run Module ή να χρησιμοποιήσουμε τη συντόμευση πληκτρολογίου F5.

3.9.10 Έξοδος Εντολών

Για να «κλείσουμε» τη διεπαφή γραμμής των εντολών Python όταν τελειώνουμε κάθε φορά στη γραμμή εντολών της, γράφουμε:

```
exit()
```

3.9.11 Εσοχή

Η εσοχή στην Python αναφέρεται στον κενό χαρακτήρα στην αρχή μιας γραμμής εντολών ή αλλιώς του κώδικα, όπου είναι πολύ σημαντική σε αντίθεση με άλλες γλώσσες προγραμματισμού που η εσοχή στον κώδικα είναι μόνο για αναγνωσιμότητα.

Η Python χρησιμοποιεί εσοχή για να προσδιορίσει την ομαδοποίηση των εντολών, να υποδείξει ένα μπλοκ κώδικα και μπορεί να δώσει σφάλμα εάν παραλείψουμε την εσοχή.

ΠΑΡΑΔΕΙΓΜΑ 2

Η Python δεν θα δώσει συντακτικό λάθος αφού δεν παραλείψαμε την εσοχή, όπως:

```
if  5 > 2 :
        print (“Bravo!”)
```

Ενώ η Python θα δώσει συντακτικό λάθος εάν παραλείψουμε την εσοχή, όπως:

```
if  5 > 2 :
print (“Bravo!”)
```

Ο αριθμός των διαστημάτων εξαρτάται από το χρήστη ως προγραμματιστή, αλλά πρέπει να είναι τουλάχιστον ένα. Η πιο συνηθισμένη χρήση είναι τέσσερα κενά διαστήματα.
Δηλαδή,

```
if  5 > 2 :
  print (“Bravo!”)
if  5 > 2 :
        print (“Bravo!”)
```

Προσοχή

Πρέπει να χρησιμοποιήσουμε τον ίδιο αριθμό κενών στο ίδιο μπλοκ, αλλιώς η Python θα δώσει σφάλμα.

Δηλαδή είναι συντακτικό λάθος:

```
if  5 > 2 :
  print (“Bravo!”)
        print (“Bravo!”)
```

3.9.12 Σχόλια

Η Python μας δίνει την δυνατότητα να γράψουμε σχόλια, κείμενο μίας ή περισσοτέρων γραμμών, με σκοπό να τεκμηριώσουμε ή να εξηγήσουμε τον κώδικα Python και συνεπώς ο κώδικας να γίνει πιο ευανάγνωστος. Στα σχόλια είναι εφικτό να χρησιμοποιηθούν σημειώσεις και εντολές για τον αναγνώστη του προγράμματος για να μπορεί να κατανοήσει τη λειτουργία του προγράμματος ή ακόμα και για να αποτραπεί η εκτέλεση κατά τη δοκιμή ενός προγράμματος.
Τα σχόλια ξεκινούν με ένα σύμβολο # γράφοντας πάντα από δεξιά αυτού του συμβόλου και η Python δεν λαμβάνει υπόψιν τα σχόλια, δηλαδή θα αγνοήσει την γραμμή ή τις γραμμές που ακολουθούν μετά, αποδίδοντας την υπόλοιπη γραμμή ως σχόλιο.
Μπορούμε να ξεκινήσουμε τα σχόλια αντί για # γράφοντας τριπλά εισαγωγικά “ “ “ τα σχόλια που θέλουμε μιας ή περισσοτέρων γραμμών και κλείνοντας πάλι με τριπλά εισαγωγικά.

ΠΑΡΑΔΕΙΓΜΑ 3

Αν γράψουμε:

```
#Αυτό είναι ένα σχόλιο
#Γραμμένο σε
print(“Hello, World!”)
```

Όταν το εκτελέσουμε, θα εμφανιστεί:

```
Hello, World!
```

Ή αλλιώς, χρησιμοποιώντας “ “ “

```
“ ” ”
Αυτό είναι ένα σχόλιο
γραμμένο σε
“ ” ”
print(“Hello, World!”)
```

Όταν το εκτελέσουμε, θα εμφανιστεί:

```
Hello, World!
```

Είναι εφικτό ένα σχόλιο να τοποθετηθεί και στο τέλος μιας γραμμής εντολών, αγνοώντας η Python όλη τη γραμμή κατά την εκτέλεσή της.

ΠΑΡΑΔΕΙΓΜΑ 4

Αν γράψουμε:

```
print(“Hello, World!”) #Αυτό είναι ένα σχόλιο
```

Όταν το εκτελέσουμε, θα εμφανιστεί πάλι:

```
Hello, World!
```

3.9.13 Αναζήτηση Βοήθειας – help ( )

Αν χρειαστεί μπορούμε γρήγορα να αναζητήσουμε σχετικές πληροφορίες, χρησιμοποιώντας την ενσωματωμένη συνάρτηση της Python, τη help( ), που είναι ιδιαίτερα χρήσιμη, ειδικά αν χρησιμοποιούμε την κονσόλα του διερμηνευτή. Με το πλήκτρο q «κλείνουμε» τη βοήθεια.

ΠΑΡΑΔΕΙΓΜΑ 5

Αν γράψουμε:

```
help(print)
```

Όταν το εκτελέσουμε, θα δώσει πληροφορίες για τη συνάρτηση print που χρησιμοποιείται για την εκτύπωση στην οθόνη.

3.9.14 Κυριολεκτικές Σταθερές (Literal Constants)

Κυριολεκτική σταθερά ονομάζεται έτσι γιατί κυριολεκτεί, δηλαδή η τιμή της ορίζεται κυριολεκτικά. Κυριολεκτική σταθερά είναι ένας αριθμό όπως:
3, 1.29, 10.45e-4 ή μια συμβολοσειρά.

Έτσι για παράδειγμα ο αριθμός 3 αναπαριστά τον εαυτό του, η τιμή του δεν αλλάζει και αποτελεί μία σταθερά.

3.10 Συμβολοσειρές (Strings)

Εκτός από τους αριθμούς, η Python μπορεί χειριστεί και αλφαριθμητικά, τα οποία εκφράζονται με διαφορετικούς τρόπους.

• Μονό ή διπλά εισαγωγικά

Οι συμβολοσειρές στην Python περιλαμβάνονται είτε από μεμονωμένα εισαγωγικά είτε από διπλά εισαγωγικά . Μια μεταβλητή μπορεί να είναι αριθμός αλλά και αλφαριθμητικό. Όταν θέτουμε μια μεταβλητή το αλφαριθμητικό το βάζουμε μέσα σε εισαγωγικά μονά ή διπλά (“ “, ‘ ‘). Έτσι οι μεταβλητές συμβολοσειράς μπορούν να δηλωθούν χρησιμοποιώντας μονά ή διπλά εισαγωγικά.

Δηλαδή, το ‘γεια’ είναι ίδιο με το “γεια”.

• Οι μεταβλητές δεν χρειάζεται να δηλωθούν με κάποιο συγκεκριμένο τύπο και αν χρειαστεί μπορούν ακόμη και να αλλάξουν τύπο αφού έχουν ορισθεί.
Επίσης, μπορούμε να εμφανίσουμε μια συμβολοσειρά με τη συνάρτηση print(). Η συνάρτηση print( ) παράγει μια πιο ευανάγνωστη έξοδο, παραλείποντας τα εισαγωγικά και εκτυπώνοντας ειδικούς χαρακτήρες.

ΠΑΡΑΔΕΙΓΜΑ 6

Έχουμε:

```
print(“Hello”) #Με διπλά εισαγωγικά
```

```
print(‘Hell’) #Με μονά εισαγωγικά
```

ΠΑΡΑΔΕΙΓΜΑ 7

Έχουμε:

```
x=”Hello”
#Είναι το ίδιο  με
x=’Hello’
print(x)
```

• Καθορισμός τύπων

Σε μια μεταβλητή μπορούμε να καθορίσουμε και τον τύπο δεδομένων μιας μεταβλητής.

Έτσι, μπορούμε να αποθηκεύσουμε στην μνήμη του υπολογιστή αριθμούς και αντικείμενα διαφόρων ειδών.

Οι πιο συνηθισμένοι τύποι δεδομένων της Python είναι οι παρακάτω:

- **Int – Ακέραιος (π.χ. 4, 20, 100).**
- **Float – Πραγματικός, κινητής υποδιαστολής (π.χ. 1.6, 5.0, 12.01).**
- **Bool - Λογική μεταβλητή, η οποία παίρνει τις τιμές False ή True (εκφράσεις).**
- **NoneType – παίρνει την τιμή None, (δηλώνει απουσία τιμής).**
- **Complex - Μιγαδικοί (π.χ. 5 + 3j, 9j).**

ΠΑΡΑΔΕΙΓΜΑ 8

Έχουμε:

```
number=2
not_number=’3’
print(number+4)
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
6
```

Προσοχή

Αν γράψουμε:

```
print(not_number+4)
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

Θα εμφανίσει Error, θα μας δώσει σφάλμα για το print(not_number + 5), αφού δεν γίνεται να προσθέσουμε έναν ακέραιο (int) αριθμό με αλφαριθμητικό (string).

ΠΑΡΑΔΕΙΓΜΑ 9

Έχουμε τρεις μεταβλητές στην Python x= 4, y=5 και z=7.
Αν γράψουμε:

```
x=str(4) #Το x θα είναι 4
y=int(5) Το y θα είναι 5
z=float(7) Το z θα είναι 7.0
w=2.14+j5.2 Το w θα είναι 2.14+j*5.2
j=1
print(x)
print(y)
print(z)
print(w)
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
4
5
7.0
7.34
```

ΠΑΡΑΔΕΙΓΜΑ 10

```
x=4
x=”Hello!” #Το x άλλαξε και είναι τύπου str
print(x)
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Hello!
```

• Συνάρτηση type – Απόκτηση τύπο

Για να μας «δώσει» τον τύπο δεδομένων μιας μεταβλητής, τότε χρησιμοποιούμε τη **συνάρτηση type( ).**

ΠΑΡΑΔΕΙΓΜΑ 11

```
x=4
y=”Hello!”
print(type(x))
print(type(y))
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
<class ‘int’ >
<class ‘str’>
```

• Διάκριση πεζών και κεφαλαίων

Τα ονόματα των μεταβλητών έχουν διάκριση πεζών κεφαλαίων.

ΠΑΡΑΔΕΙΓΜΑ 12

```
a=4
A=”Hello!” #Το α δεν θα αντικατασταθεί από το Α
print(a)
print(A)
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
4
Hello!
```

3.11 Αριθμοί

Όταν θέλουμε να καθορίσουμε έναν τύπο σε μια μεταβλητή, η Python παρόλο που είναι μια αντικειμενοστραφείς γλώσσα προγραμματισμού, χρησιμοποιεί κλάσεις για να ορίσει τύπους δεδομένων, συμπεριλαμβανομένων των πρωτογενών της τύπων.

Οι αριθμητικοί τύποι όπως προαναφέραμε στην Python είναι τρεις:

- ✓ int
- ✓ float
- ✓ complex

Επίσης:

- **int**: κατασκευάζει γενικά έναν ακέραιο αριθμό από μια σταθερή τιμή του κώδικα, ή από έναν float δεκαδικό αριθμό (αφαιρώντας όλα τα δεκαδικά), ή από ένα string (με την προϋπόθεση ότι η συμβολοσειρά αντιπροσωπεύει έναν ακέραιο αριθμό).
- **float**: κατασκευάζει γενικά έναν δεκαδικό αριθμό από έναν ακέραιο αριθμό, μια σταθερή τιμή του κώδικα, ή από ένα string (με την προϋπόθεση ότι η συμβολοσειρά αντιπροσωπεύει έναν δεκαδικό αριθμό).
- **str**: κατασκευάζει μια συμβολοσειρά από μια ποικιλία τύπων δεδομένων, συμπεριλαμβανομένων συμβολοσειρών, ακέραιων κυριολεκτικών και κυλιόμενων αριθμών.

Ας δούμε ένα παράδειγμα με τους τρεις αριθμητικούς τύπους.

```
x = 4      # int
y = 1.8    # float
z = 2j     #complex
```

• Ο αριθμητικός τύπος Int

Το Int ή ακέραιος, είναι ένας ακέραιος αριθμός, θετικός ή αρνητικός. Χωρίς δεκαδικά, απεριόριστου μήκους.

ΠΑΡΑΔΕΙΓΜΑ 1

Ακέραιοι αριθμοί:

```
x=2
y=424347512551454
z=-1352455
print(type(x))
print(type(y))
print(type(z))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
<class 'int'>
<class 'int'>
<class 'int'>
```

• Ο αριθμητικός τύπος float

Το float ή «αριθμός κινητής υποδιαστολής» είναι ένας αριθμός, θετικός ή αρνητικός που περιέχει ένα ή περισσότερα δεκαδικά ψηφία. Επίσης, μπορεί να είναι αριθμοί με “e” για να υποδηλώνει τη δύναμη του 10.

ΠΑΡΑΔΕΙΓΜΑ 2

Floats αριθμοί:

```
x=2.10
y=2.0
z=-22.36
k=21e4
print(type(x))
print(type(y))
print(type(z))
print(type(k))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
<class ‘float’>
<class ‘float’>
<class ‘float’>
<class ‘float’>
```

• Ο αριθμητικός τύπος complex

Η Python έχει ενσωματωμένη υποστήριξη για μιγαδικούς αριθμούς και ως κατάληξη για το «φανταστικό μέρος» χρησιμοποιεί το j ή το J πίσω από τον αριθμό.

ΠΑΡΑΔΕΙΓΜΑ 3

Οι μιγαδικοί αριθμοί γράφονται:

```
x=1+4j
y=2j
z=-3j
print(type(x))
print(type(y))
print(type(z))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
<class ‘complex’>
<class ‘complex’>
<class ‘complex’>
```

3.12 Μετατροπή Τύπου

Υπάρχουν μέθοδοι int(), float() και complex(), που μπορούμε να εφαρμόσουμε για να μετατρέψουμε έναν τύπο σε κάποιον άλλο.

Προσοχή

Δεν μπορούμε να μετατρέψουμε τους μιγαδικούς αριθμούς σε άλλο τύπο.
Ας δούμε κάποια παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε:

```
x=2   #int
y=3.6 #float
z=2j  #complex

#Μετατροπή από ακέραιο αριθμό σε δεκαδικό:
a=float(x)

#Μετατροπή από δεκαδικό αριθμό σε ακέραιο:
b=int(y)

#Μετατροπή από ακέραιο αριθμό σε μιγαδικό:
c=complex(x)
print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
2.0
3
2+0j

<class ‘float’>
<class ‘int’>
<class ‘complex’>
```

3.13 Τυχαίοι Αριθμοί (Random Number)

Η Python έχει μια ενσωματωμένη και πολύ χρήσιμη βιβλιοθήκη για τη δημιουργία τυχαίων αριθμών που είναι η random.

Οι δυνατότητες που παρέχει η βιβλιοθήκη είναι οι παρακάτω:

- **random ( )** - παράγει ένα τυχαίο πραγματικό αριθμό στο διάστημα [0.0, 1.0].
- **randint(a, b)** - παράγει έναν τυχαίο ακέραιο αριθμό στο διάστημα [a,b].
- **choise(x)** - επιστρέφει έναν τυχαίο ακέραιο αριθμό μιας λίστας τιμών x.
- **sample(x, k)** - επιστρέφει μια λίστα k τυχαίων στοιχείων μιας λίστας τιμών x.

ΠΑΡΑΔΕΙΓΜΑ 2

Εμφανίστε έναν τυχαίο αριθμό μεταξύ 1 και 9, εισάγοντας την τυχαία μονάδα.

Έχουμε:

```
import random
print(random.randint(1,9))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
8
```

Για κάθε φορά που «τρέχουμε» αυτό το πρόγραμμα, θα εμφανίζεται στην οθόνη ένας αριθμός από το 1 έως και το 9.

ΠΑΡΑΔΕΙΓΜΑ 3

Να εκτυπώσετε τους τύπους δεδομένων των παρακάτω μεταβλητών:

```
print(type(x))
print(type(y))
print(type(z))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
<class ‘int’>
<class ‘float>
<class ‘complex’>
```

3.14 Μεταβλητές (Variables)

Να τονίσουμε ότι η Python δεν έχει εντολή για δήλωση μεταβλητής. Στην Python οι μεταβλητές όπως χρησιμοποιούνται για την αποθήκευση μιας τιμής δεδομένων, ουσιαστικά είναι μια θέση μνήμης, απλά τμήματα της μνήμης του Η/Υ, που τη δεσμεύουμε για να «κρατήσουμε», να αποθηκεύσουμε μια πληροφορία. Μια μεταβλητή δημιουργείται τη στιγμή που της εκχωρείται για πρώτη φορά μια τιμή, δηλαδή δηλώνονται αυτόματα στην Python, μόλις τους αποδοθεί κάποια τιμή (x= 4, y=”Hello”).

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε:

```
message = (“Hello World”)
print(message)
```

Δίνουμε εντολή στην Python να δεσμεύσει ένα χώρο στη μνήμη με όνομα message, αποθηκεύοντας σε αυτή για τιμή με όνομα “Hello World”. Ανάμεσα από το όνομα αποθήκευσης message και την τιμή αποθήκευσης υπάρχει το σύμβολο της ισότητας (=) που χρησιμοποιείται για να εκχωρηθεί μια τιμή σε μία μεταβλητή και ονομάζεται τελεστής εκχώρησης ή τελεστής καταχώρησης ή αλλιώς τελεστή ανάθεσης. Αν χρησιμοποιήσουμε μια μεταβλητή που δεν ορίζεται, τότε θα μας εμφανίσει σφάλμα (error).
Ουσιαστικά, αυτός ο τελεστής πρέπει να έχει στα αριστερά του μια μεταβλητή (το όνομα που δίνουμε για τη δέσμευση χώρου μνήμης) και στα δεξιά του μια τιμή (δηλαδή αυτό που θέλουμε να αποθηκεύσουμε).
Συνεπώς, όταν θα καλούμε αυτή την μεταβλητή (message) θα αναζητούμε το χώρο μνήμης που τον ονομάσαμε με το όνομα της μεταβλητής (message) και αποθήκευέ μου αυτήν την τιμή που σου δίνω εντολή “Hello World”. Η Python πηγαίνει στο χώρο μνήμης και γράφει “Hello World”.

Στη συνέχεια του προγράμματος μπορούμε να επικαλεστούμε αυτό το χώρο μνήμης (message) να μας φέρει το περιεχόμενο που βρίσκεται μέσα σε αυτό το χώρο μνήμης που ονομάσαμε message και είναι το “Hello World”.
Και με την εντολή print(message) ζητάμε να μας τυπώσει το περιεχόμενο που βρίσκεται στο χώρο μνήμης που το ονομάσαμε message, δηλαδή θα εμφανιστεί στην οθόνη μας Hello World.

ΠΑΡΑΔΕΙΓΜΑ 2

Έχουμε δύο μεταβλητές στην Python x= 4 και y=”Hello, World!”.

Αν γράψουμε:

```
x=4
y=”Hello, World!”
```

```
print(x)   #Θα μας τυπώσει την τιμή της μεταβλητής x.
print(y)   #Θα μας τυπώσει την τιμή της μεταβλητής y.
```

Τότε όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
4
Hello, World!
```

3.15 Ονόματα Μεταβλητών

Οι μεταβλητές στην Python μπορεί να είναι ονόματα σε συνδυασμό Αγγλικών χαρακτήρων και αριθμών, αλλά δεν γίνεται να ξεκινάνε με αριθμό. Ακόμα, η Python τους κεφαλαίους με τους πεζούς χαρακτήρες δεν τους αντιλαμβάνεται το ίδιο.

Μια μεταβλητή είναι δυνατό να έχει ένα σύντομο όνομα, όπως x και y ή ένα αντιπροσωπευτικό όνομα πιο περιγραφικό όπως για παράδειγμα ηλικία, carname, total, volume.

Ας δούμε κάποιους κανόνες για μεταβλητές Python

• Ένα όνομα μεταβλητής πρέπει να ξεκινά με ένα γράμμα (κεφαλαίο ή πεζό ASC‖) ή τον χαρακτήρα υπογράμμισης (κάτω παύλα _).

• Ένα όνομα μεταβλητής είναι αδύνατο να ξεκινά με αριθμό.

• Ένα όνομα μεταβλητής μπορεί να περιέχει μόνο αλφαριθμητικούς χαρακτήρες και κάτω παύλες όπως Az, 0-9, και _ .

• Τα ονόματα μεταβλητών έχουν διάκριση πεζών – κεφαλαίων, όπως για παράδειγμα η ηλικία, η ηλικία και η ηλικία είναι τρεις διαφορετικές μεταβλητές.

ΠΑΡΑΔΕΙΓΜΑ 1

Το όνομα John, μπορούμε σύμφωνα με τους κανόνες της Python να το δηλώσουμε σωστά:

myvar = ” John”

my_var = ” John”

_my_var = ” John”

myVar = ” John”

MYVAR = ” John”

Myvar2 = ” John”

ΠΑΡΑΔΕΙΓΜΑ 2

Ας δούμε κάποια παραδείγματα δήλωσης ονομάτων, μη σωστών κανόνων της Python:

2myvar = ” John”
my-var = ” John”
my var = ” John”

• Ονόματα μεταβλητών πολλών λέξεων

Τα ονόματα των μεταβλητών με περισσότερες από μία λέξεις μπορεί να είναι δύσκολο να διαβαστούν.

Ωστόσο, υπάρχουν τεχνικές που μπορούμε να χρησιμοποιήσουμε ώστε να τις κάνουμε πιο ευανάγνωστες.

Όταν έχουμε περισσότερα ονόματα μεταβλητών από ένα, τότε:

Κάθε λέξη ξεκινάει με κεφάλαιο γράμμα, εκτός από την πρώτη λέξη

Κάθε λέξη αρχίζει από κεφάλαιο γράμμα.

Κάθε λέξη χωρίζεται με έναν ειδικό χαρακτήρα υπογράμμισης.

ΠΑΡΑΔΕΙΓΜΑ 3

myVariableName = ” John”

MyVariableName = ” John”

my_variable_name = ” John”

3.16 Εκχώρηση πολλών Μεταβλητών (Variables)

Η Python μας επιτρέπει να εκχωρούμε τιμές σε πολλές μεταβλητές σε μια γραμμή.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε τρεις μεταβλητές x= Orange, y=Banana, z=Cherry.

Τότε είναι:

```
x=”Orange”, y=”Banana”, z=”Cherry”
print(x)
```

```
print(y)
print(z)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Orange
Banana
Cherry
```

Προσοχή

Αν ο αριθμός των μεταβλητών δεν ταιριάζει με τον αριθμό των τιμών, τότε θα εμφανιστεί σφάλμα.

• Μια τιμή σε πολλές μεταβλητές

Μια τιμή μπορούμε να την αντιστοιχίσουμε σε πολλές μεταβλητές σε μία γραμμή.

ΠΑΡΑΔΕΙΓΜΑ 2

Έχουμε τρεις μεταβλητές x, y, z.

Τότε μπορούμε:

```
x=y=z=”Orange”
print(x)
print(y)
print(z)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Orange
Orange
Orange
```

• Unpack σε μια συλλογή

Σε περίπτωση που έχουμε μια συλλογή τιμών σε μια λίστα ή αλλιώς πλειάδα κ.λπ. η Python μας επιτρέπει να εξάγουμε τιμές σε μεταβλητές και γράφουμε unpacking.

ΠΑΡΑΔΕΙΓΜΑ 3

Έχουμε τρεις επιλογές φρούτων.

```
fruits=[”apple”, ”banana”, ”cherry”]
x,y,z=fruits
print(x)
print(y)
print(z)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
apple
banana
cherry
```

3.17 Βιβλιοθήκες Math – Random

Η Βιβλιοθήκη math αποτελεί μια πολύ χρήσιμη βιβλιοθήκη της Python γιατί υποστηρίζει μαθηματικά με συναρτήσεις, την εύρεση τετραγωνικής ρίζας και πολλές άλλες.

Οι βιβλιοθήκες της Python έτσι και η math για να είναι εφικτό να περιληφθούν σε ένα πρόγραμμα, πρέπει να ακολουθήσουν τις διαδικασίες:

• import math: πρέπει να γράφεται math για όποια συνάρτηση πρόκειται να χρησιμοποιηθεί.
Για παράδειγμα math.sqrt(25) = 5.

• from math import: Εισάγονται όλες οι συναρτήσεις που θα χρησιμοποιηθούν χωρίς να χρειάζεται να γράψουμε μπροστά το math.

Για παράδειγμα math.sqrt(4) = 2.

• from math import: συγκεκριμένες συναρτήσεις ή μία, οι οποίες θα χρησιμοποιηθούν χωρίς το πρόθεμα μπροστά math.

Για παράδειγμα from math import cos, sin, sqrt.

Προσοχή

Για τις τριγωνομετρικές συναρτήσεις το όρισμα δεν είναι οι μοίρες αλλά τα ακτίνια.

Για παράδειγμα το cos(pi/2) ισοδυναμεί με το cos (90).

Ένας τύπος για να μετατρέψουμε μοίρες θ σε ακτίνια α είναι:

$$\alpha = \frac{\pi \, x \, \theta}{180}$$

Επίσης, η βιβλιοθήκη random αποτελεί μια πολύ χρήσιμη βιβλιοθήκη της Python, για την παραγωγή τυχαίων αριθμών που προαναφέραμε.

3.17.1 Ας ξαναθυμηθούμε αυτές τις δυνατότητες

• **seed ([a])** : αρχικοποιεί την γεννήτρια τυχαίων αριθμών, χωρίς παράμετρο χρησιμοποιεί το ρολόι του συστήματος.

• **random ( )** : παράγει έναν τυχαίο αριθμό στο διάστημα [0.0, 1.0].

• **randint (a,b)** : παράγει έναν τυχαίο ακέραιο αριθμό στο διάστημα [a, b].

• **choise (x)** : επιστρέφει έναν τυχαίο στοιχείο μιας λίστας τιμών, έστω x.

• **sample (x, k)** : επιστρέφει μία λίστα k τυχαίων στοιχείων μιας λίστας τιμών x.

Όπως αναφέραμε παραπάνω, η δυνατότητα που μας δίνει η βιβλιοθήκη random είναι παραγωγή τυχαίων αριθμών οι οποίοι ακολουθούν μια συγκεκριμένη κατανομή και μια από αυτές που μας ενδιαφέρει είναι η:

- Τριγωνική κατανομή: random.triangular(low,high), για τυχαίες τιμές στο διάστημα low,high.

3.18 Τελεστές (Operands)

Οι τελεστές είναι λειτουργίες όπου απαιτούν κάποια δεδομένα για να λειτουργήσουν και μπορούν να αναπαρασταθούν με ειδικά σύμβολα όπως το + καθώς και με ειδικές λέξεις – κλειδιά. Οι τελεστές χρησιμοποιούνται για την εκτέλεση εργασιών και πράξεων σε μεταβλητές και τιμές.

Τα σύμβολα των τελεστών είναι:

- + (πρόσθεση)
- - (αφαίρεση)
- * (πολλαπλασιασμός)
- / (διαίρεση)
- ** (δύναμη)
- // (διαίρεση επιστρέφοντας τον ακέραιο αριθμό του αποτελέσματος της διαίρεσης)
- % (υπόλοιπο της διαίρεσης)

Η Python τους τελεστές τους κατηγοριοποιεί στις παρακάτω ομάδες:

- Αριθμητικοί τελεστές
- Χειριστές ανάθεσης
- Σύγκριση τελεστών
- Λογικοί τελεστές
- Τελεστές ταυτότητας
- Χειριστές μελών
- Χειριστές bitwise

• Αριθμητικοί Τελεστές

Οι αριθμητικοί τελεστές χρησιμοποιούνται για την εκτέλεση κοινών μαθηματικών πράξεων, με αριθμητικές τιμές.

Operator	Name	Example
+	Πρόσθεση	x + y
-	Αφαίρεση	x - y
*	Πολλαπλασιασμός	x * y
/	Διαίρεση	x / y
%	Modulus	x % y
**	Δύναμη-Exponentiation	x ** y
//	Διαίρεση (στρογγυλοποιημένη προς τα κάτω)Floor division	x // y

Ας δούμε αναλυτικά τους τελεστές με κάποια παραδείγματα.

Τελεστής	Όνομα	Εξήγηση	Παράδειγμα
+	Συν	Προσθέτει δύο αντικείμενα.	• 3 + 5 = 8 • Το ‘a’ + ‘b’ δίνει ’ab’
-	Μείον	Αφαιρεί έναν αριθμό από κάποιον άλλο, είτε δίνει έναν αρνητικό αριθμό.	• 50 – 15 = 35 • Το - 5.21 δίνει έναν αρνητικό αριθμό
*	Επί	Δίνει το γινόμενο δύο αριθμών ή μια συμβολοσειρά (string) επαναλαμβανόμενη τόσες φορές.	• 3 * 5 = 15 • Το ‘la’ * 3 δίνει ‘lalala’
**	Δύναμη	Επιστρέφει το x υψωμένο στη δύναμη y.	• Το 2 * * 3 δίνει 2 * 2 * 2 = 8

/	Διά	Διαιρείται το x με y.	• Το 8 / 6 δίνει 1.3333333
//	Διαίρεση στρογγυλοποιημένη προς τα κάτω	Επιστρέφει τον κοντινότερο (προς τα κάτω) ακέραιο στο πηλίκο.	• Το 4 // 3 δίνει 1
%	Υπόλοιπο	Επιστρέφει το υπόλοιπο της διαίρεσης.	•Το 8 % 3 δίνει 2 •Το – 25.5 % 2.25 δίνει 1.5
<<	Αριστερή μετάθεση	Μεταθέτει τα δυαδικά ψηφία (bits, 0 ή 1) του αριθμού προς τα αριστερά κατά το πλήθος των θέσεων που καθορίστηκε.	Το 2 << 2 δίνει τη θέση 8. Το 2 αναπαριστάται σε bits ως 10, όπου η μετάθεση προς τα αριστερά κατά 2 bits μας δίνει 1000 που είναι το δεκαδικό 8.
>>	Δεξιά Μετάθεση	Μεταθέτει τα δυαδικά ψηφία bits (0 ή 1) του αριθμού προς τα αριστερά κατά το πλήθος των θέσεων που καθορίστηκε.	Το 2 << 2 δίνει τη θέση 8. Το 2 αναπαριστάται σε bits ως 1011 που όταν μετατεθούν δεξιά κατά 1 bit μας δίνει 101 που είναι το δεκαδικό 5.
&	Δυαδικό ΚΑΙ	Δυαδικό ΚΑΙ των αριθμών.	Το 5 & 3 δίνει 1.
|	Δυαδικό Ή	Δυαδικό Ή των αριθμών.	Το 5 | 3 δίνει 7.
^	Δυαδικό αποκλειστικό Ή	Δυαδικό αποκλειστικό Ή των αριθμών.	Το 5 ^ 3 δίνει 6.
~	Δυαδική αντιστροφή	Δυαδικό αντίστροφο του x είναι –(x+1).	Το ~ 5 δίνει 6.
<	Μικρότερο από	Επιστρέφει το αν το x είναι μικρότερο από το y. Όλοι οι τελεστές σύγκρισης επιστρέφουν True (Αληθής) ή False (Ψευδής) & τα ονόματα ξεκινούνε με κεφαλαίο γράμμα.	Το 5 > 3 δίνει False και το 3 < 5 δίνει True. Μπορούμε να το γράψουμε και 3 < 5 < 7 δίνει True.
>	Μεγαλύτερο από	Επιστρέφει το αν το x είναι μεγαλύτερο από το y	Το 5 > 3 επιστρέφει True. Αν οι τελεστές είναι και οι δύο αριθμοί μετατρέπονται σε έναν κοινό τύπο,

			αλλιώς επιστρέφει πάντα False.
<=	Μικρότερο ή ίσο	Επιστρέφει το αν το x είναι μικρότερο από ή ίσο με το y	Το x=3; y=6; x<=y; επιστρέφει True.
>=	Μεγαλύτερο ή ίσο	Επιστρέφει το αν το x είναι μεγαλύτερο από ή ίσο με το y	Το x=4; y=3; x>=y; επιστρέφει True.
==	Ίσο	Συγκρίνει αν τα αντικείμενα είναι ίσα	Το x=2; y=2; x==y; επιστρέφει True. Το x='str'; y='stR'; x==y; επιστρέφει False. Το x='str'; y='str'; x==y; επιστρέφει True.
!=	Διαφορετικό	Συγκρίνει αν τα αντικείμενα ΔΕΝ είναι ίσα	Το x=2; y=3; x!=y; επιστρέφει True.
not	Λογικό ΌΧΙ	Αν το x είναι True, επιστρέφει False, αλλιώς αν είναι False επιστρέφει True.	x=True; not x επιστρέφει False
and	Λογικό ΚΑΙ	Το x and y επιστρέφει False αν το x είναι False, αλλιώς επιστρέφει την υπολογιζόμενη τιμή του y.	x=False; y-True; x and y επιστρέφει False αφού το x είναι False. Συγκεκριμένα, η Python αφού γνωρίζει ότι η αριστερή έκφραση της 'and' είναι False, δε θα ελέγξει την τιμή του y. Έτσι ανεξάρτητα από τις άλλες τιμές ολόκληρη η έκφραση θα είναι False.
or	Λογικό Ή	Αν το x είναι True, επιστρέφει True, αλλιώς επιστρέφει την τιμή του y.	Το x = True; y=False; x or y επιστρέφει True

Προσοχή

Η χρήση των παρενθέσεων επιτρέπονται και χρησιμοποιούνται για να καθορίσουν την προτεραιότητα εκτέλεσης των πράξεων που προηγούνται, ακολουθούν οι πράξεις του πολλαπλασιασμού και της διαίρεσης και τελευταίες οι πράξεις της πρόσθεσης και της αφαίρεσης.

Το πιο σπουδαίο με την Python είναι ότι οι όλοι αυτοί τελεστές μπορούν να εφαρμοστούν και σε μεταβλητές τύπου μιγαδικού αριθμού.

ΠΑΡΑΔΕΙΓΜΑ 1

Για την πρόσθεση δύο τιμών χρησιμοποιούμε τον τελεστή +.

```
print(10+4)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
14
```

ΠΑΡΑΔΕΙΓΜΑ 2

Αν για x = 4 – 2j και y = -1.3 + 3.6j, τότε να υπολογίσετε τις παρακάτω πράξεις:

Είναι,

- x + y = 2.7 + 1.6j
- x – y = 5.3 - 5.6j
- x * y = - 1.3 – 7.2j
- x / y = - 3.07692307692 – 0.5555555555j

ΠΑΡΑΔΕΙΓΜΑ 3

Δίνετε x = 8 και y = 2. Να υπολογίσετε και να εκτυπώσετε τα αποτελέσματά από τις παρακάτω πράξεις.

Έχουμε:

```
x=8
y=2
print(x+y)
print(x-y)
print(x*y)
print(x/y)
print(x%y) #Το υπόλοιπο της διαίρεσης 8 δια 2
print(x**y) #Η δύναμη 8^2, δηλαδή 8*8
print(x//y)     #Στρογγυλοποιεί το αποτέλεσμα της διαίρεσης
          στον πλησιέστερο ακέραιο αριθμό
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
10
6
16
4
64
4
```

• Χειριστές Ανάθεσης

Οι τελεστές εκχώρησης χρησιμοποιούνται για την εκχώρηση τιμών σε μεταβλητές.

Operator	Example
=	x = 3
+ =	x +=5 είναι το ίδιο με: x = x + 5
- =	x -= 5 είναι το ίδιο με: x = x - 5
* =	x /= 5 είναι το ίδιο με: x = x * 5
/ =	x /= 5 είναι το ίδιο με: x = x / 5
% =	x %= 5 είναι το ίδιο με: x = x % 5
// =	x //= 5 είναι το ίδιο με: x = x // 5
** =	x **= 5 είναι το ίδιο με: x = x ** 5
& =	x &= 5 είναι το ίδιο με: x = x & 5

Ας δούμε κάποια παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 4

Να υπολογίσετε για x = 7 και να εκτυπώσετε τα αποτελέσματά από τις παρακάτω πράξεις.
Έχουμε:

• Για x=7

```
x=7
print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
7
```

• Για πρόσθεση

```
x+=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
12
```

• Για αφαίρεση

```
x-=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
2
```

• Για γινόμενο

```
x*=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
35
```

• Για διαίρεση

```
x/=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
1.4
```

• Για

```
x%=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
1
```

•Για

```
x//=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
1.75
```

• Για

```
x**=5
print(x)
```

Όταν το εκτελέσουμε και στις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
21.875
```

• Συσχέτιση (Associativity)

Οι τελεστές με την ίδια προτεραιότητα υπολογίζονται από τα αριστερά προς τα δεξιά. Για παράδειγμα, το 2 + 3 + 5 υπολογίζεται και ως (2 + 3) + 5. Μερικοί τελεστές ανάθεσης υπολογίζονται από τα δεξιά προς τα αριστερά, για παράδειγμα το a = b = c είναι το ίδιο με a = (b = c).

• Τελεστές Σύγκρισης

Οι τελεστές σύγκρισης χρησιμοποιούνται για να συγκρίνουμε δύο τιμές.

Operator	Name	Example
= =	Ίσο	x = = y
! =	Διαφορετικό	x ! = y
>	Μεγαλύτερο από	x > y
<	Μικρότερο από	x < y
> =	Μικρότερο ή ίσο	x > = y
< =	Μικρότερο ή ίσο με	x < = y

Ας δούμε κάποια παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 5

Να υπολογίσετε για x = 3 και y = 4 να εκτυπώσετε τα αποτελέσματά από τα παρακάτω:
Έχουμε,

• Ίσο

```
print(x==y)
#Θα επιστρέψει False γιατί το 3 δεν είναι ίσο με 4
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
```

• **Διαφορετικό**

```
print(x!=y)
#Θα επιστρέψει True γιατί το 3 δεν είναι ίσο με 4
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

• **Μεγαλύτερο από**

```
print(x>y)
#Θα επιστρέψει False γιατί το 3 είναι μικρότερο από το 4
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
```

• **Μικρότερο από**

```
print(x<y)
#Θα επιστρέψει True γιατί το 3 είναι μεγαλύτερο από το 4
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

• **Μεγαλύτερο ή ίσο**

```
print(x>=y)
#Θα επιστρέψει False γιατί το 3 δεν είναι μεγαλύτερο ή ίσο με
          4
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
```

• **Μικρότερο ή ίσο**

```
print(x<=y)
#Θα επιστρέψει True γιατί το 3 είναι μικρότερο ή ίσο με 4
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

• Λογικοί Τελεστές

Οι λογικοί τελεστές χρησιμοποιούνται για συνδυάσουν εντολές υπό όρους.

Operator	Name	Example
and	Επιστρέφει True εάν και οι δύο προτάσεις είναι αληθείς	x < 5 and x < 10
or	Επιστρέφει True εάν μία από τις προτάσεις είναι αληθής	x < 5 and x < 10
not	Επιστρέφει False εάν το αποτέλεσμα είναι αληθής	not(x < 5 and x < 10)

Ας δούμε κάποια παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 6

Να υπολογίσετε για x = 3 και y = 4 να εκτυπώσετε τα αποτελέσματά από τα παρακάτω:
Έχουμε,

• And

```
x=3
print(x>5 and x<2)
#Θα επιστρέψει False αφού και οι δύο συνθήκες είναι ψευδής (το
           3
δεν είναι μεγαλύτερο από το 5, αλλά και το 3 δεν είναι
           μικρότερο
από το 2)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
```

• Or

```
x=3
print(x>2 or x<4)
#Θα επιστρέψει True επειδή μία από τις συνθήκες είναι αληθής
           (το
3 είναι μεγαλύτερο από το 2, αλλά και το 3 είναι μικρότερο από
           το
4)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

• Not

```
x=3
print(not x>5 and x<10)
#Θα επιστρέψει False επειδή το Not χρησιμοποιείται για την
αντιστροφή του αποτελέσματος
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
```

• Χειριστές Μελών Python

Οι τελεστές μέλους χρησιμοποιούνται για να ελέγξουν εάν μια ακολουθία θα παρουσιάζεται σε ένα αντικείμενο.

Operator	Name	Example
in	Επιστρέφει True εάν υπάρχει μία ακολουθία με την καθορισμένη τιμή στο αντικείμενο	x in y
not in	Επιστρέφει False εάν μια ακολουθία με την καθορισμένη τιμή δεν υπάρχει στο αντικείμενο	x not in y

Ας δούμε κάποια παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 7

Έχουμε για in:

```
x=[“apple”, “banana”]
print(“banana” in x))
#Θα επιστρέψει True επειδή μια ακολουθία με την τιμή “banana”
βρίσκεται στη λίστα
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

Έχουμε για not in:

```
x=[“apple”, “banana”]
print(“cherry” not in x))
#Θα επιστρέψει True επειδή μια ακολουθία με την τιμή “cherry”
          δεν βρίσκεται στη λίστα
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

3.19 Μεταβλητή Εξόδου – (Print)

Για την έξοδο από των μεταβλητών στην Python, χρησιμοποιούμε τη συνάρτηση print().

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε τη μεταβλητή x και θέλουμε να «βγούμε» από αυτή, τότε:

```
x=”Η Python είναι μια εκπληκτική γλώσσα προγραμματισμού”
print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Η Python είναι εκπληκτική γλώσσα προγραμματισμού
```

• Πολλές μεταβλητές στην print ( )

Στη συνάρτηση print(), γίνετε να βγάλουμε πολλές μεταβλητές χωρισμένες με κόμμα.

ΠΑΡΑΔΕΙΓΜΑ 2

Έχουμε τρεις μεταβλητές x, y, z :

```
x=”Η Python”
y=“είναι”
z=”εκπληκτική”
print(x,y,z)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Η Python είναι εκπληκτική
```

• Τελεστής + στην print ( )

Για την έξοδο πολλαπλών μεταβλητών μπορούμε να χρησιμοποιήσουμε τον τελεστή +. Ωστόσο, ο καλύτερος τρόπος (για να μην υπάρξει σύγχυση με την πράξη της πρόσθεσης), είναι να διαχωρίσουμε τις μεταβλητές με κόμμα (,) .

ΠΑΡΑΔΕΙΓΜΑ 3

Είναι:

```
x=”Η Python”
y=“είναι”
z=”εκπληκτική”
print(x+y+z)
```

Παρατηρούμε ότι και στις δύο παραπάνω περιπτώσεις, όταν εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Η Python είναι εκπληκτική
```

Προσοχή

Για τους αριθμούς ο τελεστής + λειτουργεί ως μαθηματικός τελεστής (προσθέτει)

ΠΑΡΑΔΕΙΓΜΑ 4

Είναι:

```
X=5
y=3
print(x+y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
8
```

Προσοχή

Στη συνάρτηση print() θα εμφανιστεί σφάλμα αν χρησιμοποιήσουμε τον τελεστή (+) και όχι το κόμμα (,) σε συνδυασμό συμβολοσειρών και αριθμών.

ΠΑΡΑΔΕΙΓΜΑ 5

Έχουμε:

```
x=”Η Python”
y=3
print(x+y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Error
```

ΠΑΡΑΔΕΙΓΜΑ 6

Να υπολογίσετε εμβαδόν και την περίμετρο ενός ορθογωνίου παραλληλογράμμου.

Έχουμε:

```
length = 4      # μήκος του ορθογωνίου
breadth = 2          #πλάτος του ορθογωνίου

area = length * breadth
perim = 2 * (length + breadth)
print (“Το εμβαδόν του ορθογωνίου είναι:”,area)
print (“Η περίμετρος του ορθογωνίου είναι:”,perim)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το εμβαδόν του ορθογωνίου είναι: 8
Η περίμετρος του ορθογωνίου είναι: 12
```

Να δούμε πώς λειτουργεί το παραπάνω πρόγραμμα.

Χρησιμοποιούμε τις μεταβλητές length και breadth για να αποθηκεύσουμε το μήκος και το πλάτος αντίστοιχα του ορθογωνίου. Αποθηκεύονται το αποτέλεσμα της πράξης length * breadth στη μεταβλητή area και το αποτέλεσμα της πράξης 2 * (length + breadth) στη μεταβλητή perim, για να υπολογίσουμε το εμβαδό και την περίμετρό του. Τέλος με τη συνάρτηση print τυπώνουμε τα αποτελέσματα στην οθόνη.

Σημείωση

Παρατηρούμε ότι η Python «τυπώνει όμορφα» τα αποτελέσματα και παρόλο που δεν έχουμε καθορίσει κενά ανάμεσα στη μεταβλητή area και perim το εφαρμόζει η Python από μόνη της για εμάς. Έτσι το πρόγραμμα είναι πολύ πιο κομψό και ευανάγνωστο, κάνοντας τη ζωή του προγραμματιστή πιο εύκολη.

3.20 Ενσωματωμένοι Τύποι Δεδομένων

• Ενσωματωμένοι Τύποι

Στον προγραμματισμό, ο τύπος δεδομένων είναι μια σημαντική έννοια. Οι μεταβλητές γίνεται να αποθηκεύουν δεδομένα διαφορετικών τύπων και διαφορετικοί τύποι μπορούν να κάνουν διαφορετικές εργασίες.

Η Python διάφορους τύπους δεδομένων ενσωματωμένους από προεπιλογή και είναι:

• Τύπος κειμένου: str

• Αριθμητικοί τύποι:	int, float, complex
• Τύποι ακολουθιών:	list, tuple, range
• Τύπος χαρτογράφησης:	dict
• Τύποι συνόλου:	set, frozenset
• Τύπος Boolean:	bool
• Δυαδικοί τύποι:	bytes, bytearray, memoryview
• Κανένας τύπος:	NoneType

• Λήψη του Τύπου Δεδομένων – ΣυνάρτησηType ( )

Όπως είχαμε προαναφέρει η συνάρτηση type( ) μπορεί να μας «δώσει» τον τύπο δεδομένων οποιαδήποτε αντικειμένου.

ΠΑΡΑΔΕΙΓΜΑ 7

Να εκτυπώσετε τον τύπο δεδομένων της μεταβλητής x:

```
x=15
print(type(x))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
<class 'int'>
```

• Ρύθμιση του Τύπου Δεδομένων

Ο τύπος δεδομένων στην Python, ορίζεται όταν εκχωρείται μια τιμή σε μια μεταβλητή.

ΠΑΡΑΔΕΙΓΜΑ	**Data Type**
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana"", "cherry"]	list
x = ("apple", "banana"", "cherry")	`tuple

x = range(6)	range
x = {“name” : “John””, “age” : 36}	dict
x = {“apple”, “banana””, “cherry”}	set
x = frozenset({“apple”, “banana””, “cherry”})	frozenset
x = True	bool
x = b “Hello”	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

• Ρύθμιση του Συγκεκριμένου Τύπου Δεδομένων

Για να καθορίσουμε τον τύπο δεδομένων, μπορούμε να χρησιμοποιήσουμε τις ακόλουθες συναρτήσεις:

ΠΑΡΑΔΕΙΓΜΑ	**Data Type**
x = str (“Hello World”)	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list([“apple”, “banana””, “cherry”])	list
x = tuple((“apple”, “banana””, “cherry”))	tuple
x = range(6)	range
x = dict(“name” : “John””, “age” : 36)	dict
x = set((“apple”, “banana””, “cherry”))	set
x = frozenset((“apple”, “banana””, “cherry”))	frozenset

x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

• Αντιστοίχιση String - Variable

Η αντιστοίχιση μιας συμβολοσειράς σε μια μεταβλητή γίνεται με το όνομα της μεταβλητής ακολουθούμενο από ένα σύμβολο ίσου και τη συμβολοσειρά:

3.21 Booleans

Στον προγραμματισμό κάποιες φορές είναι αναγκαίο να γνωρίζουμε εάν μία έκφραση είναι true (αλήθεια) ή false (ψέμα).

Έτσι οι Booleans αντιπροσωπεύουν μία από τις δύο τιμές false ή true.

Η bool( ) συνάρτηση μας επιτρέπει να αξιολογήσουμε οποιαδήποτε τιμή και να μας επιστρέψει την τιμή true ή false.

Στην Python μπορούμε να χρησιμοποιήσουμε οποιαδήποτε έκφραση, λαμβάνοντας μία από τις δύο απαντήσεις false ή true.

Η Python επιστρέφει true ή false, όταν εκτελείται μια συνθήκη σε μια εντολή If.

Ας δούμε μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 8

Έχουμε:

```
print(10>8)
print(10==8)
print(10<8)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Σωστό
Λάθος
```

```
Σωστό
```

ΠΑΡΑΔΕΙΓΜΑ 9

Εκτυπώστε ένα μήνυμα σχετικά με το αν μια συνθήκη είναι true ή false.

Έχουμε:

```
a=110
b=25
if b>a:
     print(“Το b είναι μεγαλύτερο του a”)
   else:
     print(“Το b δεν είναι μεγαλύτερο του a”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το b δεν είναι μεγαλύτερο του α
```

ΠΑΡΑΔΕΙΓΜΑ 10

Δημιουργήστε και εκτυπώστε ένα μήνυμα εμφανίζοντας αν είναι true ή false μια συνθήκη.

Έχουμε:

```
print(bool(“Hello”))
print(bool(15))
```

Ή χρησιμοποιώντας δύο μεταβλητές x, y:

```
x=“Hello”
y=15
print(bool(“x”))
print(bool(“y”))
```

Όταν το εκτελέσουμε και για τις δύο περιπτώσεις θα εμφανιστεί στην οθόνη:

```
true
true
```

3.22 True – Τιμές ή Αντικείμενο

Ένα μήνυμα σχετικά με το αν μια συνθήκη είναι True, όταν:

- ✓ Συνήθως οποιαδήποτε τιμή, εάν έχει κάποιο είδος περιεχομένου.
- ✓ Σχεδόν όλες οι συμβολοσειρές, εκτός από τις κενές.
- ✓ Οποιοσδήποτε αριθμός, εκτός από το μηδέν (0).
- ✓ Οποιαδήποτε λίστα, σύνολο και λεξικό εκτός από τα κενά.

ΠΑΡΑΔΕΙΓΜΑ 11

Επιστρέφουν τιμές Boolean και συγκεκριμένα True, τα παρακάτω :

```
bool(“abc”)
bool(123)
bool([“apple”, “banana”, “orange”])
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
true
true
true
```

3.23 False – Τιμές ή Αντικείμενο

Ένα μήνυμα σχετικά με το αν μια συνθήκη είναι True, όταν:

- ✓ Όλες οι κενές τιμές, όπως ( ), [ ], { }, “ “.
- ✓ Ο αριθμός μηδέν (0).
- ✓ Η τιμή None.
- ✓ Η αξία false σε false.
- ✓ Όταν ένα αντικείμενο δημιουργείτε από μία κλάση με len συνάρτηση που επιστρέφει 0 ή False.

ΠΑΡΑΔΕΙΓΜΑ 12

Επιστρέφουν τιμές Boolean και συγκεκριμένα False, τα παρακάτω :

```
bool (None )
bool ( 0 )
bool ( “ “ )
bool ( ( ) )
bool ( [ ] )
bool ( { } )
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
False
False
False
False
False
False
```

ΠΑΡΑΔΕΙΓΜΑ 13

Έχουμε:

```
class myclass():
    def_len(self):
        return 0
myobj=myclass()
print(bool(myobj))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
False
```

Οι συναρτήσεις μπορούν να επιστρέφουν ένα Boolean, δηλαδή αν είναι true ή false μια συνθήκη.

ΠΑΡΑΔΕΙΓΜΑ 14

Να δημιουργήσετε συναρτήσεις που να επιστρέφουν τιμές Boolean και να τις εκτυπώσετε.

Έχουμε:

```
def myFunction():
     return True
print(myFunction())
def myFunction()
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
True
```

Ακόμα, μπορούμε να εκτελέσουμε ποια απάντηση θα επιστρέφει μια συνάρτηση με βάση τη Boolean.

ΠΑΡΑΔΕΙΓΜΑ 15

Αν μια συνάρτηση επιστρέφει True να εκτυπώσετε “ΝΑΙ!”, αλλιώς εκτυπώστε “ΝΟ!”.

Έχουμε:

```
def myFunction():
     return True
if myFunction():
     print(“YES!”)
   else:
     print(“NO!”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
YES!
```

Η Python περιλαμβάνει κάποιες συναρτήσεις που επιστρέφουν μια τιμή Boolean, όπως είναι η isinstance ( ), η οποία μπορεί να προσδιορίσει ένα αντικείμενο συγκεκριμένο τύπο δεδομένων.

ΠΑΡΑΔΕΙΓΜΑ 16

Να ελέγξτε αν ένα αντικείμενο είναι ακέραιος ή όχι.

Έχουμε:

```
χ=20
print(isinstance(x,int))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
true
```

3.24 Δομές Ελέγχου και Επανάληψης

Η Python στα προγράμματα που είδαμε στα προηγούμενα μαθήματα, εκτελεί πιστά πάντα μια ίδια σειρά εντολών. Όμως, αν θέλουμε να αλλάξουμε τη σειρά εκτέλεσης του προγράμματος παίρνοντας αποφάσεις το ίδιο το πρόγραμμα ώστε να εκτελέσει κάποιες διαφορετικές ενέργειες, υπό διαφορετικές συνθήκες και προϋποθέσεις, μπορεί να επιτευχθεί χρησιμοποιώντας εντολές ελέγχου ροής. Στην Python υπάρχουν τρεις εντολές ελέγχου ροής, η if, η for και η while.

3.25 Δομές Ελέγχου – if – else – elif

Η εντολή if επιτρέπει την εκτέλεση μίας ή περισσοτέρων εντολών υπό συνθήκη, χρησιμοποιείται δηλαδή για να ελεγχθεί μια συνθήκη και εάν αυτή η συνθήκη (if) είναι αληθής, τότε εκτελείται ένα σύνολο εντολών, διαφορετικά (else) γίνεται επεξεργασία ενός άλλου συνόλου εντολών. Με τον όρο συνθήκη θεωρούμε μία μεταβλητή που έχει μία λογική τιμή. Μία απλή συνθήκη ορίζεται με τη χρήση σχεσιακών και λογικών τελεστών όπως και λογικών μεταβλητών. Η χρήση του όρου else είναι προαιρετική.

Η εντολή if συνδυάζεται με else και με elif και αποτελεί μοναδική δομή ελέγχου της Python με πάρα πολλές δυνατότητες, η οποία πλεονεκτεί σημαντικά σε σχέση με άλλες γλώσσες προγραμματισμού.

Το μειονέκτημά της if θα μπορούσαμε να πούμε ότι κατά τη σύνταξή της χρειάζεται μεγάλη προσοχή στα blocks των προτάσεων, προσέχοντας έτσι τον αριθμό των εσοχών μπροστά από τις εντολές.

Οι εσοχές στην Python συμπεριφέρονται όπως οι { … } σε άλλες γλώσσες προγραμματισμού, όπως είναι η C και ή η Java.

Η απλούστερη μορφή της If… είναι:

if συνθήκη:

εντολή - 1

εντολή - 2

…….

επόμενη - εντολή

Εάν η συνθήκη είναι True (αληθής), τότε η Python εκτελεί το μπλοκ εντολών (που είναι στοιχισμένες πιο δεξιά), εντολή - 1, εντολή - 2, κλπ. Εάν η συνθήκη είναι False (ψευδής), τότε εκτελείται το επόμενο μπλοκ εντολών, επόμενη – εντολή, που η στοίχισή του πρέπει να είναι αντίστοιχη, κάτω ακριβώς του if.

ΠΑΡΑΔΕΙΓΜΑ 17

Να γραφεί πρόγραμμα που θα επιλύει την εξίσωση πρώτου βαθμού.

Για την εξίσωση ax + b = 0, έχουμε:

```
a=int(input('a='))
b=int(input('b='))
if a==0:
     print('Αδύνατη')
   else:
     x=-b/a
     print('x=',x)
```

Η Python υποστηρίζει από τα μαθηματικά τις παρακάτω επικρατέστερες λογικές συνθήκες:

- a = = b ⟶ Ισούται με
- a ! = b ⟶ Διαφορετικό από (Όχι ίσον)
- a < b ⟶ Μικρότερο από
- a < = b ⟶ Μικρότερο ή ίσο με
- a > b ⟶ Μεγαλύτερο από
- a > = b ⟶ Μεγαλύτερο ή ίσο με

Οι παραπάνω λογικές συνθήκες χρησιμοποιούνται με ποικίλους τρόπους, αλλά συνήθως σε δηλώσεις "if" και βρόχος.

ΠΑΡΑΔΕΙΓΜΑ 18

Έχουμε δύο μεταβλητές a = 15 και b = 100, τις οποίες τις χρησιμοποιούμε ως μέτρο της εντολής if για να ελέγξουμε αν το b είναι μεγαλύτερο από το a. Ο αριθμός 100 είναι μεγαλύτερος από το 15, επομένως το b είναι μεγαλύτερο από το a και έτσι εκτυπώνουμε στην οθόνη ("Το b είναι μεγαλύτερο από το a").
Έτσι είναι:

```
a=int(input('a='))
a=15
b=100
if b>a:
    print("Το b είναι μεγαλύτερο από το a")
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το b είναι μεγαλύτερο από το a
```

Μπορούμε αν έχουμε μία μόνο εντολή για εκτέλεση, να γράψουμε στην ίδια γραμμή με την εντολή if.

ΠΑΡΑΔΕΙΓΜΑ 19

Έχουμε:

```
a=15
b=100
if b>a: print("Το b είναι μεγαλύτερο από το a")
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το b είναι μεγαλύτερο από το a
```

Προσοχή

Η Python για να ορίσει το εύρος στον κώδικα χρησιμοποιεί την εσοχή, δηλαδή κενό χώρο στην αρχή της γραμμής. Εάν δεν χρησιμοποιήσουμε κενό στη δήλωση, τότε θα εμφανίσει σφάλμα η Python.

Άλλες γλώσσες προγραμματισμού χρησιμοποιούν αγκύλες για να ορίσουν το εύρος στον κώδικα.

ΠΑΡΑΔΕΙΓΜΑ 20

Έχουμε τη δήλωση, χωρίς κενό στην αρχή της γραμμής:

```
a=15
b=100
if b>a:
    print(“Το b είναι μεγαλύτερο από το a”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη ένα σφάλμα (error).

Η δομή if συνθήκη: …else και if…, elif…, else επιτρέπει τον έλεγχο αρκετών συνθηκών και η σύνταξή τους έχει την παρακάτω μορφή:

```
if συνθήκη1:
        εντολές – α
elif συνθήκη2 Then
        εντολές – β
……..
else:
        εντολές – γ
```

Οι else και elif επιτρέπουν τον ορισμό πολλών συνθηκών σε μια δομή ελέγχου if που χρησιμοποιούνται μόνο μέσα σε αυτή και όχι εκτός. Οι συνθήκες ελέγχονται με τη σειρά, αν είναι η συνθήκη είναι αληθής εκτελούνται οι επόμενες εντολές που βρίσκονται στο elif ή else. Αλλιώς αν η συνθήκη είναι ψευδής, τότε εκτελείται η επόμενη elif και else με «υπερπήδηση» των εντολών που ακολουθούν τη ψευδή συνθήκη. Τέλος, αν δεν είναι αληθής καμιά από τις εντολές, τότε εκτελούνται οι εντολές που βρίσκεται μετά το else.

• el if

Χρησιμοποιούμε στην Python την εντολή elif στον κώδικα όταν:
“Αν οι προηγούμενες συνθήκες δεν ήταν αληθείς, τότε δοκιμάστε αυτήν την συνθήκη”.

ΠΑΡΑΔΕΙΓΜΑ 21

Έχουμε δύο μεταβλητές a = 22 και b = 22, οι οποίες είναι ίσες.

```
a=22
b=22
if b>a:
    print(“Το b είναι μεγαλύτερο από το a”)
elif a==b:
    print(“Το a και το b είναι ίσα”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το a και το b είναι ίσα
```

Στο παράδειγμα αυτό αφού οι δύο μεταβλητές είναι ίσες, η πρώτη συνθήκη δεν είναι αληθής, αλλά είναι αληθής η συνθήκη elif και επομένως όταν εκτελέσουμε το πρόγραμμα αυτό θα εμφανιστεί στην οθόνη: Το a και το b είναι ίσα.

- else

Χρησιμοποιούμε την εντολή else όταν για οτιδήποτε δεν εντοπίζεται από τις προηγούμενες συνθήκες.

ΠΑΡΑΔΕΙΓΜΑ 22

Έχουμε δύο μεταβλητές a = 100 και b = 22, όπου η μεταβλητή a είναι μεγαλύτερη από τη b.

Έχουμε:

```
a=100
b=22
if b>a:
    print(“Το b είναι μεγαλύτερο από το a”)
elif a==b:
    print(“Το a και το b είναι ίσα”)
else:
    print(“Το a είναι μεγαλύτερο του b”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη

```
Το a είναι μεγαλύτερο του b
```

Στο παράδειγμα αυτό πρώτη συνθήκη δεν είναι αληθής, αλλά είναι αληθής (αφού a > b), το ίδιο και η συνθήκη elif δεν είναι αληθής, επομένως το πρόγραμμα εκτελείται για τη συνθήκη else και εκτυπώνουμε στην οθόνη: Το a είναι μεγαλύτερο του b.

• If else

Είναι δυνατό να υπάρχει ένα else χωρίς το elif.

ΠΑΡΑΔΕΙΓΜΑ 23

Έχουμε:

a=100

```
b=22
if b>a:
    print(“Το b είναι μεγαλύτερο από το a”)
else:
    print(“Το b δεν είναι μεγαλύτερο του a”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το b δεν είναι μεγαλύτερο του a
```

Αν έχουμε μία μόνο εντολή για εκτέλεση, μία για το if και μία για την άλλη, μπορούμε να τα γράψουμε όλα στην ίδια γραμμή.

ΠΑΡΑΔΕΙΓΜΑ 24

Έχουμε:

```
a=5
b=150
print(“A”) if a>b else print(“B”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
B
```

Επίσης, μπορούμε να έχουμε πολλές δηλώσεις στην ίδια γραμμή.

ΠΑΡΑΔΕΙΓΜΑ 25

```
a = 150
b = 150
print(“A”) if a > b  else print(“=”) if a==b else print(“B)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
=
```

• And

Η and είναι ένας λογικός τελεστής και χρησιμοποιείται για τον συνδυασμό εντολών υπό όρους.

ΠΑΡΑΔΕΙΓΜΑ 26

Δημιουργείστε πρόγραμμα που να ελέγχει εάν το a είναι μεγαλύτερο του b και εάν το c είναι μεγαλύτερο του a.

Έχουμε:

```
a = 300
b = 25
c = 400
if a > b  and  c > a:
   print(“Και οι δύο συνθήκες είναι αληθής»)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Και οι δύο συνθήκες είναι αληθείς
```

• Or

Η or είναι ένας λογικός τελεστής και χρησιμοποιείται για τον συνδυασμό δηλώσεων υπό όρους.

ΠΑΡΑΔΕΙΓΜΑ 27

Δημιουργείστε πρόγραμμα που να ελέγχει εάν το a είναι μεγαλύτερο του b ή εάν το a είναι μεγαλύτερο του c.

Έχουμε:

```
a = 300
b = 25
c = 400
if a > b  or  a > c:
   print(“Τουλάχιστον μία από τις δύο συνθήκες είναι αληθής”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Τουλάχιστον μία από τις δύο συνθήκες είναι αληθείς.
```

ΠΑΡΑΔΕΙΓΜΑ 28

Να γραφεί πρόγραμμα το οποίο να διαβάζει από το πληκτρολόγιο τη θερμοκρασία του περιβάλλοντος και να εμφανίζει τα παρακάτω σχετικά μηνύματα:

Εάν η θερμοκρασία είναι μικρότερη ή ίση με 8 βαθμούς, τότε να εμφανίζει το μήνυμα: «Κάνει παγωνιά».
Εάν η θερμοκρασία είναι μεγαλύτερη από 8 βαθμούς και μικρότερη ή ίση από 15, τότε να εμφανίζει το μήνυμα: «Κάνει ψύχρα».
Εάν η θερμοκρασία είναι μεγαλύτερη των 15 και μικρότερη ή ίση από 28 βαθμούς, τότε να εμφανίζει το μήνυμα: «Έχει καλό καιρό».
Εάν η θερμοκρασία είναι μεγαλύτερη από 28 βαθμούς,, τότε να εμφανίζει το μήνυμα: «Κάνει πολύ ζέστη».

Έχουμε:

```
thermokrasia = input(“Δώσε τη θερμοκρασία του περιβάλλοντος”)
thermokrasia = float(thermokrasia)
if thermokrasia < = 8 :
    print(“Κάνει παγωνιά”)
elif thermokrasia < = 15:
    print(“Κάνει ψύχρα”)
elif thermokrasia < 28:
    print(“Έχει καλό καιρό”)
else:
    print(“Κάνει ζέστη”)
```

3.26 Φωλιασμένο if

Είναι δυνατό να έχουμε μία if μέσα σε μια άλλη if εντολές και αυτό λέγεται ένθετες if εντολές. Η μια if μέσα στην άλλη, ονομάζεται φωλιασμένη if.

ΠΑΡΑΔΕΙΓΜΑ 29

Έχουμε:

```
x=62
if x>10:
     print(“Πάνω από δέκα,”)
if x>20:
     print(“και επίσης πάνω από είκοσι!”)
   else:
     print(“αλλά όχι πάνω από είκοσι.”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Πάνω από δέκα,
Αλλά και πάνω από είκοσι!
```

• If – pass

Οι δηλώσεις If δεν μπορούν να είναι κενές, αλλά αν υπάρχει κάποια if χωρίς περιεχόμενο, τότε για να αποφύγουμε σφάλμα μπορούμε να βάλουμε pass.

ΠΑΡΑΔΕΙΓΜΑ 30

Έχουμε:

```
a=5
b=85
if b>a:
   pass
```

Όταν το εκτελέσουμε δεν θα εμφανιστεί στην οθόνη τίποτα.
Βάζοντας υπάρχει κενή δήλωση if , θα δημιουργούσε σφάλμα χωρίς τη δήλωση pass.

Ας εξηγήσουμε αναλυτικά ένα παράδειγμα με την εντολή if.

ΠΑΡΑΔΕΙΓΜΑ 31

Έχουμε:

```
number = 25
quess = int(input(“Εισάγετε έναν ακέραιο αριθμό:”))

if quess == number:
   print(“Μπράβο, μαντέψατε σωστά!”)

elif quess < number:
  print(“ Όχι, ο αριθμός που πληκτρολογήσατε είναι
μεγαλύτερος.”)

else:
  print(“Όχι, ο αριθμός που πληκτρολογήσατε είναι
μικρότερος.”)

print (“Τέλος”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Εισάγετε έναν ακέραιο αριθμό: 40
```

```
Όχι, ο αριθμός που πληκτρολογήσατε είναι μεγαλύτερος.
Τέλος
```

Ας δούμε αναλυτικά πώς λειτουργεί το πρόγραμμα.

Στο παραπάνω πρόγραμμα, ο χρήστης μαντεύει αριθμούς και ελέγχουμε αν ο κάθε αριθμός που δίνει αντιστοιχεί στον αριθμό που έχουμε ορίσει. Χρησιμοποιούμε τη μεταβλητή number ορίζουμε τον τυχαίο ακέραιο αριθμό που πληκτρολογούμε. Στη συνάρτηση input( ) παίρνουμε την πρόγνωση του χρήστη. Μόλις εισάγουμε κάτι και πατήσουμε enter, η συνάρτηση input( ) επιστρέφει αυτό που εισάγαμε ως συμβολοσειρά (string), μετατρέποντάς την σε ακέραιο αριθμό, με τη χρήση της εντολής int και αποθηκεύεται στην μεταβλητή quess. Στη συνέχεια συγκινούμε τον αριθμό που δίνει ο χρήστης με τον αριθμό που ορίσαμε από την αρχή. Εάν οι αριθμοί είναι ίδιοι τυπώνεται στην οθόνη ένα μήνυμα επιτυχίας που ορίζουμε εμείς. Αν ο αριθμός είναι μικρότερος από τον αριθμό μας, τότε εμφανίζεται ένα μήνυμα σχετικό με το ότι ο αριθμός είναι μικρότερος. Ο όρος elif που χρησιμοποιούμε στην πραγματικότητα συνδυάζει δύο σχετιζόμενες εντολές, if else-if else σε μια εντολή if-elif-else, κάνοντας το πρόγραμμά μας ευκολότερο και έτσι μειώνεται ο αριθμός των εσοχών που απαιτούνται. Πολύ προσοχή, οι εντολές elif και else να έχουνε στο τέλος της λογικής γραμμής άνω και κάτω τελεία : .

ΠΑΡΑΔΕΙΓΜΑ 32

Στο προηγούμενο παράδειγμα που είδαμε με τη θερμοκρασία του περιβάλλοντος, μπορούμε να δούμε έναν άλλο τρόπο προγράμματος με εμφωλευμένα if.

Έχουμε:

```
thermokrasia = input(“Δώσε τη θερμοκρασία του περιβάλλοντος”)
thermokrasia = float(thermokrasia)
if thermokrasia < = 8 :
    print(“Κάνει παγωνιά”)
else thermokrasia > 8  and  thermokrasia < = 15:
    print(“Κάνει ψύχρα”)
if thermokrasia > 15 and thermokrasia < = 28:
    print(“Έχει καλό καιρό”)
if thermokrasia > 28:
    print(“Κάνει ζέστη”)
```

Προσοχή

Θα πρέπει να είμαστε πάρα πολύ προσεκτικοί στο πώς χρησιμοποιούμε τα επίπεδα των εσοχών (των αριθμό των κενών διαστημάτων) που δηλώνουμε στην Python και είναι απίστευτα σημαντικές καθώς ορίζουν ποιες εντολές ανήκουν σε πιο μπλοκ (πλοκάδα) εντολών.

Έτσι αφού η Python τελειώσει την εκτέλεση ολόκληρης της εντολής if κι των συσχετιζόμενων όρων elif και else, συνεχίζει στο επόμενο μπλοκ (πλοκάδα) εντολών που περιέχει την εντολή if ξεκινώντας την εκτέλεση του προγράμματος. Τέλος η Python με την εντολή print ("Τέλος") βλέπει το τέλος του προγράμματος και απλά τελειώνει εκεί.

3.27 Δομές Επανάληψης

3.27.1 While & for Loops

Η Python διαθέτει τις συνηθισμένες επαναληπτικές εντολές, έχοντας ακόμα πολλές άλλες λειτουργίες.

Η εντολή while μας επιτρέπει να εκτελούμε επανειλημμένα, για όσο μια συνθήκη είναι αληθής και αποκαλείται αλλιώς εντολή βρόχου (looping statement). Είναι δηλαδή μια εντολή επανάληψης στην οποία ο έλεγχος της επανάληψης πραγματοποιείται μέσω μίας λογικής πρότασης. Η εντολή while προαιρετικά μπορεί να έχει else.

Η γενική της δομή είναι:

While λογική πρόταση:

Εντολή - 1

Εντολή – 1

. . . . .

Εντολή – ν

Επόμενη εντολή

Προσοχή

Επίσης, όπως και στη σύνταξη δομής ελέγχου και εδώ χρειάζεται προσοχή κατά τη σύνταξή της στα blocks των προτάσεων, προσέχοντας έτσι τον αριθμό των εσοχών μπροστά από τις εντολές.

Η Python έχει δύο πρωτόγονες επαναληπτικές εντολές βρόχο:

- while loops
- for loops

• While loop

Το βρόγχο while το χρησιμοποιούμε όταν μια συνθήκη είναι αληθής και θέλουμε να εκτελέσουμε ένα σύνολο εντολών.

Ο βρόχο while απαιτεί κάποιες σχετικές μεταβλητές να είναι έτοιμες, όπως για παράδειγμα να ορίσουμε μια μεταβλητή ευρετηρίου ή την αρχική τιμή ενός μετρητή.

ΠΑΡΑΔΕΙΓΜΑ 1

Να γραφεί πρόγραμμα που να εκτυπώνεται το i εφόσον είναι μικρότερος από τον αριθμό 5.

Έχουμε:

```
i=1 #Ορίζουμε αρχική τιμή του μετρητή i ίσο με 1
while i<5:
      print(i)
      i+=1 #Είναι το ίδιο με i=i+1, αυξάνεται το i κατά ένα
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
1
2
3
4
```

Προσοχή

Το i πρέπει να το αυξάνουμε κατά ένα (i + = 1), αλλιώς ο βρόχο θα συνεχιστεί για πάντα.

ΠΑΡΑΔΕΙΓΜΑ 2

Να γραφεί πρόγραμμα για με πρώτη υπο-ακολουθία της σειράς που να εκτυπώνεται το i εφόσον είναι μικρότερος από τον αριθμό 5.

Έχουμε:

```
i+=1 #Είναι το ίδιο με i=i+1, αυξάνεται το i κατά ένα
i=1 #Ορίζουμε αρχική τιμή του μετρητή i ίσο με 1
while i<5:
      print(i)
      i+=1 #Είναι το ίδιο με i=i+1, αυξάνεται το i κατά ένα
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
1
2
3
```

```
4
```

ΠΑΡΑΔΕΙΓΜΑ 3

Έχουμε,

```
number = 28
running = True

while running:
     quess = int(input("Εισάγετε έναν ακέραιο αριθμό:"))

     if quess == number:
          print ("Μπράβο, μαντέψατε σωστά.")
          running = False  # σταματάει το βρόγχο while εδώ
     elif quess < number:
          print (" Όχι, ο αριθμός είναι μεγαλύτερος.")
     else:
          print (" Όχι, ο αριθμός είναι μικρότερος.")

print("Τέλος")
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Εισάγετε έναν ακέραιο αριθμό: 50
Όχι, ο αριθμός είναι μεγαλύτερος.
Εισάγετε έναν ακέραιο αριθμό: 19
Όχι, ο αριθμός που πληκτρολογήσατε είναι μικρότερος.
Εισάγετε έναν ακέραιο αριθμό: 28
Μπράβο, μαντέψατε σωστά!
Τέλος
```

Ας δούμε αναλυτικά πώς λειτουργεί το πρόγραμμα.

Το παραπάνω πρόγραμμα αποτελεί ένα καλό παράδειγμα χρήσης της εντολής while. Συγκεκριμένα, παίζουμε πάλι ένα παιχνίδι πρόγνωσης αλλά πλεονεκτεί στο ότι επιτρέπει στο χρήστη να μαντεύει αριθμούς μέχρι να βρει τελικά τον σωστό αριθμό που ορίσαμε αρχικά. Δηλαδή, το πρόγραμμα εκτελείται επανειλημμένα για κάθε πρόγνωση, όπως το προηγούμενο παράδειγμα(με την εντολή (if –elif-else).

Τις εντολές input και if τις μετακινούμε εντός βρόχο while και πριν τον βρόγχο while ορίζουμε στη μεταβλητή running σε True, προχωρώντας στην εκτέλεση των μπλοκ (πλοκάδα) εντολών while.

Στη συνέχεια, η προϋπόθεση ελέγχεται και πάλι (η μεταβλητή running). Εάν η τιμή της συνεχίζει να είναι true (αληθές), εκτελείται και πάλι ολόκληρη η πλοκάδα εντολών (while –block), διαφορετικά εκτελείται η προαιρετική πλοκάδα εντολών (else-block) προχωρώντας στην επόμενη εντολή.

Η πλοκάδα else εκτελείται όταν η προϋπόθεση while αποκτά την τιμή False (ψευδές).

Προσοχή

Όταν υπάρχει ένας όρος else για ένα βρόγχο while εκτελείται πάντοτε, εκτός και αν διακοπεί ο βρόχο με την εντολή break.

• Break

Με την εντολή break, μπορούμε να σταματήσουμε έναν βρόχο ακόμα κι αν η συνθήκη while είναι αληθής, δηλαδή μπορούμε να διακόψουμε την εκτέλεση της εντολής βρόχο ακόμη κι αν η προϋπόθεση του βρόχο δεν έχει γίνει False (ψευδές) ή δεν έχει επαναληφθεί.

Προσοχή

Είναι σημαντικό να αναφέρουμε, ότι εάν διακόψουμε έναν βρόχο for ή while με αυτόν τον τρόπο, τότε οποιαδήποτε πλοκάδα βρόχο else δεν εκτελείται.

ΠΑΡΑΔΕΙΓΜΑ 4

Να γραφεί πρόγραμμα ώστε όταν είναι ίσο με 2 να πραγματοποιεί έξοδο από το βρόγχο.

Έχουμε:

```
i=1 #Ορίζουμε αρχική τιμή του μετρητή i ίσο με 1
while i<6:
      print(i)
      if i==2:
         break
      i+=1 #Είναι το ίδιο με i=i+1, αυξάνεται το i κατά ένα
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
1
2
```

• Continue

Με την εντολή continue, μπορούμε να σταματήσουμε την τρέχουσα επανάληψη και να συνεχίσουμε με το επόμενο.

ΠΑΡΑΔΕΙΓΜΑ 5

Να γραφεί πρόγραμμα που εάν το i είναι 4, να συνεχίζετε η επόμενη επανάληψη.

Έχουμε:

```
i=0 #Ορίζουμε αρχική τιμή του μετρητή i ίσο με μηδέν
while i<6:
     i+=1 #Είναι το ίδιο με i=i+1, αυξάνεται το i κατά ένα
     if i==4:
        continue
     print(i)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
1
2
3
4
5
6
```

- Else

Με την εντολή else, μπορούμε να εκτελέσουμε ένα μπλοκ κώδικα μία φορά, όταν η συνθήκη πλέον δεν είναι αληθής.

ΠΑΡΑΔΕΙΓΜΑ 6

Να γραφεί πρόγραμμα που να εμφανίζετε ένα μήνυμα όταν η συνθήκη είναι ψευδής.

Έχουμε:

```
i=1 #Ορίζουμε αρχική τιμή του μετρητή i ίσο με 1
while i<6:
     print(i)
     i+=1 #Είναι το ίδιο με i=i+1, αυξάνεται το i κατά ένα
else:
     print("Το i δεν είναι πλέον μικρότερο από 6")
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
1
2
3
4
5
i δεν είναι πλέον μικρότερο από 6
```

• Βρόχο For

Η εντολή for είναι μια άλλη εντολή βρόχο, η οποία επαναλαμβάνεται σε μια ακολουθία εντολών εκτελώντας κάθε αντικείμενο σε μια ακολουθία. Ακολουθία είναι μια ταξινομημένη συλλογή αντικειμένων.

Ένας βρόχο for χρησιμοποιείται για επανάληψη σε μια ακολουθία, δηλαδή είτε σε λίστα, σύνολο ή συμβολοσειρά κλπ.

Με το βρόγχο for μπορούμε να εκτελέσουμε ένα σύνολο εντολών, μία φορά για κάθε στοιχείο σε μια λίστα, σύνολο κλπ. και δεν χρειάζεται να οριστεί από πριν μια μεταβλητή ευρετηρίου.

ΠΑΡΑΔΕΙΓΜΑ 7

Να γραφεί πρόγραμμα που να εκτυπώνει κάθε φρούτο σε μία λίστα φρούτων.

Έχουμε:

```
fruits=[“apple, “banana”, “cherry”]
for x in fruits:
     print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
apple
banana
cherry
```

Μπορεί και οι συμβολοσειρές να είναι επαναλαμβανόμενα αντικείμενα και να περιέχουν ακολουθία χαρακτήρων.

ΠΑΡΑΔΕΙΓΜΑ 8

Να γραφεί πρόγραμμα που να εκτυπώνει κατακόρυφα τα γράμματα ενός φρούτο.

Έχουμε:

```
fruits=[“apple, “banana”, “cherry”]
for x in “banana”:
     print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
b
a
n
a
n
a
```

Με τη δήλωση break μπορούμε να σταματήσουμε τον βρόγχο πριν περάσει όλα τα στοιχεία.

ΠΑΡΑΔΕΙΓΜΑ 9

Να γραφεί πρόγραμμα που όταν x είναι “μπανάνα” να πραγματοποιείται έξοδος από τον βρόγχο.

Έχουμε:

```
fruits=[“apple, “banana”, “cherry”]
for x in fruits:
     print(x)
     if x==”banana”:
        break
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
apple
```
banana

ΠΑΡΑΔΕΙΓΜΑ 10

Να γραφεί πρόγραμμα που όταν x είναι “μπανάνα” να πραγματοποιείται έξοδος από τον βρόγχο, αλλά πριν την εκτύπωση.

Έχουμε:

```
fruits=[“apple, “banana”, “cherry”]
for x in fruits:
     if x==”banana”:
        break
     print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
apple
```

Μπορούμε να διακόψουμε την τρέχουσα επανάληψη του βρόχου με την εντολή continue.

ΠΑΡΑΔΕΙΓΜΑ 11

Να γραφεί πρόγραμμα που να εκτυπώνονται τα φρούτα εκτός από το φρούτο “μπανάνα”.

Έχουμε:

```
fruits=[“apple, “banana”, “cherry”]
for x in fruits:
     if x==”banana”:
        continue
     print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
apple
cherry
```

ΠΑΡΑΔΕΙΓΜΑ 12

Να γραφεί πρόγραμμα που να υπολογίζει και να τυπώνει όλους τους πρώτους αριθμούς, στο διάστημα [a, b] το οποίο να το εισάγει ο χρήστης.

Σημείωση: Ένας φυσικός αριθμός λέγεται πρώτος, όταν διαιρείται μόνο με τον εαυτό του και τη μονάδα.
Έχουμε:

```
a = int (input(‘Αρχή=’))          # Εύρεση πρώτων αριθμών
b = int (input(‘Τέλος=’))         Εισαγωγή διαστήματος αναζήτησης πρώτων
counter = 0
for  x  in range (a, b +1):
      prime = True        # Ελέγχουμε αν βρέθηκε διαιρέτης
      dieretis = 2        # Πιθανοί διαιρέτες
     while prime and dieretis <= x//2
          if x % dieretis = = 0
              prime = False   # Δεν είναι πρώτος αριθμός
         diairetis + - 1
        if prime:
           counter +=1
           print (counter,  x)
```

ΠΑΡΑΔΕΙΓΜΑ 13

Να γραφεί πρόγραμμα που να υπολογίζει και να τυπώνει το άθροισμα των ακεραίων αριθμών από το 1 μέχρι το 10.

Έχουμε,

```
for k in range (1, 11):
     sum + = sum + k
print (sum)
```

ΠΑΡΑΔΕΙΓΜΑ 14

Έχουμε,

```
for i in range (1, 5):
     print (i)
else:
     print (“Ο βρόχος loop τερματίστηκε”)

print(“Τέλος”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
1
2
3
4
0
Ο βρόχος loop τερματίστηκε
```

Ας δούμε αναλυτικά πώς λειτουργεί το πρόγραμμα.

Στο παράδειγμα αυτό τυπώνουμε μια ακολουθία αριθμών, παράγοντας αυτή την ακολουθία με την ενσωματωμένη συνάρτηση range.

Συγκεκριμένα, παρέχουμε δύο αριθμούς και η range επιστρέφει μια ακολουθία αριθμών, αρχίζοντας από τον πρώτο αριθμό έως το δεύτερο αριθμό. Έτσι η range (1, 5) θα μας δώσει την ακολουθία [1, 2, 3, 4] (δηλαδή μέχρι 5-1=4). Εάν δώσουμε έναν τρίτο αριθμό στην range (1, 5, 2) τότε θα μας δώσει ως αποτέλεσμα [1, 3] (δηλαδή το εύρος των αριθμών δεν συμπεριλαμβάνει τον δεύτερο αριθμό).

Στη συνέχεια, ο βρόχος for επαναλαμβάνεται σε αυτό το εύρος (for i in range (1, 5)) που είναι το αντίστοιχο του for i in [1, 2, 3, 4], σαν να αντιστοιχούμε κάθε αριθμό της ακολουθίας στο i ξεχωριστά και στην συνέχεια εκτελείται η πλοκάδα εντολών για κάθε τιμή της i και τυπώνετε η τιμή τους.

Το τμήμα else είναι προαιρετικό και όταν χρησιμοποιείται εκτελείται πάντα μόλις τερματιστεί ο βρόχος for (εκτός αν υπάρχει ενδιάμεσα η εντολή break).

Σημείωση

Ο βρόγχος for . . in δουλεύει για οποιαδήποτε ακολουθία. while εκτελείται πάντοτε, εκτός και αν διακοπεί ο βρόχο με την εντολή break.

3.28 Συναρτήσεις

Οι συναρτήσεις είναι επαναχρησιμοποιούμενα μέρη προγραμμάτων και αποτελούν το πιο σπουδαίο δομικό στοιχείο μη στοιχειώδους προγράμματος. Μας επιτρέπουν να δίνουμε κάποιο όνομα σε ένα σύνολο εντολών, «τρέχοντας» εκείνο το σύνολο εντολών χρησιμοποιώντας το όνομά τους, όποτε θέλουμε, όσες φορές θέλουμε, οπουδήποτε μέσα στο πρόγραμμα και είναι γνωστές σα κλήση (calling) της συνάρτησης.

Με λίγα λόγια, μια συνάρτηση είναι ένα μπλοκ κώδικα που εκτελείται μόνο όταν καλείται. Μας επιτρέπει μια συνάρτηση να μεταβιβάζουμε δεδομένα, γνωστά ως παράμετροι και ακόμα ως αποτέλεσμα μπορεί να επιστρέψει δεδομένα.

Σε προηγούμενες ενότητες έχουμε δει ενσωματωμένες συναρτήσεις, όπως τη len και τη range.

Οι συναρτήσεις ορίζονται χρησιμοποιώντας τη λέξη κλειδί def, ακολουθώντας ένα όνομα που να ταυτοποιεί αυτή την συνάρτηση και στη συνέχεια ακολουθεί ένα ζευγάρι παρενθέσεων που μέσα περικλείονται ονόματα μεταβλητών. Αξίζει να επισημάνουμε ότι η γραμμή που συντάσσεται μια μεταβλητή τελειώνει με άνω και κάτω τελεία : .

3.29 Η συνάρτηση range ( )

Τη συνάρτηση range ( ) τη χρησιμοποιούμε για να παράγει μια αριθμητική πρόοδο, εάν χρειάζεται να μετακινηθούμε σε μια ακολουθία αριθμών.

Η συνάρτηση range ( ) επιστρέφει μια ακολουθία αριθμών, αρχίζοντας από το 0 από προεπιλογή και αυξάνεται κατά 1 και τελειώνει σε έναν καθορισμένο αριθμό.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε:

```
for x in range(4):
     print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
0
1
2
3
```

Προσοχή

Το εύρος (3) δεν σημαίνει τιμές από 0 έως 6, αλλά εννοούμε τις τιμές 0 έως 3.

Ας δούμε ένα παραδείγματα αναλυτικά.

3.30 Δημιουργία Συνάρτησης

Στην Python μια συνάρτηση ορίζεται χρησιμοποιώντας τη λέξη κλειδί def.

Για να καλέσουμε μια συνάρτηση, χρησιμοποιούμε το όνομα της συνάρτησης που θέλουμε γράφοντας αμέσως μετά παρένθεση.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε:

```
def sayHello():
     print (“Hello World!!!”)
# σύνολο εντολών που ανήκουν στη συνάρτηση
# Τέλος της συνάρτησης

sayHello()      # κλήση της συνάρτησης
sayHello() # κλήση της συνάρτησης πάλι
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Hello World!!!
Hello World!!!
```

Ας δούμε αναλυτικά πώς λειτουργεί το πρόγραμμα.

Αρχικά, ορίζουμε μια συνάρτηση με το όνομα sayHello και δεν δηλώνουμε καθόλου μεταβλητές, αφού η συνάρτηση δεν έχει παραμέτρους.

Οι παράμετροι στις συναρτήσεις είναι απλά η είσοδος στη συνάρτηση ώστε να μπορούμε να περνάμε διαφορετικές τιμές στη συνάρτηση και να παίρνουμε τα αντίστοιχα αποτελέσματα.

Σημείωση

Είναι εφικτό να καλούμε μια συνάρτηση δύο φορές, χωρίς να χρειάζεται να γράφουμε τον κώδικα δύο φορές.

3.31 Επιχειρήματα – (Arguments ή συντομευμένα arg)

Οι πληροφορίες είναι δυνατό να εκχωρηθούν σε συναρτήσεις ως ορίσματα. Τα ορίσματα καθορίζονται μετά το όνομα της συνάρτησης, μέσα σε παρενθέσεις και να υπάρχουν περισσότερα από ένα τα διαχωρίζουμε με κόμμα.

Μια παράμετρος είναι η μεταβλητή που παρατίθεται μέσα στις παρενθέσεις που ορίζεται μια συνάρτηση, ενώ ένα όρισμα είναι η τιμή που αποστέλλεται στην συνάρτηση όταν καλείται και μπορούν να χρησιμοποιηθούν και οι δύο για να μεταβιβάζουν πληροφορίες σε μια συνάρτηση.

Προσοχή

Μια συνάρτηση θα πρέπει να καλείται με τον σωστό αριθμό ορισμάτων, δηλαδή αν η συνάρτησή μας έχει δύο ορίσματα, τότε πρέπει να την καλέσουμε με δύο ορίσματα και όχι με περισσότερα ή λιγότερα γιατί αλλιώς θα μας εμφανίσει σφάλμα.

ΠΑΡΑΔΕΙΓΜΑ 2

Έχουμε μια συνάρτηση με ένα όρισμα (fname). Όταν η συνάρτηση καλείται, δίνουμε ένα όνομα (το μικρό), το οποίο χρησιμοποιείται στη συνάρτηση για να τυπώνουμε ολόκληρο το όνομα.

```
def myHello(fname):
     print (fname + “Refsnes”)
myHello(“Anna”)
myHello(“Zaxaraki”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Anna Refsnes
Zaxaraki Refsnes
```

ΠΑΡΑΔΕΙΓΜΑ 3

Έχουμε μια συνάρτηση η οποία αναμένει δύο ορίσματα και λαμβάνει δύο ορίσματα.

```
def myHello(fname, 1name):
     print(fname + “    “ + 1name)

myHello(“Anna”, “Zaxaraki”)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Anna Zaxaraki
```

Σημείωση

Χρησιμοποιούμε το * πριν από το όνομα μιας παραμέτρου στον ορισμό της συνάρτησης όταν δεν γνωρίζουμε πόσα ορίσματα θα περάσουν στη συνάρτησή μας και με αυτό τον τρόπο θα «λαμβάνει» ένα μπλοκ εντολών και έχοντας πρόσβαση στα στοιχεία η συνάρτηση.

Για παράδειγμα γράφουμε def myHello(*kids):

3.31.1 Λέξεις Κλειδιά (Keyword)

Μπορούμε να μεταφέρουμε ορίσματα με τη σύνταξη key = value, χωρίς να έχει σημασία η σειρά.

ΠΑΡΑΔΕΙΓΜΑ 4

```
def my_function(car3, car2, car1):
  print("Το ακριβότερο αυτοκίνητο είναι " + car3)

my_function(car1 = "Mercedes", car2 = "Nissan", car3 =
"Mercedes")
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Το ακριβότερο αυτοκίνητο είναι Mercedes
```

ΠΑΡΑΔΕΙΓΜΑ 5

Να γραφεί πρόγραμμα που να εμφανίζει σχετικό μήνυμα στην οθόνη τέσσερις χώρες καταγωγής:

```
def myHello(country = "Ελλάδα"):
  print("Κατάγομαι από " + country)

myHello("Κορέα")
myHello("Σερβία")
myHello()
myHello("Ιρλανδία")
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Κατάγομαι από Κορέα
Κατάγομαι από Σερβία
Κατάγομαι από Ελλάδα
Κατάγομαι από Ιρλανδία
```

ΠΑΡΑΔΕΙΓΜΑ 6

Είναι:

```
def my_function(travel):
  for x in travel:
    print(x)

country  = ["Κορέα", "Σερβία", "Ελλάδα", "Ιρλανδία"]

my_function(country)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Κορέα
Σερβία
Ελλάδα
Ιρλανδία
```

3.31.2 Return Values

Τη return τη χρησιμοποιούμε για να μπορέσει η συνάρτηση να μας επιστρέψει μια τιμή.

ΠΑΡΑΔΕΙΓΜΑ 7

```
def myHello(x):
     return 4 * x
print(myHello(2))
print(myHello(5))
print(myHello(10))
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
8
20
40
```

3.31.3 Function – pass

Οι ορισμοί δεν επιτρέπεται να είναι κενοί και για να αποφύγουμε τυχόν λάθη σε περίπτωση που έχουμε ένα κενό περιεχόμενο γράφουμε τη λέξη pass.

ΠΑΡΑΔΕΙΓΜΑ 8

```
def myHello():
     pass
# επειδή ο ορισμός της συνάρτησης είναι κενός και θα
δημιουργούσε ένα σφάλμα γράφουμε τη λέξη pass για να το
αποφύγουμε.
```

3.32 Λίστες

Η πιο χρήσιμη δομή δεδομένων για τον κβαντικό προγραμματισμό είναι οι λίστες. Οι λίστες αποτελούν κάποιους βασικούς τύπους δεδομένων που διαθέτει η Python και τις υποστηρίζει με πλήθος μεθόδων και συναρτήσεων.

Οι λίστες χρησιμοποιούνται για να αποθηκεύσουμε πολλά στοιχεία σε μία μόνο μεταβλητή, οι οποίες αποτελούν έναν από τους τέσσερις ενσωματωμένους τύπους δεδομένων στην Python για να μπορούμε να αποθηκεύουμε κάποια συλλογή δεδομένων.

Οι λίστες μπορεί να περιέχουν αντικείμενα διαφορετικών τύπων, διαχωριζόμενα με κόμμα και δημιουργούνται μέσα σε αγκύλες. Όπως τα αλφαριθμητικά έτσι και οι λίστες μπορούν να αναπροσαρμόζονται και να διαιρούνται σε τμήματα.

ΠΑΡΑΔΕΙΓΜΑ 9
Δημιουργήστε μια λίστα με τέσσερα φρούτα.
Έχουμε:

```
mylist=[“apple, “banana”, “cherry”]
     print(mylist)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[“apple”, “banana”, “cherry”]
```

3.33 Στοιχεία Λίστας

Τα στοιχεία της λίστας είναι ευρετηριασμένα, ταξινομημένα καθώς αλλάζουν και επιτρέπουν διπλότυπες τιμές. Το πρώτο στοιχείο έχει ευρετήριο [0], το δεύτερο έχει ευρετήριο [1] κ.λπ. Μια κενή λίστα συμβολίζεται με α = [ ].

Ταξινομημένες λίστες, είναι οι λίστες που τα στοιχεία της έχουν μια συγκεκριμένη σειρά και δεν πρόκειται να αλλάξει.

Όλες οι πράξεις επιστρέφουν τα ζητούμενα στοιχεία σε μια καινούργια λίστα.

Σε αντίθεση με τα string που είναι αμετάβλητα, οι λίστες είναι μεταβλητού τύπου, δηλαδή μπορεί το περιεχόμενό τους να αλλάξει, προσθέτοντας ή και αφαιρώντας στοιχεία στη λίστα αφού δημιουργηθεί. Αν χρειαστεί να προσθέσουμε νέα στοιχεία στη λίστα, τότε αυτά θα τοποθετηθούν στο τέλος της λίστας.

Το μέγεθος μιας λίστας μπορεί να αλλάξει ή ακόμα και να καταργηθεί εντελώς.

Οι λίστες μπορεί να είναι ομογενείς και ετερογενείς. Ομογενείς λίστες είναι εκείνες, οι οποίες περιέχουν στοιχεία του ίδιου τύπου (αριθμοί, ονόματα, string, κλπ.). Ετερογενείς λίστες είναι εκείνες οι οποίες περιέχουν στοιχεία διαφορετικού τύπου (αριθμοί και ονόματα και string, κ.λπ.).

3.34 Δομές δεδομένων

Ο τύπος δεδομένων μιας λίστας έχει κάποιες μεθόδους, που είναι οι παρακάτω.

• **append( )** - εισαγωγή νέου αντικειμένου στο τέλος της λίστας.
Π.χ. list.append(x) και ισοδυναμεί με: a[len(a) :] = [x] .

• **extend( )** - Επεκτείνει τη λίστα με την παράθεση όλων των στοιχείων στη συγκεκριμένη λίστα.
Π.χ. list.extend(L) και ισοδυναμεί με: a[len(a) :] = L .

• **insert(k,x)** - Τοποθέτηση ενός στοιχείου σε μια δεδομένη θέση, δηλαδή εισαγωγή του νέου αντικειμένου x στην k θέση της λίστας.

• **remove( )** - Διαγραφή αντικειμένου από τη λίστα.
Π.χ. list.remove(x): Αφαιρείται το πρώτο στοιχείο της λίστα του οποίου η αξία του είναι x και σε περίπτωση που δεν υπάρχει εμφανίζει σφάλμα.

• **clear( )** - διαγράφει όλα τα αντικείμενα της λίστας, δηλαδή ισοδυναμεί με del a [:]
Π.χ. list.clear( ).

• **pop( )** - διαγράφει το τελευταίο αντικείμενο της λίστας, ενώ
list.pop([i]) - Αφαιρεί το στοιχείο στη δεδομένη θέση στη λίστα και το επιστρέφει. Οι αγκύλες γύρο από το i , δηλώνουν ότι είναι προαιρετική η παράμετρος.

• **sort( )** - Ταξινομεί τα στοιχεία της λίστας σε αύξουσα διάταξη (εφόσον, περιέχει αντικείμενα ίδιου τύπου δεδομένων).
Π.χ. list.sort( ).

• **count( )** - Π.χ. list.count(x): Επιστρέφει τον αριθμό σχετικά με το πόσες φορές εμφανίζεται το x.

• **index( )** - list.index(x): Επιστρέφει το δείκτη στη λίστα του πρώτου στοιχείου με τιμή χ. Αν δεν υπάρχει τέτοιο στοιχείο εμφανίζει σφάλμα.

• **reverse( )** - Αντιστρέφει τα στοιχεία μιας λίστας σε σειρά.

• **copy( )** - Επιστρέφει ένα αντίγραφο της λίστας και ισοδυναμεί με: a [:] .

Ας δούμε κάποια παραδείγματα για τις παραπάνω μεθόδους.

ΠΑΡΑΔΕΙΓΜΑΤΑ 10

```
a = [55.16, 444, 444, 1, 6524.14]

print(a.count(444), a.count(1), a.count(55.16), a.count("x"))

Όταν το «τρέξουμε» θα μας εμφανίσει στην οθόνη:
2  1  1  0

a.insert(7, -2)
a.append(444)
a
Όταν το «τρέξουμε» θα μας εμφανίσει στην οθόνη:
[66.25, 333, -1, -1, 333, 1, 1234.5, -2, -5, 444, 444]
```

```
a.index(444)

Όταν το «τρέξουμε» θα μας εμφανίσει στην οθόνη:
9
```

```
a.remove(444)
a

Όταν το «τρέξουμε» θα μας εμφανίσει στην οθόνη:
[66.25, 333, -1, -1, 333, 1, 1234.5, -2, -5, 444]
```

```
a.reverse()
a
```

```
Όταν το «τρέξουμε» θα μας εμφανίσει στην οθόνη:
[444, -5, -2, 1234.5, 1, 333, -1, -1, 333, 66.25]
```

```
a.sort()
a

Όταν το «τρέξουμε» θα μας εμφανίσει στην οθόνη:
[-5, -2, -1, -1, 1, 66.25, 333, 333, 444, 1234.5]
```

ΠΑΡΑΔΕΙΓΜΑ 15

Δημιουργείστε λίστες που να επιτρέπουν διπλότυπα τιμές.

Έχουμε:

```
mylist= [“apple”, “banana”, “cherry”, “banana”]
print(mylist)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[“apple”, “banana”, “cherry”, “banana”]
```

• Μήκος Λίστας – len ( )

Η ενσωματωμένη συνάρτηση len( ) όπως προαναφέραμε γενικά επιστρέφει το μήκος μιας συμβολοσειράς και ισχύει και για τις λίστες για να προσδιορίζουμε πόσα στοιχεία έχει μία λίστα.

ΠΑΡΑΔΕΙΓΜΑ 16

Έχουμε:

```
x = “superpython”
len(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
11
```

ΠΑΡΑΔΕΙΓΜΑ 17

Να προσδιορίστε των αριθμό των στοιχείων της λίστας mylist = [“apple”, “banana”, “cherry”] και να τα εκτυπώσετε.
Έχουμε:

```
mylist= [“apple”, “banana”, “cherry”]
print(len(mylist))
#Θα μας δώσει το πλήθος των στοιχείων της λίστας.
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
3
```

• Τύπος Λίστας

Τα στοιχεία της λίστας όπως είπαμε μπορεί να είναι οποιαδήποτε τύπου δεδομένων, όπως συμβολοσειράς, int, boolean κλπ.

ΠΑΡΑΔΕΙΓΜΑ 18

Δημιουργήστε και εκτυπώστε μια λίστα που να περιέχει στοιχεία διαφορετικού τύπου δεδομένων, όπως συμβολοσειράς, int και boolean.

Έχουμε:

```
list1 = [“apple”, “banana”, “cherry”]
list2 = [5, 3, 1, 9, 7]
list3 = [False, True, True, False]
list4 =  [“apple”, 100, True, 30, “abc”, 2-3j]

print(list1)
print(list2)
print(list3)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[“apple”, “banana”, “cherry”]
```

```
[5, 3, 1, 9, 7]
[False, True, True, False]
[“apple”, 100, True, 30 “abc”]
```

• type ( ) – Λίστας

Όπως είδαμε σε προηγούμενες ενότητες, χρησιμοποιούμε type() για τον τύπο δεδομένων μιας λίστας.

• list ( ) – Λίστας

Χρησιμοποιούμε list( ) για να δημιουργήσουμε μία λίστα.

ΠΑΡΑΔΕΙΓΜΑ 19

Δημιουργήστε μια λίστα και τυπώστε την.

Έχουμε:

```
thislist = list((“apple”, “banana”, “cherry”))

#σημειώστε τις διπλές στρογγυλές αγκύλες

print(thislist)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[“apple”, “banana”, “cherry”]
```

• add ( ) – Λίστας

Μόλις δημιουργηθεί ένα σύνολο είδαμε ότι δεν μπορούμε να αλλάξουμε τα στοιχεία του, αλλά μπορούμε να προσθέσουμε νέα στοιχεία.

Για να προσθέσουμε ένα στοιχείο σε ένα σύνολο χρησιμοποιούμε την add( ).

ΠΑΡΑΔΕΙΓΜΑ 20

Έχουμε:

```
thisset- = {“apple”, “banana”, “cherry”}
mylist = [“kiwi”, “orangr”]
thisset.update(mylist)
print(thisset)
```

3.35 Πίνακες – (Arrays)

Οι πίνακες (arrays) είναι μια πολύτιμη δομή δεδομένων που αποτελεί ένα πολύ χρήσιμο εργαλείο για τον προγραμματισμό. Ένας πίνακας θα μπορούσαμε να πούμε ότι είναι μία ειδική μεταβλητή, η οποία μπορεί να κρατήσει πιο πολλές από μία τιμή τη φορά και συνήθως χρησιμοποιούνται για την αποθήκευση πολλαπλών τιμών σε μία μεμονωμένη μεταβλητή.

Η Python δεν έχει ενσωματωμένη υποστήριξη για πίνακες (Arrays), αλλά χρησιμοποιώντας τις λίστες (lists) με κατάλληλο τρόπο, εισάγοντας συγκεκριμένες βιβλιοθήκες, είναι δυνατόν να δημιουργήσουμε πίνακες όποιων διαστάσεων επιθυμούμε και να τους χρησιμοποιήσουμε εύκολα.

Οι πίνακες αφού στην ουσία είναι λίστες, διατηρούν όλες τις ιδιότητες των λιστών, όπως για παράδειγμα το πέρασμα των παραμέτρων σε συναρτήσεις ισχύει ακριβώς το ίδιο που είδαμε στην ενότητα με τις λίστες.

Το μεγάλο «όπλο» της Python για να αναπαραστήσουμε πίνακες είναι οι βιβλιοθήκες, οι οποίες μπορούν να ικανοποιήσουν απόλυτα την πιο απαιτητική εφαρμογή. Συγκεκριμένα θα χρησιμοποιήσουμε την πολύ γνωστή βιβλιοθήκη Numpy.

3.36 Δομή Δεδομένων Πινάκων

Η έννοια του μονοδιάστατου πίνακα (όπως είδαμε στο προηγούμενο κεφάλαιο), σχετίζεται άμεσα με την έννοια του διανύσματος στα μαθηματικά, αφού μπορούμε να αναπαραστήσουμε ένα διάνυσμα τιμών, με μια απλή λίστα.

Ας δούμε ένα παραδείγματα.

Έχουμε μία λίστα L = [2, 5, 7, 9] που ισοδυναμεί με έναν μονοδιάστατο πίνακα 4 θέσεων, με πρώτο στοιχείο το L[0], δεύτερο στοιχείο το L[1], τρίτο στοιχείο το L[3] κοκ.

Αρχικοποιούμε μια λίστα με μηδενικά N = 4 θέσεων και έχουμε:

A = [0 for I in range(N)]

Ένας δισδιάστατος πίνακας αποτελεί μια λίστα από λίστες και ισχύει ότι και στους μονοδιάστατους πίνακες προσαρμόζοντας ανάλογα. Ένας NxN πίνακας αποτελείται από μια λίστα N θέσεων και ορίζεται:

A = [0 for I in range(N) for j in range(N)]]

Περιέχει μέσα της N λίστες, N θέσεων η καθεμία και η προσπέλαση των στοιχείων του πίνακα γίνεται, με πρώτο στοιχείο στη θέση 0,0 να είναι το A[0[0] και το τελευταίο στοιχείο στην τελευταία γραμμή και στήλη να είναι το A[N-1][N-1].

Ας δούμε αναλυτικά με τη σειρά.

Ένας πίνακας είναι δυνατό να περιέχει αρκετές τιμές σε ένα όνομα (αποθηκευμένες), έτσι ώστε ανατρέχοντας διάφορα ευρετήρια αποκτάμε πρόσβαση στις τιμές αυτές. Αυτό μας εξυπηρετεί σημαντικά και μας δίνει λύση, όταν θέλουμε για παράδειγμα να κάνουμε μια αναζήτηση για να βρούμε κάτι συγκεκριμένο σε ένα ευρετήριο εγγραφών με 2.000 αυτοκίνητα και όχι με δύο τρία αυτοκίνητα.

Μια λίστα στοιχείων με ονόματα αυτοκινήτων, η αποθήκευσή τους σε μεμονωμένες μεταβλητές θα μπορούσε να είναι:

car1 = “Opel”
car2 = “Nissan”
car3 = “Ford”
car4 = “BMW”

ΠΑΡΑΔΕΙΓΜΑ 1

Να δημιουργήστε έναν πίνακα που να περιέχει κάποιες μάρκες αυτοκινήτων.

Έχουμε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
print(cars)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, τα ονόματα των αυτοκινήτων που καταχωρήσαμε:

```
[“Opel”, “Nissan”, “Ford”, “BMW”]
```

ΠΑΡΑΔΕΙΓΜΑ 2

Έχουμε:

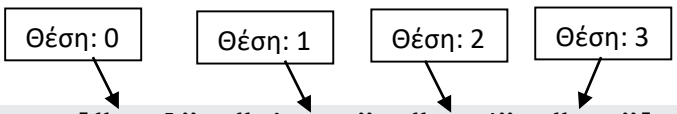
```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
x = cars [1]
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, το δεύτερο όνομα του αυτοκινήτου που καταχωρήσαμε κατά σειρά.

```
Nissan
x = cars [1]
```

Σημείωση

Ο αριθμός καταχώρησης στην Python ξεκινάει με τον αριθμό θέσης μηδέν. Η θέση 0 αντιστοιχεί στην καταχώρηση του αυτοκινήτου με το πρώτο όνομα που καταχωρήσαμε, δηλαδή Opel, η θέση 1 αντιστοιχεί στην καταχώρηση του αυτοκινήτου με όνομα Nissan, η θέση 2 αντιστοιχεί στην καταχώρηση του αυτοκινήτου με όνομα Ford, κλπ.




```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
```

3.37 Τροποποίηση καταχώρησης σε ένα πίνακα

Μπορούμε να αλλάξουμε σε έναν πίνακα το όνομά ενός στοιχείο του.

ΠΑΡΑΔΕΙΓΜΑ 3

Δημιουργείστε πρόγραμμα που να τροποποιήσουμε το όνομα του πρώτου στοιχείου του πίνακα cars, από Opel σε Toyota.

Τότε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
cars[0] = “Toyota”
print(cars)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, η καινούργια καταχώρηση με όνομα Toyota (αντί για Opel που ήταν προηγουμένως), στην πρώτη θέση του πίνακα.

```
cars = [“Toyota”, “Nissan”, “Ford”, “BMW”]
```

3.38 Ορισμός μήκους σε έναν πίνακα

Για να επιστρέψουμε το μήκος ενός πίνακα, δηλαδή τον αριθμό των στοιχείων ενός πίνακα, γράφουμε len( ).

ΠΑΡΑΔΕΙΓΜΑ 4

Δημιουργούμε έναν πίνακα cars, που να περιέχει τις μάρκες αυτοκινήτων, Opel, Nissan, Ford, BMW και να τυπώνεται στην οθόνη ο συνολικός τους αριθμός.

Έχουμε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
x = len(cars)
print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, το σύνολο των αυτοκινήτων που καταχωρήσαμε (σε αριθμό):

```
4
```

3.39 Στοιχεία πίνακα βρόγχου – for in

Μπορούμε να δημιουργήσουμε έναν βρόχο για όλα τα στοιχεία ενός πίνακα, χρησιμοποιώντας τον βρόχο for in.

ΠΑΡΑΔΕΙΓΜΑ 5

Δημιουργείστε βρόχο for in σε έναν πίνακα cars που να περιέχει τις μάρκες αυτοκινήτων, Opel. Nissan, Ford, BMW και κάθε στοιχείο του πίνακα cars να τυπώνονται στην οθόνη.

Έχουμε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
for x in cars:
print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, τα ονόματα των αυτοκινήτων που καταχωρήσαμε κάθετα:

```
Opel
Nissan
Ford
BMW
```

3.40 Προσθήκη στοιχείων σε πίνακα – Append ( )

Για να προσθέσουμε ένα στοιχείο σε έναν πίνακα (στο τέλος), χρησιμοποιούμε τη append() μέθοδο.

ΠΑΡΑΔΕΙΓΜΑ 6

Δημιουργείστε πρόγραμμα που να προσθέτει μία ακόμη μάρκα αυτοκινήτου στον cars πίνακα.

Έχουμε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”]
cars.append(“Toyota”)
print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, τα ονόματα των αυτοκινήτων από τον cars πίνακα προσθέτοντας στο τέλος του πίνακα το όνομα του αυτοκινήτου που ζητήσαμε να καταχωρηθεί:

```
[“Opel”, “Nissan”, “Ford”, “BMW”, “Toyota”]
```

3.41 Αφαίρεση στοιχείων από πίνακα – Pop ( ) & Remove ( )

Για να αφαιρέσουμε το τελευταίο στοιχείο σε έναν πίνακα, χρησιμοποιούμε τη pop ( ) ή τη remove ( ) μέθοδο.

ΠΑΡΑΔΕΙΓΜΑ 7

Δημιουργείστε πρόγραμμα που να αφαιρεί από τον cars πίνακα, μια καταχώρηση με τη pop( ) μέθοδο.

Έχουμε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”, “Toyota”]
cars.pop(1)
print(cars)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, τα ονόματα των αυτοκινήτων από τον cars πίνακα αφαιρώντας το στοιχείο από τη 2η θέση του πίνακα:

```
[“Opel”, “Ford”, “BMW”, “Toyota”]
```

ΠΑΡΑΔΕΙΓΜΑ 8

Δημιουργείστε πρόγραμμα που να αφαιρεί από τον cars πίνακα, μια καταχώρηση με τη remove( ) μέθοδο.

Έχουμε:

```
cars = [“Opel”, “Nissan”, “Ford”, “BMW”, “Toyota”]
cars.remove(“Ford”)
print(cars)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, τα ονόματα των αυτοκινήτων από τον cars πίνακα αφαιρώντας το στοιχείο από τη 2η θέση του πίνακα:

```
[“Opel”, “Nissan”, “BMW”, “Toyota”]
```

Ας θυμηθούμε συνοπτικά κάποιες μεθόδους ενός πίνακα που είναι είδαμε και είναι ίδιες και στις λίστες.

3.42 Μέθοδοι Πίνακα

Μέθοδος	Περιγραφή
append( )	Προσθέτει ένα στοιχείο στο τέλος του πίνακα.
clear( )	Καταργεί όλα τα στοιχεία από τη λίστα.
copy( )	Επιστρέφει ένα αντίγραφο της λίστας.
count( )	Επιστρέφει τον αριθμό των στοιχείων με την καθορισμένη τιμή.
extend( )	Προσθέτει τα στοιχεία μιας λίστας (ή οποιαδήποτε επαναληπτικού), στο τέλος της τρέχουσας λίστας.
index( )	Επιστρέφει το ευρετήριο του πρώτου στοιχείου με την καθορισμένη τιμή.
insert( )	Προσθέτει ένα στοιχείο στην καθορισμένη θέση.
pop( )	Αφαιρεί το στοιχείο στην καθορισμένη θέση.
remove( )	Καταργεί το πρώτο στοιχείο με την καθορισμένη τιμή.
reverse( )	Αντιστρέφει τη σειρά της λίστας.
sort( )	Ταξινομεί τη λίστα.

3.43 Η μαθηματική έννοια του πίνακα

Για να ορίσουμε έναν δισδιάστατο πίνακα, γράφουμε:

```
list = [[2, 5, 7, 9], [1, 0, 5, 4]]
print list [0] [2]        # τυπώνουμε γραμμή 0,   στήλη 2 == 7
print list[0]          # τυπώνουμε γραμμή 0 == [2, 5, 7, 9]
```

	0	1	2	3
0	**2**	**5**	**7**	**9**
1	**1**	**0**	**5**	**4**

3.43.1 Δισδιάστατα λίστα

```
list = [[2, 5, 7], [1, 0, 5]]
for i in range(2):
   for j in range(5):
        print list[i][j]
   print          # αρχίζει από την επόμενη γραμμή
```

Όταν το «τρέξουμε» θα εμφανιστεί στην οθόνη:

```
2 5 7
1 0 5
```

3.43.2 Πράξεις με ακολουθίες

Τελεστής	**Αποτέλεσμα**
<seq> + <seq>	Σύνδεση
<seq> * <int-expr>	Επανάληψη
<seq> [ ]	Δείκτης
len(<seq>)	Μήκος ακολουθίας
<seq>[:]	Slicing
for <var> in <seq>	Επανάληψη
<expr> in <seq>	Συμμετοχή (Boolean)

ΠΑΡΑΔΕΙΓΜΑ 9

Δημιουργείστε μια συνάρτηση η οποία όταν καλείται, να δημιουργεί ένα μονοδιάστατο πίνακα όσων θέσεων επιθυμούμε και να αποδίδεται σε όλα τα στοιχεία του ως αρχική τιμή το μηδέν.

Έχουμε:

```
def monodiast_array(size):
sqlist=[]
for x in range(0,size):
    splist.append(0)
return
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, τα ονόματα των αυτοκινήτων από τον cars πίνακα αφαιρώντας το στοιχείο από τη 2η θέση του πίνακα:

Ας εξηγήσουμε αναλυτικά το παράδειγμα.

Αρχικά, ορίζουμε μια συνάρτηση με παράμετρο size, την monodiast_array(size).

Με την splist =[ ] ορίζουμε μια κενή λίστα, η οποία θα είναι και ο πίνακάς μας.

Με την for x in range(0,size) επαναλαμβάνουμε το block των εντολών της «for…» μέχρι την τιμή της παραμέτρου, που είναι και το μέγεθος του πίνακα.

Ας δούμε ένα παράδειγμα ακόμη.

ΠΑΡΑΔΕΙΓΜΑ 10

Να γραφεί πρόγραμμα που να υπολογίζει και να τυπώνει στην οθόνη την προπαίδεια κάποιου αριθμού (από το 1 έως το 10).

Έχουμε:

```
arithmo=int(input("Δώσε τον αριθμό: "))
for i in range(1,10):
   print("Η προπαίδεια του",arithmo,"είναι:i,"*",arithmo,"= ",
i*arithmo)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
Δώσε τον αριθμό: 5

Η προπαίδεια του 5 είναι:  1 * 5 =  5
Η προπαίδεια του 5 είναι:  2 * 5 =  10
Η προπαίδεια του 5 είναι:  3 * 5 =  15
Η προπαίδεια του 5 είναι:  4 * 5 =  20
Η προπαίδεια του 5 είναι:  5 * 5 =  25
Η προπαίδεια του 5 είναι:  6 * 5 =  30
Η προπαίδεια του 5 είναι:  7 * 5 =  35
Η προπαίδεια του 5 είναι:  8 * 5 =  40
Η προπαίδεια του 5 είναι:  9 * 5 =  45
```

3.44 Βιβλιοθήκες numpy - matplotlib

Η βιβλιοθήκη numpy χρησιμοποιείται κυρίως για τη διαχείριση και την υποστήριξη μαθηματικών υπολογισμών μεταξύ πινάκων. Η βιβλιοθήκη αυτή παρέχει ένα σύνολο μεθόδων και συναρτήσεων που μας δίνουν τη δυνατότητα να διαχειριστούμε πίνακες πολλών διαστάσεων με ευχρηστία. Η δομή πινάκων επιτρέπει τις συνηθισμένες πράξεις μεταξύ διανυσμάτων στη γραμμική άλγεβρα, ενώ για τις λίστες αυτό δεν ισχύει. Στη numpy οι πράξεις του πολλαπλασιασμού (*), της διαίρεσης (/) καθώς και η ύψωση σε δύναμη (**), ορίζονται για πίνακες με συγκεκριμένο τρόπο, εφαρμόζοντας τις πράξεις μεταξύ κάθε δύο αντίστοιχων στοιχείων δύο πινάκων. Η ίδια διαδικασία ισχύει εφαρμόζοντας την κάθε πράξη σε κάθε στοιχείο του πίνακα πραγματοποιούνται και όλες οι πράξεις (+, -, *, /, **) μεταξύ των αριθμών και των πινάκων.

Η εισαγωγή της βιβλιοθήκης numpy, γίνεται γράφοντας:
import numpy as np

Οι πιο συνηθισμένες μονάδες της βιβλιοθήκης numpy είναι:

- Δημιουργία πίνακα μιας διάστασης,

  a=np.array([1,2,3,4])

- Δημιουργία πίνακα δύο διαστάσεων,
  b=np.array([[0,1,2],[3,4,5]])
- Εύρεση διαστάσεων πίνακα:
  a.ndim=1, b.ndim=2

- Εύρεση σχήματος πίνακα:
  b.nshape=(2,3)

- Πλήθος στοιχείων πίνακα:
  a.size=4, b.size=6

- Πράξεις με μονοδιάστατους πίνακες:
  A π B με π: +, -, *, /, %

- Αυτόματη δημιουργία πινάκων:
  - Μηδενικός πίνακας $A_{n \times m} \leftarrow$ np.zeros((n,m))
  - Πίνακας με όλα τα στοιχεία του 1: $A_{n \times m} \leftarrow$ np.ones((n,m))
  - Μοναδιαίος πίνακας (τετραγωνικός πίνακας με στοιχεία της κύριας διαγώνιου 1 και τα υπόλοιπα 0): $I_{n \times n} \leftarrow$ np.eye((n,n))

- Γινόμενο πινάκων: C=np.dot(A,B)

- Τανυστικό γινόμενο πινάκων: C=np.kron(A,B)

- Ανάστροφος πίνακας: B=A.transpose()

Η βιβλιοθήκη matplotlib χρησιμοποιείται για τη δημιουργία γραφικών παραστάσεων και περιέχει πληθώρα συναρτήσεων οι οποίες υποστηρίζουν και δίνουν τη δυνατότητα δημιουργίας πολλών διαφορετικών γραφικών παραστάσεων κάθε είδους, όπως ιστόγραμμα, ραβδόγραμμα, πίτα κλπ. καθώς και πολλαπλών γραφημάτων ενσωματωμένων στο ίδιο γράφημα. Ακόμα μπορούμε να δημιουργήσουμε τρισδιάστατα γραφήματα και με κίνηση (animation).

Η εισαγωγή της βιβλιοθήκης matplotlib, γίνεται γράφοντας:
import matplotlib .pyplot as plt

Οι πιο βασικές μονάδες της βιβλιοθήκης numpy είναι:

- plt.xlabel('Τίτλος του άξονα x'), Π.χ.:
  plt.xlabel('Ταχύτητα')
- plt.ylabel('Τίτλος του άξονα y'), Π.χ.:
  plt.ylabel('Χρόνος')
- plt.legend(synartiseis), Π.χ.:
  synartiseis=['y=2*x', 'y=x*x']
- plt.title('Τίτλος'), Π.χ.:
  plt.title('Κίνηση με σταθερή ταχύτητα')
- plt.grid( ): Για εμφάνιση πλέγματος σε ένα γράφημα
- plot3D(y,z,x): Για τρισδιάστατα γραφήματα
- plt.plot(x,y): Για προετοιμασία γραφήματος
- plt.show( ): Για εμφάνιση του γραφήματος

3.45 Πράξεις Πινάκων

• Πρόσθεση Πινάκων

Έτσι αν προσθέσουμε δύο array παίρνουμε ως αποτέλεσμα για τα διανύσματα αυτό που γνωρίζουμε από τα μαθηματικά.

ΠΑΡΑΔΕΙΓΜΑ 11

Να γραφεί πρόγραμμα που να υπολογίζει το άθροισμα των δύο πινάκων $[2 \quad 5 \quad 3]$, $[1 \quad 4 \quad 2]$ και να τυπώνει το αποτέλεσμά τους στην οθόνη.

Έχουμε:

```
import numy as np
```

```
x = np.array([2,5,3])
y = np.array([1,4,2])
print(x+y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, το άθροισμά των στοιχείων των δύο πινάκων:

```
[3 9 5]
```

Παρατήρηση

Για το παραπάνω παράδειγμα, η ίδια πράξη έχει διαφορετικό αποτέλεσμα για λίστες όπως είδαμε. Δηλαδή,

```
a = [2,5,3]
b = [1,4,2]
print(a+b)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[2, 5, 3, 1, 4, 2]
```

- Αφαίρεση Πινάκων

Αν θέλουμε τη διαφορά δύο array παίρνουμε ως αποτέλεσμα για τα διανύσματα αυτό που γνωρίζουμε από τα μαθηματικά.

ΠΑΡΑΔΕΙΓΜΑ 12

Να γραφεί πρόγραμμα που να υπολογίζει τη διαφορά των δύο πινάκων $[2 \quad 5 \quad 3]$, $[1 \quad 4 \quad 2]$ και να τυπώνει το αποτέλεσμά τους στην οθόνη.

Έχουμε:

```
import numy as np
x = np.array([2,5,3])
y = np.array([1,4,2])
print(x-y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, το άθροισμά των στοιχείων των δύο πινάκων:

```
[1 1 1]
```

Προσοχή

Η πράξη της αφαίρεσης για τις λίστες δεν ορίζεται.

• Πολλαπλασιασμός Πινάκων

Η πράξη του πολλαπλασιασμού δεν ορίζεται για τα διανύσματα, αλλά μόνο για πίνακες.

ΠΑΡΑΔΕΙΓΜΑ 13

Να γραφεί πρόγραμμα που να υπολογίζει το γινόμενο δύο πινάκων $[2 \quad 5 \quad 3]$, $[1 \quad 4 \quad 2]$ και να τυπώνει το αποτέλεσμά τους στην οθόνη.

Έχουμε:

```
import numy as np
x = np.array([2,5,3])
y = np.array([1,4,2])
print(x*y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, το αποτέλεσμα του γινομένου των αντίστοιχων στοιχείων των δύο πινάκων:

```
[2 20 6]
```

Παρατήρηση

Για το παραπάνω παράδειγμα, η ίδια πράξη έχει διαφορετικό αποτέλεσμα για λίστες όπως είδαμε. Δηλαδή,

```
a = [2,5,3]
x = a*3
print(x)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[2, 5, 3, 2, 5, 3, 2, 5, 3]
```

• Διαίρεση Πινάκων

ΠΑΡΑΔΕΙΓΜΑ 14

Να γραφεί πρόγραμμα που να υπολογίζει το γινόμενο δύο πινάκων $[2 \quad 5 \quad 3]$, $[1 \quad 4 \quad 2]$ και να τυπώνει το αποτέλεσμά τους στην οθόνη.

Έχουμε:

```
import numy as np
x = np.array([2,5,3])
y = np.array([1,4,2])
print(x/y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη, το αποτέλεσμα της διαίρεσης των αντίστοιχων στοιχείων των δύο πινάκων:

```
[2  1.25  1.5]
```

- Ύψωση σε δύναμη

Έχουμε:

```
import numy as np
x = np.array([2,5,3])
y = np.array([1,4,2])
print(x**y)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[2  625  9]
```

ΠΑΡΑΔΕΙΓΜΑ 15

Να γραφεί πρόγραμμα που να δημιουργήσουμε μια συνάρτηση η οποία να δέχεται ως όρισμα δύο διανύσματα και να ελέγχει αν αυτά είναι ορθογώνια και να τυπώνει στην οθόνη σχετικό μήνυμα.

Έχουμε:

```
A = np.array([[1,2], [3,4]])
x = np.array([1,0])
print(np.dot(A,x))
#Πολλαπλασιασμός πίνακα 2x2 με διάνυσμα δίνει διάνυσμα

x = np.array([[1],[0]])
print(np.dot(A,x)
#Πολλαπλασιασμός πίνακα 2x2 με πίνακα 2x1 δίνει διάνυσμα
πίνακα 2x1
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[1 3]
[[1]
  [3]]
```

Σημείωση

Η πράξη του πολλαπλασιασμού θα πρέπει να είναι δυνατόν να πραγματοποιηθούν και οι διαστάσεις των πινάκων να είναι συμβατές.

ΠΑΡΑΔΕΙΓΜΑ 16

Έχουμε:

```
x = np.eye
A = np.array([[1,2], [3,4]])
x = np.array([1,0])
print(np.dot(A,x))
#Πολλαπλασιασμός πίνακα 2x2 με διάνυσμα δίνει διάνυσμα

x = np.array([[1],[0]])
print(np.dot(A,x)
#Πολλαπλασιασμός πίνακα 2x2 με πίνακα 2x1 δίνει διάνυσμα
πίνακα 2x1
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[1 3]
[[1]
  [3]]
```

- Ειδικοί Πίνακες
  - Για πίνακα διαστάσεων N x M,… με όλα τα στοιχεία του μηδέν, γράφουμε np.zeros(N,M,..)).
  - Για πίνακα διαστάσεων N x M,… με όλα τα στοιχεία του ένα, γράφουμε np.ones((N,M,..)).
  - Για πίνακα δύο διατάσεων N x M όπου στην κύρια διαγώνιο έχουμε τον αριθμό 1, τότε γράφουμε np.eye(N,M,k) και ένα στην κύρια διαγώνιο (default k: κύρια διαγώνιο).
  - Για τυχαίες τιμές στο διάστημα ο έως 1 γράφουμε, np.random.random((N,M,…)).

ΠΑΡΑΔΕΙΓΜΑ 17

Να δημιουργηθεί πίνακας 3x4 με τυχαίες ακέραιες τιμές στο διάστημα από 10 έως 100.

Είναι:

```
A = np.random.random((3,4))
A = A * 90 + 10
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[1 3]
[[1]
  [3]]
```

3.46 Εσωτερικό Γινόμενο

Για να ορίσουμε το εσωτερικό γινόμενο μεταξύ διανυσμάτων (μόνο για μονοδιάστατους πίνακες), θα πρέπει να χρησιμοποιήσουμε την εντολή **dot** της Numpy.

Μπορούμε με δύο τρόπους να γράψουμε τον κώδικα.

ΠΑΡΑΔΕΙΓΜΑ 18

Να γραφεί πρόγραμμα που να υπολογίζει το εσωτερικό γινόμενο των πινάκων $[1 \quad 2 \quad 3]$, $[4 \quad 5 \quad 6]$ και να τυπώνει το αποτέλεσμά τους στην οθόνη.

Α΄Τρόπος

```
x = np.array([1,2,3])
y = np.array([4,5,6])
a = np.dot(x,y)
print(a)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
32
```

Β΄Τρόπος

```
x = np.array([1,2,3])
y = np.array([4,5,6])
a = x.dot(y)
print(a)
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
32
```

ΠΑΡΑΔΕΙΓΜΑ 19

Να γραφεί πρόγραμμα που να εμφανίζει στην οθόνη τους μονοδιάστατους πίνακες $[0 \quad 1 \quad 2]$, $[0 \quad 1 \quad 2]$, $[0 \quad 1 \quad 2]$, τον ένα κάτω από τον άλλο

Έχουμε:

```
array = [[1,2,3], [1,2,3], [1,2,3]]
for i in range(3):
   print(array[i])
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[1,2,3]
[1,2,3]
[1,2,3],
```

ΠΑΡΑΔΕΙΓΜΑ 20

Να γραφεί πρόγραμμα που να εμφανίζει στην οθόνη τους μονοδιάστατους πίνακες [0 1 2], [3 4 5], [6 7 8], τον ένα κάτω από τον άλλο

Έχουμε:

```
array = [[1,2,3], [4,5,6], [7,8,9]]
count=0
for i in range(3):
    temp=[]
    for j in range(count)
    count + = 1
    array.append(temp)
   for I in range(3):
      print(array[i])
cpll=[row[1] for row in array]
print coll
```

Όταν το εκτελέσουμε θα εμφανιστεί στην οθόνη:

```
[1,4,7]
```

3.46.1 Μετατροπή Μοίρες σε radians

Για να μετατρέψουμε μια γωνία x σε radians, γράφουμε radians(x).

3.46.2 Υπολογισμός Ημιτόνου – Συνημίτονο – Εφαπτομένη (σε radians)

- Η συνάρτηση για την εύρεση του ημιτόνου (sin) είναι η math.sin(x), η οποία επιστρέφει το ημίτονο των x radians.
- Η συνάρτηση για την εύρεση του συνημίτονου (cos) είναι η math.cos( ), η οποία επιστρέφει το συνημίτονο των x radians.
- Η συνάρτηση για την εύρεση της εφαπτομένης (tan) είναι η math.tan( ), η οποία επιστρέφει την εφαπτομένη των x radians.

ΠΑΡΑΔΕΙΓΜΑ 21

Υπολογισμός ημίτονο, συνημίτονο και εφαπτομένη 30 μοιρών (σε radians).

Έχουμε:

```
sin(30)=math.sin(math.radians(30))
print(sin(30))
cos(30)=math.cos(math.radians(30))
print(cos(30))
tan(30)=math.tan(math.radians(30))
print(tan(30))
```

3.47 Βιβλιοθήκες Qutip – Qiskit

Η βιβλιοθήκη qutip χρησιμοποιείται για την περιστροφή της σφαίρας Bloch, την απεικόνιση διανυσμάτων κλπ. που θα δούμε στα επόμενα κεφάλαια, τα οποία σχετίζονται με την Κβαντική Υπολογιστική και τον κβαντικό προγραμματισμό.

Επίσης, η βιβλιοθήκη Qiskit αποτελεί ένα πολύτιμο και σπουδαίο εργαλείο καθώς μας δίνει τη δυνατότητα να προγραμματίσουμε πολύπλοκους κβαντικούς αλγορίθμους. Αναπτύχθηκε από την IBM και αρκετοί ερευνητές και επιστήμονες της αποδίδουν τον χαρακτηρισμό «ο ελβετικός σουγιάς του κβαντικού προγραμματισμού». Στην Qiskit θα αναφερθούμε και θα ασχοληθούμε περισσότερο στο επόμενο κεφάλαιο.

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Εισάγετε το τμήμα του κώδικα που λείπει παρακάτω για να εμφανιστεί το κείμενο “Hello, World!”

```
(“Hello, World!”)
```

2) Να γράψετε τον ειδικό χαρακτήρα για σχόλια στην Python, στην πρόταση:

```
Αυτό είναι ένα σχόλιο
```

3) Δημιουργήστε μια μεταβλητή με όνομα carname και εκχωρήστε την τιμή Volvo σε αυτήν.

```
= ‘ ‘
```

4) Στα παρακάτω προγράμματα παρατηρήστε τι τύπο δεδομένων του x θα εκτυπώσει αντίστοιχα για το καθένα και ποιος τύπος δεδομένων θα ήταν αυτός;

```
x=5
print (int(x))
```

```
x=20.5
print (type(x))
```

```
x=True
print (type(x))
```

5) Στα παρακάτω προγράμματα εισάγετε τη σωστή σύνταξη για το καθένα αντίστοιχα για να μετατρέψετε το x σε:

- Ακέραιο αριθμό,

```
x=10.5
x=      (x)
```

- Σε αριθμό κινητής υποδιαστολής,

```
x=10
x=      (x)
```

- Σε μιγαδικό αριθμό,

```
x=10
x=      (x)
```

6) Ελέγξτε τις παρακάτω δηλώσεις και γράψτε αν θα εκτυπώσει μια Boolean τιμή, True ή False.

```
print (20 > 8)
```

```
print (20 = =  8)
```

```
print (bool (‘ abc ’) )
```

```
print (bool ( 0 ) )
```

7) Να γράψετε κώδικα, ο οποίος να εκτελεί και να τυπώνει το αποτέλεσμα των παρακάτω πράξεων:

- Πολλαπλασιάστε το 10 με το 5.

- Διαιρέστε το 10 με το 2.

- Να υπολογίζει το εμβαδό τετραγώνου πλευράς ίσο με α = 2 (Το εμβαδό τετραγώνου ισούται με α^2).

8) Συμπληρώστε το κενό που λείπει, στον παρακάτω κώδικα, ώστε να ελέγχει ένα υπάρχει “μήλο” στο fruits αντικείμενο.

```
fruits = [ “apple” , “banana”]
if “apple”          fruits:
    print (“Yes, apple is a fruit!”)
```

9) Συμπληρώστε το κενό που λείπει, με τον σωστό τελεστή σύγκρισης, στον παρακάτω κώδικα, ώστε να ελέγχει ένα υπάρχει “μήλο” στο fruits αντικείμενο.

```
fruits = [ “apple” , “banana”]
if “apple”          fruits:
    print (“Yes, apple is a fruit!”)
```

10) Συμπληρώστε το κενό που λείπει, με τον σωστό τελεστή σύγκρισης, στον παρακάτω κώδικα, ώστε να ελέγχει εάν το 5 δεν ισούται με το 10.

```
if  5         10:
    print (“Ο αριθμός 5 και 10 δεν είναι ίσοι”)
```

11) Συμπληρώστε το κενό που λείπει, με τον σωστό λογικό τελεστή, στον παρακάτω κώδικα, ώστε να ελέγχει εάν μία από τις δύο προτάσεις είναι True.

```
if  5 = = 10:         4 = = 4 :
    print (“Τουλάχιστον μία από τις δηλώσεις είναι αληθής”)
```

12) Στο μάθημα του προγραμματισμού του τμήματος Ψηφιακών Συστημάτων, έχουμε 1000 εγγεγραμμένους φοιτητές και στις εξετάσεις παρευρίσκονται οι 400 από αυτούς. Οι βαθμολογίες καταχωρούνται με τη βοήθεια της Python. Για να καλύψουμε τη βαθμολογία και των 1000 φοιτητών, γράψτε όλους τους τύπους μεταβλητών που θα χρειαστούμε.

13) Γράψτε κάποια παραδείγματα περιπτώσεων που θα ήταν χρήσιμοι οι τύποι int, float, bool, NoneType:

14) Συμπληρώστε το κενό που λείπει, στον παρακάτω κώδικα, για κάθε περίπτωση:

- Εκτυπώστε “Hello World” εάν a είναι μεγαλύτερο από το b.

```
a=80
b=20
    a          b:
    print(“Hello World”)
```

- Εκτυπώστε “Hello World” εάν a δεν είναι ίσο με b.

```
a=80
b=20
    a          b:
    print(“Hello World”)
```

- Εκτυπώστε “Ναι” εάν a είναι ίσο με b, διαφορετικά εκτυπώστε “Όχι”.

```
a=80
b=20
    a = = b:
    print(“Ναι”)
```

```
    print(“Όχι”)
```

- Εκτυπώστε “Hello” εάν a είναι ισούται με b και c ισούται με d.

```
if  a = = b        c = = d:
    print(“Hello”)
```

- Εκτυπώστε “Hello” εάν a είναι ϊσο με b, ή εάν c είναι ίσο με d.

```
if  a = = b        c = = d:
    print(“Hello”)
```

15) Συμπληρώστε το κενό που λείπει, στον παρακάτω κώδικα, για κάθε περίπτωση:

- Εκτυπώστε i εφόσον i είναι μικρότερο από 6.

```
i = 1
      i < 6:
    print(i)
    i + = 1
```

- Σταματήστε το βρόχο εάν i είναι ίσο με 3.

```
i = 1
      i < 6:
    if  i = = 3:

    i + = 1
```

- Όταν το i ισούται με 3 στον βρόχο, μεταβείτε απευθείας στην επόμενη επανάληψη

```
i = 0
while  i < 6:
    i + = 1
    if  i = = 3:

    print(i)
```

- Εκτυπώστε ένα μήνυμα όταν η συνθήκη είναι False.

```
i = 1
while  i < 6:
    print(i)
    i + = 1
    :
    print (“Δεν είναι μικρότερο από 6”)
```

16) Να γράψετε πρόγραμμα που να εμφανίζει το όνομα του μικρότερου παιδιού από πέντε ονόματα, χρησιμοποιώντας τη συνάρτηση def, myName και child1, child2, child3, child4, child5 g=για να καταχωρήσετε τα πέντε ονόματα αντίστοιχα.

17) Να γραφεί πρόγραμμα που να διαβάζει την ηλικία ενός ανθρώπου. Αν είναι κάτω των 18 ετών, να εμφανίζει ένα μήνυμα με τη λέξη «ΑΝΗΛΙΚΟΣ». Αν είναι πάνω από 18 ετών, να εμφανίζει μήνυμα με τη λέξη «ΕΝΗΛΙΚΟΣ» και τέλος, αν είναι άνω των 70 ετών, να εμφανίζει τη λέξη «ΗΛΙΚΙΩΜΕΝΟΣ».

18) Να γράψετε πρόγραμμα που να εμφανίζει τέσσερα ονόματα φρούτων, χρησιμοποιώντας τη συνάρτηση def_function(food) και fruits για τα φρούτα της αρεσκείας σας.

19) Να γραφεί πρόγραμμα που να δίνεται το μήκος και το πλάτος ενός ορθογωνίου παραλληλογράμμου και στη συνέχεια να υπολογίζει και να εμφανίζει την περίμετρο του.

20) Να γραφεί πρόγραμμα που να υπολογίζει και να εμφανίζει την απόλυτη τιμή ενός αριθμού.

21) Να γραφεί πρόγραμμα που να ζητείται και εμφανίζει την προπαίδεια ενός ζυγού αριθμού από το 1 μέχρι το 10.

22) Να γραφεί πρόγραμμα που να δημιουργεί και να εμφανίζει με κλήση μιας συνάρτησης με παράμετρο1, να επιστρέφει το άθροισμα των ακεραίων αριθμών από το 1 μέχρι το 100.

23) Να γραφεί πρόγραμμα που να δημιουργεί και να εμφανίζει μια λίστα με όνομα nums με ακέραιες τιμές από 0έως 100. Ακόμα, να δημιουργεί και να εμφανίζει μια λίστα με όνομα odds που να περιέχει περιττούς αριθμούς της λίστας nums τιμές από 0 έως 100.

24) Να γραφεί πρόγραμμα που να εισάγουμε 200 ονόματα φοιτητών σε μια λίστα classList.

25) Να δημιουργήστε συνάρτηση, η οποία όταν την καλείτε, να εμφανίζει το μέγιστο στοιχείο ενός μονοδιάστατου πίνακα καθώς και τη θέση του μέγιστου στοιχείου του.

26) Να δημιουργήσετε συνάρτηση, η οποία όταν την καλείτε, να ταξινομεί ένα μονοδιάστατο πίνακα.

27) Να γραφεί πρόγραμμα το οποίο να υπολογίζει και να εμφανίζει στην οθόνη το άθροισμα των άρτιων αριθμών από 100 έως και 1000.

28) Να υπολογιστεί το άθροισμα των ακέραιων αριθμών από το 5 μέχρι το 50 με βήμα 5.

29) Να γραφεί πρόγραμμα που να υπολογίζει και να εμφανίζει στην οθόνη τους τυχερούς αριθμούς του Τζόκερ. Από τους 45 αριθμούς κληρώνονται μόνο 5 αριθμοί που βρίσκονται μέσα στην κληρωτίδα και από τους 20 αριθμούς Τζόκερ κληρώνεται 1 αριθμός που βρίσκεται μέσα σε άλλη κληρωτίδα.

30) Να γραφεί πρόγραμμα το οποίο να διαβάζει από την οθόνη, το επώνυμο ενός φοιτητή, την βαθμολογία του σε ένα μάθημα, ελέγχοντας ότι ο βαθμός παίρνει τιμές στο διάστημα από μηδέν έως και 10. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να δοθεί ως επώνυμο μαθητή η λήξε “Τέλος”.

31) Να δημιουργήσετε μία συνάρτηση η οποία να υπολογίζει αν ένας πίνακας είναι άνω τριγωνικός.

32) Να δημιουργήσετε μία συνάρτηση η οποία να υπολογίζει αν ένας πίνακας είναι άνω τριγωνικός.

33) Να δημιουργήσετε μία συνάρτηση η οποία παίρνει σαν είσοδο έναν τετραγωνικό πίνακα 3x3 και να υπολογίζει την ορίζουσά του.

34) Να γράψετε πρόγραμμα και μία συνάρτηση οποία να ταξινομεί τις γραμμές ενός δισδιάστατου πίνακα με βάση το πρώτο στοιχείο της γραμμής. Αν τα πρώτα στοιχεία δύο πρώτων γραμμών είναι ίσα, τότε να ελέγχει τα δεύτερα στοιχεία, αν είναι και αυτά ίσα, τότε να ελέγχει τα τρίτα κοκ.

35) Να γραφεί πρόγραμμα που να διαβάζει το μήκος της ακτίνας ενός κύκλου και να τυπώνει τη διάμετρο, το μήκος και το εμβαδόν αυτού του κύκλου. Η διάμετρος του κύκλου δίνεται από τον τύπο d=2*r, η περίμετρος ισούται με p=2*π*r και το εμβαδόν με $E=\pi*r^2$.

36) Να γραφεί πρόγραμμα όπου για τους πίνακες (στήλη), $A = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ και $A = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, να υπολογίζει:

α) Το άθροισμα των πινάκων Α + Β, Β + Γ.

β) Το γινόμενο των πινάκων Α • Γ, Γ • Β.

γ) Το γινόμενο 3 • Α, 2 • Γ και -1 • (Α • Γ).

37) Να γραφεί πρόγραμμα όπου για τους πίνακες Α = $\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$, Β = $\begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix}$, Γ = $[2 \quad 0 \quad -1]$ και Δ = $[-1 \quad 0 \quad 1]$, να υπολογίζει το τανυστικό γινόμενο των: i. Α $\otimes$ Β, ii. Α $\otimes$ Γ και iii. Β $\otimes$ Δ.

38) Να γραφεί πρόγραμμα που να προσδιορίζει τα x, y ϵ $\mathbb{R}$, ώστε οι παρακάτω μιγαδικοί αριθμοί:

i.(x + 2y) + (4x - 3y)i και 1-2i να είναι ίσοι.

ii. z_1= (4x - y) - 3i και z_2 = - 1 + 2x - 3y)i να είναι συζυγείς.

39) Να γραφεί πρόγραμμα ώστε να υπολογίζει τα μέτρα των παρακάτω μιγαδικών αριθμών:

i. z_1 = 2 + 3i, ii. z_2 = - 1 και iii. z_3 = 6i.

40) Να γραφεί πρόγραμμα που να υπολογίζει και να εμφανίζει στην οθόνη, το ημίτονο, το συνημίτονο και την εφαπτομένη των 30, 45 και 90 μοιρών.

ΚΕΦΑΛΑΙΟ: 4ο - Εισαγωγή στον κβαντικό προγραμματισμό

ΚΕΦΑΛΑΙΟ 4ο

Εισαγωγή στον Κβαντικό Προγραμματισμό

Σύνοψη

Σε αυτό το κεφάλαιο αρχικά, γίνεται μια αναφορά στις βιβλιοθήκες του κβαντικού προγραμματισμού, δίνεται η ερμηνεία τι είναι το Qiskit, γιατί χρησιμοποιούμε Qiskit και ακολουθούν σχετικές οδηγίες για την εγκατάσταση της Qiskit. Περιγράφονται οι βασικές εντολές και η σύνταξη του Qiskit που θα χρησιμοποιήσουμε και θα εισάγουμε σε αυτό το κεφάλαιο τη δημιουργία απλών κβαντικών προγραμμάτων. Στη συνέχεια, επισημαίνεται η διαφορά του bit με το qubit, τι είναι κβαντικός καταχωρητής και παρουσιάζονται τα κβαντικά συστήματα δύο καταστάσεων, οι βασικές τους καταστάσεις, η έννοια της υπέρθεσης και οι ορισμοί του συμβολισμού Dirac. Επιπλέον, γίνεται ανάλυση των τελεστών και των δράσεων τους στα διανύσματα κατάστασης και προβάλλονται οι κβαντικές πύλες που δρουν σε ένα qubit (Hadamard, Αδράνειας) και σε κβαντικούς καταχωρητές των δύο qubits (CNOT, SWAP). Τέλος, δίνονται εφαρμογές, λυμένα παραδείγματα και προγράμματα καθώς και ασκήσεις για λύση από το μαθητή, για την αποτελεσματική εκμάθηση και εμπέδωσή του σε απαραίτητες γνώσεις ανάπτυξης κβαντικών προγραμμάτων. Αυτό το κεφάλαιο απευθύνεται σε μαθητές Γ' Γυμνασίου καθώς και μαθητές Λυκείου.

4.1 Βιβλιοθήκες Κβαντικού Προγραμματισμού

Όπως αναφέραμε στο προηγούμενο κεφάλαιο, η Python είναι μία από τις λίγες γλώσσες προγραμματισμού που υποστηρίζει τον κβαντικό προγραμματισμό έμμεσα με ειδικές βιβλιοθήκες. Τα τελευταία χρόνια ισχυρές εταιρίες όπως η IBM, η Google, η Microsoft, η Riggetti και άλλες, έχουν αναπτύξει πολλές βιβλιοθήκες κβαντικού προγραμματισμού, μερικές από τις οποίες είναι η Qiskit, η Circ και η pyQuil. Η βιβλιοθήκη Circ της Python η οποία αναπτύχθηκε από την Google, μπορεί να εκτελέσει κβαντικά προγράμματα όχι μόνο σε περιβάλλον του Jupyter notebook. Η βιβλιοθήκη pyQuil της οποίας η λογική ανάπτυξη των προγραμμάτων της μοιάζει αρκετά με της Qiskit, με κάποιες διαφορές στη σύνταξη.

Εμείς θα ασχοληθούμε με το Qiskit.

4.2 Γιατί Qiskit

Οι βιβλιοθήκες κυκλωμάτων στην ουσία είναι μια συλλογή από κυκλώματα, πύλες και οδηγίες. Αυτά τα κυκλώματα μπορούν να χρησιμεύσουν και ως δομικά στοιχεία για αλγορίθμους και μπορούν να προσομοιωθούν κλασικά. Κάθε στοιχείο μπορεί να συνδεθεί σε ένα κύκλωμα επιτρέποντας στους χρήστες να προγραμματίζουν σε υψηλότερα επίπεδα.

Επομένως, θα χρησιμοποιήσουμε τη βιβλιοθήκη Qiskit, γιατί αυτή μας επιτρέπει να κατασκευάσουμε κβαντικά προγράμματα και συμπεριλαμβάνει εξομοιωτή για να «τρέχουμε» τα προγράμματά μας τοπικά, σε πραγματικούς κβαντικούς υπολογιστές.

Εικόνα: 1 – Λογότυπο Qiskit

Πηγή εικόνας: https://github.com/Qiskit/qiskit.org

4.3 Τι Είναι το Qiskit

Το Qiskit (Quantum Information Science Kit) αποτελεί ένα τεράστιο εργαλείο ανάπτυξης κβαντικού λογισμικού καθώς μας δίνει τη δυνατότητα να προγραμματίσουμε πολύπλοκους κβαντικούς αλγορίθμους. Αναπτύχθηκε από την IBM και κυκλοφόρησε το Δεκέμβρη του 2018. Επειδή οι κβαντικοί υπολογιστές είναι δύσκολο να βρεθούν, μπορούμε να χρησιμοποιήσουμε μέσω online πλατφόρμας, έναν πάροχο cloud, όπως το κιτ εργαλείων Qiskit της IBM Q. Η IBM Quantum experience είναι μια βιομηχανία, μια υπηρεσία της IBM που παρέχει τη δυνατότητα στο χρήστη να «τρέχει» τα προγράμματά του σε έναν πραγματικό κβαντικό υπολογιστή της IBM. Δημιουργήθηκε με σκοπό την κατασκευή κβαντικών υπολογιστών για επιχειρήσεις αλλά και για επιστημονική έρευνα, η οποία παρέχει τη δυνατότητα στους χρήστες να έχουν πρόσβαση (μέσω του cloud) στους κβαντικούς επεξεργαστές της. Επιπλέον παρέχει ένα διαδικτυακό εκπαιδευτικό υλικό και ένα φόρουμ για συζήτηση, σχετικά με θέματα κβαντικής υπολογιστικής και προγραμματισμό. Το Qiskit αποτελείται από μια συλλογή προγραμμάτων και εργαλείων λογισμικού, που χρησιμοποιούνται από προγραμματιστές για την δημιουργία λογισμικού για συγκεκριμένες πλατφόρμες. Είναι ένα λογισμικό ανοιχτού

κώδικα, δηλαδή το λογισμικό του παρέχεται δωρεάν και ελεύθερα. Αυτό το ψηφιακό εγχειρίδιο, χρησιμοποιείται για τη διευκόλυνση της μελέτης των εννοιών του κβαντικού υπολογισμού, παρέχοντας εργασίες που σχετίζονται με κβαντικούς υπολογιστές, σε επίπεδο παλμών, κυκλωμάτων και λειτουργικών μονάδων εφαρμογής.

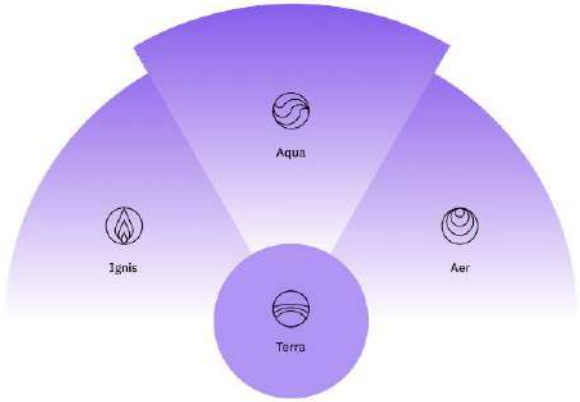
4.4 Τι Μπορεί να Κάνει το Qiskit

Το Qiskit διευκολύνει και επιτρέπει στους χρήστες να σχεδιάζουν κβαντικά προγράμματα και κυκλώματα υλοποιώντας τα σε προσομοιωτές αλλά ακόμα και σε αληθινούς υπολογιστές. Περιλαμβάνει ένα ολοκληρωμένο σύνολο κβαντικών πυλών και μια ποικιλία πολύτιμων προκατασκευασμένων κυκλωμάτων, ώστε να μπορούν οι χρήστες κάθε επιπέδου να εξερευνήσουν τις δυνατότητές της καθώς και να χρησιμοποιήσουν εύκολα και αποτελεσματικά την εφαρμογή αυτή, για έρευνα και ανάπτυξη εφαρμογών. Θα κατανοήσουμε καλύτερα τις δυνατότητες της βιβλιοθήκης Qiskit κατά την πορεία αυτού του κεφαλαίου.

4.5 Qiskit Elements

Το Qiskit αποτελείται από τέσσερα θεμελιώδη στοιχεία, βασισμένα στα τέσσερα στοιχεία της φύσης:

- Terra (Γη): η βάση κώδικα, για τη σύνθεση κβαντικών προγραμμάτων σε επίπεδο κυκλωμάτων και παλμών.
- Aqua (Νερό): για την κατασκευή αλγορίθμων και εφαρμογών.
- Ignis (Φωτιά): για αντιμετώπιση θορύβου και σφαλμάτων.
- Aer (Αέρας): για την επιτάχυνση της ανάπτυξης μέσω προσομοιωτών εξομοιωτών και εντοπισμών σφαλμάτων.



Εικόνα: 2 – Θεμελιώδη στοιχεία του Qiskit

Πηγή εικόνας: https://medium.com/@rajneesh.aggarwal/quantum-computing-using-qiskit-ibm-q-experience-and-python-2fea95dbf2e9

4.6 Ρύθμιση περιβάλλοντος Qiskit

Το Qiskit υποστηρίζει Python 3.7 ή νεότερη έκδοση. Εμείς χρησιμοποιούμε το Jupyter Notebook το οποίο περιλαμβάνεται στο Anaconda και συνιστάται για αλληλεπίδραση με το Qiskit καθώς προτείνεται κυρίως για επιστημονικές εφαρμογές.

Το Qiskit υποστηρίζεται στα 64-bit συστήματα (Windows – Uduntu 16.04 και macOs 10.12.6).

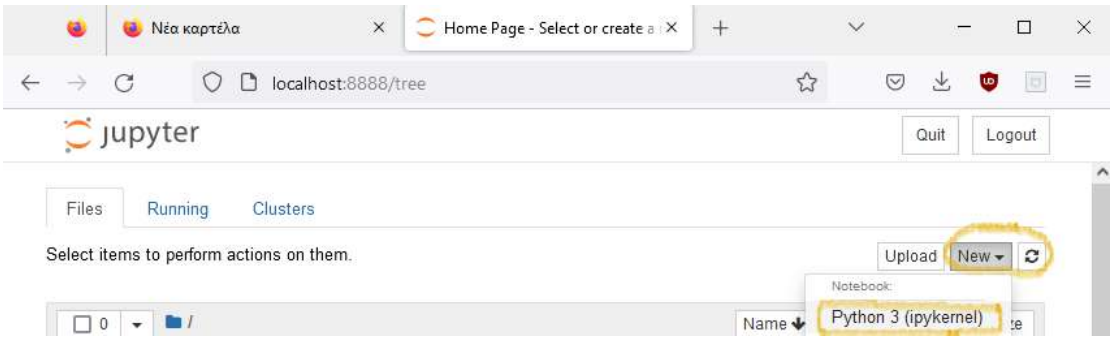
Για να εγκαταστήσουμε το πακέτο Qiskit, ακολουθούμε τις οδηγίες που αναφέρονται στο παρακάτω link:

https://qiskit.org/documentation/getting_started.html .

Για να ελέγξουμε ότι έγινε σωστά η εγκατάσταση (όπου αναφέραμε στο προηγούμενο κεφάλαιο) ακολουθούμε τις παρακάτω ενέργειες:

- ✓ Πληκτρολογούμε στη γραμμή εντολών Jupyter notebook και μας μεταφέρει στον φυλλομετρητή (browser).

- ✓ Επιλέγουμε ⟶ new ⟶ Python, όπως φαίνεται παρακάτω:



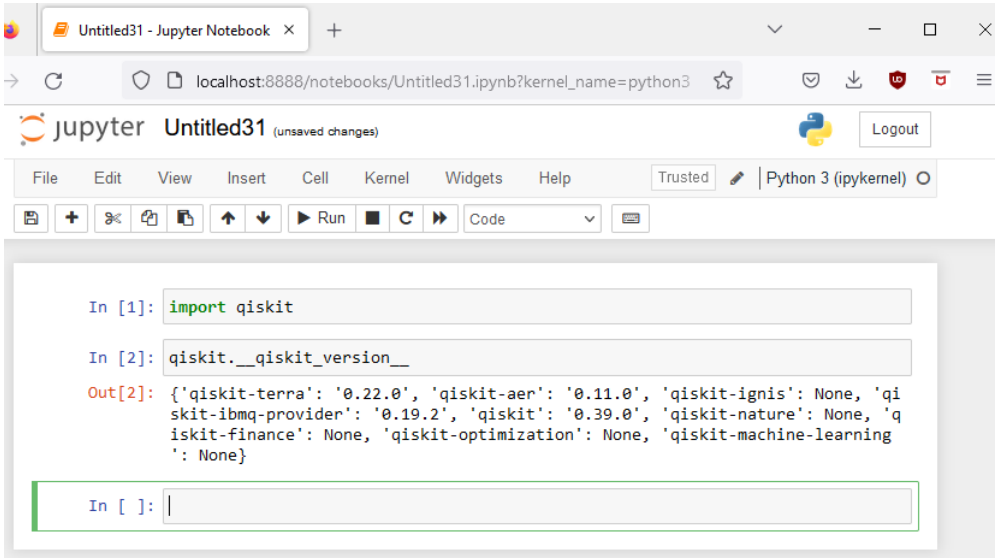
Για να ελέγξουμε την έκδοση του Qiskit που έχει εγκατασταθεί πληκτρολογούμε στο πλαίσιο την εντολή:

- ✓ import qiskit

qiskit_version_ (θα μας εμφανίσει την έκδοση της qiskit)

```
import qiskit
qiskit.__qiskit_version__
```

- ✓ Για να «τρέξουμε» δηλαδή να εκτελέσουμε ένα πρόγραμμα επιλέγουμε Run ή χρησιμοποιούμε το συνδυασμό πλήκτρων Shift + Enter.



Προσοχή

Σε περίπτωση που δεν πραγματοποιήθηκε η εγκατάσταση του πακέτου Qiskit, πληκτρολογούμε στο πλαίσιο του Jupyter Notebook την εντολή:

```
pip install qiskit
```

Έτσι σε αυτό το κεφάλαιο είμαστε έτοιμοι για να ξεκινήσουμε τα πρώτα κβαντικά μας προγράμματα.

4.7 Εισαγωγή στο Qiskit

Το Qiskit όταν χρησιμοποιείται για τη δημιουργία ενός κβαντικού προγράμματος, αυτό μπορεί να αναλυθεί σε μια σειρά έξι βημάτων, όπως αναφέρονται παρακάτω:

1. Εισαγωγή πακέτων
2. Αρχικοποίηση μεταβλητών
3. Προσθέστε πύλες
4. Οραματιστείτε το κύκλωμα
5. Προσομοίωση του πειράματος
6. Οραματιστείτε το αποτέλεσα

Ακόμα, τα βασικά στοιχεία που πρέπει να ακολουθούνται για μια διαδοχική εργασία, αυτή αποτελείται από τα παρακάτω βήματα:

- **Κατασκευή**: Σχεδιασμός ενός κβαντικού κυκλώματος για αναζήτηση λύσης ενός συγκεκριμένου προβλήματος.

- **Compile**: Μεταγλώττιση κυκλώματος.

- **Εκτέλεση**: Εκτέλεση του μεταγλωττισμένου κυκλώματος στις καθορισμένες κβαντικές υπηρεσίες.

- **Ανάλυση**: Οπτικοποίηση του πειράματος, υπολογίζοντας συνοπτικά στοιχεία, για παράδειγμα η απεικόνιση των αποτελεσμάτων μιας εργασίας σε ένα ιστόγραμμα.

Σημείωση

Με ένα ιστόγραμμα μπορούμε να παρουσιάσουμε τις πιθανότητες των κβαντικών καταστάσεων.

Ας δούμε ποιες εντολές θα χρειαστούμε και πώς συντάσσονται στο Qiskit για τη δημιουργία απλών προγραμμάτων.

• Εισαγωγή κατάλληλων βιβλιοθηκών στο πρόγραμμά μας

Τα βασικά στοιχεία που πρέπει να υπάρχουν αρχικά, όταν γράφουμε ένα πρόγραμμα, συντάσσονται ως εξής:

```
1. import numy as nq
2. from qiskit import QuantumCircuit
3. from qiskit.providers.aer import QasmSimulator
```

```
4. from qiskit.visualization import plot_histogram
```

Ας εξηγήσουμε κάποιες εντολές ανά γραμμή (βλέποντας μπροστά τον αριθμό της γραμμής που βρίσκεται η κάθε εντολή αντίστοιχα):

```
Γραμμή 2: QuantumCircuit: Κρατάει τις κβαντικές μας οδηγίες.
```

```
Γραμμή 3: QasmSimulator: Είναι ο προσομοιωτής κυκλώματος
                         απόδοσης Aer
```

```
Γραμμή 4: plot_histogram: Δημιουργεί ιστογράμματα, γραφικές
παραστάσεις
```

Ακόμα, καλείται:

- circuit: το κύκλωμα (το χρησιμοποιήσουμε για όνομα του κυκλώματος, αλλά μπορούμε να το ονομάσουμε και αλλιώς, για παράδειγμα και με μια μεταβλητή).
- QuantumCircuit: το κβαντικό κύκλωμα

Για να προβάλλουμε ένα κύκλωμα που σχεδιάσαμε με τις διάφορες μορφές που χρησιμοποιούνται σε διάφορα βιβλία και επιστημονικά άρθρα, μπορούμε να γράψουμε:

```
Qiskit.circuit.QuantumCircuit.draw()
```

Πριν δημιουργήσουμε απλά κβαντικά προγράμματα, να θυμηθούμε πρώτα και να δούμε κάποιες καινούργιες έννοιες.

4.8 Κβαντικός Υπολογισμός

4.8.1 Bits – Qubits

Στο πρώτο κεφάλαιο είδαμε ότι η στοιχειώδης μονάδα πληροφορίας στους κλασικούς υπολογιστές είναι το bit το οποίο περιγράφει ένα δισδιάστατο κλασικό σύστημα και μπορεί να βρίσκεται σε μία από τις δύο καταστάσεις, είτε στην κατάσταση 0 είτε στην κατάσταση 1. Παραδείγματα τέτοιων συστημάτων που εκφράζονται με 1 bit είναι:

- Ένας διακόπτης είναι ανοικτός ή κλειστός.
- Μία πρόταση είναι «αληθής» ή «ψευδής».

Ένα bit μπορεί να αναπαρασταθεί ως πίνακας 2 x 1 , όπου η πρώτη γραμμή του πίνακα αντιπροσωπεύει την κατάσταση 0 του συστήματος και η δεύτερη γραμμή την κατάσταση 1.

Δηλαδή,

- Η κατάσταση 0 συμβολίζεται ως [1 0].
- Η κατάσταση 1 συμβολίζεται ως [0 1].

Η στοιχειώδης κβαντική μονάδα πληροφορίας που χρησιμοποιείται για την περιγραφή ενός κβαντικού συστήματος είναι το κβαντικό bit, ή αλλιώς το qubit. Ένα qubit όπως ένα bit, παριστάνεται ως ένας πίνακας 2 x 1 αλλά περιλαμβάνει μιγαδικούς αριθμούς.

4.8.2 Κβαντική Υπέρθεση

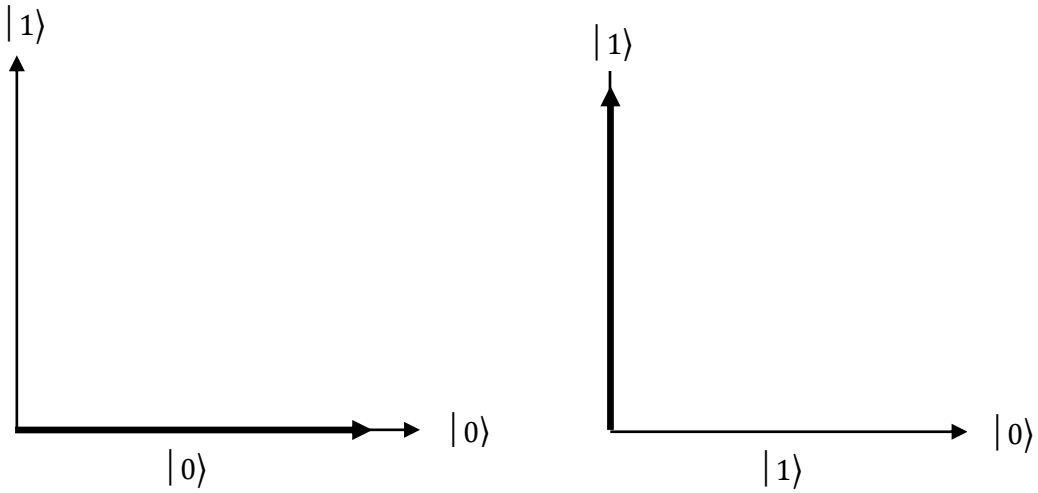
Τα κβαντικά συστήματα σε αντίθεση με τα κλασικά συστήματα, μπορούν να βρίσκονται ταυτόχρονα και στις δύο πιθανές καταστάσεις, δηλαδή σε κβαντική υπέρθεση (Quantum Superposition), χωρίς η μία να αποκλείει την άλλη. Για παράδειγμα σε κάποια συστήματα ένας αντίστοιχος διακόπτης θα ήταν την ίδια στιγμή ανοιχτός και κλειστός, δηλαδή ένα κβαντικό σύστημα μπορεί να είναι ταυτόχρονα στην κατάσταση 0 και στην κατάσταση 1.

Άρα, η θεμελιώδης διαφορά του qubit με το κλασικό bit είναι ότι ένα bit μπορεί να βρίσκεται σε μόνο μία από τις δύο δυνατές καταστάσεις 0 ή 1, ενώ ένα qubit μπορεί να βρίσκεται και στις δύο καταστάσεις ταυτόχρονα ή σε κάποια ενδιάμεση κατάσταση μεταξύ του 0 και του 1, όταν βρίσκεται σε υπέρθεση.

Δηλαδή, είναι η πιθανότητα για μια ενδεχόμενη κατάσταση του qubit και το μόνο που γνωρίζουμε πριν το μετρήσουμε είναι οι πιθανότητές του, να βρεθεί σε κάποια κατάσταση.

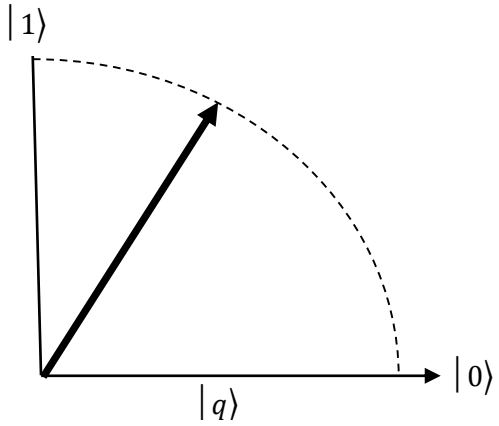
4.8.3 Διάνυσμα Κατάστασης

Το qubit, μπορεί όταν το μετρήσουμε να είναι 0 ή 1, αλλά πριν το δούμε και το μετρήσουμε, μπορεί να βρεθεί σε πολλές διαφορετικές καταστάσεις, οι οποίες αναπαρίστανται με το διάνυσμα κατάστασης και τις συμβολίζουμε με $|0\rangle$ και $|1\rangle$. Τα διανύσματα κατάστασης μπορούν να αναπαρασταθούν σε ένα σύστημα αξόνων, με οριζόντιο άξονα να είναι το $|0\rangle$ που αντιπροσωπεύεται σε αυτό το σύστημα με μια γραμμή που εφάπτεται στον οριζόντιο άξονα και κάθετο άξονα να είναι το $|1\rangle$ με μια γραμμή να εφάπτεται στον κάθετο άξονα. Παρακάτω βλέπουμε την απεικόνιση αυτών των διανυσμάτων κατάστασης.



Σχήμα: 1 – Γραφικές απεικονίσεις των διανυσμάτων κατάστασης 0 και 1

Όπως προαναφέραμε εκτός από τις καταστάσεις $|0\rangle$ και $|1\rangle$ υπάρχουν και οι ενδιάμεσες καταστάσεις και αυτό να αναπαριστάνεται με ένα διάνυσμα κατάστασης με διεύθυνση ανάμεσα στους δύο άξονες, όπως φαίνεται παρακάτω.



Σχήμα: 2 – Το διάνυσμα κατάστασης του $|q\rangle$

Το διάνυσμα κατάστασης εκφράζει τις πιθανότητες για το ποιο θα είναι το αποτέλεσμα όταν το qubit μετρηθεί.

Η κατάσταση $|q\rangle$ ονομάζεται υπέρθεση των καταστάσεων $|0\rangle$ και $|1\rangle$. Η προβολή του διανύσματος της κατάστασης της υπέρθεσης στον άξονα $|0\rangle$ έχει μήκος α και στον άξονα $|1\rangle$ έχει μήκος b.

Συνεπώς, τα qubits μπορούν να βρεθούν σε οποιαδήποτε υπέρθεση των βασικών καταστάσεων και ένα qubit έχει γενική μορφή:

$$|q\rangle = \alpha|0\rangle + b|1\rangle$$

Κατά τη μέτρηση ενός qubit, υπάρχει μία συγκεκριμένη πιθανότητα για κάθε μία από τις δύο καταστάσεις που βρίσκεται το qubit. Το τετράγωνο του μέτρου του πλάτους πιθανότητας είναι αυτό που αντιπροσωπεύει την πιθανότητα εμφάνισης του συστήματος σε μία συγκεκριμένη κατάσταση.

Δηλαδή,

- με πιθανότητα $|\alpha|^2$ το σύστημα θα βρίσκεται στην κατάσταση 0
- με πιθανότητα $|\beta|^2$ το σύστημα θα βρίσκεται στην κατάσταση 1

Σύμφωνα με τα παραπάνω, πρέπει οι πιθανότητες του συστήματος να έχουν άθροισμα ίσο με 1 (ή 100%) και έτσι ισχύει:

$$|\alpha|^2 + |b|^2 = 1\,.$$

Προσοχή

Ένα qubit δεν είναι ισοδύναμο με ένα κλασικό bit, ακόμα και αν η πιθανότητα να είναι στην κατάσταση 0 ή 1 είναι ίση με την πιθανότητα το qubit όταν μετρηθεί να βρεθεί στην ίδια κατάσταση.

Οι δύο βασικές καταστάσεις ενός qubit όπως αναφέραμε παραπάνω συμβολίζονται με $|0\rangle$ και $|1\rangle$ και μπορούν να γραφούν ως πίνακες με μία στήλη, όπως φαίνεται παρακάτω:

$$\bullet\ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{και} \quad \bullet\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Η υπέρθεση καταστάσεων, κάθε άλλη κβαντική κατάσταση δίνεται από τη σχέση:

$$|q\rangle = \alpha|0\rangle + b|1\rangle = \alpha\begin{bmatrix} 1 \\ 0 \end{bmatrix} + b\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix},$$

όπου α και b είναι μιγαδικοί αριθμοί (α, β ϵ $\mathbb{C}$) και ονομάζονται πλάτη πιθανότητας. Το διάνυσμα κατάστασης του qubit είναι ένα διάνυσμα στο χώρο Hilbert, που έχει δύο διαστάσεις.

Συνοψίζοντας

Το διάνυσμα κατάστασης του qubit $|q\rangle$ εκφράζει τα παρακάτω:

• Το qubit δηλώνεται με το σύμβολο ket, δηλαδή $|q\rangle$ που θα δούμε παρακάτω.

• Το διάνυσμα κατάστασης είναι ένα διάνυσμα δύο στοιχείων τα οποία ονομάζονται πλάτη πιθανότητας και εκφράζουν την πιθανή εξέλιξη του αποτελέσματος της μέτρησης του qubit.

• Το άθροισμα των πιθανοτήτων είναι πάντα ίσο με 1.

4.9 Κβαντικά Συστήματα Δύο Καταστάσεων

4.9.1 Περιγραφή – Συμβολισμός Dirac

Τα κβαντικά συστήματα με τα οποία θα ασχοληθούμε σε αυτή την ενότητα είναι των δύο καταστάσεων και για να μπορέσουμε να παραστήσουμε την κβαντική κατάσταση ενός συστήματος μαζί με τις ιδιότητές τους θα χρησιμοποιήσουμε βασικούς μαθηματικούς όρους. Δηλαδή, για να καταφέρουμε να παρακολουθήσουμε τη συμπεριφορά των qubits όταν εφαρμόζουμε πύλες, θα στηριχθούμε στη χρήση της μαθηματικής γλώσσας των διανυσμάτων και των πινάκων.

Στην Κβαντική Μηχανική, ο συμβολισμός Dirac (bra-ket) χρησιμοποιείται για την περιγραφή των κβαντικών καταστάσεων (quantum states). Ένα κβαντικό σύστημα μπορεί να περιγραφεί από το διάνυσμα κατάστασης, που στην ουσία είναι ένας γραμμικός συνδυασμός καταστάσεων. Ένα διάνυσμα κατάστασης περιγράφεται με τη βοήθεια συμβόλου Ket $|\ \rangle$.

• Διάνυσμα Bra και Ket

Για να περιγράψουμε κβαντικούς υπολογισμούς και αλγορίθμους χρησιμοποιούμε τα διανύσματα Bra και Ket, όπου το όνομά τους προέρχεται από την αγγλική λέξη Bracket και σημαίνει αγκύλη, με σύμβολα (> και <) μέσα στα οποία περικλείουμε φράσεις ή μέρη πληροφορίας. Αυτό τον συμβολισμό τον εισήγαγε ο διάσημος φυσικός Paul Dirac, που συνέβαλε στην ανάπτυξη της κβαντικής μηχανικής. Τα διανύσματα κατάστασης των κβαντικών συστημάτων τα συμβόλισε τοποθετώντας τα μέσα σε μισές αγκύλες $|\ \rangle$ και $\langle\ |$. Από τη λέξη bracket, χρησιμοποίησε τα τρία πρώτα γράμματα για το σύμβολο bra $|\ \rangle$ και τα τρία τελευταία γράμματα για το σύμβολο ket $|\ \rangle$.

Εμείς θα ασχοληθούμε μόνο με τα διανύσματα Ket.

• Διάνυσμα Ket

Τα διανύσματα Ket γράφονται ως πίνακας με μία στήλη και ονομάζονται πίνακες κατάστασης. Τα κβαντικά συστήματα δύο καταστάσεων, έχουν γενικά δύο στοιχεία στους πίνακες $\begin{bmatrix} a \\ b \end{bmatrix}$.

Μπορούμε να πούμε ότι γενικά όπως προαναφέραμε, ότι το διάνυσμα κατάστασης $|K\rangle$ γράφεται ως πίνακας:

$|K\rangle = a\,|H\rangle + b\,|T\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$.

Έτσι, η κατάσταση 0 συμβολίζεται με το Ket $|0\rangle$ και η κατάσταση 1 συμβολίζεται με Ket $|1\rangle$. Κάθε κατάσταση Ket αντιστοιχεί σε ένα διάνυσμα ενός πιο κατάλληλου χώρου Hilbert.

Έτσι, όπως είδαμε τα Kets μπορούμε να τα γράψουμε σε διανυσματικές στήλες:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{και} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Σημείωση

Για την περιγραφή των κβαντικών χρησιμοποιούμε στοιχεία της θεωρίας των μιγαδικών χώρων Hilbert.
Γενικά μπορούμε να πούμε ότι ένα qubit βρίσκεται σε μια κατάσταση $|y\rangle$ η οποία περιγράφεται από τη σχέση:

$|y\rangle = c_0\,|0\rangle + c_1\,|1\rangle$

Τα c_0, c_1 ονομάζονται πλάτη πιθανότητας κα είναι μιγαδικοί αριθμοί, για τους οποίους ισχύει:

$$|y\rangle = |\boldsymbol{c_0}|^2 + |\boldsymbol{c_1}|^2 = 1.$$

Σε κάθε Ket $|y\rangle$ αντιστοιχεί ένα bra $\langle y|$ ώστε μα ισχύει η παρακάτω σχέση:
$|y\rangle = c_0^*\,|0\rangle + c_1^*\,\langle 1|$
Τα c_0^* και c_1^* είναι οι συζυγείς μιγαδικοί των c_0 και c_1 αντίστοιχα.

Ας δούμε τι είναι ο χώρος Hilbert.

• Χώρος Hilbert

Ο χώρος Hilbert είναι ένας διανυσματικός χώρος στον οποίο έχει οριστεί ένα εσωτερικό γινόμενο και συμβολίζεται με H_n, n διαστάσεων. Στην ουσία πρόκειται για ένα τεχνικό ορισμό που ασχολούνται οι μαθηματικοί και οι φυσικοί και όσον αφορά τη θεωρία του κβαντικού υπολογισμού μας διευκολύνει πάρα πολύ, αφού οι καταστάσεις που περιγράφουν το σύστημα είναι απλά διανύσματα και οι τελεστές που επιδρούν στο σύστημα είναι απλοί πίνακες. Η χρονική εξέλιξη των κβαντικών συστημάτων περιγράφεται από μοναδιαίους τελεστές, δηλαδή από μοναδιαίους πίνακες με μιγαδικά στοιχεία και αυτό γιατί οι μοναδιαίοι πίνακες έχουν πολλές χρήσιμες ιδιότητες. Μια πολύ σημαντική ιδιότητα είναι ότι διατηρούν το μέγεθος των διανυσμάτων στα οποία επιδρούν και ακόμα, έχουν αντίστροφο πίνακα που το γινόμενό τους ισούται με 1.

Ας δούμε μερικά παράδειγμα.

ΠΑΡΑΔΕΙΓΜΑ 1

Να βρεθεί τι τιμή πρέπει να έχει το x, στο διάνυσμα κατάστασης $\begin{bmatrix} x \\ 0{,}2 \end{bmatrix}$, ώστε να αποτελεί αποδεκτή κβαντική κατάσταση.

Έχουμε:

Για να αποτελεί κβαντική κατάσταση, θα πρέπει να ισχύει:

$$x^2 + (0{,}2)^2 = 1 \Rightarrow x^2 + 0{,}04 = 1 \Rightarrow x^2 = 1 - 0{,}04 \Rightarrow x^2 = 0{,}96 \Rightarrow$$

$$x = \sqrt{0{,}96} \Rightarrow x \approx 0{,}98 \,.$$

ΠΑΡΑΔΕΙΓΜΑ 2

Να εξηγήσετε αν το qubit $|q\rangle = \frac{2}{\sqrt{2}}|0\rangle + \frac{3}{\sqrt{2}}|1\rangle$ βρίσκεται ή όχι σε αποδεκτή κβαντική κατάσταση.

Έχουμε,

Για να βρίσκεται το qubit σε κβαντική κατάσταση, θα πρέπει να ισχύει:

$$\left(\frac{2}{\sqrt{2}}\right)^2 + \left(\frac{3}{\sqrt{2}}\right)^2 = 1 \Rightarrow \frac{2^2}{(\sqrt{2})^2} + \frac{3^2}{(\sqrt{2})^2} = 1 \Rightarrow \frac{4}{2} + \frac{9}{2} = 1 \Rightarrow \frac{13}{2} \neq 1 \,.$$

Επομένως το qubit q, δεν βρίσκεται σε κβαντική κατάσταση.

ΠΑΡΑΔΕΙΓΜΑ 3

Να βρείτε ποια είναι η πιθανότητα μέτρησης 1 του qubit $|q\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)}{\sqrt{2}}|1\rangle$.

Έχουμε,

Για να υπολογίσουμε την πιθανότητα μέτρησης 1 του qubit, θα τετραγωνίσουμε το συντελεστή της κατάστασης $|1\rangle$ και είναι:

$\left(\frac{-1}{\sqrt{2}}\right)^2 = \frac{(-1)^2}{(\sqrt{2})^2} = \frac{1}{2}$.

Επομένως, το $\frac{1}{2}$ είναι το μισό και η πιθανότητα μέτρησης του 1 είναι 50%.

4.10 Η Σφαίρα Bloch

Ένα qubit μπορεί σχεδόν με όλα του τα χαρακτηριστικά να αναπαρασταθεί από τη σφαίρα του Bloch.

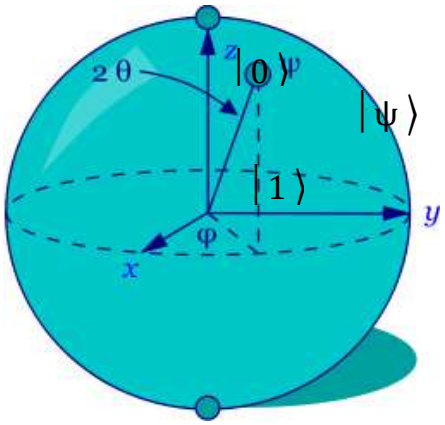
Η σφαίρα του Bloch είναι μια γεωμετρική αναπαράσταση του χώρου τριών διαστάσεων καθαρής κατάστασης ενός κβαντομηχανικού συστήματος δύο επιπέδων και πήρε την ονομασία της από τον Ελβετό φυσικό Felix Bloch (Φέλιξ Μπλοχ 1905-1983). Χρησιμοποιείται για να αναπαραστήσει ένα κβαντικό σύστημα δύο καταστάσεων ή αλλιώς ενός κβαντικού καταχωρητή ενός qubit, με όλα τα χαρακτηριστικά του αλλά και τις διεργασίες πάνω σε αυτό. Η σφαίρα Bloch αποτελεί ένα πολύ σπουδαίο εργαλείο, η οποία μας παρέχει μια πολύ χρήσιμη εικονογράφηση της κατάστασης ενός και μόνο απλού qubit. Επίσης, τελεστές που δρουν πάνω σε ένα qubit περιγράφονται και κατανοούνται πολύ ικανοποιητικά με τη βοήθεια της σφαίρας Bloch. Πρόκειται για μια σφαίρα με ακτίνα ίση με 1 και κάθε διάνυσμα που θα απεικονιστεί σε αυτή, θα έχει και αυτό μήκος 1 καθώς η αρχή του βρίσκεται στην αρχή των αξόνων και το τελικό σημείο οπουδήποτε στην επιφάνεια της σφαίρας.

Ας σκεφτούμε την κατάσταση ενός qubit ως μια θέση σε μια σφαίρα, όπως η επιφάνεια της γης, με τον Βόρειο Πόλο να είναι “0” και τον Νότιο Πόλο να είναι “1”. Ο χώρος των ακτινών σε ένα δισδιάστατο χώρο είναι η προβολή μιγαδικής γραμμής, η οποία είναι ισομορφική ως προς τη σφαίρα.

Με τη βοήθεια της σφαίρας Bloch μπορούν να απεικονιστούν οι βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ καθώς και οποιαδήποτε άλλη κατάσταση υπέρθεσης.

Έτσι, για μια κβαντική κατάσταση ψ, ισχύει $|\psi\rangle = \alpha|0\rangle + b|1\rangle$ και μπορεί να γραφεί ως μια μιγαδική υπέρθεση των διανυσμάτων Ket $|0\rangle$ και $|1\rangle$ και η αναπαράσταση του ψ είναι:

$|\psi\rangle = cos\theta\,|0\rangle + e^{i\varphi} sin\theta\,|1\rangle = cos\theta\,|0\rangle + (cos\varphi + i\,sin\varphi)sin\theta\,|1\rangle$, με $0 \le \theta \le \frac{\pi}{2}$, $0 \le \varphi \le 2\pi$ και ϕ, θ είναι πραγματικοί αριθμοί.



Σχήμα: 3 - Η αναπαράσταση ενός qubit -Το διάνυσμα στη σφαίρα Bloch
Πηγή εικόνας:
https://el.wikipedia.org/wiki/%CE%A3%CF%86%CE%B1%CE%AF%CF%81%CE%B1_%CE%9C%CF%80%CE%BB%CE%BF%CF%87

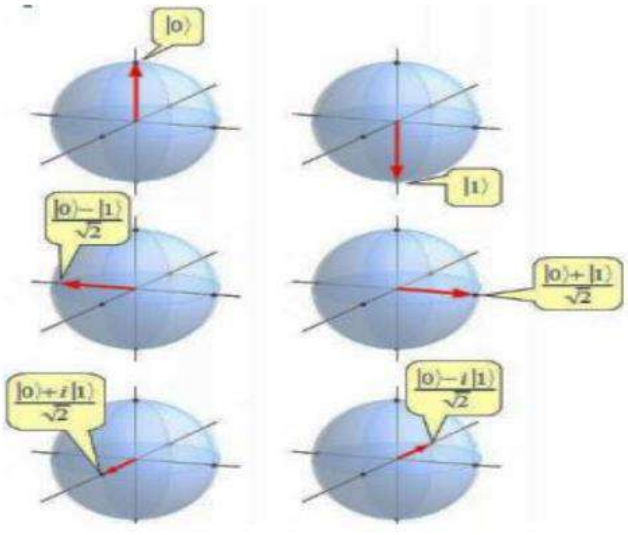
Οι παράμετροι, οι γωνίες ψ και θ προσδιορίζουν μοναδικά ένα σημείο στην επιφάνεια της σφαίρας Bloch (το οποίο είναι και το τέλος του διανύσματος $|\psi\rangle$), του οποίου οι συντεταγμένες του είναι (x, z, y) και ισούνται:

- x = sin2θ x cosϕ
- y = sin2θ x sinϕ
- z = cos2θ

Το διάνυσμα κατάστασης του qubit όπως περιγράφεται από την παραπάνω εξίσωση μπορεί να αναπαρασταθεί σε τρεις διαστάσεις με τη χρήση της σφαίρας του Bloch.

Η σφαίρα Bloch όπως αναφέραμε παραπάνω, έχει ακτίνα ίση με τη μονάδα και συγκεκριμένα (για το σχήμα: 2) το διάνυσμα κατάστασης $|\psi\rangle$ του qubit, έχει την αρχή του στο κέντρου της σφαίρας και το τέλος του σε κάποιο σημείο της επιφάνειας της σφαίρας που καθορίζεται από τις γωνίες θ και ϕ. Το μήκος του $|\psi\rangle$ είναι ίσο με τη μονάδα.

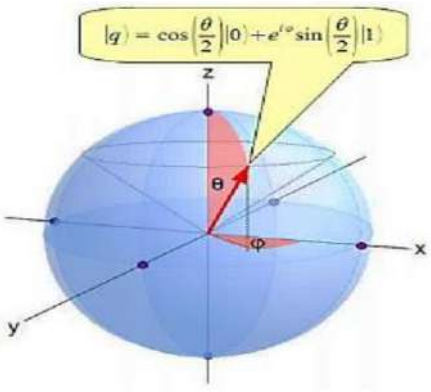
Η απεικόνιση του ενός qubit σε κάθε άξονα



Σχήμα: 4 - Η αναπαράσταση ενός qubit –στους άξονες x, y, z
Πηγή εικόνας:
file:///C:/Users/User/Downloads/%CE%A0%CE%A4%CE%A5%CE%A7%CE%99%CE%91%CE%9A%CE%97%20%CE%95%CE%A1%CE%93%CE%91%CE%A3%CE%99%CE%91%20%CE%94%CE%99%CE%91%CE%9A%CE%91%CE%9A%CE%97%CE%A3%20%CE%93%CE%95%CE%A9%CE%A1%CE

Για την απεικόνιση ενός qubit σε κάθε άξονα, τότε αυτό περιστρέφεται σε αυτούς, με γωνία θ και ϕ να λαμβάνονται θετικές από τον άξονα +z στο qubit και από το +y στο +x. Η γωνία θ ονομάζεται πολική και καθορίζει τις τιμές των πλατών πιθανότητας, δηλαδή η γωνία θ είναι πολύ σημαντική γιατί μας δίνει την πιθανότητα να παρατηρήσουμε το qubit στην κατάσταση “0” ή στην κατάσταση “1”. Η γωνία θ μετριέται από τα θετικά του άξονα x προς τα θετικά του άξονα y. Η γωνία ϕ ονομάζεται γωνία φάσης, η οποία μετριέται από τα θετικά του άξονα z, προς τα κάτω, μέχρι εκεί

που βρίσκεται το διάνυσμά μας q και είναι πάντα θετική (με τους παραπάνω περιορισμούς).



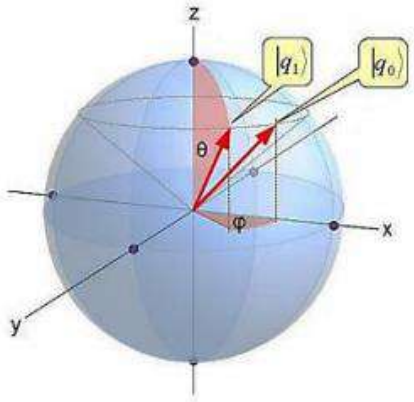
Σχήμα: 5 - Η αναπαράσταση ενός qubit στη σφαίρα Bloch

Ένα κλασικό bit θα μπορούσε να βρεθεί μόνο στους πόλους της σφαίρας.

Παρατηρούμε ότι όλα τα διανύσματα πάνω στον κώνο που σχηματίζεται (δηλαδή υπάρχουν πάρα πολλά σημεία) τα οποία έχουν την ίδια γωνία θ, αλλά διαφορετική γωνία φ (φάση) και η z καθορίζει τις πιθανότητες παρατήρησης. Όλα αυτά τα διανύσματα σχηματίζουν την επιφάνεια ενός κώνου και έχουν ακριβώς την ίδια πιθανότητα να τα παρατηρήσουμε στην κατάσταση “0” ή “1”.

Η απεικόνιση των δύο qubit σε κάθε άξονα

Όταν θέλουμε να παρατηρήσουμε και να απεικονίσουμε δύο qubit σε κάθε άξονα, δεν μπορούμε με μία μέτρηση μόνο να ξεχωρίσουμε 2 qubit με διαφορά φάσης φ. Εάν επαναλάβουμε το πείραμα και φέρουμε τα διανύσματα σε αυτές τις θέσεις πολλές φορές, η πιθανότητα αυτά τα δύο διανύσματα να εμφανίσουν “0” ή “1” είναι ακριβώς η ίδια.



Σχήμα: 6 - Η αναπαράσταση δύο qubit στη σφαίρα Bloch

Αυτές οι καταστάσεις διαφέρουν μόνο στη γωνία φ και δεν είναι δυνατόν να διακριθούν με παρατήρηση (η παρατήρηση πρέπει να είναι επαναληπτική, γιατί αν επαναλάβουμε το πείραμα μια φορά δεν μας μεταφέρει πληροφορία).

Στο σχήμα 3, απεικονίζονται τα 2 qubit στη σφαίρα Bloch, τα οποία έχουν την ίδια γωνία θ, αλλά διαφορετική φάση και συγκεκριμένα, το $|q_1\rangle$ έχει γωνία φάσης φ ενώ το $|q_2\rangle$ έχει γωνία φάσης μηδέν.

Οι καταστάσεις των δύο qubit περιγράφονται:

• $|q_1\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$ και • $|q_2\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$.

Αν θέσουμε $|e^{i\varphi}| = 1$ στην παραπάνω σχέση, τότε η ολική πιθανότητα θα είναι:

$$\left|\cos\frac{\theta}{2}\right|^2 + \left|e^{i\varphi}\sin\frac{\theta}{2}\right|^2 = \left|\cos\frac{\theta}{2}\right|^2 + \left|\sin\frac{\theta}{2}\right|^2 = 1.$$

Άρα, τα πλάτη των πιθανοτήτων είναι $cos\frac{\theta}{2}$ και $sin\frac{\theta}{2}$, ενώ οι πιθανότητες είναι $\left|cos\frac{\theta}{2}\right|^2$ και $\left|sin\frac{\theta}{2}\right|^2$ που θα εμφανιστεί σε μια μέτρηση η βασική κατάσταση $|0\rangle$ και $|1\rangle$.

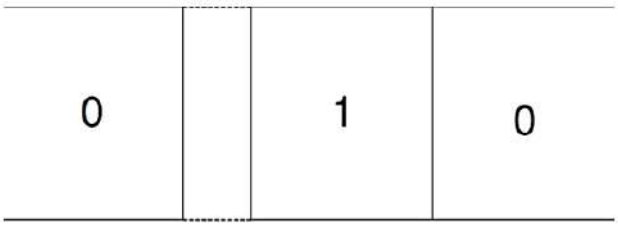
Συνεπώς, για την αναπαράσταση μιας κβαντικής κατάστασης 1 qubit χρειαζόμαστε μόνο τρεις συντελεστές, τις γωνίες θ, φ και το μέτρο του διανύσματος το οποίο πάντα είναι ίσο με την μονάδα.

Παρατήρηση

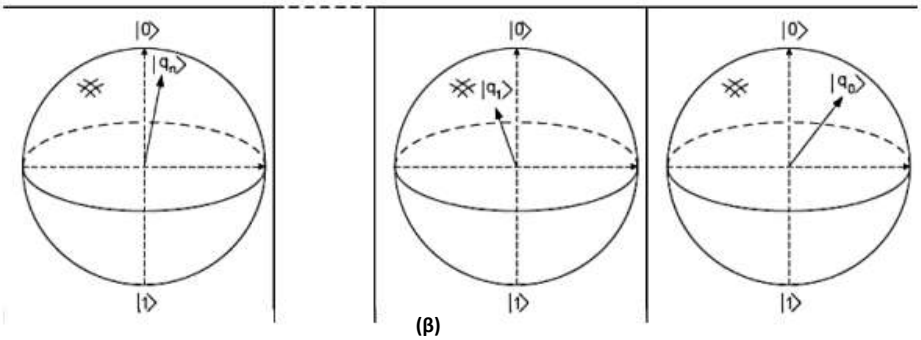
Δεν είναι δυνατό σε μία μόνο μέτρηση να διακρίνουμε δύο qubits τα οποία διαφέρουν μόνο κατά γωνία φάσης (φ) και αυτό οφείλεται στο γεγονός ότι οι πιθανότητες εμφάνισης μιας βασικής κατάστασης και τα πλάτη αυτών (τα οποία είναι μετρήσιμα μεγέθη), δεν περιέχουν τη γωνία φάσης φ. Όμως η γωνία φάσης, αποτελεί πολύ σημαντικό ρόλο.

4.11 Κβαντικός Καταχωρητής

Στους κλασικούς υπολογιστές οι καταχωρητές (registers), είναι διατάξεις στις οποίες αποθηκεύονται μικρά μέρη πληροφορίας με τη μορφή bits. Επομένως, ένα σύνολο bits αποτελεί έναν καταχωρητή, στον οποίο αποθηκεύονται οι τιμές κάποιων μεταβλητών. Οι κβαντικοί υπολογιστές όπως προαναφερθήκαμε βασίζονται σε qubits και όπως ο κλασικός καταχωρητής του ψηφιακού υπολογιστή, έτσι και ο καταχωρητής ενός κβαντικού υπολογιστή αποτελείται από ένα σύστημα πολλών qubits, που συνήθως είναι τοποθετημένα σε σειρά (μετρώντας από δεξιά προς τα αριστερά, σε αντίθεση με τα bits του κλασικού καταχωρητή). Δηλαδή ένα σύνολο qubits αποτελεί έναν κβαντικό καταχωρητή. Σε έναν κβαντικό υπολογιστή μπορούμε να αποθηκεύσουμε πολύ περισσότερη πληροφορία σε σχέση με έναν κλασικό καταχωρητή καθώς τα δεδομένα αποθηκεύονται σε κβαντικούς καταχωρητές, στους οποίους τα qubit είναι διατεταγμένα σε σειρά.



(α)



(β)

Σχήμα: 7 – (α) Κλασικός καταχωρητής – (β) Κβαντικός καταχωρητής

Πηγή εικόνας: http://ikee.lib.auth.gr/record/323911/files/Gatsou.pdf

Οι κβαντικοί υπολογιστές εκτελούν υπολογισμούς και μας ενδιαφέρει η έκφραση του τανυστικού γινομένου με μορφή πινάκων.

Αρχικά, πρέπει να οριστεί το τανυστικό γινόμενο (όπως είδαμε σε προηγούμενο κεφάλαιο). Θα ορισθεί για πίνακες με μια στήλη και δύο γραμμές, όπως το διάνυσμα ket που μελετάται.

Εφαρμογή 1

Έχουμε έναν κβαντικό καταχωρητή C, ο οποίος αποτελείται από δύο qubits q_1 και q_0, με διάνυσμα κατάστασης να είναι (πίνακας μιας στήλης και δύο γραμμών) αντίστοιχα $|q_1\rangle = \alpha|0\rangle + b|1\rangle = \begin{bmatrix} \alpha \\ b \end{bmatrix}$ και $|q_0\rangle = c|0\rangle + d|1\rangle = \begin{bmatrix} c \\ d \end{bmatrix}$. Τότε ο καταχωρητής των δύο qubits q_1 και q_0 περιγράφεται από τη σχέση:

$$C = A \otimes B = \begin{bmatrix} \alpha \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \alpha \cdot c \\ \alpha \cdot d \\ b \cdot c \\ b \cdot d \end{bmatrix}.$$

Έτσι, το αποτέλεσμα του τανυστικού γινομένου των δύο πινάκων είναι ένας νέος πίνακας με πλήθος στοιχείων ίσο με το άθροισμα των στοιχείων των δύο προηγούμενων πινάκων.

Συνεπώς, για τον περιβάλλον των qubits, μπορούμε με τη χρήση του τανυστικού γινομένου να βρούμε τις βασικές καταστάσεις ενός κβαντικού καταχωρητή, ο οποίος αποτελείται από n qubits καθώς και τα πλάτη της πιθανότητας της κάθε κατάστασης.

Ας δούμε μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 4

Έχουμε έναν κβαντικό καταχωρητή που αποτελείται από 2 qubits, με αρχική κατάσταση 0 που γράφεται $|00\rangle$. Η κατάσταση του καταχωρητή δίνεται από το τανυστικό γινόμενο που συμβολίζεται με $\otimes$ των καταστάσεών τους (των 2 qubits) που περιέχει και είναι:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}1\\0\end{bmatrix}\\0\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}.$$

Με φυσικά πλάτη πιθανοτήτων $1^2 + 0^2 + 0^2 + 0^2 = 1$.

Επομένως, για τις βασικές καταστάσεις ενός καταχωρητή με 2 qubits ισχύει:

- $$|01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}0\\1\end{bmatrix}\\0\begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}.$$

- $$|10\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\begin{bmatrix}1\\0\end{bmatrix}\\1\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}.$$

- $$|11\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\begin{bmatrix}0\\1\end{bmatrix}\\1\begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

Το ίδιο θα ισχύει και όταν έχουμε παραπάνω από δύο qubits, δηλαδή:

- $$|011\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix}.$$

Όλοι πιθανοί συνδυασμοί των 3 qubits συνολικά είναι:

- $$|000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}.$$

- $|001\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

- $|010\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

- $|011\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

- $|100\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

- $|101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$.

• $\left| 110 \right\rangle = \left| 1 \right\rangle \otimes \left| 1 \right\rangle \otimes \left| 0 \right\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$.

• $\left| 111 \right\rangle = \left| 1 \right\rangle \otimes \left| 1 \right\rangle \otimes \left| 1 \right\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$.

Συνεπώς, για να βρούμε την τελική κατάσταση ενός καταχωρητή όταν τα qubits βρίσκονται σε οποιαδήποτε κατάσταση, αρκεί να πραγματοποιήσουμε τους παραπάνω υπολογισμούς.

Έτσι, η πληροφορία που αποθηκεύεται σε έναν κβαντικό καταχωρητή είναι πολύ περισσότερη από όση σε ένα κλασικό καταχωρητή, αφού με τον κλασικό καταχωρητή των 2 bit μπορούμε να αποθηκεύσουμε μια τιμή από τις τέσσερις (ή 00 ή 01 ή 10 ή 11) ενώ στον κβαντικό καταχωρητή δύο καταστάσεων είναι δυνατό να αποθηκεύσει τέσσερις αριθμούς ταυτόχρονα (και το 00 και το 01 και το 10 και το 11).

Γενικά

Η κατάσταση ενός κβαντικού καταχωρητή ο οποίος αποτελείται από n qubits δίνεται από το τανυστικό γινόμενο των καταστάσεων των n qubits που τις αποτελούν και δίνεται από την παρακάτω σχέση:

$$\left| q_R \right\rangle = \left| q_{n-1} \right\rangle \otimes \dots \otimes \left| q_1 \right\rangle \otimes \left| q_0 \right\rangle = \left| q_{n-1} \dots q_1 q_0 \right\rangle$$

Ας δούμε μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 5

Έχουμε έναν κβαντικό καταχωρητή που περιέχει 2 qubits $\left| \kappa \lambda \right\rangle$, τα οποία βρίσκονται στις καταστάσεις $\left| \kappa \right\rangle = \frac{3}{5} \left| 0 \right\rangle + \frac{4}{5} \left| 1 \right\rangle$ και $\left| \lambda \right\rangle = \frac{1}{\sqrt{2}} \left| 0 \right\rangle + \frac{1}{\sqrt{2}} \left| 1 \right\rangle$, να βρεθεί η τελική κατάσταση του καταχωρητή.

Για να βρούμε την κατάσταση του καταχωρητή, θα υπολογίσουμε το τανυστικό γινόμενο:

• $|\kappa\,\lambda\rangle = |\kappa\rangle \otimes |\lambda\rangle = \begin{bmatrix} \frac{3}{5} \\ \frac{4}{5} \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{3}{5}\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \\ \frac{4}{5}\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \frac{3}{5\sqrt{2}} \\ \frac{3}{5\sqrt{2}} \\ \frac{4}{5\sqrt{2}} \\ \frac{4}{5\sqrt{2}} \end{bmatrix} =$

$$= \frac{3}{5\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{3}{5\sqrt{2}}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \frac{4}{5\sqrt{2}}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \frac{4}{5\sqrt{2}}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Έτσι οι παραπάνω πίνακες ισοδυναμούν με τις καταστάσεις:

$$\frac{3}{5\sqrt{2}}|00\rangle + \frac{3}{5\sqrt{2}}|01\rangle + \frac{4}{5\sqrt{2}}|10\rangle + \frac{4}{5\sqrt{2}}|11\rangle\,.$$

Όμως, για να ισχύει η παραπάνω κβαντική κατάσταση με τις αντίστοιχες πιθανότητες εμφάνισης των $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, θα πρέπει το άθροισμα των τετραγώνων των συντελεστών καταστάσεων να ισούται με τη μονάδα.
Οπότε έχουμε:

$$\left(\frac{3}{5\sqrt{2}}\right)^2 + \left(\frac{3}{5\sqrt{2}}\right)^2 + \left(\frac{4}{5\sqrt{2}}\right)^2 + \left(\frac{4}{5\sqrt{2}}\right)^2 =$$

$$= \frac{9}{25\cdot 2} + \frac{9}{25\cdot 2} + \frac{16}{25\cdot 2} + \frac{16}{25\cdot 2} = \frac{50}{50} = 1.$$

Σημείωση

Γενικά σε ένα πραγματικό κβαντικό υπολογιστή η αρχική κατάσταση των qubits είναι εξ ορισμού $|0\rangle$. Όταν έχουμε πολλά qubit για την υλοποίηση ενός καταχωρητή η Python τα αρχικοποιεί πάντα στην κατάσταση $|0\rangle$, εκτός σε διαφορετική περίπτωση που δηλωθεί κάποια κατάσταση διαφορετική.

Προσοχή!!!

Κατά τη μέτρηση ενός κυκλώματος τα qubits καταστρέφονται και επομένως θα χρειαστούμε ανάλογο αριθμό κλασικών bits, για να αποθηκευτεί η κβαντική τους κατάσταση και επομένως να πάρουμε τη μέτρηση του κυκλώματος.

Ας δούμε ποιες εντολές θα χρειαστούμε και πώς συντάσσονται, για το σχεδιασμό απλών κβαντικών κυκλωμάτων.

• Υλοποίηση κβαντικών κυκλωμάτων

Κάποια βασικά στοιχεία που πρέπει να υπάρχουν αρχικά, όταν γράφουμε ένα πρόγραμμα, είναι τα QuantumRegister, ClassicalRegister QuantumCircuit και συντάσσονται ως εξής:

```
from qiskit import QuantumRegister, ClassicalRegister,
QuantumCircuit
```

• Δημιουργία κλασικών κυκλωμάτων – ClassicalRegister

Για να δημιουργήσουμε ένα κλασικό κύκλωμα, γράφουμε ClassicalRegister, συντάσσοντάς την όπως φαίνεται παρακάτω:

```
QuantumRegister(n)
```

Όπου n είναι ο αριθμός των bit, ο οποίος αποτελείται ο καταχωρητής.

• Δημιουργία κβαντικών κυκλωμάτων – QuantumRegister

Για να δημιουργήσουμε ένα κλασικό κύκλωμα, γράφουμε QuantumRegister, συντάσσοντάς την όπως φαίνεται παρακάτω:

```
QuantumRegister(n)
```

Όπου n είναι ο αριθμός των qubit, ο οποίος αποτελείται ο καταχωρητής.

Εφαρμογή 2

Για τη δημιουργία των register χρειαζόμαστε τις παρακάτω εντολές:

```
1. qr = QuantumRegister(2)
2. cr = ClassicalRegister(2)
3. qc = QuantumCircuit(q, c)
```

Ας δούμε αναλυτικά πώς δουλεύει η κάθε εντολή (ανά γραμμή).

- Γραμμή 1: Δημιουργεί δύο qubit αρχικοποιημένα στην κατάσταση $|0\rangle$.
- Γραμμή 2: Δημιουργεί το κλασικό κύκλωμα δύο bit στα οποία θα αποθηκευτούν τα αποτελέσματα των μετρήσεων για κάθε qubit αντίστοιχα.
- Γραμμή 3: Δημιουργεί το κβαντικό κύκλωμα με περιεχόμενα το quantum register (2 qubits) και το classical register (2 qubits).

Θα μπορούσαμε να γράψουμε τις παραπάνω εντολές εν συντομία, ως μια εντολή όπως φαίνεται παρακάτω:

```
qc = QuantumCircuit(q,c)
```

• Μέτρηση κυκλωμάτων – Measure

Κατά την μέτρηση ενός κυκλώματος, γράφουμε measure, για να καταχωρήσουμε το αποτέλεσμα της μέτρησης του qubit το οποίο αποθηκεύεται (η κβαντική κατάσταση των qubits) στα αντίστοιχα κλασικά bits, συντάσσοντάς την όπως φαίνεται παρακάτω:

```
Circuit.measure(qr,cr)
```

όπου:

qr: qubit για μέτρηση

cr: κλασικό bit για να αποθηκεύσουμε τη μέτρηση (της κβαντικής κατάστασης των qubits).

Αλλιώς η παραπάνω εντολή μπορεί να συνταχθεί ως:

```
circuit.measure(range(n),range(n))
```

Εφαρμογή 3

Για τη μέτρηση ενός κυκλώματος συντάσσουμε την παρακάτω εντολή:

```
circuit.measure([0,1], [0,1])
```

Αλλιώς μπορεί να συνταχθεί η παραπάνω εντολή ως:

```
circuit.measure(range(2),range(2))
```

Η εντολή αυτή εφαρμόζει και στα δύο qubit από μία κβαντική πύλη μέτρησης. Συγκεκριμένα, το πρώτο μέρος της παρένθεσης (**[0,1],** [0,1]) μας δείχνει σε ποια qubit θα εφαρμοστεί η πύλη μέτρησης. Το δεύτερο μέρος της παρένθεσης ([0,1], **[0,1]**) μας δείχνει σε ποια bit θα αποθηκευτούν τα αποτελέσματα για το κάθε qubit αντίστοιχα. Συνεπώς, το αποτέλεσμα της μέτρησης του πρώτου qubit θα αποθηκευτεί στο πρώτο bit και το αποτέλεσμα της μέτρησης του δεύτερου qubit θα αποθηκευτεί στο δεύτερο bit, τα οποία θα κατανοήσουμε καλύτερα στα επόμενα κεφάλαια.

Παρατήρηση

Αξίζει να τονίσουμε ότι κατά τη μέτρηση του κυκλώματος, τα qubits καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα κλασικά bits.

• Μέθοδος – draw ( )

Για να σχεδιάσουμε ένα κύκλωμα, γράφουμε draw( ), συντάσσοντάς την όπως φαίνεται παρακάτω:

```
Circuit.draw ( )
```

• Προσομοίωση κυκλώματος – Simulator

Για να προσομοιώσουμε ένα κύκλωμα, γράφουμε simulator, συντάσσοντάς την όπως φαίνεται παρακάτω:

```
Circuit.measure(qr,cr)
```

Εφαρμογή 4

Για τη μέτρηση ενός κυκλώματος συντάσσουμε τις παρακάτω εντολές:

```
1. qr = QuantumRegister(2)
2. cr = ClassicalRegister(2)
3. qc = QuantumCircuit(q, c)
```

• Απεικόνιση αποτελεσμάτων σε ιστόγραμμα– Plot_histogram

Για να προσομοιώσουμε ένα κύκλωμα, γράφουμε plot_histogram, συντάσσοντάς την όπως φαίνεται παρακάτω:

```
plot_histogram(counts)
```

όπου: counts μία μεταβλητή.

Το Qiskit παρέχει εργαλεία για την απεικόνιση των αποτελεσμάτων μιας εργασίας μας, όπως για παράδειγμα για τη δημιουργία ιστογράμματος πιθανοτήτων των κβαντικών καταστάσεων. Έτσι, για να εμφανιστεί το ιστόγραμμα πιθανοτήτων μιας κβαντικής κατάστασης, πληκτρολογούμε την εντολή plot_histogram( ) η οποία συντάσσεται και ως εξής:

```
from qiskit. Visualization import plot_histogram
plot_histogram(counts)
```

Έτσι, εμφανίζεται ένα ιστόγραμμα, παρουσιάζοντας τις πιθανότητες των κβαντικών καταστάσεων κατά την εκτέλεση ενός κβαντικού κυκλώματος (με κάποιων αριθμό δοκιμών).

• Εντολή Initialize

Η initialize είναι μια εντολή και όχι μια πύλη, καθώς αποτελεί μια εντολή επαναφοράς, η οποία δεν είναι ενιαία.

```
initialize(params, num_qubits = None)
```

όπου:

• params καλείται παράμετροι (str, list, int ή statedector):

- ✓ list : διάνυσμα μιγαδικών πλατών προς αρχικοποίηση
- ✓ string: ετικέτες των βασικών καταστάσεων
- ✓ int: ένας ακέραιος αριθμός που μας δείχνει σε ποια qubit να υλοποιηθεί η αρχικοποίηση
- ✓ statedector: για αρχικοποίηση

• num_qubits (int): Η παράμετρος αρχικοποιείται μόνο για παραμέτρους int. Προσδιορίζει τον συνολικό αριθμό των qubits στην κλήση αρχικοποίησης.

Ας δούμε μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 6

Να δημιουργηθεί πρόγραμμα που περιγράφει ακριβώς το πώς να αρχικοποιήσουμε στον καταχωρητή 2 qubits.

Έχουμε:

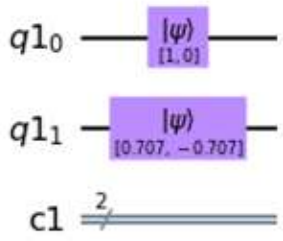
```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  from math import pi, sqrt
4.  %matplotlib inline
5.  initial_state_0 = [1,0]
6.  initial_state_1 = [1/sqrt(2),-1/sqrt(2)]
7.  qr = QuantumRegister(2)
8.  cr = ClassicalRegister(2)
9. circuit = QuantumCircuit(qr,cr)
10. circuit.initialize(initial_state_0,0)
11. circuit.initialize(initial_state_1,1)
12. circuit.draw(output='mpl')
```

Να δούμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- ➢ Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- ➢ Γραμμή 5 και 6: Αποδοχή των τιμών των ζητούμενων αρχικών καταστάσεων στις αντίστοιχες μεταβλητές του προγράμματος.
- ➢ Γραμμή 7: Δημιουργία κβαντικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των qubits και με μια μεταβλητή) .

- Γραμμή 8: Δημιουργία κλασικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των bits και με μια μεταβλητή) .
- Γραμμή 9: Δημιουργία κυκλώματος από 2 qubit και 2 bits.
- Γραμμές 10 και 11: Αρχικοποίηση των qubits.
- Γραμμή 12: Οπτικοποίηση του κυκλώματος. Γράφοντας 'mpl' μας δίνει το αποτέλεσμα του κυκλώματος χρωματισμένο. Εμφανίζεται το αποτέλεσμα του προγράμματος σε μορφή ιστογράμματος, ώστε με τη γραφική παράσταση να μπορούμε να έχουμε μια ολοκληρωμένη εικόνα για το συγκεκριμένο πρόγραμμά μας.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί:



Σχήμα: 8 – Κύκλωμα με αρχικές καταστάσεις

Ας δούμε το παραπάνω παράδειγμα όχι με 2 qubit, αλλά με 3 qubits.

ΠΑΡΑΔΕΙΓΜΑ 7

Να δημιουργηθεί ένα πρόγραμμα που περιγράφει ακριβώς το πώς να αρχικοποιήσουμε στον καταχωρητή 3 qubits.

Έχουμε:

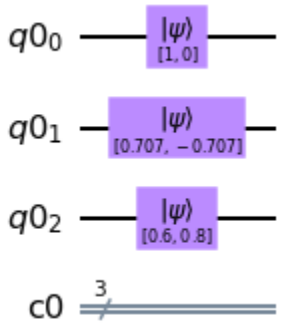
```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  from math import pi, sqrt
4.  %matplotlib inline
5.  initial_state_0 = [1,0]
6.  initial_state_1 = [1/sqrt(2),-1/sqrt(2)]
7.  initial_state_2 = [3/5,4/5]
8.  qr = QuantumRegister(3)
9.  cr = ClassicalRegister(3)
10. circuit = QuantumCircuit(qr,cr)
11. circuit.initialize(initial_state_0,0)
12. circuit.initialize(initial_state_1,1)
13. circuit.initialize(initial_state_2,2)
```

```
14. circuit.draw(output='mpl')
```

Το πρόγραμμα συντάσσεται ακριβώς το ίδιο με το πρόγραμμα των 2 qubit, με τη διαφορά:

- Γραμμή 7: Αποδοχή της επιπλέον τιμής της ζητούμενης αρχικής κατάστασης στην αντίστοιχη μεταβλητή του προγράμματος.
- Γραμμή 8 : Δημιουργία κβαντικού καταχωρητή αποτελούμενο από 3 qubits (μπορούμε να αποδώσουμε το πλήθος των qubits και με μια μεταβλητή) .
- Γραμμή 9: Δημιουργία κλασικού καταχωρητή αποτελούμενο από 3 qubits (μπορούμε να αποδώσουμε το πλήθος των bits και με μια μεταβλητή) .
- Γραμμή 10: Δημιουργία κυκλώματος από 3 qubit και 3 bits.
- Γραμμή 13: Αρχικοποίηση του (επιπλέον) qubit.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί:



Σχήμα: 9 – Καταχωρητής με τρία qubits

• Μέτρηση Πιθανοτήτων

Για να μετρήσουμε τις πιθανότητες από τα ενδεχόμενα των αποτελεσμάτων θα πρέπει να προσθέσουμε στο πρόγραμμα του παραπάνω παραδείγματος κάποιες επιπλέον εντολές.

Στο πρόγραμμα για την αρχικοποίηση του καταχωρητή των 2 qubits γράψαμε το παρακάτω πρόγραμμα:

```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  from math import pi, sqrt
4.  %matplotlib inline
5.  initial_state_0 = [1,0]
6.  initial_state_1 = [1/sqrt(2),-1/sqrt(2)]
```

```
7.  qr = QuantumRegister(2)
8.  cr = ClassicalRegister(2)
9. circuit = QuantumCircuit(qr,cr)
10. circuit.initialize(initial_state_0,0)
11. circuit.initialize(initial_state_1,1)
12. circuit.draw(output='mpl')
```

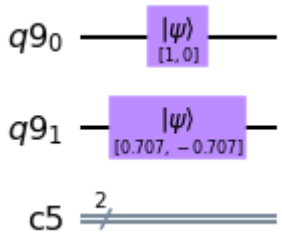
Οι επιπλέον εντολές που χρειάζεται να προσθέσουμε στο παραπάνω πρόγραμμα για να μετρήσουμε τις πιθανότητες των ενδεχομένων αποτελεσμάτων, είναι οι παρακάτω:

```
13. circuit.measure(qr,cr)
14. simulator = Aer.get_backend('qasm_simulator')
15.
result=execute(circuit,backend=simulator,shots=1000).result()
16. plot_histogram(result.get_counts(circuit))
```

Να δούμε αναλυτικά τις παραπάνω τρεις γραμμές, της συνέχειας του προγράμματος:

- Γραμμές 13: Μέτρηση του κυκλώματος (μετά, τα qubit καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα bits).
- Γραμμή 14: Επιλογή προσομοιωτή.
- Γραμμή 15: Αποδίδεται στη μεταβλητή result το αποτέλεσμα της εκτέλεσης του κυκλώματος, που καλούμε να εκτελεσθεί το πρόγραμμα 1000 φορές τυχαία.
- Γραμμή 16: Εμφανίζεται το αποτέλεσμα του προγράμματος σε μορφή ιστογράμματος και αυτό για το λόγο ότι με μία γραφική παράσταση μπορούμε να έχουμε μια ολοκληρωμένη εικόνα για το συγκεκριμένο πρόγραμμά μας.

Όταν «τρέξουμε» το πρόγραμμα (μέχρι και τη γραμμή 12), θα εμφανιστεί το αποτέλεσμα του παραπάνω προγράμματος:

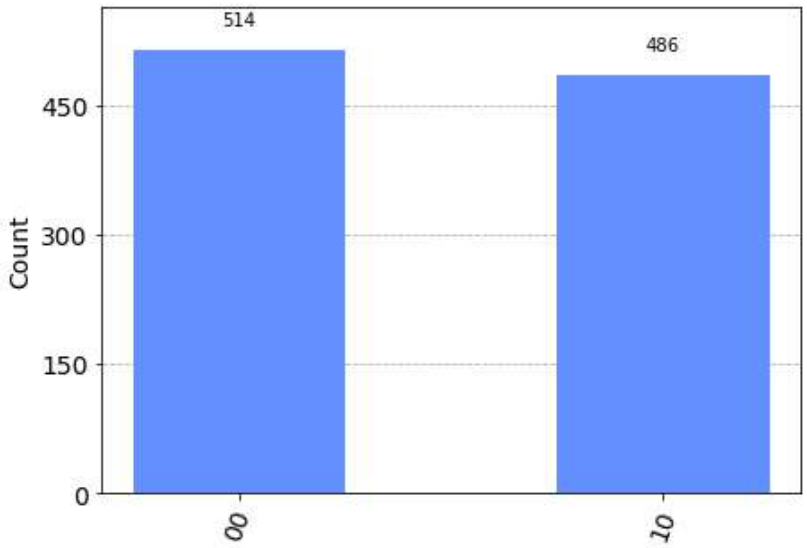


Σχήμα: 10 – Καταχωρητής με δύο qubits

Παρατήρηση

Όταν ένα πρόγραμμα το εκτελούμε σε πραγματικό κβαντικό υπολογιστή και όχι σε εξομοιωτή, τότε όσες περισσότερες φορές το εκτελέσουμε, τόσες περισσότερες είναι οι πιθανότητες το κύκλωμα να έχει καλύτερη προσέγγιση στην πραγματικότητα. Γι' αυτό και στο παραπάνω παράδειγμα γράψαμε να εκτελεσθεί το πρόγραμμα πολλές φορές (Γραμμή 15: 1.000 φορές).

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το ιστόγραμμα των κβαντικών καταστάσεων:



Σχήμα: 11 – Ιστόγραμμα κβαντικών καταστάσεων

Το παραπάνω ιστόγραμμα παρουσιάζει τις πιθανότητες εμφάνισης των κβαντικών καταστάσεων κατά την πρώτη εκτέλεση του κβαντικού κυκλώματος με τις χίλιες δοκιμές.

Χρησιμοποιώντας την ίδια μέθοδο, μπορούμε να συγκρίνουμε δύο διαφορετικές εκτελέσεις ενός κβαντικού κυκλώματος.

Ας δούμε τα δύο παραπάνω παραδείγματα (αρχικοποίηση των 2 qubits και των 3 qubits) προσπαθώντας να αναλύσουμε μαθηματικά αυτό το κβαντικό κύκλωμα.

Έχουμε προαναφέρει ότι όταν έχουμε παραπάνω από 2 qubits σε ένα κβαντικό καταχωρητή, τότε χρησιμοποιούμε το τανυστικό γινόμενο (⊗)των αντίστοιχων διανυσμάτων τους.

Επομένως, έχουμε:

$$|abc\rangle = |\alpha\rangle \otimes |\mathrm{b}\rangle \otimes |\mathrm{c}\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} \frac{3}{5} \\ \frac{4}{5} \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ 0 & \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \otimes \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \\ -1 \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \\ 0 \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \\ 0 \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \end{bmatrix} = = \frac{1}{\sqrt{2}} \begin{bmatrix} 0.6 \\ 0.8 \\ -0.6 \\ -0.8 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Άρα,

$$|abc\rangle = \frac{1}{\sqrt{2}} (0.6 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 0.6 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 0.8 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}).$$

Άρα,

$$|abc\rangle = \frac{1}{\sqrt{2}} (0.6\ |000\rangle + 0.8\ |001\rangle - 0.6\ |010\rangle - 0.8\ |011\rangle .$$

Για να είναι αποδεκτή η κβαντική κατάσταση θα πρέπει το άθροισμα των τετραγώνων των καταστάσεων να είναι ίσο με τη μονάδα.

Δηλαδή,

$$(\frac{0.6}{\sqrt{2}})^2 + (\frac{0.8}{\sqrt{2}})^2 + (-\frac{0.8}{\sqrt{2}})^2 + (-\frac{0.6}{\sqrt{2}})^2 = 0.18 + 0.32 + 0.18 + 0.32 = 1 .$$

Οι πιθανότητες εμφάνισης είναι τα $|000\rangle$, $|010\rangle$ με 0.18 και αντίστοιχα τα $|001\rangle$ και $|011\rangle$ με 0.32 όπως έδειξε και το αποτέλεσμα της εκτέλεσης του προγράμματος.

Προσοχή

Τα αποτελέσματα του ιστογράμματος διαβάζονται ανάποδα, δηλαδή από τα δεξιά προς τα αριστερά. Συνεπώς, το πρώτο qubit είναι αυτό που εμφανίζεται τελευταίο.

4.12 Κβαντικές Πύλες

Για να κατανοήσουμε καλύτερα τι είναι κβαντικές πύλες, ας θυμηθούμε πρώτα τις πύλες στους κλασικούς υπολογιστές. Οι λογικές πύλες στους κλασικούς υπολογιστές αποτελούν τον βασικό τρόπο χειρισμού των bits, είναι φυσικά συστήματα κατασκευασμένα από πυρίτιο και αποτελούνται από πολλά τρανζίστορ από τα οποία διαρρέουν οι πληροφορίες. Συγκεκριμένα, οι κλασικοί υπολογιστές αποτελούνται από αγωγούς και λογικές πύλες οι οποίες συγκροτούν κυκλώματα και επεξεργαστές. Οι αγωγοί μεταφέρουν την πληροφορία μέσω του ρεύματος από πύλη σε πύλη, οι οποίες την επεξεργάζονται και μετατρέπουν την πληροφορία κατά την είσοδό τους με βάσει του πίνακα αληθείας της λογικής πύλης.

Αντίθετα, στους κβαντικούς υπολογιστές, οι αντίστοιχες πύλες που ονομάζονται κβαντικές πύλες δεν είναι φυσικά συστήματα, αλλά αντιπροσωπεύουν μετασχηματισμούς, δράσεις που ασκούνται πάνω σε ένα ή πολλά qubits ή και σε κβαντικούς καταχωρητές. Οι κβαντικές λογικές πύλες δέχονται ως είσοδο ένα ή παραπάνω qubits και εξάγουν ένα ή παραπάνω qubits. Αυτό σημαίνει ότι έχουν τον ίδιο αριθμό εισόδων και εξόδων. Μια πύλη η οποία δρα σε n qubits αντιπροσωπεύεται από έναν ορθομοναδιαίο πίνακα διαστάσεων 2^k x 2^k. Το αποτέλεσμα της δράσης μιας πύλης σε qubit βρίσκεται με τον πολλαπλασιασμό του πίνακα της πύλης με τον πίνακα του διανύσματος κατάστασης. Οι κβαντικοί υπολογισμοί είναι δράσεις στα κβαντικά συστήματα και αντιπροσωπεύονται από τελεστές που έχουν ως αποτέλεσμα την περιστροφή διανυσμάτων στο χώρο του Hilbert και περιγράφονται με τη βοήθεια πινάκων. Τα διανύσματα αυτά παριστάνουν τις κβαντικές καταστάσεις των κβαντικών καταχωρητών. Συνεπώς, μια κβαντική πύλη είναι ένας τελεστής (operator) ο οποίος ενεργεί πάνω στα qubits. Έτσι, οι κβαντικές πύλες είναι τελεστές του χώρου Hilbert οι οποίες περιστρέφουν τα διανύσματα κατάστασης των qubits και των κβαντικών καταχωρητών χωρίς να αλλάζουν το μήκος τους, το οποίο είναι πάντα ίσο με τη μονάδα. Πολύ σημαντικό να επισημάνουμε ότι η πληροφορία αποθηκεύεται στα qubits ή στους κβαντικούς καταχωρητές παραμένοντας εκεί και δεν περνά μέσα από τις κβαντικές πύλες όπως στους κλασικούς υπολογιστές, αλλά εφαρμόζονται πάνω στα qubits ή στους καταχωρητές. Επομένως, οι κβαντικές πύλες μπορούν να αναπαρασταθούν ως πίνακες και η πραγματοποίησή τους μπορεί να γίνει με τον πολλαπλασιασμό πινάκων μεταξύ του πίνακα που αντιπροσωπεύει την κβαντική πύλη και του πίνακα n x 1 που αντιπροσωπεύει τα qubits (διάνυσμα στήλης).

Ένα παράδειγμα που αναφέραμε και σε προηγούμενο κεφάλαιο, παρομοιάζουμε ένα qubit με ένα νόμισμα την στιγμή που το στρίβουμε και αιωρείται στον αέρα, το οποίο μπορούμε με κάποιους μηχανισμούς που ονομάζονται πύλες να το επηρεάσουμε ώστε να μην έχουμε πάντα ίσες πιθανότητες εμφάνισης αποτελεσμάτων φυσικά όπως το επιθυμούμε εμείς. Σε αυτό το γεγονός βασίζονται όλοι οι κβαντικοί αλγόριθμοι, δηλαδή με ποιον τρόπο πρέπει να επηρεάσουμε τα quits ώστε να μας οδηγήσουν στην λύση του προβλήματος.

Οι κβαντικές πύλες πρέπει να είναι αντιστρέψιμες, οι οποίες έχουν αντιστοίχιση ένα προς ένα μεταξύ των καταστάσεων εισόδου και εξόδου, κάτι που αυτό δεν ισχύει στις κλασικές λογικές πύλες. Εφαρμόζοντας (ισοδύναμα) την πύλη για δεύτερη φορά στο ίδιο bit, τότε το σύστημα επιστρέφει στην αρχική του κατάσταση.

4.13 Κβαντικές Πύλες που δρουν σε ένα Qubit

Οι πύλες που δρουν σε ένα qubit περιστρέφουν το διάνυσμα κατάστασης του qubit, περιγράφοντας τη δράση μιας κβαντικής πύλης σε ένα qubit από έναν πίνακα 2x2 στοιχείων. Η κύρια λειτουργία τους είναι η περιστροφή του διανύσματος κατάστασης, το οποίο επιτυγχάνεται με τη μεταβολή των γωνιών φ και θ μέσω των πράξεων που εκτελούν οι πύλες πάνω στα qubits όπως θα δούμε στη συνέχεια. Ακόμη, αν με μια κβαντική πύλη αλλάξουμε την κατάσταση ενός qubit, μπορούμε να επαναφέρουμε το qubit στην αρχική του κατάσταση χρησιμοποιώντας την ίδια πύλη.

Οι περιστροφές που είναι δυνατό να συμβούν γενικά είναι άπειρες με αποτέλεσμα να υπάρχουν πάρα πολλές κβαντικές πύλες που δρουν σε ένα qubit. Εμείς θα ασχοληθούμε με τρεις πύλες, την πύλη Hadamard, την πύλη αδράνειας και την πύλη Pauli-X (NOT.

Ας δούμε αναλυτικά τη λειτουργία της κάθε πύλης.

4.14 Κβαντική Πύλη Hadamard

Η πύλη Hadamard δρα σε ένα μόνο qubit, συμβολίζεται με H και αποτελεί μία πολύ σημαντική πύλη καθώς και ένα βασικό και κρίσιμο εργαλείο για το σχεδιασμό κβαντικών κυκλωμάτων. Αυτό οφείλεται στο γεγονός ότι η πύλη έχει ως είσοδο της ένα μόνο qubit (που δε βρίσκεται σε κατάσταση υπέρθεσης), μετατρέποντάς το κατάλληλα ώστε να βρίσκεται σε μια τέτοια κατάσταση. Δηλαδή, μπορεί να μεταφέρει τα qubits από τις βασικές τους καταστάσεις, γίνοντας αντιληπτό από το ανθρώπινο μάτι, σε υπέρθεση καταστάσεων, που είναι χαρακτηριστικό μόνο των κβαντικών υπολογιστών. Έτσι, γίνεται αντιληπτή αυτή η σπουδαία σημασία που αποκτούν οι κβαντικοί υπολογιστές καθώς βασίζονται στην υπέρθεση των qubits για τις περισσότερες πράξεις που θα πραγματοποιηθούν.

Όμως δημιουργεί μια αβεβαιότητα για το αποτέλεσμα που λαμβάνουμε μετά τη μέτρηση του qubit και αυτό μπορεί να προσδιοριστεί μόνο πιθανολογικά.

Όταν η πύλη Hadamard δρα σε qubits που βρίσκεται σε μία από τις δύο καταστάσεις (0 ή 1), τότε τα θέτει σε μία κατάσταση που είναι υπέρθεση των βασικών

καταστάσεων, ενώ όταν δρα σε qubits που βρίσκονται σε υπέρθεση καταστάσεων, επιστρέφει τα qubits στις βασικές τους καταστάσεις.

Η πύλη Hadamard μαθηματικά περιγράφεται από τον ακόλουθο πίνακα:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Η επίδραση της πύλης Hadamard στις βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ ενός qubit, είναι:

- $H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle.$
- $H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle.$

Άρα, η πιθανότητα να εμφανιστεί μια από τις δύο βασικές καταστάσεις, είτε στην κατάσταση $|0\rangle$, είτε στην κατάσταση $|1\rangle$ (ύστερα από κάποια μέτρηση), είναι ίδια και έχει τιμή:

$$\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1^2}{(\sqrt{2})^2} = \frac{1}{2} = 0{,}5 = 50\ \%.$$

Στον παρακάτω πίνακα, βλέπουμε συνοπτικά τις ιδιότητες της πύλης Hadamard (H):

Κατάσταση των qubits πριν τη δράση της πύλης $\|q_1\rangle$	**Κατάσταση των qubits μετά τη δράση της πύλης $\|q_0\rangle$**
$\|0\rangle$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
$\|1\rangle$	$\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$
$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\|0\rangle$
$\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	$\|1\rangle$

Σχήμα: 12 – Πίνακας αντιστοίχισης της πύλης Hadamard

Συνεπώς, η πύλη Hadamard αντιστοιχίζει:

- την κατάσταση $|0\rangle$ στην κατάσταση $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
- την κατάσταση $|1\rangle$ στην κατάσταση $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
- την κατάσταση $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ στην κατάσταση $|0\rangle$
- την κατάσταση $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ στην κατάσταση $|1\rangle$

Παρατήρηση

Εάν εφαρμοσθούν δύο διαδοχικές πύλες Hadamard, τότε η δεύτερη πύλη (H) εξουδετερώνει την πρώτη πύλη (H).

Δηλαδή,

• $H(H\,|\,0\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}2\\ 0\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |\,0\rangle.$

• $H(H\,|\,1\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0\\ 2\end{bmatrix} = \begin{bmatrix}0\\ 1\end{bmatrix} = |\,1\rangle.$

Έτσι, ισχύει:

$(H\,|\,0\rangle) \longrightarrow |\,0\rangle.$

$(H\,|\,1\rangle) \longrightarrow |\,1\rangle.$

Οι πύλες Hadamard αναμένεται ότι θα αποτελέσουν το κύριο στοιχείο για τη διασύνδεση κλασικών και κβαντικών υπολογιστών.

• Προσθήκη Πυλών - Circuit

Όταν γράφουμε κώδικα και θέλουμε να σχεδιάσουμε μία πύλη, γράφουμε το όνομα του κυκλώματος που επιθυμούμε να χρησιμοποιήσουμε, παράδειγμα circuit. και δίπλα συνήθως γράφουμε το αρχικό γράμμα από το όνομα της πύλης που επιδρά στο qubit και μέσα σε παρένθεση τον αριθμό των qubit που εφαρμόζεται στην πύλη αυτή.

Συγκεκριμένα, για να σχεδιάσουμε σε ένα κύκλωμα μια πύλη Hadamard στο πρόγραμμα, συντάσσεται:

- circuit.h (n)

Δηλαδή, γράφουμε δίπλα από το όνομα του κυκλώματος circuit. το αρχικό γράμμα “h” της λέξη Hadamard και όπου n είναι ο αριθμός των qubit, σε ποιο qubit θέλουμε να εφαρμόζεται.

Να δούμε πώς συντάσσεται ένα πρόγραμμα για να μας εμφανίζει σχεδιαστικά την πύλη Hadamard.

Εφαρμογή 5

Θέλουμε να σχεδιάσουμε ένα κύκλωμα με ένα qubit (q), που να εφαρμόζεται σε αυτό η πύλη Hadamard, οπότε συντάσσουμε το παρακάτω πρόγραμμα:

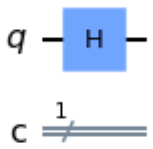
```
1. from qiskit import QuantumCircuit
2. from qiskit.visualization import *
```

```
3. circuit = QuantumCircuit(1,1)
4. circuit.h(0)
5. circuit.draw(output='mpl')
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 4: Προσθήκη πύλης Hadamard, με αριθμό qubit μηδέν.
- Γραμμή 5: Οπτικοποίηση του κυκλώματος . Εμφανίζει (σχεδιάζει) την συγκεκριμένη πύλη (H) του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το κύκλωμα με την πύλη Hadamard που εφαρμόζεται στο qubit q:



Σχήμα: 13 – Κύκλωμα με την πύλη Hadamard να επιδρά στο qubit

Η πύλη Hadamard επιδρά σε οποιαδήποτε κατάσταση ενός qubit.

Ας δούμε μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 8

Έχουμε μια κατάσταση α ενός qubit, δηλαδή $|\alpha\rangle = \begin{bmatrix} 0{,}6 \\ 0{,}8 \end{bmatrix}$ και επιδρά μια πύλη Hadamard.

Τότε, είναι:

$$H|\alpha\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0{,}6 \\ 0{,}8 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 0{,}6+0{,}8 \\ 0{,}6-0{,}8 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1{,}4 \\ -0{,}2 \end{bmatrix} = \begin{bmatrix} \frac{1{,}4}{\sqrt{2}} \\ -\frac{0{,}2}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0{,}99 \\ -0{,}14 \end{bmatrix} =$$

$$= \begin{bmatrix} 0{,}99 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -0{,}14 \end{bmatrix} = 0{,}99\begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0{,}14\begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0{,}99\,|0\rangle - 0{,}14\,|1\rangle.$$

Τετραγωνίζουμε τους συντελεστές των καταστάσεων $|0\rangle$ και $|1\rangle$ αντίστοιχα για να βρούμε την πιθανότητα εμφάνισής τους.

Είναι:

• $(0{,}99)^2 = 0{,}98$ και • $(0{,}14)^2 = 0{,}02$.

Επομένως, η πιθανότητα εμφάνισης στην κατάσταση $|0\rangle$ είναι 98% και η πιθανότητα εμφάνισης στην κατάσταση $|1\rangle$ είναι 2%.

Η υλοποίηση του και η μέτρηση του καταχωρητή δίνεται από το παρακάτω πρόγραμμα.

Έχουμε:

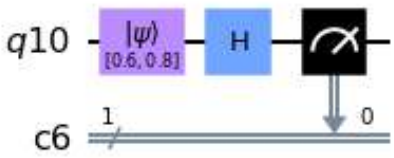
```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  %matplotlib inline
4.  initial_state = [0.6,0.8]
5.  qr = QuantumRegister(1)
6.  cr = ClassicalRegister(1)
7.  circuit = QuantumCircuit(qr,cr)
8.  circuit.initialize(initial_state,0)
9.  circuit.h(0)
10. circuit.measure(qr,cr)
11. circuit.draw(output='mpl')
12. circuit.measure(qr,cr)
13. simulator = Aer.get_backend('qasm_simulator')
14. result = execute(circuit,
backend=simulator,shots=1010).result()
15. plot_histogram(result.get_counts(circuit))
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 4: Ορισμός των τιμών των ζητούμενων αρχικών καταστάσεων στις αντίστοιχες μεταβλητές.
- Γραμμή 5: Δημιουργία κβαντικού καταχωρητή, ο οποίος αποτελείται από 1 qubit (το πλήθος των qubits μπορεί να αποδίδεται και από μια μεταβλητή).
- Γραμμή 6: Δημιουργία κλασικού καταχωρητή, ο οποίος αποτελείται από 1 bit (το πλήθος των bits μπορεί να αποδίδεται και από μια μεταβλητή).
- Γραμμή 7: Δημιουργία κυκλώματος από 1 qubit και 1 bit.
- Γραμμή 8: Αρχικοποίηση των qubits.
- Γραμμή 9: Εφαρμόζουμε την πύλη Hadamard και αυξάνεται στην συνέχεια ο αριθμός ων qubits που θέλουμε να εφαρμοστεί.
- Γραμμή 10: Μέτρηση του κυκλώματος. (Τα qubits στην ουσία καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα bits.

- Γραμμή 11: Οπτικοποίηση του κυκλώματος. Εμφανίζει (σχεδιάζει) την συγκεκριμένη πύλη (H) του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.
- Γραμμή 12: Τελειώνουμε το πρόγραμμά μας εδώ για να δούμε τα αποτελέσματά του, ώστε να εμφανισθεί το γράφημα του κυκλώματος και στη συνέχεια γράφουμε το υπόλοιπο μέρος του προγράμματος.
- Γραμμή 13: Επιλογή του προσομοιωτή.
- Γραμμή 14: Αποδίδουμε το αποτέλεσμα της εκτέλεσης του κυκλώματος στην μεταβλητή result και όπως είδαμε, επειδή το πρόγραμμά μας δεν εκτελείται σε πραγματικό υπολογιστή αλλά σε προσομοιωτή, το «τρέχουμε» πολλές φορές για να έχουμε καλύτερη πιθανολογική προσέγγιση του αποτελέσματος (ζητάμε από το πρόγραμμα να εκτελεσθεί τυχαία 1010 φορές το κύκλωμα).
- Γραμμή 15: Εμφανίζεται το αποτέλεσμα του προγράμματος με τη μορφή ιστογράμματος. Παρατηρούμε στο γράφημα τις πολύ μικρές διαφορές στις δύο μετρήσεις (της τάξης 3%).

Όταν «τρέξουμε» το πρόγραμμα μέχρι και τη γραμμή 11, θα εμφανιστεί το αποτέλεσμα:

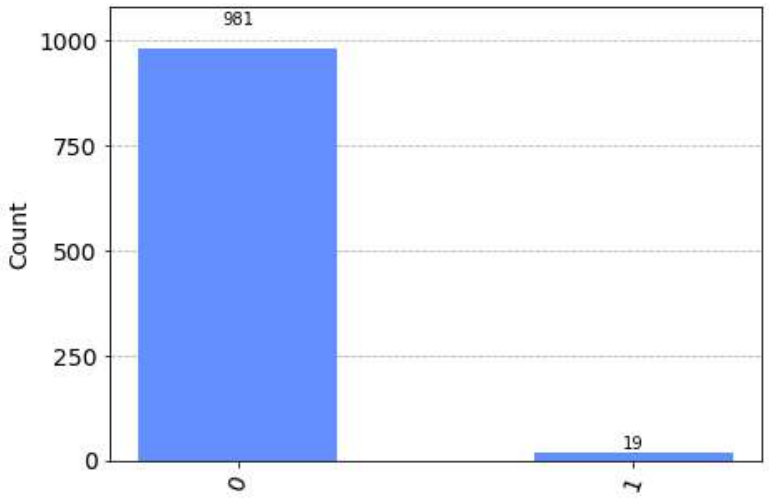


Σχήμα: 14 – Κύκλωμα με την πύλη Hadamard να επιδρά στο 1ο qubit που βρίσκεται σε κατάσταση $|\alpha\rangle$ και πύλη αδράνειας στο 2ο qubit

Όταν «τρέξουμε» ολόκληρο το πρόγραμμα (μέχρι και τη γραμμή 15), θα εμφανιστεί το ιστόγραμμα με την πύλη Hadamard.

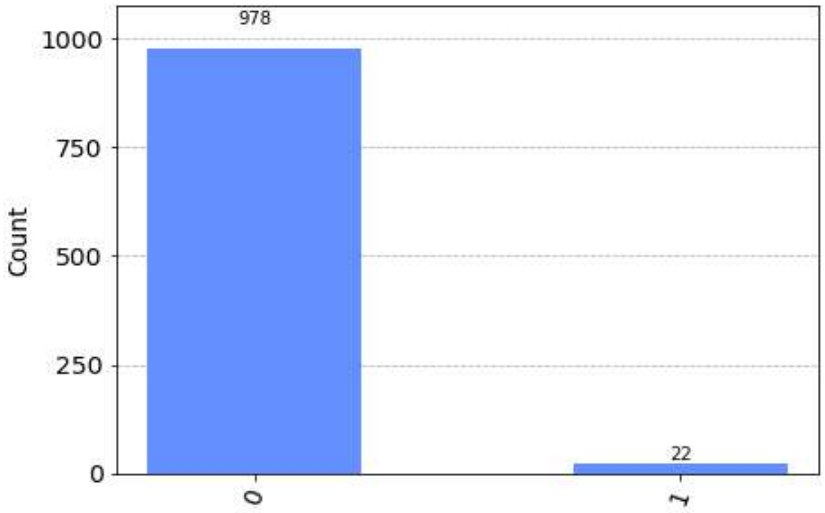
Τα αποτελέσματα μετά από δύο μετρήσεις, διακρίνονται οι μικρές διαφορές οι οποίες φαίνονται παρακάτω:

1η Μέτρηση:



Σχήμα: 15 – Ιστόγραμμα αποτελεσμάτων εφαρμογής πύλης Hadamard 1η μέτρηση

2η Μέτρηση:



Σχήμα: 16 – Ιστόγραμμα αποτελεσμάτων εφαρμογής πύλης Hadamard 2η μέτρηση

ΠΑΡΑΔΕΙΓΜΑ 9

Έχουμε μια κατάσταση κ ενός qubit, δηλαδή $|\kappa\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ και επιδρά μια πύλη Hadamard. Να υπολογιστεί η εφαρμογή της πύλης (H) στο qubit.

Έχουμε:

Η εφαρμογή μιας πύλης Hadamard σε ένα qubit δίνεται από τον υπολογισμό του γινομένου των παρακάτω πινάκων:

$$H\,|\,\kappa\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1^2}{(\sqrt{2})^2}\begin{bmatrix}\frac{1}{\sqrt{2}}\\ -\frac{0{,}2}{\sqrt{2}}\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1+1\\ 1-1\end{bmatrix} =$$

$$= \frac{1}{2}\begin{bmatrix}2\\ 0\end{bmatrix} = \begin{bmatrix}\frac{2}{2}\\ \frac{0}{2}\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |\,0\rangle.$$

Έτσι το αποτέλεσμα της μέτρησης είναι $|\,0\rangle$, δηλαδή με πιθανότητα εμφάνισης 100%.

4.15 Δύο Qubit – Εφαρμόζεται πύλη μόνο στο ένα Qubit

4.15.1 Κβαντική Πύλη Αδράνειας (I)

Όταν έχουμε ένα κύκλωμα με δύο qubit και μόνο στο πρώτο έχουμε μια δράση, εφαρμόζεται για παράδειγμα μια πύλη Hadamard και στο δεύτερο qubit δεν έχουμε καμία δράση, τότε στο πρώτο qubit όπως είδαμε πραγματοποιούμε τις παραπάνω πράξεις.

Στο δεύτερο qubit αφού δεν εφαρμόζεται κάτι, τότε χρησιμοποιούμε μια συγκεκριμένη πύλη που ονομάζεται πύλη αδράνειας ώστε να μπορέσουμε να πραγματοποιήσουμε και να επιλύσουμε τις πράξεις μας στα κβαντικά κυκλώματα.

Η πύλη αδράνειας επιδρά σε ένα qubit, η οποία περιγράφει έναν τελεστή και το χαρακτηριστικό της πύλης αυτής είναι ότι αφήνει αμετάβλητη την κατάσταση του qubit. Η πύλη αδράνειας καλείται και ως μοναδιαίος πίνακας γι' αυτό και ονομάζεται αλλιώς μοναδιαία πύλη και συμβολίζεται με (I).

Η πύλη αδράνειας (I) μαθηματικά περιγράφεται από τον ακόλουθο πίνακα:

$$I = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \text{ και ισχύει και } I\,|\,q\,\rangle = |\,q\rangle.$$

Πύλες σαν αυτή περιστρέφουν το διάνυσμα κατάστασης ενός qubit μέσα στη σφαίρα Bloch, δηλαδή μεταβάλουν τις γωνίες θ και φ.

Παρακάτω μπορούμε να διαπιστώσουμε ότι πραγματικά η πύλη αδράνειας (I) δεν μεταβάλει την κατάσταση του qubit στο οποίο επιδρά.

Δηλαδή,

- $I\,|\,0\rangle = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |\,0\rangle.$

• $\mathrm{I}\,|1\rangle = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} = |1\rangle.$

Στον παρακάτω πίνακα, βλέπουμε συνοπτικά τις ιδιότητες της πύλης αδράνειας (I):

Κατάσταση των qubits πριν τη δράση της πύλης $\|q_1\rangle$	**Κατάσταση των qubits μετά τη δράση της πύλης** $\|q_0\rangle$
$\|\mathbf{0}\rangle$	$\|\mathbf{0}\rangle$
$\|\mathbf{1}\rangle$	$\|\mathbf{1}\rangle$
$\|q\rangle$	$\|q\rangle$

Σχήμα: 17 – Πίνακας αντιστοίχισης της πύλης Αδράνειας (I)

Συνεπώς, η πύλη αδράνειας (I) αντιστοιχίζει:

- την κατάσταση $|0\rangle$ στην κατάσταση $|0\rangle$.
- την κατάσταση $|1\rangle$ στην κατάσταση $|1\rangle$.
- την κατάσταση $|q\rangle$ στην κατάσταση $|q\rangle$.

Για να σχεδιάσουμε σε ένα κύκλωμα την πύλη αδράνειας, στο πρόγραμμα συντάσσεται:

- circuit.i (n)

Δηλαδή, γράφουμε δίπλα από το όνομα του κυκλώματος circuit. το γράμμα "i" με το οποίο συμβολίζουμε την πύλη αδράνειας και όπου n είναι ο αριθμός των qubit που θέλουμε να εφαρμοστεί.

Ας δούμε μερικά παραδείγματα.

Εφαρμογή 6

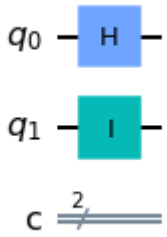
Θέλουμε να σχεδιάσουμε ένα κύκλωμα με δύο qubit, που στο πρώτο να εφαρμόζεται η πύλη Hadamard και στο δεύτερο η πύλη αδράνειας (I), οπότε συντάσσουμε το παρακάτω πρόγραμμα:

```
1. from qiskit import QuantumCircuit
2. from qiskit.visualization import *
3. circuit = QuantumCircuit(2,2)
4. circuit.h(0)
5. ciscuit.i(1)
6. circuit.draw(output='mpl')
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 4: προσθήκη πύλης Hadamard, με αριθμό qubit μηδέν.
- Γραμμή 5: εμφανίζει (σχεδιάζει) την συγκεκριμένη πύλη αδράνειας (I) του προγράμματος.
- Γραμμή 6: εμφανίζει (σχεδιάζει) τις συγκεκριμένες πύλες (H) και (I) αντίστοιχα του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το κύκλωμα με τις πύλες Hadamard και αδράνειας (I) που εφαρμόζονται στα qubit q0 και q1 αντίστοιχα:



Σχήμα: 18 – Κύκλωμα με την πύλη Hadamard να επιδρά στο 1ο qubit και πύλη αδράνειας στο 2ο qubit

ΠΑΡΑΔΕΙΓΜΑ 10

Έχουμε δύο qubit, από τα οποία το πρώτο βρίσκεται σε μια κατάσταση α, δηλαδή $|\alpha\rangle = \begin{bmatrix} 0{,}6 \\ 0{,}8 \end{bmatrix}$ που επιδρά σε αυτό μια πύλη Hadamard και το δεύτερο qubit βρίσκεται σε μια κατάσταση b, δηλαδή $|b\rangle = \begin{bmatrix} 0{,}17 \\ 0{,}83 \end{bmatrix}$.

Τότε, για το πρώτο qubit έχουμε:

• $H|\alpha\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0{,}6 \\ 0{,}8 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 0{,}6+0{,}8 \\ 0{,}6-0{,}8 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1{,}4 \\ -0{,}2 \end{bmatrix} = \begin{bmatrix} \frac{1{,}4}{\sqrt{2}} \\ -\frac{0{,}2}{\sqrt{2}} \end{bmatrix} = = \begin{bmatrix} 0{,}99 \\ -0{,}14 \end{bmatrix} =$

$\begin{bmatrix} 0{,}99 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -0{,}14 \end{bmatrix} = 0{,}99\begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0{,}14\begin{bmatrix} 0 \\ 1 \end{bmatrix} =$

$= 0{,}99\,|0\rangle - 0{,}14\,|1\rangle.$

Για το δεύτερο qubit στο οποίο δεν επιδρά κάτι, τότε εφαρμόζουμε την πύλη αδράνειας και έχουμε:

• $I|b\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0{,}17 \\ 0{,}83 \end{bmatrix} = \begin{bmatrix} 0{,}17+0 \\ 0+0{,}83 \end{bmatrix} = \begin{bmatrix} 0{,}17 \\ 0{,}83 \end{bmatrix} = 0{,}17\begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0{,}83\begin{bmatrix} 0 \\ 1 \end{bmatrix} =$

$= 0{,}17\,|0\rangle + 0{,}83\,|1\rangle$.

Τετραγωνίζουμε τους συντελεστές των καταστάσεων $|0\rangle$ και $|1\rangle$ αντίστοιχα για να βρούμε την πιθανότητα εμφάνισής τους.
Έχουμε:

• $(0{,}17)^2 = 0{,}30$ και • $(0{,}83)^2 = 0{,}70$.

Επομένως, η πιθανότητα εμφάνισης του $|0\rangle$ είναι 30% και η πιθανότητα του $|1\rangle$ είναι 70%.

Η υλοποίηση του και η μέτρηση του καταχωρητή δίνεται από το παρακάτω πρόγραμμα.
Έχουμε:

```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  %matplotlib inline
4.  initial_state = [0.6,0.8]
5.  qr = QuantumRegister(2)
6.  cr = ClassicalRegister(2)
7.  circuit = QuantumCircuit(qr,cr)
8.  circuit.initialize(initial_state,0)
9.  circuit.h(0)
10. circuit.i(1)
11. circuit.measure(qr,cr)
12. circuit.draw(output='mpl')
13. simulator = Aer.get_backend('qasm_simulator')
14. result = execute(circuit,
backend=simulator,shots=1024).result()
15. plot_histogram(result.get_counts(circuit))
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

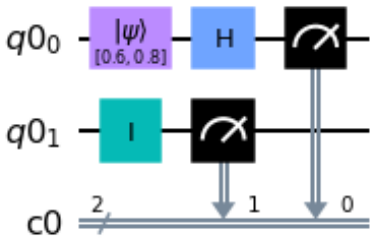
- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 4: Ορισμός των τιμών των ζητούμενων αρχικών καταστάσεων στις αντίστοιχες μεταβλητές.
- Γραμμή 5: Δημιουργία κβαντικού καταχωρητή, ο οποίος αποτελείται από 2 qubit (το πλήθος των qubits μπορεί να αποδίδεται και από μια μεταβλητή).
- Γραμμή 6: Δημιουργία κλασικού καταχωρητή, ο οποίος αποτελείται από 2 bit (το πλήθος των bits μπορεί να αποδίδεται και από μια μεταβλητή).
- Γραμμή 7: Δημιουργία κυκλώματος από 2 qubit και 2 bit.
- Γραμμή 8: Αρχικοποίηση των qubits.
- Γραμμή 9: Εφαρμόζουμε την πύλη Hadamard και αυξάνεται στην συνέχεια ο αριθμός ων qubits που θέλουμε να εφαρμοστεί.

- Γραμμή 10: Εφαρμόζουμε τη μοναδιαία πύλη αδράνειας (I) .
- Γραμμή 11: Μέτρηση του κυκλώματος. (Τα qubits στην ουσία καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα bits.
- Γραμμή 12: Οπτικοποίηση του κυκλώματος. Εμφανίζει (σχεδιάζει) την συγκεκριμένη πύλη (H) του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.

Τελειώνουμε το πρόγραμμά μας εδώ για να δούμε τα αποτελέσματά του, ώστε να εμφανισθεί το γράφημα του κυκλώματος και στη συνέχεια γράφουμε το υπόλοιπο μέρος του προγράμματος.

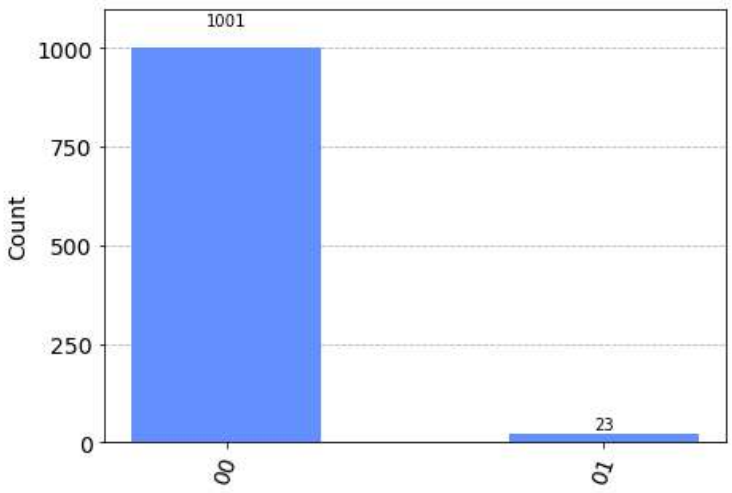
- Γραμμή 13: Επιλογή του προσομοιωτή.
- Γραμμή 14: Αποδίδουμε το αποτέλεσμα της εκτέλεσης του κυκλώματος στην μεταβλητή result και όπως είδαμε, επειδή το πρόγραμμά μας δεν εκτελείται σε πραγματικό υπολογιστή αλλά σε προσομοιωτή, το «τρέχουμε» πολλές φορές για να έχουμε καλύτερη πιθανολογική προσέγγιση του αποτελέσματος (ζητάμε από το πρόγραμμα να εκτελεσθεί τυχαία 1024 φορές το κύκλωμα).
- Γραμμή 15: Εμφανίζεται το αποτέλεσμα του προγράμματος με τη μορφή ιστογράμματος. Παρατηρούμε στο γράφημα τις πολύ μικρές διαφορές στις δύο μετρήσεις (της τάξης 3%).

Όταν «τρέξουμε» το πρόγραμμα μέχρι και τη γραμμή 11, θα εμφανιστεί το αποτέλεσμα:



Σχήμα: 19 – Κύκλωμα με την πύλη Hadamard να επιδρά στο 1ο qubit που βρίσκεται σε κατάσταση $|\alpha\rangle$ και πύλη αδράνειας στο 2ο qubit

Όταν «τρέξουμε» ολόκληρο το πρόγραμμα (μέχρι και τη γραμμή 15), θα εμφανιστεί το αποτέλεσμα:



Σχήμα: 20 – Ιστόγραμμα αποτελεσμάτων εφαρμογής πύλης Hadamard στο 1ο qubit

Σημείωση

Όταν σε ένα κύκλωμα έχουμε δύο qubit τα οποία βρίσκονται σε κάποια κατάσταση και στο πρώτο ασκείται μία δράση και στο δεύτερο qubit δεν επιδρά κάτι, στην ουσία αυτό το δεν επιδρά κάτι δεν ισχύει στο χώρο των κβαντικών. Εφαρμόζουμε την πύλη αδράνειας (I) η οποία μας βοηθάει στην επίλυση κβαντικών κυκλωμάτων .

ΠΑΡΑΔΕΙΓΜΑ 11

Έχουμε ένα κύκλωμα με δύο qubits, με αρχική τιμή $|0\rangle$. Τότε:
α). Να εφαρμόσουμε μια πύλη Hadamard στο πρώτο qubit.
β). Να εφαρμόσουμε μια πύλη Hadamard πάλι στο πρώτο.
γ). Να υπολογιστεί το αποτέλεσμα μετά την μέτρηση του πιο πάνω κυκλώματος.
δ). Να γράψουμε πρόγραμμα ώστε να εμφανίζεται το παραπάνω κύκλωμα.

Έχουμε:

Τα δύο qubit έχουν αρχική κατάσταση $|0\rangle$ και βρίσκουμε τη μορφή του κβαντικού καταχωρητή, υπολογίζοντας το τανυστικό γινόμενων των πινάκων :

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0\begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

α). Εφαρμόζουμε Hadamard σε πρώτο qubit και τίποτα στο δεύτερο qubit, το οποίο ισοδυναμεί με πύλη αδράνειας στο δεύτερο qubit, οπότε έχουμε:

$$H \otimes I = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & -1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix} 1\cdot 1 & 1\cdot 0 & 1\cdot 1 & 1\cdot 0 \\ 1\cdot 0 & 1\cdot 1 & 1\cdot 0 & 1\cdot 1 \\ 1\cdot 1 & 1\cdot 0 & -1\cdot 1 & -1\cdot 0 \\ 1\cdot 0 & 1\cdot 1 & -1\cdot 0 & -1\cdot 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

Εφαρμόζουμε το παραπάνω αποτέλεσμα στο αποτέλεσμα της αρχικής κατάστασης και έτσι υπολογίζουμε το γινόμενο των παρακάτω πινάκων:

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1\cdot 1 + 0\cdot 0 + 1\cdot 0 + 0\cdot 0 \\ 0\cdot 1 + 1\cdot 0 + 0\cdot 0 + 1\cdot 0 \\ 1\cdot 1 + 0\cdot 0 - 1\cdot 0 + 0\cdot 0 \\ 0\cdot 1 + 1\cdot 0 + 0\cdot 0 - 1\cdot 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}}\left(\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}\left(|00\rangle + |10\rangle\right).$$

β). Εφαρμόζουμε Hadamard σε πρώτο qubit και τίποτα στο δεύτερο qubit, δηλαδή αυτό σημαίνει ότι εφαρμόζουμε την πύλη αδράνειας στο δεύτερο qubit και ο υπολογισμός είναι ο ίδιος όπως φαίνεται παρακάτω:

$$H \otimes I = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

Εφαρμόζουμε τον παραπάνω υπολογισμό στην προηγούμενη κατάσταση και έχουμε:

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \left(\frac{1}{\sqrt{2}}\right)^2 \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} =$$

$$= \frac{1}{(\sqrt{2})^2}\begin{bmatrix} 1\cdot 1 + 0\cdot 0 + 1\cdot 1 + 0\cdot 0 \\ 0\cdot 1 + 1\cdot 0 + 0\cdot 1 + 1\cdot 0 \\ 1\cdot 1 + 0\cdot 0 + (-1)\cdot 1 + 0\cdot 0 \\ 0\cdot 1 + 1\cdot 0 + 0\cdot 1 - 1\cdot 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |00\rangle.$$

γ). Έτσι το αποτέλεσμα της μέτρησης είναι:

$1|00\rangle + 0|01\rangle + 0|10\rangle + 0|11\rangle.$

Δηλαδή οι υπόλοιπες βασικές καταστάσεις δεν υπάρχουν. Έχουμε 100% πιθανότητα εμφάνισης της κατάστασης $|00\rangle$ (μετά από τις παραπάνω δράσεις πραγματοποιήσαμε).

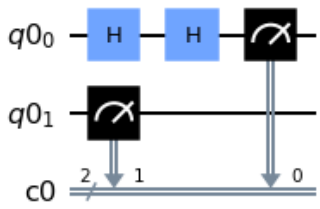
δ). Θέλουμε να σχεδιάσουμε ένα κύκλωμα το παραπάνω κύκλωμα, οπότε συντάσσουμε το παρακάτω πρόγραμμα:

```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  %matplotlib inline
4.  qr = QuantumRegister(2)
5.  cr = ClassicalRegister(2)
6.  circuit = QuantumCircuit(qr,cr)
7.  circuit.h(qr[0])
8.  circuit.h(qr[0])
9.  circuit.measure(qr,cr)
10. circuit.draw(output='mpl')
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 4: Δημιουργία κβαντικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των qubits και με μια μεταβλητή) .
- Γραμμή 5: Δημιουργία κλασικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των bits και με μια μεταβλητή) .
- Γραμμή 6: Δημιουργία κυκλώματος από 2 qubit και 2 bits.
- Γραμμές 7 και 8: Προσθήκη δύο πύλες Hadamard, με αριθμό qubit μηδέν.
- Γραμμή 9: Τελειώνουμε το πρόγραμμά μας εδώ για να δούμε τα αποτελέσματά του, ώστε να εμφανισθεί το γράφημα του κυκλώματος και στη συνέχεια γράφουμε το υπόλοιπο μέρος του προγράμματος.
- Γραμμή 10: Οπτικοποίηση του κυκλώματος. Εμφανίζει (σχεδιάζει) τις συγκεκριμένες πύλες (H) και (H) αντίστοιχα του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το ζητούμενο κύκλωμα:



Σχήμα: 21 – Κύκλωμα με δύο πύλες Hadamard στο 1ο qubit και καμία δράση στο 2ο qubit

4.16 Κβαντική Πύλη Pauli-X-Y-Z (πύλη NOT)

Η κβαντική πύλη Pauli (ή πύλη NOT) είναι η αντίστοιχη της πύλης NOT των κλασικών υπολογιστών και ανήκει στην ομάδα των τεσσάρων πυλών η οποία δρα σε ένα qubit και συμβολίζεται ανάλογα με τα γράμματα X, Y, Z (NOT). Στην ουσία περιστρέφουν το διάνυσμα του qubit γύρω από τους άξονες X, Y, Z (στη σφαίρα Bloch).

Ας δούμε ξεχωριστά τις πύλες Pauli X-Y-Z.

• Η πύλη Pauli-X (NOT)

Η κβαντική πύλη Pauli-X αντιστρέφει την κατάσταση του qubit και στην ουσία, πρόκειται για μια περιστροφή του διανύσματος γύρω από τον άξονα X κατά 180 μοίρες (π ακτίνια), γι' αυτό και συμβολίζεται με X.

Η πύλη Pauli-X (NOT) μαθηματικά περιγράφεται από τον ακόλουθο πίνακα:

$$H = NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Η επίδραση της πύλης Pauli-X (NOT) στις βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ ενός qubit, είναι:

• $X|0\rangle = \text{NOT}|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle.$

• $\text{X}|1\rangle = \text{NOT}|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle.$

Στον παρακάτω πίνακα, βλέπουμε συνοπτικά τις ιδιότητες της πύλης Pauli-X (NOT) :

Κατάσταση των qubits πριν τη δράση της πύλης $\|q_1\rangle$	**Κατάσταση των qubits μετά τη δράση της πύλης** $\|q_0\rangle$
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$

Σχήμα: 22 – Πίνακας αντιστοίχισης της πύλης Pauli-X (NOT)

Συνεπώς, η πύλη Pauli-X (NOT) αντιστοιχίζει:

- την κατάσταση $|0\rangle$ στην κατάσταση $|1\rangle$
- την κατάσταση $|1\rangle$ στην κατάσταση $|0\rangle$.

• Η πύλη Pauli-Y

Η κβαντική πύλη Pauli-Y αντιστρέφει σε περιστροφή του διανύσματος την κατάσταση του qubit κατά 180 μοίρες (π ακτίνια), γύρω από τον άξονα Y, γι' αυτό και συμβολίζεται με Y. Στην ουσία μετατρέπει το διάνυσμα από την κατάσταση $|0\rangle$ σε $i|1\rangle$ και την κατάσταση $|1\rangle$ σε $i|0\rangle$.

Η πύλη Pauli-Y (NOT) μαθηματικά περιγράφεται από τον ακόλουθο πίνακα:

$$Y = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix}.$$

Η επίδραση της πύλης Pauli-X (NOT) στις βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ ενός qubit, είναι:

• $Y|0\rangle = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} . \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix} = 0|0\rangle + i|1\rangle = i|1\rangle .$

• $Y|1\rangle = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} . \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix} = -i|0\rangle + 0|1\rangle = -i|0\rangle .$

Στον παρακάτω πίνακα, βλέπουμε συνοπτικά τις ιδιότητες της πύλης Pauli-Y:

Κατάσταση των qubits πριν τη δράση της πύλης $\|q_1\rangle$	**Κατάσταση των qubits μετά τη δράση της πύλης $\|q_0\rangle$**
$\|\mathbf{0}\rangle$	$\boldsymbol{i}\|\mathbf{1}\rangle$
$\|\mathbf{1}\rangle$	$-\boldsymbol{i}\|\mathbf{0}\rangle$

Σχήμα: 23 – Πίνακας αντιστοίχισης της πύλης Pauli-X (NOT)

Συνεπώς, η πύλη Pauli-Y αντιστοιχίζει:

- την κατάσταση $|0\rangle$ στην κατάσταση $i|1\rangle$
- την κατάσταση $|1\rangle$ στην κατάσταση $-i|0\rangle$.

• Η πύλη Pauli-Z

Η κβαντική πύλη Pauli-Z περιστρέφει το διάνυσμα κατά 180 μοίρες (π ακτίνια), γύρω από τον άξονα Z, γι' αυτό και συμβολίζεται με Z. Στην ουσία, αφήνει αμετάβλητη την κατάσταση $|0\rangle$ σε $i|1\rangle$ και μετατρέπει την κατάσταση $|1\rangle$ σε $-|1\rangle$.

Η πύλη Pauli-Z μαθηματικά περιγράφεται από τον ακόλουθο πίνακα:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Η επίδραση της πύλης Pauli-X (NOT) στις βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ ενός qubit, είναι:

• $Z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1|0\rangle + 0|1\rangle = |0\rangle$.

• $Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = 0|0\rangle + (-1)|1\rangle = -1|1\rangle$.

Στον παρακάτω πίνακα, βλέπουμε συνοπτικά τις ιδιότητες της πύλης Pauli-X (NOT) :

Κατάσταση των qubits πριν τη δράση της πύλης $\|q_1\rangle$	**Κατάσταση των qubits μετά τη δράση της πύλης** $\|q_0\rangle$
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-1\|1\rangle$

Σχήμα: 24 – Πίνακας αντιστοίχισης της πύλης Pauli-X (NOT)

Συνεπώς, η πύλη Pauli-X (NOT) αντιστοιχίζει:

- την κατάσταση $|0\rangle$ στην κατάσταση $|0\rangle$
- την κατάσταση $|1\rangle$ στην κατάσταση $-1|1\rangle$.

4.17 Κβαντικές Πύλες που δρουν δύο Qubit

Οι κβαντικές πύλες μπορεί να δράσουν σε περισσότερα από ένα qubit. Εμείς θα ασχοληθούμε με τις κβαντικές πύλες που δρουν μόνο σε δύο και τρία qubits και συγκεκριμένα, θα ασχοληθούμε με δύο πύλες, την πύλη CNOT (cX) και την πύλη Swap.

4.18 Σχεδιασμός σφαίρας Bloch - Επίδραση πυλών στη σφαίρα Bloch

Βιβλιοθήκη QutiP

Για να σχεδιάσουμε ή να περιστρέψουμε μια σφαίρα Bloch, να απεικονίσουμε διανύσματα και πολλά άλλα, χρησιμοποιούμε τη βιβλιοθήκη qutip. Η βιβλιοθήκη QuTiP γενικά προσφέρει πάρα πολλά εργαλεία σχεδίαση που αφορούν τη σφαίρα Bloch.

Προσοχή

Σε περίπτωση που δεν πραγματοποιήθηκε η εγκατάσταση της βιβλιοθήκης QuTiP, πληκτρολογούμε στο πλαίσιο του Jupyter Notebook την εντολή:

```
pip install qutip
```

Ας δούμε μερικά παράδειγμα.

ΠΑΡΑΔΕΙΓΜΑ 1

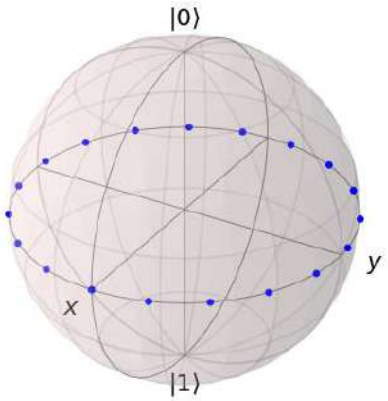
Να δημιουργήσουμε πρόγραμμα που να εμφανίζει μια απλή γραφική απεικόνιση της σφαίρα Bloch, αποτυπώνοντας κάποια σημεία, αλλά και των βασικών καταστάσεων της κβαντικής αναπαράστασης μνήμης που συμβολίζονται με $|0\rangle$ και $|1\rangle$. Τότε το πρόγραμμα συντάσσεται ως εξής:

```
1.  import qutip as qt
2.  import numpy as np
3.  bloch = qt.Bloch()
4.  bloch.make_sphere()
5.  gonia = np.linspace(0, 2*np.pi, 20)
6.  x = np.cos(gonia)
7.  y = np.sin(gonia)
8.  z = np.zeros(20)
9.  simeia=[x,y,z]
10. bloch.add_points(simeia)
11. bloch.show()
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1 και 2: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμές 3 και 4: Δημιουργία σφαίρας Bloch.
- Γραμμές 5, 6, 7, 8, 9 και 10: Εμφάνιση σημείων επί της σφαίρας.
- Γραμμή 11: Εμφάνιση γραφήματος σφαίρας Bloch

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το ζητούμενο κύκλωμα:



Σχήμα: 25 – Σφαίρα Bloch - qubit

ΠΑΡΑΔΕΙΓΜΑ 2

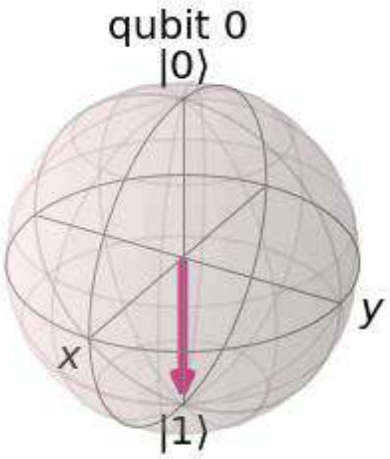
Να εμφανιστεί η σφαίρα Bloch που επιδρούν οι πύλες X, Y, Z στα $|0\rangle$ και $|1\rangle$ συντάσσεται ως εξής:

```
1.  from qiskit import *
2.  from qiskit.quantum_info import Statevector
3.  from math import pi
4.  from qiskit.visualization import plot_bloch_multivector
5.  qc = QuantumCircuit(1,1)
6.  qc.x(0)
7.  state = Statevector.from_instruction(qc)
8.  plot_bloch_multivector(state)
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2, 3 και 4: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 5: Δημιουργία κυκλώματος από 1 qubit και 1 bits.
- Γραμμή 6: Δημιουργία πύλης X.
- Γραμμή 8: Εμφάνιση γραφήματος σφαίρας Bloch

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το ζητούμενο κύκλωμα:



Σχήμα: 26 – Σφαίρα Bloch μετά από επίδραση της πύλης X

4.19 Κβαντική Πύλη Ελεγχόμενης Άρνησης CNOT (cX)

Η κβαντική πύλη ελεγχόμενης άρνησης, με αγγλικό όρο της πύλης Controlled-Not και συμβολίζεται από τα αρχικά της με (CNOT). Η πύλη αυτή (CNOT) δρα σε δύο qubits, δηλαδή δέχεται για είσοδο δύο qubits και εξάγει δύο qubits. Το πρώτο qubit εισόδου ονομάζεται qubit ελέγχου (control qubit) και συμβολίζεται με c και το άλλο qubit καλείται στόχος (target qubit) και συμβολίζεται με t. Οι καταστάσεις των δύο qubits πριν τη δράση της πύλης είναι $|c_1\rangle$ και $|t_1\rangle$, ενώ μετά τη δράση της πύλης είναι $|c_0\rangle$ και $|t_0\rangle$.

Η πύλη CNOT αλλάζει την κατάσταση του qubit στόχου (target qubit), όταν η κατάσταση του qubit ελέγχου είναι $|1\rangle$, ενώ αφήνει αμετάβλητη την κατάσταση του qubit στόχου όταν η κατάσταση του qubit ελέγχου είναι $|0\rangle$. δηλαδή όταν η κατάσταση του qubit ελέγχου είναι $|0\rangle$ δεν αλλάζει την κατάσταση του qubit, ενώ όταν η κατάσταση είναι $|1\rangle$ τότε την αλλάζει.

Η κατάσταση του πρώτου qubit (qubit ελέγχου) $|c_1\rangle$ δε μεταβάλλεται και ισχύει πάντα $|c_1\rangle = |c_0\rangle$.

Αν οι τιμές εισόδου είναι μία από τις βασικές καταστάσεις $|0\rangle$ και $|1\rangle$ και για τα δύο qubits, η έξοδος του δεύτερου qubit (target qubit) της πύλης CNOT αντιστοιχεί στο αποτέλεσμα της κλασικής πύλης XOR (exclusive OR). Η λογική πράξη της XOR συμβολίζεται ως ”⊕”.

Το σύμβολο της κβαντικής πύλης CNOT είναι:

Σχήμα: 27 – Σύμβολο κβαντικής πύλης CNOT

Η κβαντική πύλη ελέγχου CNOT μαθηματικά περιγράφεται από ακόλουθο πίνακα:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Η γενική περιγραφή της δράσης της πύλης είναι:

$\text{CNOT} \left| c_1 t_1 \right\rangle = \left| c_0 t_0 \right\rangle$.

Η επίδρασή της πύλης ελεγχόμενης άρνησης στον καταχωρητή $\left| 10 \right\rangle$, είναι:

• $\text{cX} \left| 10 \right\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \left| 11 \right\rangle$.

Έτσι, ανάλογα για τον καταχωρητή φαίνεται η επίδραση της πύλης CNOT στον παρακάτω πίνακα :

Κατάσταση των qubits πριν τη δράση της πύλης $\left\| c_1 t_1 \right\rangle$	**Κατάσταση των qubits μετά τη δράση της πύλης** $\left\| c_0 t_0 \right\rangle$
$\left\| 00 \right\rangle$	$\left\| 00 \right\rangle$
$\left\| 01 \right\rangle$	$\left\| 01 \right\rangle$
$\left\| 10 \right\rangle$	$\left\| 11 \right\rangle$
$\left\| 11 \right\rangle$	$\left\| 10 \right\rangle$

Σχήμα: 28 – Πίνακας αντιστοίχισης της πύλης CNOT

Προσοχή

Το πρώτο είναι το qubit ελέγχου λειτουργεί σαν NOT στο δεύτερο qubit μόνο εάν το πρώτο είναι $|1\rangle$, αλλιώς στην ουσία δεν κάτι τίποτα.

Έτσι, η πύλη CNOT αντιστοιχίζει:

- την κατάσταση $|00\rangle$ στην κατάσταση $|00\rangle$
- την κατάσταση $|01\rangle$ στην κατάσταση $|01\rangle$
- την κατάσταση $|10\rangle$ στην κατάσταση $|11\rangle$
- την κατάσταση $|11\rangle$ στην κατάσταση $|10\rangle$

Για να σχεδιάσουμε σε ένα κύκλωμα την πύλη ελεγχόμενης άρνησης CNOT (cX), στο πρόγραμμα συντάσσεται:

- circuit.cx (i j)

Δηλαδή, γράφουμε δίπλα από το όνομα του κυκλώματος circuit. τα γράμματα “cX” με το οποίο συμβολίζουμε την πύλη ελεγχόμενης άρνησης CNOT, με i > j και όπου i, j είναι ο αριθμός των qubit που θέλουμε να εφαρμοστεί.

Εφαρμογή 7

Θέλουμε να σχεδιάσουμε ένα κύκλωμα με δύο qubit, που στο πρώτο δεν υπάρχει κάποια δράση και στο δεύτερο επιδρά η πύλη CNOT), οπότε συντάσσουμε το παρακάτω πρόγραμμα:

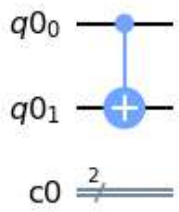
```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  %matplotlib inline
4.  qr = QuantumRegister(2)
5.  cr = ClassicalRegister(2)
6.  circuit = QuantumCircuit(qr,cr)
7.  circuit.cx(qr[0],[1])
8.  circuit.measure(qr,cr)
9. circuit.draw(output=’mpl’)
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 4: Δημιουργία κβαντικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των qubits και με μια μεταβλητή) .
- Γραμμή 5: Δημιουργία κλασικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των bits και με μια μεταβλητή) .

- Γραμμή 6: Δημιουργία κυκλώματος από 2 qubit και 2 bits.
- Γραμμές 7: Προσθήκη της πύλης CNOT (cX).
- Γραμμή 8: Μέτρηση του κυκλώματος. Με την καταστροφή των qubits η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα κλασικά bits.
- Γραμμή 9: Οπτικοποίηση του κυκλώματος. Εμφανίζει το αποτέλεσμα του προγράμματος, δύο qubit, που δρα στο δεύτερο qubit η πύλη CNOT (cX). Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το κύκλωμα με την πύλη CNOT:



Σχήμα: 29 – Υλοποίηση πύλης cX με qubit ελέγχου το 2ο qubit

Όταν θέλουμε το qubit ελέγχου να είναι δεύτερο, δηλαδή όταν το δεύτερο qubit βρίσκεται στην κατάσταση 1, να αλλάζει το πρώτο qubit, τότε αυτό είναι εφικτό να πραγματοποιηθεί με ένα συνδυασμό πυλών Hadamard και CNOT.

Παράδειγμα 12

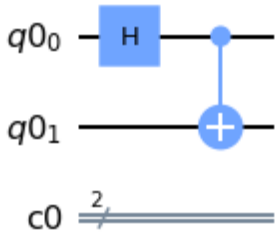
Θέλουμε να σχεδιάσουμε ένα κύκλωμα με δύο qubit, που στο πρώτο qubit να εφαρμόζεται η πύλη Hadamard και στο δεύτερο qubit η πύλη CNOT, οπότε συντάσσουμε το παρακάτω πρόγραμμα:

```
1. from qiskit import *
2. from qiskit.visualization import plot_histogram
3. %matplotlib inline
4. qr = QuantumRegister(2)
5. cr = ClassicalRegister(2)
6. circuit = QuantumCircuit(qr,cr)
7. circuit.h(0)
8. circuit.cx(qr[0],[1])
9. circuit.draw(output='mpl')
```

Το πρόγραμμα συντάσσεται ακριβώς το ίδιο με την παραπάνω εφαρμογή που είδαμε, προσθέτοντας μόνο:

- Γραμμή 7: Προσθήκη μιας πύλης Hadamard, με αριθμό qubit μηδέν.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το κύκλωμα με την πύλη Hadamard στο πρώτο qubit και την πύλη CNOT στο δεύτερο qubit:

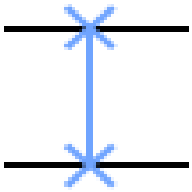


Σχήμα: 30 – Υλοποίηση πύλης Hadamard στο 1ο qubit και πύλη cX με qubit ελέγχου στο 2ο qubit

4.20 Κβαντική Πύλη Εναλλαγής (SWAP)

Η κβαντική πύλη εναλλαγής qubit συμβολίζεται με SWAP και είναι μια κβαντική πύλη που εφαρμόζεται σε δύο qubit, εναλλάσσοντας τα περιεχόμενά τους. Μετά την εφαρμογή της πύλης SWAP, τα δύο qubit που δέχθηκαν τη δράση της πραγματοποίησαν στην ουσία εναλλάξει τις καταστάσεις που βρίσκονταν πριν.

Το σύμβολο της κβαντικής πύλης SWAP είναι:



Σχήμα: 31 – Σύμβολο κβαντικής πύλης SWAP

Η κβαντική πύλη ελέγχου SWAP μαθηματικά περιγράφεται από ακόλουθο πίνακα:

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Όπως είδαμε οι πύλες που δρουν σε ένα qubit ο πίνακας της πύλης είναι ένας πίνακας 2x2, ενώ ο πίνακας μιας πύλης που δρα σε δύο qubits αποτελείται από έναν πίνακα μιας στήλης τεσσάρων στοιχείων.

Εφαρμογή 8

Ας δούμε πώς επιδρά η πύλη SWAP σε δύο qubits q_1 και q_2 με

$|q_1\rangle = \alpha|0\rangle + b\ |1\rangle$ και $|q_2\rangle = c|0\rangle + \mathrm{d}\ |1\rangle$ και είναι:

$$|q_1\, q_2\rangle = |q_1\rangle \otimes |q_2\rangle = \begin{bmatrix} \alpha \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a\begin{bmatrix} c \\ d \end{bmatrix} \\ b\begin{bmatrix} c \\ d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a \cdot c \\ a \cdot d \\ b \cdot c \\ b \cdot d \end{bmatrix}.$$

$$\text{SWAP}(|q_1\, q_2\rangle) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \cdot c \\ a \cdot d \\ b \cdot c \\ b \cdot d \end{bmatrix} = \begin{bmatrix} a \cdot c \\ b \cdot c \\ a \cdot d \\ b \cdot d \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} \otimes \begin{bmatrix} a \\ b \end{bmatrix} = |q_2\, q_1\rangle.$$

Στον παρακάτω πίνακα, βλέπουμε συνοπτικά τις ιδιότητες της πύλης SWAP:

Κατάσταση των qubits $\|\mathbf{q}\rangle$ πριν τη δράση της πύλης	**Κατάσταση των qubits $\|\mathbf{q}\rangle$ μετά τη δράση της πύλης**
$\|\boldsymbol{q_1 q_0}\rangle$	$\|\boldsymbol{q_0 q_1}\rangle$
$\|\boldsymbol{q_0 q_1}\rangle$	$\|\boldsymbol{q_1 q_0}\rangle$

Σχήμα: 32 – Πίνακας αντιστοίχισης της πύλης SWAP

Συνεπώς, η πύλη Pauli-X (NOT) αντιστοιχίζει:

- την κατάσταση $|\boldsymbol{q_1 q_0}\rangle$ στην κατάσταση $|\boldsymbol{q_0 q_1}\rangle$
- την κατάσταση $|\boldsymbol{q_0 q_1}\rangle$ στην κατάσταση $|\boldsymbol{q_1 q_0}\rangle$.

Η γενική μορφή της πύλης SWAP, είναι:

$\text{SWAP}|t_1 t_2\rangle = |t_2 t_1\rangle$.

Η επίδρασή της πύλης SWAP στον καταχωρητή $|10\rangle$, είναι:

$$\bullet\ \text{SWAP}|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle.$$

Παρατήρηση

Η επίδραση της πύλης SWAP παρουσιάζει ενδιαφέρον σε καταχωρητή με διαφορετική αρχική κατάσταση των $|00\rangle$, $|01\rangle$, $|10\rangle$ και $|11\rangle$.

Η πύλη SWAP δρα σε 2 qubit και τα ανταλλάσσει, δηλαδή:

$$|\psi, \phi\rangle \longrightarrow |\phi, \psi\rangle$$

Έτσι, η πύλη SWAP αντιστοιχίζει:

- την κατάσταση $|00\rangle$ στην κατάσταση $|00\rangle$
- την κατάσταση $|01\rangle$ στην κατάσταση $|10\rangle$
- την κατάσταση $|10\rangle$ στην κατάσταση $|01\rangle$
- την κατάσταση $|11\rangle$ στην κατάσταση $|11\rangle$

Για να σχεδιάσουμε σε ένα κύκλωμα την πύλη SWAP, το πρόγραμμα συντάσσεται:

- circuit.swap (i, j)

Δηλαδή, γράφουμε δίπλα από το όνομα του κυκλώματος circuit. τη λέξη της πύλης "swap", με i > j και όπου i, j είναι ο αριθμός των qubit που θέλουμε να εφαρμοστεί.

Ας δούμε μερικά παραδείγματα.

ΠΑΡΑΔΕΙΓΜΑ 13

Ας δούμε πώς επιδρά η συγκεκριμένη πύλη στα qubits $|\alpha b\rangle$, με $|\alpha\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$ και $|b\rangle = \frac{8}{17}|0\rangle + \frac{15}{17}|1\rangle$.

Αρχικά, εξετάζουμε αν τα δύο qubits βρίσκονται σε αποδεκτή κατάσταση και έχουμε:

- $\left(\frac{3}{5}\right)^2 + \left(\frac{4}{5}\right)^2 = \frac{3^2}{5^2} + \frac{4^2}{5^2} = \frac{9}{25} + \frac{16}{25} = \frac{25}{25} = 1.$
- $\left(\frac{8}{17}\right)^2 + \left(\frac{15}{17}\right)^2 = \frac{8^2}{17^2} + \frac{15^2}{17^2} = \frac{64}{289} + \frac{225}{289} = \frac{289}{289} = 1.$

Άρα τα δύο qubits είναι σε αποδεκτή κατάσταση και στη συνέχεια εφαρμόσουμε την πύλη SWAP και είναι:

$$\text{SWAP}(|\alpha b\rangle) = \text{SWAP}(|\alpha\rangle \otimes |b\rangle) = \text{SWAP}\left(\begin{bmatrix}\frac{3}{5}\\ \frac{4}{5}\end{bmatrix} \otimes \begin{bmatrix}\frac{8}{17}\\ \frac{15}{17}\end{bmatrix}\right) =$$

$$\text{SWAP} = \begin{bmatrix} \frac{3}{5}\begin{bmatrix}\frac{8}{17}\\ \frac{15}{17}\end{bmatrix} \\ \frac{4}{5}\begin{bmatrix}\frac{8}{17}\\ \frac{15}{17}\end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{24}{85} \\ \frac{45}{85} \\ \frac{32}{85} \\ \frac{60}{85} \end{bmatrix} = \frac{1}{85}\begin{bmatrix} 24 \\ 32 \\ 45 \\ 60 \end{bmatrix} =$$

$$= \frac{1}{85}\left(24\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + 32\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + 45\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + 60\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) =$$

$$= \frac{1}{85}\left(24\left|00\right\rangle + 32\left|01\right\rangle + 45\left|10\right\rangle + 60\left|11\right\rangle \right) =$$

$$= 0{,}08\left|00\right\rangle + 0{,}14\left|01\right\rangle + 0{,}28\left|10\right\rangle + 0{,}50\left|11\right\rangle.$$

Η υλοποίηση του και η μέτρηση του καταχωρητή δίνεται από το παρακάτω πρόγραμμα.

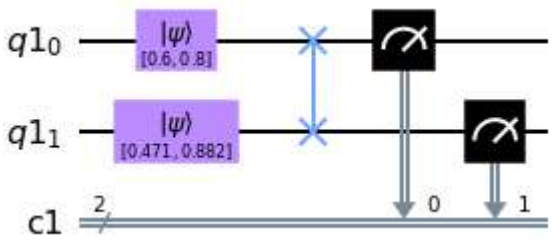
```
1. from qiskit import *
2. qr = QuantumRegister(2)
3. cr = ClassicalRegister(2)
4. initial_state_0 = [3/5, 4/5]
5. initial_state_1 = [8/17, 15/17]
6. circuit = QuantumCircuit(qr,cr)
7. circuit.initialize(initial_state_0,0)
8. circuit.initialize(initial_state_1,1)
9. circuit.swap(0,1)
10. circuit.measure(qr,cr)
11. circuit.draw(output='mpl')
12. simulator = Aer.get_backend('qasm_simulator')
13. result=execute(circuit,backend=simulator,counts=1024).resul
    t()
14. from qiskit.visualization import plot_histogram
15. plot_histogram(result.get_counts(circuit))
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμή 1: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 2: Δημιουργία κβαντικού καταχωρητή, ο οποίος αποτελείται από 2 qubit (το πλήθος των qubits μπορεί να αποδίδεται και από μια μεταβλητή).
- Γραμμή 3: Δημιουργία κλασικού καταχωρητή, ο οποίος αποτελείται από 2 bit (το πλήθος των bits μπορεί να αποδίδεται και από μια μεταβλητή).

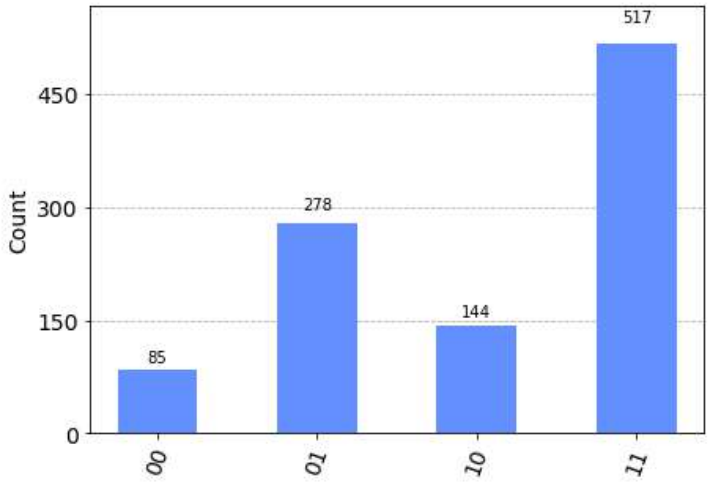
- Γραμμή 4: Ορισμός των τιμών των ζητούμενων αρχικών καταστάσεων στις αντίστοιχες μεταβλητές.
- Γραμμή 5: Ορισμός των τιμών των ζητούμενων αρχικών καταστάσεων στις αντίστοιχες μεταβλητές.
- Γραμμή 6: Δημιουργία κυκλώματος από 2 qubit και 2 bits.
- Γραμμές 7 και 8: Αρχικοποίηση των qubits.
- Γραμμές 9: Προσθήκη της πύλης SWAP.
- Γραμμή 10: Μέτρηση του κυκλώματος. (Τα qubits στην ουσία καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα bits.
- Γραμμή 11: Οπτικοποίηση του κυκλώματος. Εμφανίζει (σχεδιάζει) τις συγκεκριμένες πύλες (H) και (H) αντίστοιχα του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.
- Γραμμή 12: Επιλογή του προσομοιωτή.
- Γραμμή 13: Αποδίδουμε το αποτέλεσμα της εκτέλεσης του κυκλώματος στην μεταβλητή result και όπως είδαμε, επειδή το πρόγραμμά μας δεν εκτελείται σε πραγματικό υπολογιστή αλλά σε προσομοιωτή, το «τρέχουμε» πολλές φορές για να έχουμε καλύτερη πιθανολογική προσέγγιση του αποτελέσματος (ζητάμε από το πρόγραμμα να εκτελεσθεί τυχαία 1010 φορές το κύκλωμα).
- Γραμμές 14: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμή 15: Εμφανίζεται το αποτέλεσμα του προγράμματος με τη μορφή ιστογράμματος. Παρατηρούμε στο γράφημα τις πολύ μικρές διαφορές στις δύο μετρήσεις (της τάξης 3%).

Όταν «τρέξουμε» το πρόγραμμα μέχρι και τη γραμμή 11, θα εμφανιστεί το αποτέλεσμα :



Σχήμα: 33 – Υλοποίηση πύλης SWAP στο παραπάνω πρόγραμμα

Όταν «τρέξουμε» ολόκληρο το πρόγραμμα (μέχρι και τη γραμμή 15), θα εμφανιστεί το αποτέλεσμα :



Σχήμα: 34 – Ιστόγραμμα αποτελεσμάτων εφαρμογής πύλης SWAP

Παρατήρηση

Στο παραπάνω πρόγραμμα, αξίζει να επισημάνουμε τις γραμμές 4, 5 και 7, 8. Σε έναν εξομοιωτή είναι εφικτό με μεγάλη ευκολία να αποδώσουμε αρχικές τιμές σε ένα qubit (με αρχική κατάσταση του qubit να είναι πάνταα $|0\rangle$), ενώ σε έναν κβαντικό υπολογιστή αυτό πραγματοποιείται με την εφαρμογή κατάλληλων πυλών.

Η πύλη SWAP σε κάποιες περιπτώσεις είναι δυνατό να υλοποιηθεί άμεσα μέσω των qubits, σε άλλες περιπτώσεις αυτό είναι εφικτό να πραγματοποιηθεί με έμμεσο τρόπο χρησιμοποιώντας κάποιες απλές πύλες. Τέτοιο παράδειγμα αποτελεί η υλοποίηση της πύλης SWAP σε συνδυασμό τριών κβαντικών πυλών CNOT (cX).

Σημείωση

Όταν έχουμε να επιλύσουμε κβαντικά κυκλώματα, είναι πολύ χρήσιμο να θυμόμαστε τα εξής:

• Όταν έχουμε κύκλωμα και σε κάποια qubits δεν εφαρμόζεται καμία πύλη, τότε εφαρμόζουμε πύλη αδράνειας.

• Όταν έχουμε συνδυασμό πυλών, τότε ισοδυναμεί με τανυστικό γινόμενο των αντίστοιχων διανυσμάτων κατάστασης.

• Όταν έχουμε εφαρμογή πύλης σε qubit(s), τότε ισοδυναμεί με γινόμενο των αντίστοιχων πινάκων (των πυλών) με το διάνυσμα κατάστασης των qubits.

Στο επόμενο κεφάλαιο, θα δούμε τη διαδικασία εκτέλεσης των κβαντικών κυκλωμάτων σε κβαντικό υπολογιστή καθώς και τη μέθοδο επίλυσής τους.

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Για να αποτελεί αποδεκτή κβαντική κατάσταση τι τιμή πρέπει να έχει:

i). το α, στο διάνυσμα κατάστασης $\begin{bmatrix} \alpha \\ 0{,}4 \end{bmatrix}$.

ii). το x, στο διάνυσμα κατάστασης $\begin{bmatrix} 0{,}7 - x \\ x \end{bmatrix}$.

2) Να υπολογίσετε με ακρίβεια, ποιο από τα παρακάτω διανύσματα είναι διανύσματα κατάστασης:

i). $|K\rangle$ = 0,7071 $|H\rangle$+ 0,7071 $|T\rangle$

ii). $|K\rangle$ = 0,70 $|H\rangle$ + 0,30 $|T\rangle$

iii). $|K\rangle$ = - 0500 $|H\rangle$ + 0,866 $|T\rangle$

iv). $|K\rangle$ = 0,50 $|H\rangle$ - 0,50 $|T\rangle$

3) Να εξηγήσετε αν είναι αποδεκτές ή όχι οι παρακάτω κβαντικές καταστάσεις:

i) $|q\rangle = \frac{2}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ και ii) $|k\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$.

4) Μετρήστε τις πιθανότητες εμφάνισης 0 και 1 των παρακάτω qubits:

i) $|\alpha\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ ii) $|b\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{3}{2}|1\rangle$ και

iii). $|c\rangle = \frac{2}{2\sqrt{2}}|0\rangle + \frac{\sqrt{7}}{2\sqrt{2}}|1\rangle$.

5) Να ξεχωρίσετε την κβαντική κατάσταση των παρακάτω qubits:

i) $|\alpha\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{2}{\sqrt{2}}|1\rangle$ ii) $|b\rangle = \frac{1}{\sqrt{2}}|0\rangle - i\frac{\sqrt{3}}{2}|1\rangle$

6) Να εξηγήσετε αν αποτελεί έγκυρο κβαντικό σύστημα το $|K\rangle$. Αν αποτελεί έγκυρο κβαντικό σύστημα το $|K\rangle$, τότε να υπολογίσετε τις πιθανότητες των καταστάσεών $|00\rangle$, $|01\rangle$, $|10\rangle$ και $|11\rangle$. Δίνεται το

$$|K\rangle = 0.412311|00\rangle + 0.47958|01\rangle + 0.5831|10\rangle + 0.5099|11\rangle$$

7) Έχουμε έναν κβαντικό καταχωρητή που περιέχει 2 qubits $|x\psi\rangle$, τα οποία βρίσκονται στις καταστάσεις $|x\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$ και $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, να βρεθεί η τελική κατάσταση του καταχωρητή.

8) Έχουμε δύο qubits που βρίσκονται στην κατάσταση $|x\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|00\rangle - \frac{1}{2}|11\rangle$. Μετά τη μέτρηση, να υπολογιστούν:

i) Η πιθανότητα να βρεθούν στην κατάσταση 00.

ii) Η πιθανότητα να βρεθεί το πρώτο qubit στην κατάσταση 1.

iii) Η πιθανότητα να βρεθεί το δεύτερο qubit στην κατάσταση 0.

iv) Η πιθανότητα να βρεθούν στην κατάσταση 01.

9) Να υπολογίσετε τη δράση της πύλης CNOT στις παρακάτω βασικές καταστάσεις:

i) $|00\rangle$ ii) $|01\rangle$ και iii) $|11\rangle$.

10) Να υπολογίσετε τη δράση της κβαντικής πύλης Hadamard (H) στο παρακάτω qubit που βρίσκεται σε υπέρθεση βασικών καταστάσεων:

$|q\rangle = \alpha|0\rangle + b|1\rangle$.

11) Να υπολογίσετε τη δράση της κβαντικής πύλης CNOT στο παρακάτω κβαντικό qubit που βρίσκεται σε υπέρθεση βασικών καταστάσεων:

$|q\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle$.

12) Να υπολογίσετε κύκλωμα:

i) X H $|0\rangle$ ii) X H $|1\rangle$ και iii) CNOT X H $|1\rangle$.

Στη συνέχεια να γράψετε πρόγραμμα ώστε να επαληθευθεί η απάντησή σας.

13) Έχουμε ένα κύκλωμα με δύο qubits, με αρχική τιμή $|0\rangle$. Τότε:

α). Να εφαρμόσουμε μια πύλη Hadamard στο πρώτο qubit και καμία δράση για το δεύτερο qubit.

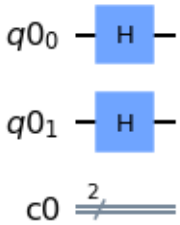
β). Να υπολογιστεί το αποτέλεσμα μετά την μέτρηση του πιο πάνω κυκλώματος.

γ). Να γράψουμε πρόγραμμα ώστε να εμφανίζεται το παραπάνω κύκλωμα.

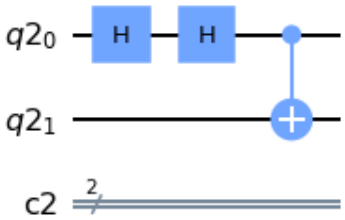
14) Να γράψετε πρόγραμμα που να εμφανίζει μια απλή γραφική απεικόνιση της σφαίρα Bloch, αποτυπώνοντας κάποια σημεία, αλλά και των βασικών καταστάσεων της κβαντικής αναπαράστασης μνήμης που συμβολίζονται με $|0\rangle$ και $|1\rangle$.

15) Να πραγματοποιήσετε τις κατάλληλες πράξεις για τα παρακάτω σχήματα και στη συνέχεια να γράψετε πρόγραμμα που να εμφανίζει τα σχήματα αυτά.

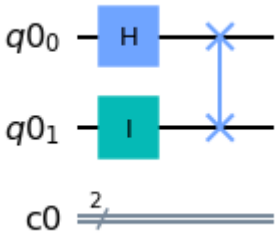
i)



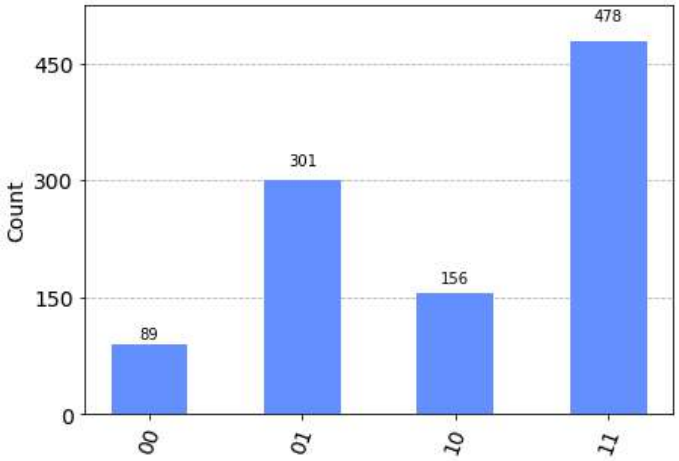
ii)



16) Να περιγράψετε το παρακάτω σχήμα και στη συνέχεια να γράψετε πρόγραμμα που να εμφανίζει το σχήμα αυτό.



17) Να σχολιάσετε το παρακάτω γράφημα:



ΚΕΦΑΛΑΙΟ: 5ο - Κβαντικά κυκλώματα

ΚΕΦΑΛΑΙΟ 5ο

Κβαντικά Κυκλώματα

Σύνοψη

Σε αυτό το κεφάλαιο μελετάμε τη διαδικασία εκτέλεσης των κβαντικών κυκλωμάτων σε κβαντικό υπολογιστή καθώς και τη μέθοδο επίλυσής τους. Αρχικά, περιγράφεται το κυκλωματικό μοντέλο των κβαντικών υπολογισμών, η αρχή της κβαντικής υπολογιστικής και δίνεται το διάγραμμα ροής. Ακολουθεί αναφορά στις έννοιες της κβαντικής διεμπλοκής των καταστάσεων Bell και της κβαντικής τηλεμεταφοράς. Ακόμα, παρουσιάζεται ποια είναι η διαφορά της κβαντικής διεμπλοκής από την κβαντική υπέρθεση και πώς μπορούμε να φέρουμε σε διεμπλοκή δύο qubits. Υποδεικνύονται αποδείξεις της διεμπλοκής και ιδιότητες του κβαντικού υπολογισμού, όπως το θεώρημα μη κλωνοποίησης και της κβαντικής τηλεμεταφοράς. Στη συνέχεια, για την μέγιστη δυνατή αφομοίωση του κεφαλαίου από το μαθητή, δίνονται λυμένα παραδείγματα και εφαρμογές προγραμμάτων κβαντικού υπολογισμού σε κβαντικά κυκλώματα και των κβαντικών παιχνιδιών καθώς και ασκήσεις για λύση από το μαθητή. Αυτό το κεφάλαιο απευθύνεται σε μαθητές Λυκείου.

5.1 Υπολογισμοί Κβαντικών Κυκλωμάτων

5.1.1 Κβαντικά κυκλώματα

Στο προηγούμενο κεφάλαιο είδαμε τις σπουδαιότερες κβαντικές πύλες που δρουν σε ένα ή δύο qubits, οι οποίες περιστρέφουν τα διανύσματα κατάστασης των qubits και κβαντικών καταχωρητών χωρίς να αλλάζουν το μήκος τους, το οποίο είναι πάντα ίσο με τη μονάδα. Δηλαδή, οι κβαντικοί υπολογισμοί είναι δράσεις τελεστών που έχουν ως αποτέλεσμα την περιστροφή διανυσμάτων στο χώρο του Hilbert. Έτσι, μπορούμε να υλοποιήσουμε συγκεκριμένους κβαντικούς υπολογισμούς χρησιμοποιώντας και διατάσσοντας κατάλληλα αυτές τις κβαντικές πύλες, δημιουργώντας το κβαντικό κύκλωμα (quantum circuit). Συνεπώς, κβαντικό κύκλωμα καλείται ένα πλήθος κβαντικών πυλών οι οποίες επιδρούν σε ένα κβαντικό καταχωρητή και αποτελούνται από qubits, κβαντικούς καταχωρητές και κβαντικές πύλες. Τα κβαντικά κυκλώματα φανερώνουν τη χρονική σειρά και τον τρόπο με τον οποίο δρουν οι κβαντικές πύλες στους κβαντικούς καταχωρητές. Επίσης, τα κβαντικά κυκλώματα μετασχηματίζουν κβαντικές καταστάσεις και οι κβαντικές πύλες των ενός και δύο qubit μπορούμε να πούμε ότι αποτελούν τη βάση για την κατασκευή ενός κβαντικού κυκλώματος.

Ακόμα, κατά την μέτρηση ενός κυκλώματος τα qubits καταστρέφονται και δεν είναι εφικτό να κρατήσουν την πληροφορία. Έτσι, τα qubit στέλνουν την πληροφορία στα bit και το αποτέλεσμα της μέτρησης (η κβαντική κατάσταση των qubits μετά τη μέτρηση) αποθηκεύεται μόνιμα στα αντίστοιχα bits.

Ένα κβαντικό κύκλωμα διακρίνεται από τις παρακάτω μετρήσεις:

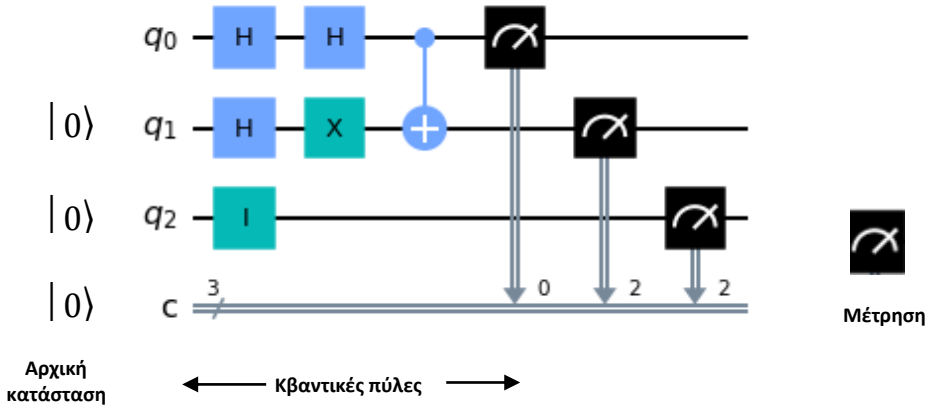
1. Τον αριθμό των N qubits που περιέχει (και χρησιμοποιούνται κατά την είσοδο), τα οποία βρίσκονται όλα στην κατάσταση $|0\rangle$.
2. Τον αριθμό των βημάτων d που απαιτείται για να υλοποιηθεί, δηλαδή το πλήθος των διαδοχικών κβαντικών πυλών σε σειρά, η οποία ονομάζεται βάθος (depth) του κυκλώματος.
3. Τον κβαντικό όγκο (quantum volume) του κυκλώματος που προσδιορίζει τις δυνατότητες και τα ίσως κάποια λάθη κάποιου κβαντικού κυκλώματος.

Χαρακτηριστικό είναι το μοντέλο προγραμματισμού κβαντικών κυκλωμάτων και περιγράφεται από την παρακάτω διαδικασία:

1. Στην αρχή έχουμε ένα πλήθος qubits (που χρησιμοποιούνται κατά την είσοδο), τα οποία βρίσκονται όλα στην κατάσταση $|0\rangle$.
2. Στη συνέχεια, όπως στις κλασικές πύλες έτσι και στις κβαντικές πύλες, η κβαντική πληροφορία μεταφέρεται μέσω καλωδίου αντίστοιχα για κάθε qubit ξεχωριστά.
3. Ακολουθεί η πραγματοποίηση των κβαντικών πυλών στα διάφορα qubit, οι οποίες εκτελούν μετασχηματισμούς στα qubit.
4. Τέλος, εφαρμόζονται κάποιες μετρήσεις (μία ή πιο πολλές), πάνω σε ένα ή περισσότερα qubit, ώστε να πάρουμε τα σχετικά αποτελέσματα.

Παράδειγμα 1

Έχουμε το παρακάτω κβαντικό κύκλωμα με τρία qubit που εφαρμόζονται οι πύλες Hadamard (H), αδράνειας (I), Pauli-X (NOT) και ελεγχόμενης άρνησης (CNOT).

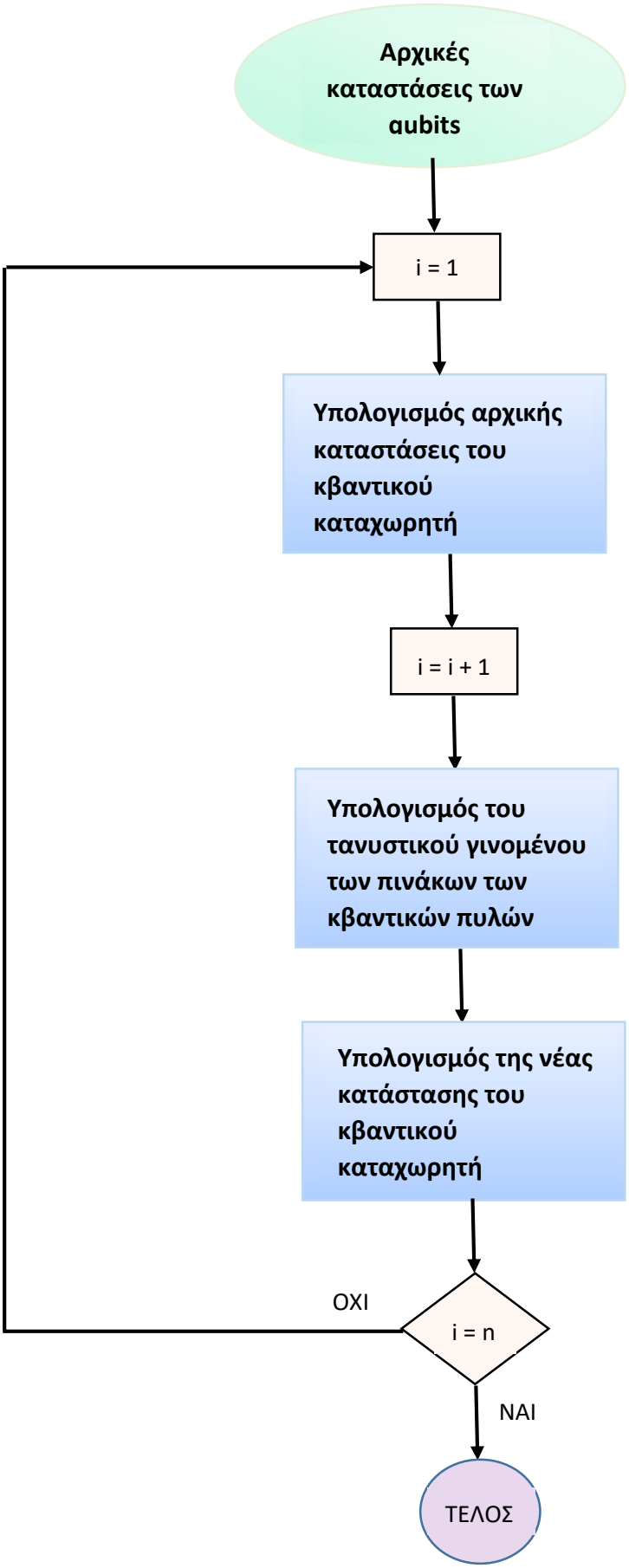


Σχήμα: 1 – Παράδειγμα κβαντικού κυκλώματος τριών qubits

Συγκεκριμένα, στο παραπάνω κύκλωμα έχει εφαρμοστεί στο πρώτο qubit δύο πύλες Hadamard και μια πύλη CNOT μεταξύ του πρώτου και δεύτερου qubit. Στο δεύτερο qubit έχει εφαρμοστεί μια πύλη Hadamard και μια πύλη Pauli - X. Στο τρίτο qubit δεν εφαρμόζεται κάτι και έχουμε την πύλη αδράνειας. Στο τέλος των qubit πραγματοποιείται μία μέτρηση αντίστοιχα. Έτσι, αφού τα qubit είναι τρία το N=3 και το βάθος του κυκλώματος είναι d=5.

5.1.2 Η Αρχή της Κβαντικής Υπολογιστικής - Επίλυση Κυκλωμάτων

Για να αναπαρασταθούν οι κβαντικοί υπολογισμοί με κάποιο μοντέλο, ύστερα από πολλές προσπάθειες μελετητών κατέληξαν σήμερα να χρησιμοποιούν το μοντέλο που καλείται κυκλωματικό μοντέλο των κβαντικών υπολογισμών. Η αρχή της κβαντικής υπολογιστικής ενός κυκλώματος στηρίζεται σε αυτό το μοντέλο το οποίο χρησιμοποιείται για τη περιγραφή, τη δημιουργία των κβαντικών υπολογισμών καθώς και την κατασκευή κβαντικών κυκλωμάτων. Η διαδικασία αυτή φαίνεται συνοπτικά στο παρακάτω διάγραμμα εκτέλεσης των κβαντικών υπολογισμών και με (i) συμβολίζεται ο αριθμός των βημάτων , ενώ με (n) συμβολίζεται ο συνολικός αριθμός βημάτων του κβαντικού υπολογισμού.



Σχήμα: 2 – Η αρχή της κβαντικής υπολογιστικής ως διάγραμμα ροής για την εκτέλεση κβαντικών υπολογισμών

Σύμφωνα με το παραπάνω σχεδιάγραμμα, περιγράφονται παρακάτω πώς εκτελούνται οι κβαντικοί υπολογισμοί για την επίλυση ενός κυκλώματος εφαρμόζοντας βήμα βήμα τις παρακάτω μετρήσεις:

1ον Αρχική κατάσταση καταχωρητή:

Αρχικά δίνεται η κβαντική κατάσταση των N qubits που αποτελούν έναν κβαντικό καταχωρητή. Για να βρούμε την αρχική κατάσταση ενός καταχωρητή και συγκεκριμένα, για να υπολογίσουμε τις βασικές καταστάσεις των δύο qubits στον καταχωρητή, χρησιμοποιούμε το τανυστικό γινόμενο των πινάκων που αντιστοιχούν στις καταστάσεις των N αυτών qubits (όπως είδαμε στο προηγούμενο κεφάλαιο). Ο πίνακας που προκύπτει αποτελεί την αρχική κατάσταση του κβαντικού καταχωρητή.

2ον Συνδυασμός πυλών:

Όταν έχουμε συνδυασμό πυλών, τότε για να υπολογίσουμε τη συνολική τους δράση χρησιμοποιούμε πάλι το τανυστικό γινόμενο των πινάκων που περιγράφουν τις κβαντικές πύλες του συγκεκριμένου κυκλώματος.

Στο προηγούμενο κεφάλαιο είδαμε και αξίζει να επισημάνουμε, ότι όταν σε ένα qubit δεν δρα κάποια πύλη τότε πάντα εφαρμόζουμε στο qubit αυτό την πύλη αδράνειας I.

Προσοχή

Το τανυστικό γινόμενο το εφαρμόζουμε στις πύλες ακριβώς με τη σειρά που εμφανίζονται οι πύλες σε ένα κύκλωμα. Αν για παράδειγμα σε ένα κύκλωμα εμφανίζεται πρώτα η πύλη Hadamard και μετά η πύλη NOT, τότε θα χρησιμοποιήσουμε το τανυστικό γινόμενο της πύλης H με τη NOT και είναι διαφορετικό με το να χρησιμοποιήσουμε το τανυστικό γινόμενο της πύλης NOT με την πύλη H. Με λίγα λόγια δεν ισχύει η αντιμεταθετή ιδιότητα.

Δηλαδή:

$$H \otimes \text{NOT} \neq \text{NOT} \otimes H \ .$$

3ον Εφαρμογή πυλών σε κβαντική κατάσταση:

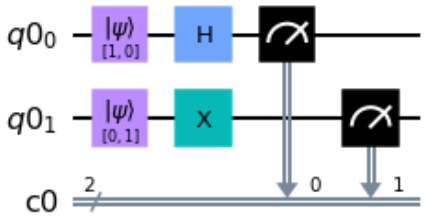
Η εφαρμογή των κβαντικών πυλών σε κβαντική κατάσταση, υπολογίζεται χρησιμοποιώντας το γινόμενο των αντίστοιχων πινάκων, προσέχοντας πάλι τη σειρά. Πολλαπλασιάζουμε τον πίνακα που προκύπτει από το τανυστικό γινόμενο των πινάκων των κβαντικών πυλών με τον πίνακα της κατάστασης του κβαντικού καταχωρητή και το αποτέλεσμα είναι ο πίνακας της νέας κατάστασης του κβαντικού καταχωρητή. Δηλαδή η σειρά είναι πίνακας πυλών επί πίνακας κατάστασης.

Επαναλαμβάνουμε τη διαδικασία 2 και 3 τόσες φορές όσα και τα βήματα του κβαντικού υπολογισμού.

Έτσι, η τελική κατάσταση του κβαντικού καταχωρητή είναι το αποτέλεσμα του κβαντικού υπολογισμού.

Εφαρμογή 1

Ο κβαντικός καταχωρητής του παρακάτω κυκλώματος αποτελείται από δύο qubits και η κατάστασή του είναι $|01\rangle$.



Σχήμα: 3 – Κβαντικού κυκλώματος δυο qubits

Για την επίλυση του παραπάνω κβαντικού κυκλώματος θα πραγματοποιήσουμε έναν κβαντικό υπολογισμό εφαρμόζοντας τα παρακάτω βήματα:

1ον <u>Τελική κατάσταση καταχωρητή</u>:

Αρχικά δίνεται η κβαντική κατάσταση των qubits που αποτελούν τον κβαντικό καταχωρητή. Για να βρούμε την τελική κατάσταση ενός καταχωρητή και συγκεκριμένα, για να υπολογίσουμε τις βασικές καταστάσεις των δύο qubits στον καταχωρητή, χρησιμοποιούμε το τανυστικό γινόμενο των πινάκων που αντιστοιχούν στις καταστάσεις των δύο αυτών qubits.

Έχουμε:

$$\bullet\ |01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 0\begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} =$$

$$= 0\,|00\rangle + 1\,|01\rangle + 0\,|10\rangle + 0\,|11\rangle.$$

Έτσι, σύμφωνα με την παραπάνω σχέση καταλαβαίνουμε ότι η πιθανότητα να μετρήσουμε την κατάσταση του κβαντικού καταχωρητή στην κατάσταση $|01\rangle$ είναι ένα, ενώ η πιθανότητα να μετρήσουμε τις άλλες τρεις καταστάσεις ($|00\rangle$, $|10\rangle$ και $|11\rangle$) είναι μηδέν.

2ον <u>Συνδυασμός πυλών:</u>

Όπως αναφέραμε παραπάνω, όταν έχουμε συνδυασμό πυλών τότε για να υπολογίσουμε τη συνολική τους δράση χρησιμοποιούμε το τανυστικό τους γινόμενο. Συγκεκριμένα, για το κύκλωμά μας εφαρμόζεται στο πρώτο qubit μια πύλη Hadamard και στο δεύτερο qubit εφαρμόζεται μια πύλη NOT (X) και γράφεται:

H (0) X (1) .

Γνωρίζουμε ότι η πύλη Hadamard και NOT(X) ισοδυναμούν αντίστοιχα:

• $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$ και • $NOT\ (X) = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$.

Έτσι, το τανυστικό τους γινόμενο θα είναι:

$$\bullet\ H \otimes NOT(X) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \otimes \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} & 1\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \\ 1\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} & -1\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}1\cdot 0 & 1\cdot 1 & 1\cdot 0 & 1\cdot 1\\ 1\cdot 1 & 1\cdot 0 & 1\cdot 1 & 1\cdot 0\\ 1\cdot 0 & 1\cdot 1 & -1\cdot 0 & -1\cdot 1\\ 1\cdot 1 & 1\cdot 0 & -1\cdot 1 & -1\cdot 0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & 0 & 1\\ 1 & 0 & 1 & 0\\ 0 & 1 & 0 & -1\\ 1 & 0 & -1 & 0\end{bmatrix}.$$

3ον <u>Εφαρμογή πυλών σε κβαντική κατάσταση:</u>

Για να υπολογίσουμε την εφαρμογή της πύλης Hadamard με την NOT στην κβαντική τους κατάσταση χρησιμοποιούμε το γινόμενο:

$$\begin{pmatrix}\text{Αποτέλεσμα τανυστικού γινομένου}\\ \text{των πυλών H με NOT που δρουν στο}\\ \text{κβαντικό κύκλωμα}\end{pmatrix}\ \text{x}\ \begin{pmatrix}\text{Αποτέλεσμα πίνακα}\\ \text{αρχικής κατάστασης}\\ \text{κβαντικού καταχωρητή}\\ \text{καταχωρητή}\end{pmatrix} =$$

$$=$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & 0 & 1\\ 1 & 0 & 1 & 0\\ 0 & 1 & 0 & -1\\ 1 & 0 & -1 & 0\end{bmatrix}\text{x}\begin{bmatrix}0\\ 1\\ 0\\ 0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0\cdot 0 + 1\cdot 1 + 0\cdot 0 + 1\cdot 0\\ 1\cdot 0 + 0\cdot 1 + 1\cdot 0 + 0\cdot 0\\ 0\cdot 0 + 1\cdot 1 + 0\cdot 0 + (-1)\cdot 0\\ 1\cdot 0 + 0\cdot 1+) - 1)\cdot 0 + 0\cdot 0\end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 0\\ 1\\ 0\end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\ 0\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 1\\ 0\end{bmatrix}\right) = \frac{1}{\sqrt{2}}(\,|\,00\,\rangle + |\,10\,\rangle).$$

Έτσι το αποτέλεσμα της μέτρησης είναι:

$\frac{1}{\sqrt{2}} \left| 00 \right\rangle + \frac{1}{\sqrt{2}} \left| 10 \right\rangle$.

Αποτελεί κβαντική κατάσταση, αφού:

$$\left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1^2}{(\sqrt{2})^2} + \frac{1^2}{(\sqrt{2})^2} = \frac{1}{2} + \frac{1}{2} = \frac{2}{2} = 1.$$

Συνεπώς, μετά την μέτρηση του κυκλώματος, έχουμε 50% πιθανότητα εμφάνισης της κατάστασης $\left| 00 \right\rangle$ και 50% πιθανότητα εμφάνισης της κατάστασης $\left| 10 \right\rangle$, δηλαδή οι πιθανότητες εμφάνισης της κατάστασης $\left| 00 \right\rangle$ ή της κατάστασης $\left| 10 \right\rangle$ είναι ίσες.

Παρακάτω συντάσσουμε σε πρόγραμμα, το παραπάνω κύκλωμα:

```
1.  from qiskit import *
2.  qr = QuantumRegister(2)
3.  cr = ClassicalRegister(2)
4.  initial_state_0 = [1,0]
5.  initial_state_1 = [0,1]
6.  circuit = QuantumCircuit(qr,cr)
7.  circuit.initialize(initial_state_0,0)
8.  circuit.initialize(initial_state_1,1)
9.  circuit.x(1)
10. circuit.h(0)
11. circuit.measure(qr,cr)
12. circuit.draw(output='mpl')
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

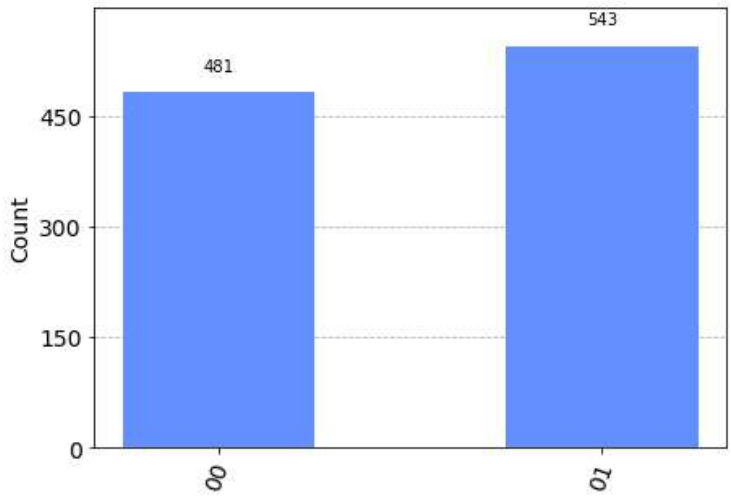
- Γραμμή 1: Ενσωμάτωση της κατάλληλης βιβλιοθήκης.
- Γραμμή 2: Δημιουργία κβαντικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των qubits και με μια μεταβλητή) .
- Γραμμή 3: Δημιουργία κλασικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των bits και με μια μεταβλητή) .
- Γραμμές 4 και 5: Ορισμός των τιμών των ζητούμενων αρχικών καταστάσεων στις αντίστοιχες μεταβλητές.
- Γραμμή 6: Δημιουργία κυκλώματος από 2 qubit και 2 bits.
- Γραμμές 9: Προσθήκη μιας πύλης NOT με αριθμό qubit ένα.
- Γραμμή 10: Προσθήκη μιας πύλης Hadamard με αριθμό qubit μηδέν.

- Γραμμή 11: Μέτρηση του κυκλώματος. (Τα qubits στην ουσία καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα bits.
- Γραμμή 12: Οπτικοποίηση του κυκλώματος. Εμφανίζει (σχεδιάζει) τις συγκεκριμένες πύλες (H) και (NOT) του προγράμματος. Γράφοντας 'mpl' μας δίνει χρωματισμένο το αποτέλεσμα.

Για να εμφανισθεί το γράφημα του κυκλώματος και στη συνέχεια γράφουμε το υπόλοιπο μέρος του προγράμματος:

```
13. simulator = Aer.get_backend('qasm_simulator')
14  result =
execute(circuit,backend=simulator,counts=1024).result()
15. plot_histogram(result.get_counts(circuit))
```

- Γραμμή 13: Επιλογή του προσομοιωτή.
- Γραμμή 14: Αποδίδουμε το αποτέλεσμα της εκτέλεσης του κυκλώματος στην μεταβλητή result και όπως είδαμε, επειδή το πρόγραμμά μας δεν εκτελείται σε πραγματικό υπολογιστή αλλά σε προσομοιωτή, το «τρέχουμε» πολλές φορές για να έχουμε καλύτερη πιθανολογική προσέγγιση του αποτελέσματος (ζητάμε από το πρόγραμμα να εκτελεσθεί τυχαία 1024 φορές το κύκλωμα).
- Γραμμή 15: Εμφανίζεται το αποτέλεσμα του προγράμματος με τη μορφή ιστογράμματος. Παρατηρούμε στο γράφημα τις πολύ μικρές διαφορές στις δύο μετρήσεις.

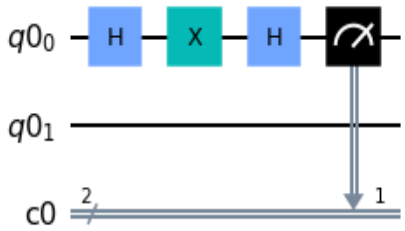


Σχήμα: 4 – Ιστόγραμμα αποτελεσμάτων κβαντικού προγράμματος

Ας δούμε ακόμα ένα παράδειγμα ακολουθώντας την παραπάνω διαδικασία που περιγράψαμε.

ΠΑΡΑΔΕΙΓΜΑ 1

Έχουμε το παρακάτω κύκλωμα των δύο qubits, με αρχική κατάσταση όλων των qubits $|00\rangle$. α) Να επιλύσετε το κύκλωμα αυτό και β). Να γράψετε κώδικα που να εμφανίζει το κύκλωμα αυτό. γ.) Να εμφανίσετε το ιστόγραμμα των αποτελεσμάτων του κβαντικού κυκλώματος.



Σχήμα: 5 – Κβαντικό κύκλωμα δύο qubits

Έχουμε:

α.) Τα δύο qubits έχουν αρχική τιμή $|0\rangle$, οπότε ο κβαντικός καταχωρητής έχει την παρακάτω μορφή:

$$\bullet\ |00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1\begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0\begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Στη συνέχεια, εφαρμόζουμε Hadamard στο πρώτο qubit και στο δεύτερο qubit αφού δεν δρα τίποτα, εφαρμόζουμε την πύλη αδράνειας (I) και το τανυστικό τους γινόμενο είναι:

$$\bullet\ H \otimes I = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & -1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix} 1\cdot 1 & 1\cdot 0 & 1\cdot 1 & 1\cdot 0 \\ 1\cdot 0 & 1\cdot 1 & 1\cdot 0 & 1\cdot 1 \\ 1\cdot 1 & 1\cdot 0 & -1\cdot 1 & -1\cdot 0 \\ 1\cdot 0 & 1\cdot 1 & -1\cdot 0 & -1\cdot 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

Εφαρμόζουμε το παραπάνω αποτέλεσμα στο αποτέλεσμα της αρχικής κατάστασης και έτσι υπολογίζουμε το γινόμενο των παρακάτω πινάκων:

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\cdot 1+0\cdot 0+1\cdot 0+0\cdot 0\\ 0\cdot 1+1\cdot 0+0\cdot 0+1\cdot 0\\ 1\cdot 1+0\cdot 0-1\cdot 0+0\cdot 0\\ 0\cdot 1+1\cdot 0+0\cdot 0-1\cdot 0\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix}.$$

Εφαρμόζουμε τώρα στο πρώτο qubit NOT(X) και πάλι πύλη αδράνειας (I) στο δεύτερο qubit και το τανυστικό τους γινόμενο είναι:

$$\bullet\ X\otimes I=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\otimes\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}=\begin{bmatrix}0\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} & 1\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\\ 1\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} & 0\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\end{bmatrix}=$$

$$=\begin{bmatrix}0\cdot 1 & 0\cdot 0 & 1\cdot 1 & 1\cdot 0\\ 0\cdot 0 & 0\cdot 1 & 1\cdot 0 & 1\cdot 1\\ 1\cdot 1 & 1\cdot 0 & 0\cdot 1 & 0\cdot 0\\ 1\cdot 0 & 1\cdot 1 & 0\cdot 0 & 0\cdot 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}.$$

Εφαρμόζω το παραπάνω αποτέλεσμα, στην προηγούμενη κατάσταση που υπολογίσαμε και έχουμε:

$$\bullet\ \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}\cdot\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}0\cdot 1+0\cdot 0+1\cdot 1+0\cdot 0\\ 0\cdot 1+0\cdot 0+0\cdot 1+1\cdot 0\\ 1\cdot 1+0\cdot 0+0\cdot 1+0\cdot 0\\ 0\cdot 1+1\cdot 0+0\cdot 1+0\cdot 0\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix}.$$

Συνεχίζουμε εφαρμόζοντας τώρα στο πρώτο qubit Hadamard και στο δεύτερο qubit αφού δεν δρα τίποτα, εφαρμόζουμε την πύλη αδράνειας (I) και το τανυστικό τους γινόμενο (το υπολογίσαμε παραπάνω), είναι:

$$\bullet\ H\otimes I=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\end{bmatrix}.$$

Εφαρμόζω το παραπάνω αποτέλεσμα, στην προηγούμενη κατάσταση που υπολογίσαμε και έχουμε:

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix}=\left(\frac{1}{\sqrt{2}}\right)^2\begin{bmatrix}1\cdot 1+0\cdot 0+1\cdot 1+0\cdot 0\\ 0\cdot 1+1\cdot 0+0\cdot 1+1\cdot 0\\ 1\cdot 1+0\cdot 0-1\cdot 1-1\cdot 0\\ 0\cdot 1+1\cdot 0+0\cdot 1-1\cdot 0\end{bmatrix}=$$

$$= \frac{1}{(\sqrt{2})^2}\begin{bmatrix}2\\0\\0\\0\end{bmatrix} = \frac{1}{2}\cdot 2\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = |00\rangle.$$

Άρα ισχύει:

$1\ |00\rangle + 0\ |00\rangle + 0\ |00\rangle + 0\ |00\rangle$, δηλαδή οι υπόλοιπες καταστάσεις στην ουσία δεν υπάρχουν και έχουμε 100 % πιθανότητα να πάμε στην κατάσταση $|00\rangle$, ύστερα από τις παραπάνω δράσεις .

β.) Η υλοποίηση του κυκλώματος δίνεται από το παρακάτω πρόγραμμα:

```
1.  from qiskit import *
2.  qr = QuantumRegister(2)
3.  cr = ClassicalRegister(2)
4.  circuit = QuantumCircuit(qr,cr)
5.  circuit.h(qr[0])
6.  circuit.x(qr[0])
7.  circuit.h(qr[0])
8.  circuit.measure(0,1)
9.  circuit.draw(output='mpl')
```

Να εξηγήσουμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμή 1: Ενσωμάτωση της κατάλληλης βιβλιοθήκης.
- Γραμμή 2: Δημιουργία κβαντικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των qubits και με μια μεταβλητή) .
- Γραμμή 3: Δημιουργία κλασικού καταχωρητή αποτελούμενο από 2 qubits (μπορούμε να αποδώσουμε το πλήθος των bits και με μια μεταβλητή) .
- Γραμμή 4: Δημιουργία κυκλώματος από 3 qubit και 3 bits.
- Γραμμή 5: Προσθήκη πύλης Hadamard, με αριθμό qubit μηδέν.
- Γραμμή 6: Προσθήκη μιας πύλης NOT με αριθμό qubit μηδέν.
- Γραμμή 7: Προσθήκη μιας πύλης Hadamard με αριθμό qubit μηδέν.
- Γραμμή 8: Μέτρηση του κυκλώματος. (Τα qubits στην ουσία καταστρέφονται και η κβαντική τους κατάσταση αποθηκεύεται στα αντίστοιχα bits.
- Γραμμή 9: Οπτικοποίηση του κυκλώματος. Εμφανίζει (σχεδιάζει) την συγκεκριμένη πύλη (H) του προγράμματος. Γράφοντας ‘mpl’ μας δίνει χρωματισμένο το αποτέλεσμα.

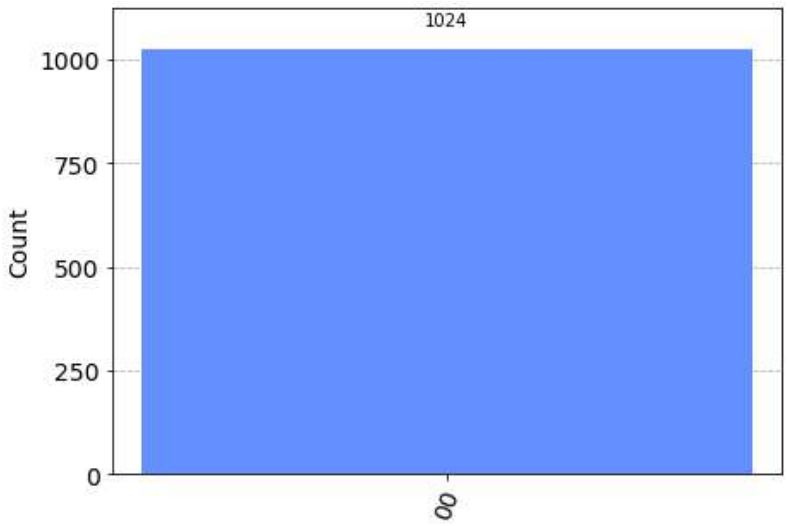
γ.) Για να εμφανισθεί το γράφημα του κυκλώματος και στη συνέχεια γράφουμε το υπόλοιπο μέρος του προγράμματος:

```
10. simulator = Aer.get_backend('qasm_simulator')
11  result =
execute(circuit,backend=simulator,counts=1024).result()
12. plot_histogram(result.get_counts(circuit))
```

Να δούμε αναλυτικά τις παραπάνω τρεις γραμμές της συνέχειας του προγράμματος:

- Γραμμές 10: Επιλογή προσομοιωτή.
- Γραμμή 11: Αποδίδεται στη μεταβλητή result το αποτέλεσμα της εκτέλεσης του κυκλώματος, που καλούμε να εκτελεσθεί το πρόγραμμα 1024 φορές τυχαία.
- Γραμμή 12: Εμφανίζεται το αποτέλεσμα του προγράμματος σε μορφή ιστογράμματος και αυτό για το λόγο ότι με μία γραφική παράσταση μπορούμε να έχουμε μια ολοκληρωμένη εικόνα για το συγκεκριμένο πρόγραμμά μας.

Το ιστόγραμμα των αποτελεσμάτων του προγράμματος φαίνεται παρακάτω:



Σχήμα: 6 – Ιστόγραμμα αποτελεσμάτων κβαντικού προγράμματος

5.2 Κβαντική Διεμπλοκή με πύλη Hadamard και CNOT

Η κβαντική διεμπλοκή (quantum entanglement), είναι μια από τις πιο χαρακτηριστικές ιδιότητες της Κβαντικής Μηχανικής που αξιοποιεί η κβαντική υπολογιστική. Συγκεκριμένα, για τους κβαντικούς υπολογιστές τη κβαντική διεμπλοκή τη χρησιμοποιούμε για να πραγματοποιήσουμε κβαντικούς υπολογισμούς και να αναπτύξουμε αλγόριθμους. Όπως είδαμε και σε προηγούμενα κεφάλαια, όταν δύο qubits βρίσκονται σε κβαντική διεμπλοκή η μέτρηση της κατάστασης του ενός καθορίζει την κατάσταση του άλλου, ακόμα και αν αυτά απομακρυνθούν έπειτα από τεράστιες αποστάσεις. Έτσι, αν ύστερα από μια μέτρηση είναι γνωστή η κατάσταση του ενός από τα δύο qubits, τότε και η κατάσταση του άλλου αυτόματα θα αλλάξει στην ίδια κατάσταση (ως συνέπεια της μέτρησης).

Έτσι, για παράδειγμα αν έχουμε ένα σύστημα το οποίο βρίσκεται στην κατάσταση $|q\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ και έχουμε δύο qubits τα οποία βρίσκονται σε κβαντική διεμπλοκή, τότε αν πραγματοποιήσουμε μια μέτρηση για το πρώτο qubit και βρεθεί στην κατάσταση $|0\rangle$, τότε αυτόματα και η κατάσταση του δεύτερου qubit θα αλλάξει στην ίδια, δηλαδή η κατάστασή του θα είναι $|0\rangle$, χωρίς να χρειαστεί να πραγματοποιήσουμε κάποια μέτρηση για το δεύτερο qubit. Ή αν το πρώτο qubit ύστερα από μέτρηση βρεθεί στην κατάσταση $|1\rangle$, τότε αυτόματα είναι γνωστή και η κατάσταση του δεύτερου qubit που βρίσκεται στην ίδια με το πρώτο qubit κατάσταση $|1\rangle$.

Ακόμα, μπορούμε να πούμε ότι δύο κβαντικά συστήματα βρίσκονται σε κβαντική διεμπλοκή, όταν η κατάστασή τους δεν είναι εφικτή να εκφραστεί με το τανυστικό γινόμενο των βασικών τους καταστάσεων.

Η διαφορά της κβαντικής διεμπλοκής από την κβαντική υπέρθεση είναι ότι όταν δύο qubit βρίσκονται σε κβαντική υπέρθεση η μέτρηση της κατάστασης του ενός qubit δεν καθορίζει την κατάσταση του άλλου qubit, ενώ όταν τα δύο qubit βρίσκονται σε κβαντική διεμπλοκή, η μέτρηση της κατάστασης του ενός qubit καθορίζει την κατάσταση του άλλου.

Ας δούμε τι σημαίνει δύο qubit να βρίσκονται σε κβαντική υπέρθεση.

Εφαρμογή 2

Έχουμε δύο qubit $|q_{s0}\rangle$ και $|q_{s1}\rangle$ που βρίσκονται σε κβαντική κατάσταση $|q_s\rangle$ και ισχύει:

$$|q_s\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle).$$

Η κβαντική κατάσταση $|q_s\rangle$ γράφεται:

$|q_s\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle) = |1\rangle \otimes \left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right]$.

Έτσι, οι καταστάσεις των $|q_{s0}\rangle$ και $|q_{s1}\rangle$ είναι:

$|q_{s0}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ και $|q_{s1}\rangle = |1\rangle$.

Άρα, η $|q_s\rangle$ γράφεται:

$|q_s\rangle = |q_{s1}\rangle \otimes |q_{s0}\rangle$.

Συνεπώς, από την παραπάνω σχέση διαπιστώνουμε ότι η $|q_s\rangle$ είναι δυνατό να γραφεί ως τανυστικό γινόμενο των καταστάσεων των δύο qubits και άρα τα $|q_{s0}\rangle$ και $|q_{s1}\rangle$ δεν βρίσκονται σε κβαντική διεμπλοκή αλλά σε υπέρθεση καταστάσεων.

Ας δούμε τώρα τι σημαίνει δύο qubit να βρίσκονται σε κβαντική διεμπλοκή.

Εφαρμογή 3

Έχουμε δύο qubit $|q_{e0}\rangle$ και $|q_{e1}\rangle$ που βρίσκονται σε κβαντική κατάσταση $|q_e\rangle$ και ισχύει:

$|q_e\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

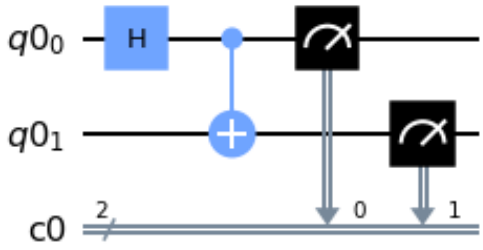
Η κβαντική κατάσταση $|q_e\rangle$ δεν είναι εφικτό να εκφραστεί ως τανυστικό γινόμενο των καταστάσεων των δύο qubits, άρα τα $|q_{e0}\rangle$ και $|q_{e1}\rangle$ βρίσκονται σε κβαντική διεμπλοκή.

Όμως για να δούμε πώς μπορούμε να παράγουμε κβαντική διεμπλοκή, δηλαδή πώς μπορούμε να φέρουμε δύο qubit σε κβαντική διεμπλοκή.

Για να το καταφέρουμε αυτό, θα χρησιμοποιήσουμε δύο μόνο κβαντικές πύλες την κβαντική πύλη Hadamard (H) και την κβαντική πύλη ελεγχόμενης άρνησης CNOT (cX). Η πύλη Hadamard αξίζει να τονιστεί ότι αποτελεί μια από τις σπουδαιότερες κβαντικές πύλες καθώς μπορεί να φέρει σε υπέρθεση ένα qubit αλλά και το επαναφέρει από υπέρθεση σε μία σταθερή κατάσταση.

Εφαρμογή 4

Έχουμε το παρακάτω κβαντικό κύκλωμα με τις δύο κβαντικές πύλες, την Hadamard και την CNOT.



Σχήμα: 7 – Κβαντικό κύκλωμα για την κβαντική διεμπλοκή με δύο qubits

Ο κβαντικός υπολογισμός για το παραπάνω κβαντικό κύκλωμα, είναι:

$\text{CNOT}(H \otimes I)\left|00\right\rangle$.

Όπου,

- $\left|00\right\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

- $H \otimes I = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \\ 1\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} & -1\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{bmatrix} =$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix} 1\cdot 1 & 1\cdot 0 & 1\cdot 1 & 1\cdot 0 \\ 1\cdot 0 & 1\cdot 1 & 1\cdot 0 & 1\cdot 1 \\ 1\cdot 1 & 1\cdot 0 & -1\cdot 1 & -1\cdot 0 \\ 1\cdot 0 & 1\cdot 1 & -1\cdot 0 & -1\cdot 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

Επομένως, η παραπάνω σχέση γίνεται:

$$\bullet\ \text{CNOT}(H \otimes I)\,|00\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle \right).$$

Είναι εφικτό να φέρουμε δύο qubits σε κβαντική διεμπλοκή και συγκεκριμένα σε τέσσερις διαφορετικές καταστάσεις. Αυτή η ασυνήθιστη δυνατότητα εφαρμογής των καταστάσεων ενός καταχωρητή, κατά την οποία δηλαδή δύο απομακρυσμένα qubits βρίσκονται σε κβαντική διεμπλοκή καλούνται «καταστάσεις Bell» (Bell states). Το ζευγάρι των qubits το οποίο δημιουργεί μία τέτοια κατάσταση ονομάζεται και «ζεύγη EPR», από το όνομα της γνωστής δημοσίευσης των Einstein Podolsky και Rosen το 1935. Τη δράση του κυκλώματος τη συμβολίζουμε με το βέλος με το E και για κάθε έναν από τους τέσσερις δυνατούς συνδυασμούς των αρχικών καταστάσεων φαίνεται παρακάτω:

$$\bullet\ |00\rangle \xrightarrow{E} \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle \right)$$

$$\bullet\ |01\rangle \xrightarrow{E} \frac{1}{\sqrt{2}} \left(|01\rangle + |10\rangle \right)$$

$$\bullet\ |10\rangle \xrightarrow{E} \frac{1}{\sqrt{2}} \left(|00\rangle - |11\rangle \right)$$

$$\bullet\ |11\rangle \xrightarrow{E} \frac{1}{\sqrt{2}} \left(|01\rangle - |10\rangle \right)$$

Τις τέσσερις παραπάνω καταστάσεις μπορούμε να τις γράψουμε και ως εξής:

$$\bullet\ |\Phi^+\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B \right)$$

$$\bullet\ |\Phi^-\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B \right)$$

$$\bullet\ |\Psi^+\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B \right)$$

• $|\psi^-\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B \right)$

Το πρόγραμμα του κβαντικού κυκλώματος Σχήμα:4 συντάσσεται:

```
1.  from qiskit import *
2.  qr=QuantumRegister(2)
3.  cr=ClassicalRegister(2)
4.  circuit=QuantumCircuit(qr,cr)
5.  circuit.h(0)
6.  circuit.cx(0,1)
7.  circuit.measure(qr,cr)
8.  %matplotlib inline
9.  circuit.draw(output='mpl')
```

Οι επιπλέον εντολές που χρειάζεται να προσθέσουμε στο παραπάνω πρόγραμμα για να μετρήσουμε τις πιθανότητες των ενδεχομένων αποτελεσμάτων, είναι οι παρακάτω:

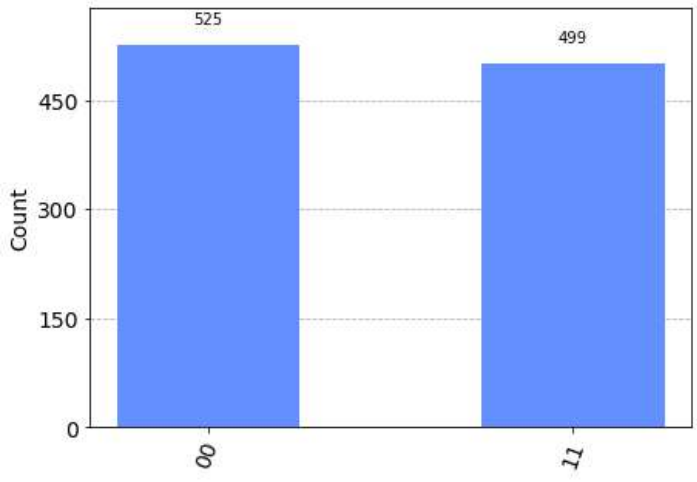
Η επιπλέον βιβλιοθήκη για να αναγνωρίζει το plot_histogram:

```
from qiskit.visualization import plot_histogram
```

Και η συνέχεια των εντολών του παραπάνω προγράμματος:

```
10. simulator=Aer.get_backend('qasm_simulator')
11. result=execute(circuit,backend=simulator).result()
12. plot_histogram(result.get_counts(circuit))
```

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί το ιστόγραμμα των κβαντικών καταστάσεων:



Σχήμα: 8 – Ιστόγραμμα κβαντικών καταστάσεων

Σημείωση

Για να δημιουργήσουμε την Bell state $|\Phi^+\rangle$, όπως είδαμε και παραπάνω, θα πρέπει σε ένα καταχωρητή με δύο qubits, να εφαρμόσουμε στο πρώτο qubit την πύλη Hadamard και στο δεύτερο qubit την πύλη NOT (cX).

Γενικά για να δημιουργήσουμε όλες τις Bell καταστάσεις μπορούμε να χρησιμοποιήσουμε τα παρακάτω κυκλώματα:

$|\Phi^+\rangle$ = H(0) ⟶ cX(0,1)

$|\Phi^-\rangle$ = X(0) ⟶ H(0) ⟶ cX(0,1)

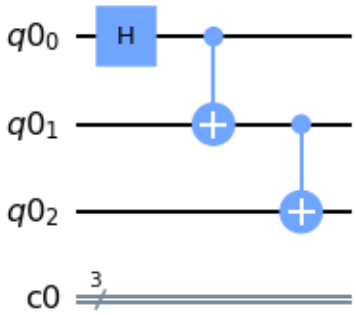
$|\Psi^+\rangle$ = X(1) ⟶ H(0) ⟶ cX(0,1)

$|\Psi^-\rangle$ = X(0) ⟶ H(0) ⟶ cX(0,1) ⟶ Z(0) ⟶ Z(1) ⟶ cX(0,1)

Το ενδιαφέρον και περίεργο της κβαντικής διεμπλοκής είναι που οι καταστάσεις Bell, καθορίζουν σε μεγάλο βαθμό δεδομένου της μέτρησης του κυκλώματος, μετρώντας το πρώτο qubit και αυτόματα μας φανερώνει τις ίδιες μετρήσεις του δεύτερου qubit. Συνεπώς, κατά την αλληλεπίδραση των δύο qubits, η οποιαδήποτε αλλαγή στο πρώτο qubit επηρεάζει και το δεύτερο ανεξάρτητα της απόστασης.

Παρομοίως, μπορούμε να φέρουμε σε κατάσταση κβαντικής διεμπλοκής περισσότερες καταστάσεις των qubits χρησιμοποιώντας τις κβαντικές πύλες Hadamard (H) και της ελεγχόμενης άρνησης CNOT (cX).

Παρακάτω φαίνεται το κβαντικό κύκλωμα που φέρνει σε κατάσταση κβαντικής διεμπλοκής τρία qubits.



Σχήμα: 9 – Κβαντικό κύκλωμα για την κβαντική διεμπλοκή με τρία qubits

Ο κβαντικός υπολογισμός που περιγράφεται στο παραπάνω κύκλωμα, εκφράζεται από τη σχέση:

$(I \otimes CNOT) (CNOT \otimes I) (H \otimes I \otimes I) \left| 000 \right\rangle = \frac{1}{\sqrt{2}} \left(\left| 000 \right\rangle + \left| 111 \right\rangle \right).$

Τέλος, η κβαντική διεμπλοκή είναι κατάλληλη για εργασίες που απαιτούν συντονισμό, συγχρονισμό ή ιδιωτικότητα. Ειδικά όσον αφορά το κβαντικό διαδίκτυο, το οποίο αποτελεί σημαντικό στοιχείο των κβαντικών υπολογιστών και των κβαντικών συστημάτων επικοινωνίας, υποστηρίζει πολυάριθμες εφαρμογές μέσα από την κβαντική διεμπλοκή. Παραδείγματα τέτοιων εφαρμογών είναι η διανομή κβαντικού κλειδιού, η ασφαλής αναγνώριση, η κρυπτογραφία και άλλες πολλές.

5.3 Θεώρημα Μη Κλωνοποίησης (No Cloning Theorem)

Στους κλασικούς υπολογιστές, γνωρίζουμε ότι μπορούμε πολύ απλά να αντιγράψουμε bits, η οποία θεωρείται πολύ συνηθισμένη, κοινότυπη διαδικασία. Αντίθετα, στους κβαντικούς υπολογιστές είναι αδύνατον να αντιγράψουμε την κατάσταση ενός qubit.

Συγκεκριμένα, δεν είναι δυνατό να αντιγράψουμε την κατάσταση ενός άγνωστου qubit, αφού δεν γνωρίζουμε την κατάστασή του. Έτσι, όταν ένα qubit βρίσκεται στις βασικές του καταστάσεις, $\left| 0 \right\rangle$ ή $\left| 1 \right\rangle$, τότε η αντιγραφή του είναι αδύνατη και αυτή η ιδιότητα είναι γνωστή ως το θεώρημα μη - κλωνοποίησης (no-cloning theorem) και εκφράζεται ως εξής:

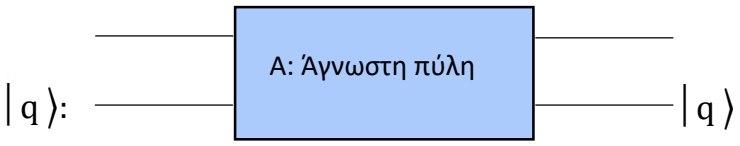
Θεώρημα μη – κλωνοποίησης (No Cloning Theorem)

Δεν μπορεί να υπάρξει μια κβαντική πύλη A τέτοια ώστε:

$$A \left| q\ 0 \right\rangle = \left| q\ q \right\rangle$$

όπου $\left| q\ q \right\rangle$ είναι ένα qubit με άγνωστη κατάσταση.

Για παράδειγμα, θέλουμε να αντιγράψουμε την κατάσταση ενός qubit, τότε χρησιμοποιούμε μια άγνωστη κβαντική πύλη A, η οποία δρα σε δύο qubits. Το πρώτο qubit βρίσκεται σε μία άγνωστη κατάσταση $\left| q \right\rangle$, ενώ το δεύτερο qubit βρίσκεται στην κατάσταση $\left| 0 \right\rangle$ και η κβαντική πύλη A αλλάζει την κατάσταση του qubit που βρίσκεται στην κατάσταση $\left| 0 \right\rangle$ για να είναι ίδια με την κατάσταση $\left| q \right\rangle$. Επομένως, μετά τη δράση της πύλης A και τα δύο qubits βρίσκονται στην ίδια κβαντική κατάσταση $\left| 0 \right\rangle$ και έτσι η κατάσταση $\left| q \right\rangle$ του πρώτου qubit αντιγράφτηκε στο δεύτερο qubit το οποίο είχε αρχική κατάσταση $\left| 0 \right\rangle$. Παρακάτω φαίνεται το κβαντικό κύκλωμα μετά την επίδραση της πύλης A.



Σχήμα: 10 – Υποθετικό κύκλωμα κλωνοποίησης.
Μια τέτοια πύλη Α είναι αδύνατον να υπάρχει.

Έχει αποδειχθεί το 1982 από τους Wootters, Zurek και Dieks, ότι δεν υπάρχει τέτοια πύλη η οποία να μπορεί να αντιγράψει την κατάσταση ενός άγνωστου qubit, αυτό είναι αδύνατον.

Αξίζει να επισημάνουμε ότι το θεώρημα της μη κλωνοποίησης αποτελεί μια σπουδαία ιδιότητα του κβαντικού υπολογισμού, στο οποίο στηρίζεται η κβαντική κρυπτογράφηση.

5.4 Κβαντική Τηλεμεταφορά

Όταν ακούμε τη φράση «τηλεμεταφορά» (teleportation) ο νους μας παραπέμπει στη μετακίνηση ενός αντικειμένου από μια τοποθεσία σε μια άλλη. Χαρακτηριστικό παράδειγμα της τηλεμεταφοράς είναι μια αμερικάνικη τηλεοπτική σειρά επιστημονικής φαντασίας διαδραματιζόμενη στο μέλλον με τίτλο«Star Trek», όπου ένα φυσικό σώμα είναι δυνατό να μετακινηθεί αστραπιαία, με ταχύτητα που πλησιάζει ή και ξεπερνάει την ταχύτητα του φωτός, σε απίστευτα μακρινές αποστάσεις.

Σύμφωνα με την κβαντική υπολογιστική, έχουμε την κβαντική τηλεμεταφορά (quantum teleportation) και μπορούμε να πούμε ότι δεν είναι φανταστική. Η κβαντική τηλεμεταφορά στην ουσία μεταφέρει μια κβαντική κατάσταση, μια κβαντική πληροφορία, από ένα σημείο σε ένα άλλο ή αλλιώς από έναν αποστολέα σε έναν απομακρυσμένο δέκτη (χωρίς να έχουμε απώλειες). Δηλαδή, στους κβαντικούς υπολογιστές γενικά, δεν έχουμε αντιγραφή, αλλά μόνο μετακίνηση. Συγκεκριμένα, με την κβαντική τηλεμεταφορά μπορούμε να μεταφέρουμε την κατάσταση ενός ή περισσοτέρων qubits από τον αποστολέα στον παραλήπτη, σε τεράστιες αποστάσεις. Έχουν υλοποιηθεί τηλεμεταφορές σε πολλές περιπτώσεις και σε πειράματα ακόμα και σε πάρα πολύ μεγάλες αποστάσεις και συγκεκριμένα μεταξύ και τελικού σημείου τηλεμεταφοράς έχουν ξεπεράσει απόσταση τα 100 χιλιόμετρα.

Ακόμα, το spooky action at a distance του A. Einstein, ίσως μας θυμίζει τις καταστάσεις Bell κατά την οποία αλλάζοντας η τιμή στο πρώτο qubit, αλλάζει ακαριαία και η τιμή του δεύτερου qubit στην ίδια, γεγονός που πραγματικά συμβαίνει.

Η κβαντική τηλεμεταφορά εκτιμάται ως ένα από τα πιο σημαντικά πρωτόκολλα στην κβαντική πληροφορία και οι καταστάσεις Bell αποτελούν πρωταγωνιστικό ρόλο σε αυτή. Ένα πρωτόκολλο διαδικτύου (Internet Protocol) αποτελεί την κυριότερη σύμβαση επικοινωνίας για μετάδοση πακέτων δεδομένων, κάνοντας τη βέλτιστη δυνατή προσπάθεια για να αποδώσει ένα πακέτο στο προορισμό του, σε ένα διαδίκτυο. Βέβαια, ένα πρωτόκολλο δεν εγγυάται ότι θα αντιμετωπίσει κάποια προβλήματα κατά την μετάδοση των πακέτων, όπως είναι η αλλοίωση δεδομένων, η απώλεια ή η επανάληψη αυτοδύναμου πακέτου, η επίδοση με καθυστέρησή του. Η κβαντική διεμπλοκή αντικαθιστά το κβαντικό κανάλι, όπως θα δούμε παρακάτω.

Ένα κλασικό κανάλι μπορεί μόνο να μεταφέρει ή να αποθηκεύσει μια κλασική πληροφορία. Αντίθετα, τα κβαντικά κανάλια, μπορούν να διαβιβάσουν κλασική και κβαντική πληροφορία ώστε να έχουμε την κβαντική αποθήκευση.

Ας περιγράψουμε τη διαδικασία της κβαντικής τηλεμεταφοράς.

Εφαρμογή 5

Η Alice θέλει να στείλει έγκαιρα μία σημαντική πληροφορία του qubit, έστω x = α $|0\rangle$ + b $|1\rangle$ στον Bob χωρίς να το αντιληφθεί κάποιος άλλος και έτσι θα χρησιμοποιήσει την κβαντική τηλεμεταφορά. Ουσιαστικά αυτή η σημαντική πληροφορία είναι οι συντελεστές από τα πλάτη πιθανοτήτων των $|0\rangle$ και $|1\rangle$, δηλαδή είναι οι αριθμοί α και b. Ένα τρίτο πρόσωπο ο Ερμής θα μας βοηθήσει για να ολοκληρωθεί με ασφάλεια αυτή η διαδικασία. Ακόμα, δεδομένο είναι ότι η Alice δεν μπορεί να αντιγράψει ώστε να στείλει την κβαντική κατάσταση του $|x\rangle$, λόγω του θεωρήματος της μη κλωνοποίησης (nom cloning theorem: είναι αδύνατο να δημιουργηθεί ένα πανομοιότυπο αντίγραφο μιας αυθαίρετης άγνωστης κβαντικής κατάστασης). Έτσι, ο Ερμής θα πρέπει να στείλει στην Alice και τον Bob μια κατάσταση διεμπλοκής (Bell state), χρησιμοποιώντας ένα κβαντικό κύκλωμα διεμπλοκής φέρνοντας δύο qubits σε κατάσταση διεμπλοκής όπου συναντώνται. Η Alice και ο Bob παίρνουν δύο qubits, που βρίσκονται και τα δύο στην κατάσταση $|0\rangle$ και τα φέρνουν σε κατάσταση διεμπλοκής. Συγκεκριμένα, ο καθένας παίρνει ένα από αυτά τα δύο qubits και ενώ η Alive παραμένει στη θέση της και με ανάλογες ενέργειες στο qubit της στέλνει τα αποτελέσματα στον Bob ο οποίος απομακρύνεται από το αρχικό σημείο.

Η παραπάνω διαδικασία περιγράφεται με τις ακόλουθες ενέργειες:

• <u>Πρώτο στάδιο του κυκλώματος, μετά την αρχικοποίηση του κυκλώματος.</u>

Ο Ερμής δημιουργεί μια κβαντική κατάσταση Bell, έστω $|b\rangle$ = A $|b_0\rangle$ + B $|b_1\rangle$

• <u>Δεύτερο στάδιο του κυκλώματος.</u>

Η Alice δρα με μια κβαντική πύλη CNOT (cX) στο qubit που έλαβε με qubit ελέγχου αυτό που επιθυμεί, δηλαδή το $|x\rangle$, αλλάζοντας την κατάσταση του κβαντικού

καταχωρητή. Στην συνέχεια η Alice δρα με μια πύλη Hadamard στο $|x\rangle$, αλλάζοντας σε μια νέα κατάσταση τον κβαντικό καταχωρητή. Υπολογίζει το αποτέλεσμα του κυκλώματος και το αποθηκεύει σε δύο κλασικά bit και στέλνει τα δύο bits στον Bob.

• Τρίτο στάδιο του κυκλώματος

Αφού ο Bob λάβει τα δύο qubits από την Alice εκτελεί ένα από τα παρακάτω σχετικά με ότι έλαβε από τον Ερμή, ανάλογα με την τιμή των δύο κλασικών bits που έλαβε και συγκεκριμένα,

00: δεν χρειάζεται να κάνει κάτι

01: εφαρμογή πύλης X

10: εφαρμογή πύλης Z

11: διαδοχική εφαρμογή των πυλών X και Z.

• Τέταρτο στάδιο του κυκλώματος

Ο Bob μετράει το δικό του κύκλωμα και το μήνυμα της Alice τηλεμεταφέρθηκε.

Ας δούμε με ένα συγκεκριμένο παράδειγμα την κβαντική διαδικασία της κβαντικής τηλεμεταφοράς και με μαθηματικούς υπολογισμούς.

Παράδειγμα 2

Η Alice θέλει να τηλεμεταφέρει μια κβαντική κατάσταση που είναι $x = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$. Ο Ερμής χρησιμοποιεί την Bell κατάσταση $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, στέλνοντας το πρώτο qubit (το qubit της Alice που θέλει να τηλεμεταφερθεί) στην Alice και το δεύτερο qubit (το κοινό qubit) στον Bob. Το τρίτο qubit είναι του Bob που θα μεταφερθεί.

Αφού η Alice και ο Bob έχουν λάβουν την κβαντική κατάσταση που στέλνει ο Ερμής, τότε έχουμε τις ακόλουθες ενέργειες:

• Υπολογίζουμε την αρχική τιμή του καταχωρητή:

$$|x\rangle\,|\Phi^+\rangle = (0.6|0\rangle + 0.8|1\rangle)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) =$$

$$= \frac{1}{\sqrt{2}}(0.6|000\rangle + 0.6|011\rangle + 0.8|100\rangle + 0.8|111\rangle)$$

• Η Δήμητρα δρά με μια πύλη ελεγχόμενου NOT (cX) (η οποία αλλάζει την τιμή του qubit στόχου, μόνο εάν το qubit ελέγχου είναι 1), με qubit ελέγχου το $|x\rangle$ στο κοινό qubit, αλλάζοντας την κατάσταση του κβαντικού καταχωρητή η οποία φαίνεται παρακάτω.

Επομένως, η παραπάνω σχέση γίνεται:

$\frac{1}{\sqrt{2}}\left(0.6\left|000\right\rangle + 0.6\left|011\right\rangle + 0.8\left|110\right\rangle + 0.8\left|101\right\rangle\right)$.

• Τέλος, η Δήμητρα επιδρά στο πρώτο qubit με μια πύλη Hadamard, αλλάζοντας πάλι την κατάσταση του κβαντικού καταχωρητή και η νέα κατάστασή του είναι:

$\frac{1}{\sqrt{2}}\left(0.6\left|000\right\rangle + 0.6\left|100\right\rangle + 0.6\left|011\right\rangle + 0.6\left|111\right\rangle + 0.8\left|010\right\rangle - 0.8\left|110\right\rangle - 0.8\left|101\right\rangle + 0.8\left|001\right\rangle\right)$.

Κάνοντας τις πράξεις, βγάζουμε κοινό παράγοντα τα δύο πρώτα κοινά qubits (που βρίσκονται στον τόπο της Alice) και η παραπάνω σχέση γίνεται:

$\frac{1}{\sqrt{2}}\Big(\left|00\right\rangle\left(0.6\left|0\right\rangle + 0.8\left|1\right\rangle\right) + \left|01\right\rangle\left(0.6\left|1\right\rangle + 0.8\left|0\right\rangle\right) + \left|10\right\rangle\left(0.6\left|0\right\rangle - 0.8\left|1\right\rangle\right) + \left|11\right\rangle\left(0.6\left|1\right\rangle - 0.8\left|0\right\rangle\right)\Big)$.

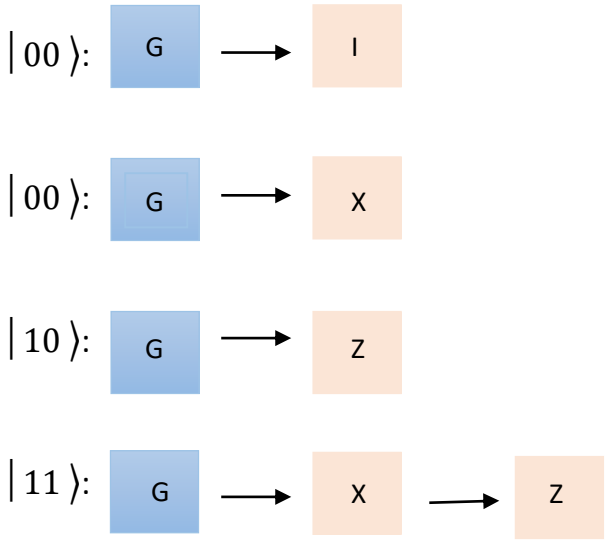
Στην παραπάνω σχέση, κάθε ένας όρους από τους τέσσερις, αποτελεί μία από τις βασικές καταστάσεις των δύο qubits της Alice και το qubit του Bob σε τέσσερις καταστάσεις υπέρθεσης, με πλάτη πιθανότητας 0.6 και 0.8. Έτσι, με τον κβαντικό υπολογισμό που πραγματοποίησε η Alice στα δύο δικά της qubit, (με τη βοήθεια της διεμπλοκής) μετέφερε τα άγνωστα πλάτη πιθανότητας στο qubit του Bob.

Η Alice υλοποιεί μέτρηση των δύο qubits της και το αποτέλεσμα αυτής της μέτρησης είναι μία από τις βασικές καταστάσεις που είναι δυνατό να βρεθούν τα δύο qubits.

Έτσι, από την παραπάνω σχέση έχουμε:

- Αν η Alice μετρήσει $\left|00\right\rangle$, τότε το qubit του Bob βρίσκεται στην κατάσταση:
$$0.6\left|0\right\rangle + 0.8\left|1\right\rangle$$
- Αν η Alice μετρήσει $\left|01\right\rangle$, τότε το qubit του Bob βρίσκεται στην κατάσταση:
$$0.6\left|1\right\rangle + 0.8\left|0\right\rangle$$
- Αν η Alice μετρήσει $\left|10\right\rangle$, τότε το qubit του Bob βρίσκεται στην κατάσταση:
$$0.6\left|0\right\rangle - 0.8\left|1\right\rangle$$
- Αν η Alice μετρήσει $\left|11\right\rangle$, τότε το qubit του Bob βρίσκεται στην κατάσταση:
$$0.6\left|1\right\rangle - 0.8\left|0\right\rangle$$

Άρα, εκτός από την πρώτη περίπτωση, ο Bob θα πρέπει να κάνει χρήση ένα δικό του κβαντικό κύκλωμα έστω "G" (ενημερώνοντας η Alice τον Bob ποια ήταν η μέτρησή της), ώστε να φέρει το qubit στην ίδια με την αρχική κατάσταση του $\left|x\right\rangle$:



• Αν η Alice ενημερώσει τον Bob ότι μέτρησε $|00\rangle$, τότε ο Bob θα κάνει χρήση της πύλης αδράνειας (I) και μόνο αυτό και η κατάσταση του qubit θα είναι:

$$0.6|0\rangle + 0.8|1\rangle = |x\rangle.$$

• Αν η Alice ενημερώσει τον Bob ότι μέτρησε $|01\rangle$, τότε ο Bob θα κάνει χρήση της κβαντικής πύλης X και η κατάσταση του qubit θα είναι:

$$X(0.6|1\rangle + 0.8|0\rangle) = 0.6|0\rangle + 0.8|1\rangle = |x\rangle.$$

• Αν η Alice ενημερώσει τον Bob ότι μέτρησε $|10\rangle$, τότε ο Bob θα κάνει χρήση της κβαντικής πύλης Z και η κατάσταση του qubit θα είναι:

$$Z(0.6|0\rangle - 0.8|1\rangle) = 0.6|0\rangle + 0.8|1\rangle = |x\rangle.$$

• Αν η Alice ενημερώσει τον Bob ότι μέτρησε $|11\rangle$, τότε ο Bob θα κάνει χρήση πρώτα της κβαντικής πύλης X και στη συνέχεια της κβαντικής πύλης Z. Και στο τέλος η κατάσταση του qubit θα είναι:

$$Z\,X(0.6|1\rangle - 0.8|0\rangle) = Z(0.6|0\rangle - 0.8|1\rangle) = 0.6|0\rangle + 0.8|1\rangle = |x\rangle.$$

Ας δούμε λίγο πιο αναλυτικά τα παραπάνω.

Εάν η Alice μετρήσει τα δύο πρώτα qubits, τα οποία αποθηκεύει και στέλνει στον Bob, τότε έχουμε τέσσερις ισοπίθανες περιπτώσεις:

1η <u>Περίπτωση</u>:

➢ 00: Αν ο Bob διαβάσει το δικό του qubit, τότε θα είναι το:

$$0.6|0\rangle + 0.8|1\rangle = |x\rangle.$$

2η Περίπτωση:

- 01: Για να βρούμε το qubit $0.6\left|0\right\rangle + 0.8\left|1\right\rangle$, θα πρέπει να εφαρμοστεί μια πύλη X στο $0.6\left|1\right\rangle + 0.8\left|0\right\rangle$ και είναι:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0 \cdot 0.8 + 1 \cdot 0.6 \\ 1 \cdot 0.8 + 0 \cdot 0.6 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} = 0.6 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 \\ 1 \end{bmatrix} =$$

$$= 0.6\left|0\right\rangle + 0.8\left|1\right\rangle = \left|x\right\rangle.$$

3η Περίπτωση:

- 10: Παρομοίως, για να βρούμε το qubit $0.6\left|0\right\rangle + 0.8\left|1\right\rangle$, θα πρέπει να εφαρμοστεί μια πύλη Z στο $0.6\left|1\right\rangle + 0.8\left|0\right\rangle$ και είναι:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 0.6 - 0 \cdot 0.8 \\ 0 \cdot 0.6 + 1 \cdot 0.8 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} = 0.6 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 \\ 1 \end{bmatrix} =$$

$$= 0.6\left|0\right\rangle + 0.8\left|1\right\rangle = \left|x\right\rangle.$$

4η Περίπτωση:

- 11: Τέλος, στην περίπτωση αυτή για να βρούμε το qubit $0.6\left|0\right\rangle + 0.8\left|1\right\rangle$, θα πρέπει να εφαρμοστούν διαδοχικά οι πύλες X και Z στο $0.6\left|1\right\rangle + 0.8\left|0\right\rangle$.

Πρώτα εφαρμόζουμε την πύλη X και έχουμε:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -0.8 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0(-0.8) + 1 \cdot 0.6 \\ 1(-0.8) + 0 \cdot 0.6 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix}.$$

Στη συνέχεια, εφαρμόζουμε την πύλη Z κα έχουμε:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 0.6 - 0 \cdot 0.8 \\ 0 \cdot 0.6 + 1 \cdot 0.8 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} = 0.6 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.8 \begin{bmatrix} 0 \\ 1 \end{bmatrix} =$$

$$= 0.6\left|0\right\rangle + 0.8\left|1\right\rangle = \left|x\right\rangle.$$

Έτσι σύμφωνα με τα παραπάνω, πραγματοποιείται η τηλεμεταφορά της πληροφορίας της Alice ακαριαία, χωρίς να προλάβει να παρέμβει ένας κακόβουλος, μη επιθυμητό πρόσωπο, ανεξάρτητα με το τι θα στείλει τελικά η Alice, ο Bib ή μη κάνοντας καμία ενέργεια ή εφαρμόζοντας στο δικό της qubit μία ή δύο απλές πύλες.

Παρατήρηση

Κατά την κβαντική κατάσταση μεταφέρεται η κατάσταση του qubit και όχι το φυσικό σύστημα που εκτελεί το άγνωστο qubit $|x\rangle$. Η Alice δεν γίνεται να αποκτήσει κάποιο αντίγραφο της κατάστασης που μετέφερε ο Bob. Ακόμα, η Alice μόλις μετρήσει τα δύο qubits της, το qubit του Bob μεταφέρεται σε μία από τις τέσσερις καταστάσεις ακαριαία, ανεξάρτητα από το πόσο μακριά βρίσκεται.

Παρακάτω φαίνεται το πρόγραμμα της εμφάνισης του κυκλώματος της κβαντικής τηλεμεταφοράς:

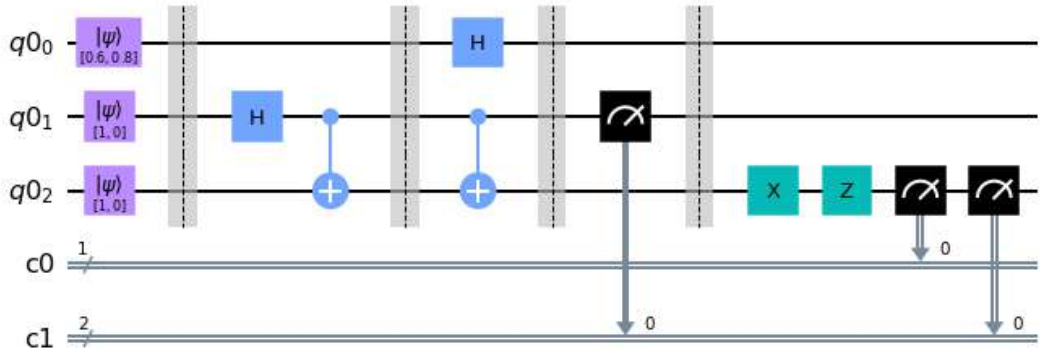
```
1.  from qiskit import *
2.  initial_state_0 = [0.6,0.8]
3.  initial_state_1 = [1,0]
4.  initial_state_2 = [1,0]
5.  qr = QuantumRegister(3)
6.  cr1 = ClassicalRegister(1)
7.  cr2 = ClassicalRegister(2)
8.  circuit = QuantumCircuit(qr,cr1,cr2)
9.  circuit.initialize(initial_state_0,0)
10. circuit.initialize(initial_state_1,1)
11. circuit.initialize(initial_state_2,2)
12. circuit.barrier()
13. circuit.h(1)
14. circuit.cx(1,2)
15. circuit.barrier()
16. circuit.cx(1,2)
17. circuit.h(0)
18. circuit.barrier()
19. circuit.measure(1,1)
20. circuit.barrier()
21. circuit.x(2)
22. circuit.z(2)
23. circuit.measure(2,0)
24. circuit.measure(2,1)
25. circuit.draw(output='mpl')
```

Να δούμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμή 1: Ενσωμάτωση της κατάλληλης βιβλιοθήκης.
- Γραμμές 2, 3 και 4: Απόδοση σε μεταβλητές της αρχικής κατάστασης των τριών qubits.
- Γραμμή 5: Δημιουργία κβαντικού καταχωρητή αποτελούμενο από τρία qubits.
- Γραμμές 6 και 7: Δημιουργία δύο κλασικών καταχωρητών.
- Γραμμή 8: Δημιουργία του κβαντικού κυκλώματος.

- Γραμμές 9, 10 και 11: Αρχικοποίηση των qubits.
- Γραμμή 12: Τοποθέτηση barrier για την καλύτερη κατανόηση του κυκλώματος, το οποίο ισχύει για όλα τα barriers του προγράμματος.
- Γραμμές 12 και 14: Ο Ερμής δημιουργεί την κβαντική κατάσταση Bell.
- Γραμμές 16 και 17: Πρωτόκολλο κβαντικής τηλεμεταφοράς.
- Γραμμές 19 και 20: Η Alice μετράει το αποτέλεσμα που θα στείλει στον Bob, των δύο πρώτων qubits.
- Γραμμές 22 και 23: Ο Bob εφαρμόζει ή όχι, ανάλογα με την περίπτωση, τις αντίστοιχες κβαντικές πύλες. Έστω ότι έλαβε 11 και εφαρμόζει διαδοχικά τις πύλες Z και X στο qubit που τις αναλογεί.
- Γραμμή 26: Εμφάνιση του κβαντικού κυκλώματος.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί:



Σχήμα: 11 – Κύκλωμα κβαντικής τηλεμεταφοράς

Παράδειγμα 3

Ο Ερμής επιλέγει μια διαφορετική κατάσταση Bell, έστω την x = $\frac{3}{5}\left|0\right\rangle + \frac{4}{5}\left|1\right\rangle$. Ο Ερμής χρησιμοποιεί την Bell κατάσταση $\left|\Psi^{-}\right\rangle = \frac{1}{\sqrt{2}}\left(\left|01\right\rangle - \left|10\right\rangle\right) = \left|11\right\rangle$ και το qubit της Alice είναι $\left|x\right\rangle = \alpha\left|0\right\rangle + b\left|1\right\rangle$.

Τότε, έχουμε τις ακόλουθες ενέργειες:

• <u>Υπολογίζουμε την αρχική τιμή του καταχωρητή:</u>

$\left|x\right\rangle\left|\Psi^{-}\right\rangle = \left(\alpha\left|0\right\rangle + b\left|1\right\rangle\right)\frac{1}{\sqrt{2}}\left(\left|01\right\rangle - \left|10\right\rangle\right) =$

$= \frac{1}{\sqrt{2}} (\alpha \left| 001 \right\rangle - \alpha \left| 010 \right\rangle + b \left| 101 \right\rangle - b \left| 110 \right\rangle).$

• Η Δήμητρα επιδρά με μια πύλη ελεγχόμενου NOT (cX) (η οποία αλλάζει την τιμή του qubit στόχου, μόνο εάν το qubit ελέγχου είναι 1), με qubit ελέγχου το $\left| x \right\rangle$ στο κοινό qubit.

Επομένως, η παραπάνω σχέση γίνεται:

$\frac{1}{\sqrt{2}} (\alpha \left| 001 \right\rangle - \alpha \left| 010 \right\rangle + b \left| 111 \right\rangle - b \left| 100 \right\rangle).$

• Τέλος, η Δήμητρα επιδρά στο πρώτο qubit με μια πύλη Hadamard και είναι:

$\frac{1}{\sqrt{2}} (\alpha \left| 001 \right\rangle + \alpha \left| 101 \right\rangle - \alpha \left| 010 \right\rangle - \alpha \left| 110 \right\rangle + b \left| 111 \right\rangle + b \left| 011 \right\rangle - b \left| 100 \right\rangle - b \left| 000 \right\rangle).$

Κάνοντας τις πράξεις, βγάζουμε κοινό παράγοντα τα δύο πρώτα κοινά qubits και η παραπάνω σχέση γίνεται:

$\frac{1}{\sqrt{2}} \left(\left| 00 \right\rangle (\alpha \left| 1 \right\rangle - b \left| 0 \right\rangle) + \left| 01 \right\rangle (-\alpha \left| 0 \right\rangle + b \left| 1 \right\rangle) + \left| 10 \right\rangle (\alpha \left| 1 \right\rangle - b \left| 0 \right\rangle) + \left| 11 \right\rangle (-\alpha \left| 0 \right\rangle + b \left| 1 \right\rangle) \right).$

• Έτσι, εάν η Alice μετρήσει τα δύο πρώτα qubits, τα οποία αποθηκεύει και στέλνει στον Bob, τότε έχουμε δύο ισοπίθανες περιπτώσεις:

1η Περίπτωση:

- 01 και 11: Αν ο Bob διαβάσει το δικό του qubit, τότε αλλάζοντας απλά το πρόσημο θα είναι το:

$$\alpha \left| 0 \right\rangle + b \left| 1 \right\rangle$$

2η Περίπτωση:

00 και 10: Σε αυτήν περίπτωση για να βρούμε το qubit $\alpha \left| 0 \right\rangle + b \left| 1 \right\rangle$, θα πρέπει πάλι, να εφαρμοστούν διαδοχικά οι πύλες X και Z.

Πρώτα εφαρμόζουμε την πύλη X και έχουμε:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0(-b) + 1 \cdot \alpha \\ 1(-b) + 0 \cdot \alpha \end{bmatrix} = \begin{bmatrix} \alpha \\ -b \end{bmatrix}.$$

Στη συνέχεια, εφαρμόζουμε την πύλη Z κα έχουμε:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ -b \end{bmatrix} . = \begin{bmatrix} 1 \cdot \alpha - 0 \cdot b \\ 0 \cdot \alpha + 1 \cdot b \end{bmatrix} = \begin{bmatrix} \alpha \\ b \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \mathrm{b} \begin{bmatrix} 0 \\ 1 \end{bmatrix} =$$

$$= \alpha \left| 0 \right\rangle + \mathrm{b} \left| 1 \right\rangle.$$

Έτσι, παρομοίως, σύμφωνα με τα παραπάνω, πραγματοποιείται ακαριαία η τηλεμεταφορά του qubit της Alice στο qubit του Bob, με τη μόνη διαφορά σε σχέση με τις διαφορετικές καταστάσεις Bell, είναι οι δραστηριότητες του Bob.

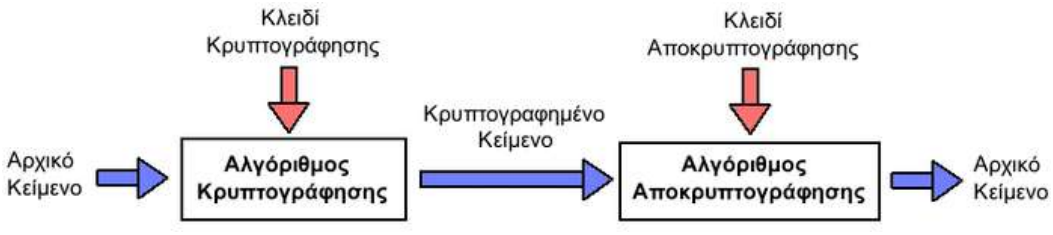
Συνεπώς, η εφαρμογή της τηλεμεταφοράς μιας πληροφορίας είναι σπουδαία διότι αν επιχειρήσει ένα τρίτο πρόσωπο, μη εξουσιοδοτημένο, να υποκλέψει κάποια πληροφορία, δεν θα κατορθώσει να το πετύχει, αφού πραγματοποιείται ακαριαία η τηλεμεταφορά της πληροφορίας, γεγονός που μεταλλάσσει τα πάντα στην κρυπτογραφία.

Η κρυπτογραφία (cryptography) είναι η επιστήμη της ασφαλούς επικοινωνίας μέσω διαδικτύου και υπολογιστικών συστημάτων. Η λέξη κρυπτογραφία διαμορφώνεται από τα συνθετικά «κρυπτός» και «γράφω». Γενικότερα, κρυπτογραφία είναι η ανταλλαγή μηνυμάτων μεταξύ αποστολέα και παραλήπτη με τέτοιο τρόπο ώστε η κατανόηση του κειμένου να είναι δυνατή μόνο από αυτούς τους δύο. Στην καθημερινή μας ζωή εφαρμόζεται πολύ συχνά και κυρίως σε θέματα τηλεπικοινωνίας και ηλεκτρονικής συναλλαγής.

Κρυπτογράφηση (encryption) ονομάζεται η διαδικασία μετασχηματισμού ενός μηνύματος σε μία μορφή ακατανόητη, με τη χρήση κάποιου κρυπτογραφικού αλγορίθμου, ώστε να μην είναι εφικτό να διαβαστεί από κάποιον άλλο εκτός του επιθυμητού παραλήπτη. Η αποκρυπτογράφηση (decryption) είναι η αντίστροφη διαδικασία, όπου το κρυπτογραφημένο κείμενο παράγει το αρχικό μήνυμα.

Η κρυπτογράφηση και η αποκρυπτογράφηση ενός μηνύματος πραγματοποιείται με τη βοήθεια ενός αλγορίθμου κρυπτογράφησης (cipher) (ο οποίος είναι γνωστός) και ενός κλειδιού κρυπτογράφησης (key) στο οποίο βασίζεται η εμπιστευτικότητα του κρυπτογραφημένου μεταδιδόμενου μηνύματος και το μέγεθος του μετριέται σε αριθμό bits. Γενικά, όσο μεγαλύτερο είναι το μέγεθος του κλειδιού κρυπτογράφησης, τόσο δυσκολότερα μπορεί να αποκρυπτογραφηθεί το κρυπτογραφημένο μήνυμα από επίδοξους εισβολείς.

Παρακάτω φαίνεται εικονικά, η γενική διαδικασία κρυπτογράφησης και αποκρυπτογράφησης ενός μηνύματος.



Σχήμα: 12 – Απεικόνιση διαδικασίας κρυπτογράφησης – αποκρυπτογράφησης.

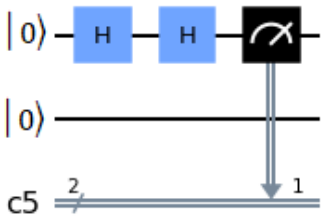
Πηγή εικόνας:
https://el.wikipedia.org/wiki/%CE%9A%CF%81%CF%85%CF%80%CF%84%CE%BF%CE%B3%CF%81%CE%B1%CF%86%CE%AF%CE%B1

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

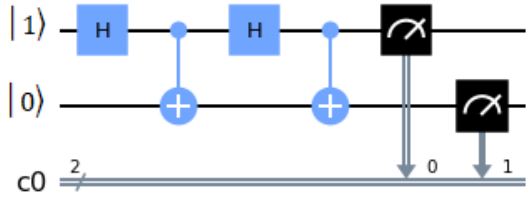
18) Να υπολογισθεί ένα κύκλωμα με τις κβαντικές πύλες X H cX $|0\rangle$. Να επαληθεύσετε την απάντηση με το αντίστοιχο πρόγραμμα.

19) Έχουμε τα παρακάτω κβαντικά κυκλώματα των δύο qubits. Σύμφωνα με τις αρχικές τιμές των δύο qubits σε κάθε κύκλωμα, να πραγματοποιηθεί ο παρακάτω κβαντικός υπολογισμός των παρακάτω κυκλωμάτων:

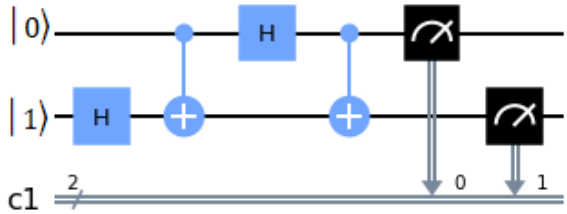
α.)



β.)



γ.)



20) Έχουμε δύο qubits τα οποία βρίσκονται στην κατάσταση:
$|q\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$. Μετά από κάποια μέτρηση, να υπολογιστούν τα παρακάτω:

α.) Την πιθανότητα να βρεθούν στην κατάσταση 00.

β.) Την πιθανότητα να βρεθούν στην κατάσταση 11.

γ.) Την πιθανότητα να βρεθεί το πρώτο qubit στην κατάσταση 0.

δ.) Την πιθανότητα να βρεθεί το δεύτερο qubit στην κατάσταση 1.

21) Έχουμε ένα κβαντικό κύκλωμα με αρχική κατάσταση $|00\rangle$.
α.) Να υπολογίσετε την κβαντική κατάσταση του κυκλώματος.

β.) Να προσθέσετε την κατάλληλη ή τις κατάλληλες κβαντικές πύλες, ώστε η τελική κατάσταση να είναι $|00\rangle$.

22) Έχουμε έναν καταχωρητής με δύο qubits. Να γράψτε πρόγραμμα στο οποίο να εφαρμόζεται τα ακόλουθα βήματα:

α.) Πύλη Hadamard σε όλα τα qubits.

β.) Πύλη Swap στο δεύτερο qubit

γ.) Πύλη Hadamard σε όλα τα qubits.

δ.) Μετρήστε το κύκλωμα και σχολιάστε τα αποτελέσματα.

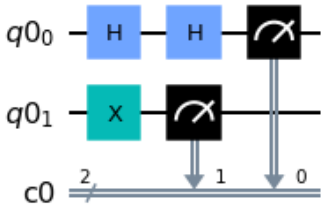
23) Υπολογίστε τις παρακάτω Bell states:

α.) $|\Phi^+\rangle$ = H (0) ⟶ cX (0,1)

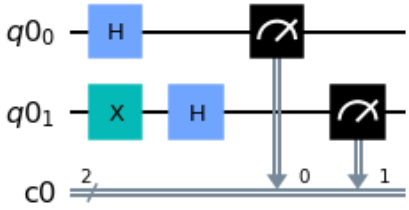
β.) $|\Phi^-\rangle$ = X (0) ⟶ H (0) ⟶ cX (0,1)

24) Δίνονται τα παρακάτω κβαντικά κυκλώματα, στα οποία η κβαντική κατάσταση όλων των qubits είναι $|0\rangle$. Να επιλυθούν με υπολογισμούς και γράφοντας τα αντίστοιχα κβαντικά προγράμματα:

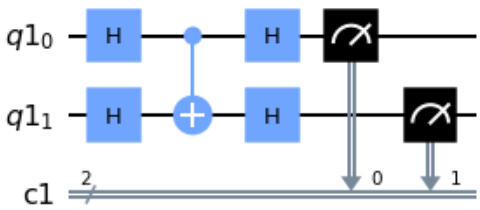
α.)



β.)

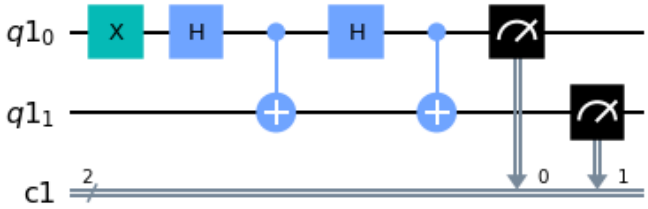


γ.)

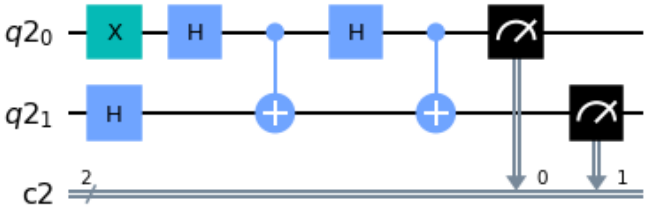


25) Δίνονται τα παρακάτω κβαντικά κυκλώματα, στα οποία η κβαντική κατάσταση όλων των qubits είναι $|0\rangle$. Να επιλυθούν με υπολογισμούς και γράφοντας τα αντίστοιχα κβαντικά προγράμματα:

α.)



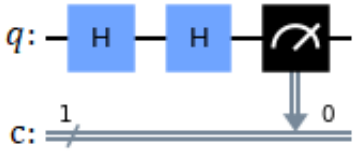
β.)



26) Αποδείξτε ότι τα δύο παρακάτω κυκλώματα δίνουν το ίδιο αποτέλεσμα.

i)

ii)



27) Μετρήστε τις πιθανότητες εμφάνισης 0 και 1 των παρακάτω qubits:

ii) $|q\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ και ii) $|b\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{3}{2}|1\rangle$.

ii) $|q\rangle = \frac{2}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ και ii) $|k\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$.

28) Έχουμε ένα qubit q, το οποίο πρέπει να τηλεμεταφερθεί.

α.) Καταστρέφεται;

β.) Η πληροφορία που περιέχει πρέπει να είναι γνωστή;

Εξηγήστε την απάντησή σας.

29) Να εξηγήσετε αναλυτικά την κάθε γραμμή εντολών του παρακάτω προγράμματος:

```
1.  from qiskit import *
2.  from qiskit.visualization import plot_histogram
3.  from math import pi, sqrt
4.  %matplotlib inline
5.  initial_state_0 = [1,0]
6.  initial_state_1 = [1/sqrt(2),-1/sqrt(2)]
7.  qr = QuantumRegister(2)
8.  cr = ClassicalRegister(2)
9.  circuit = QuantumCircuit(qr,cr)
10. circuit.initialize(initial_state_0,0)
11. circuit.initialize(initial_state_1,1)
12. circuit.h(qr[0])
13. circuit.x(0)
14. circuit.h(qr[1])
15. circuit.cx(0,1)
16. circuit.measure(qr,cr)
17. circuit.draw(output='mpl')
18. circuit.measure(qr,cr)
19. simulator = Aer.get_backend('qasm_simulator')
20.
    result=execute(circuit,backend=simulator,shots=1000).result
    ()
21. plot_histogram(result.get_counts(circuit))
```

ΚΕΦΑΛΑΙΟ: 6ο - Κβαντικά παιχνίδια

ΚΕΦΑΛΑΙΟ 6ο

Κβαντικά Παιχνίδια

Σύνοψη

Σε αυτό το κεφάλαιο αρχικά παρουσιάζεται η έννοια της τυχαιότητας, ποια είναι η σημασία της καθώς και που βρίσκει εφαρμογές γενικά σε ένα κβαντικό φαινόμενο. Στη συνέχεια, περιγράφεται μαθηματικά ένας από τους πιο αρχικούς και σημαντικότερους κβαντικούς αλγόριθμους των Bernstein-Vazirani. Επιπλέον, θίγει τα προβλήματα που μπορεί να επιλύσει, τα βήματα που εφαρμόζει ο αλγόριθμος των Bernstein-Vazirani καθώς και με ποιες εντολές συντάσσεται στο Qiskit. Τέλος, παρουσιάζονται τρία κβαντικά παιχνίδια με τα αντίστοιχα κυκλώματα και προγράμματά τους. Για την καλύτερη εκμάθηση και εμπέδωση της ύλης του μαθητή, δίνονται λυμένα παραδείγματα και εφαρμογές προγραμμάτων κβαντικού υπολογισμού των κβαντικών παιχνιδιών όπως και ασκήσεις για λύση, για την αποδοτική μελέτη και απόκτηση απαραίτητων γνώσεων ώστε να είναι σε θέση οι μαθητές να αναπτύξουν κβαντικά παιχνίδια. Αυτό το κεφάλαιο απευθύνεται σε μαθητές Λυκείου.

6.1 Τυχαιότητα

Οι υπολογισμοί που είδαμε στην τηλεμεταφορά μιας πληροφορίας, το αποτέλεσμά τους δεν είναι εντελώς τυχαίο, γιατί στην ουσία οι πράξεις στηρίζονται στην ψευδοακολουθία και όχι στην τυχαιότητα, αφού στην τυχαιότητα δεν μπορούμε να προβλέψουμε το επόμενο βήμα. Για να το κατανοήσουμε καλύτερα, ας δούμε τι σημαίνει τυχαίος αριθμός ή τυχαία ακολουθία τυχαίων αριθμών.

Γενικά, η σημασία της έκφρασης τυχαίος αριθμός ή τυχαία ακολουθία τυχαίων αριθμών, σημαίνει ότι οι αριθμοί που προκύπτουν δεν είναι αποτέλεσμα από μια σειρά απαράβατων κανόνων και έτσι δεν μπορούν να προβλεφθούν. Αντίθετα, με τον όρο ψευδοτυχαίο εννοούμε μια τυχαία ακολουθία που μπορεί να δημιουργηθεί από μια μαθηματική συνάρτηση, αλλά είναι αδύνατον να μαντέψουμε τον επόμενο όρο μιας τυχαιοκολουθίας ακόμα και στην περίπτωση που μπορεί να γνωρίζουμε κάποιους προηγούμενους όρους.

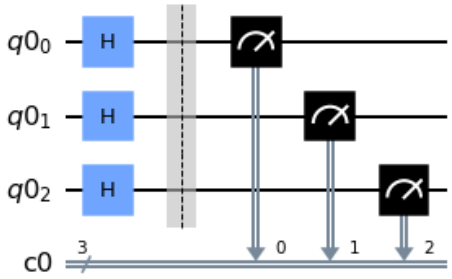
Οι μέθοδοι παραγωγής τυχαίων αριθμών διακρίνονται στις φυσικές και στις μαθηματικές μεθόδους. Παραδείγματα φυσικών μεθόδων είναι η ρίψη ενός ζαριού ή η ρίψη ενός νομίσματος.

Η παραγωγή τυχαίων αριθμών είναι μια ιδιαίτερη διαδικασία καθώς βρίσκει εφαρμογές σε προγράμματα προσομοίωσης φυσικών φαινομένων, στην κρυπτογραφία, στη δημιουργία παιχνιδιών και σε πολλά άλλες.

Όμως, με τη χρήση ενός κβαντικού φαινομένου και συγκεκριμένα της υπέρθεσης σε ένα qubit, είναι εφικτό να μας δώσει μια εντελώς τυχαία ακολουθία και όχι ψευδοτυχαία, αφού η τυχαιότητα να πάρουμε ένα από τα δύο πιθανά αποτελέσματα 0 και 1 είναι ακριβώς η ίδια, δηλαδή 50%.

Παράδειγμα 4

Έχουμε το παρακάτω πρόγραμμα με τρία qubits τα οποία θέτουμε σε υπέρθεση για να προσομοιώνει την ρίψη μιας ζαριάς. Παρακάτω φαίνεται το κύκλωμα:



Σχήμα: 1 – Κβαντικό κύκλωμα προσομοίωσης ρίψη μιας ζαριάς

Η υλοποίηση του προγράμματος είναι:

```
1.  from qiskit import *
2.  qr = QuantumRegister(3)
3.  qr = ClassicalRegister(3)
4.  circuit = QuantumCircuit(qr,cr)
5.  import matplotlib
6.  %matplotlib inline
7.  flag=0   #ελέγχει εάν το αποτέλεσμα είναι αποδεκτό
8.  while flag == 0:
9.      for i in range(3):
10.         circuit.h(qr[i])
11.     circuit.measure(qr,cr)
12.     circuit.draw(output='mpl')
13.     simulator = Aer.get_backend('qasm_simulator')
14.     result=execute(circuit,backend=simulator,shots=1000).
    result()
15.     #print(result.get_counts(circuit))
16.     a = result.get_counts(circuit)
17.     xl = list(a.keys() )
18.     x=str(xl[0])
19.     y=x[::-1]
```

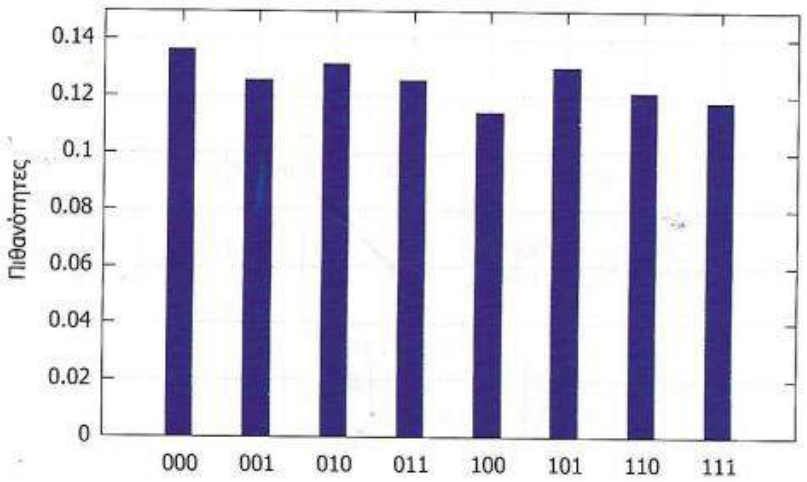
```
20.     d=0
21.     p=0
22.     for i in y:
23.         d += int(i) * (2 ** p)
24.         p += 1
25.     if d > 0  and  d < 7:
26.         print('Ζαριά = ', d )
27.         flag = 1  #αποδεκτό αποτ΄έλεσμα
28. #εισαγωγή της κατάλληλης βιβλιοθήκης
29. from qiskit.visualization import plot_histogram
30. plot_histogram(a)
```

Να δούμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμή 1: Ενσωμάτωση της κατάλληλης βιβλιοθήκης.
- Γραμμή 2: Χρησιμοποιούμε τρία qubits, διότι το ζάρι έχει 6 ενδεχόμενα, για να είναι το αποτέλεσμα της μέτρησης τριψήφιος δυαδικός αριθμός.

  Για παράδειγμα ο τριψήφιος αριθμός 101, μετατρέποντάς τον σε δεκαδικό θα πάρουμε το αποτέλεσμα της μέτρησης το οποίο είναι: $101 \rightarrow 1\ x\ 2^0 + 0\ x\ 2^1 + 1\ x\ 2^2 = 1 + 0 + 4 = 5.$
- Γραμμή 7: Η μεταβλητή flag χρησιμοποιείται για να επαληθεύσει το αποτέλεσμα της ρίψης το οποίο θα πρέπει να είναι από 1 έως 6, αποκλείοντας έτσι τις τιμές 0 και 7.
- Γραμμή 8: Εάν το αποτέλεσμα δεν είναι το επιθυμητό, δηλαδή 0 ή 1, τότε η διαδικασία της ρίψης ζαριού επαναλαμβάνεται.
- Γραμμές 9, 10: Εφαρμογή της πύλης Hadamard σε κάθε qubits.
- Γραμμές 12 και 14: Ο Ερμής δημιουργεί την κβαντική κατάσταση Bell.
- Γραμμές 17 και 18: Καθώς το αποτέλεσμα είναι σε μορφή dictionary της Python στην αρχή παίρνουμε τα κλειδιά του λεξικού τα οποία είναι ένας διψήφιος αριθμός, μετατρέποντάς το σε μία λίστα για να μπορούμε να τα επεξεργαστούμε.
- Γραμμή 19: Αντιστροφή του δυαδικού αριθμού.
- Γραμμές 20, 22, 23, 24 και 25: Μετατροπή του δυαδικού αριθμού σε δεκαδικό αριθμό.
- Γραμμές 25, 26 και 27: Ελέγχουμε αν ο αριθμός είναι ο επιθυμητός, δηλαδή να μην είναι 0 ή 7.
- Γραμμές 28 και 30: Εμφάνιση του αποτελέσματος σε μορφή ιστογράμματος.

Όταν «τρέξουμε» το πρόγραμμα, θα εμφανιστεί:



Σχήμα: 2 – Αποτέλεσμα προγράμματος ρίψης ζαριού σε μορφή ιστογράμματος

6.2 Κβαντικοί Αλγόριθμοι

Οι κβαντικοί αλγόριθμοι «τρέχουν» σε κβαντικούς υπολογιστές χρησιμοποιώντας κβαντικά φαινόμενα, όπως η κβαντική παραλληλία και η κβαντική διεμπλοκή. Όπως και στους κλασικούς υπολογιστές είναι δυνατό να λυθεί ένα πρόβλημα έτσι το ίδιο ισχύει και για τους κβαντικούς υπολογιστές, με το να πλεονεκτούν οι δεύτεροι στο ότι πραγματοποιούν και υπολογίζουν αυτούς τους αλγορίθμους με πολύ μεγάλη ταχύτητα συγκριτικά με τους κλασικούς υπολογιστές.

Όπως στους κλασικούς υπολογιστές, υπάρχουν οι αλγόριθμοι οι οποίοι εκτελούν βήμα προς βήμα μια συγκεκριμένη σειρά εντολών για την επίλυση ενός προβλήματος με τον καλύτερο και γρηγορότερο τρόπο, ομοίως και στους κβαντικούς υπολογιστές έχουμε τους κβαντικούς αλγόριθμους. Ένας κβαντικός αλγόριθμος είναι επίσης μια διαδικασία όπου ακολουθούν πιστά ένα συγκεκριμένο σύνολο βημάτων ή εντολών για την επίλυση ενός προβλήματος με άμεσο και αποτελεσματικότερο τρόπο σε ένα κβαντικό υπολογιστή, με τη διαφορά από τους κλασικούς αλγορίθμους, ότι χρησιμοποιούν κάποια βασικά χαρακτηριστικά του κβαντικού υπολογισμού, όπως η κβαντική υπέρθεση ή η κβαντική διεμπλοκή.

Αυτό που καθιστά ξεχωριστούς τους κβαντικούς αλγόριθμους είναι ότι μπορούν να λύσουν ορισμένα προβλήματα πολύ πιο γρήγορα από τους κλασικούς αλγόριθμους και αυτό γιατί οι κβαντικοί αλγόριθμοι εκμεταλλεύονται την κβαντική υπέρθεση και

την κβαντική διεμπλοκή, καθώς δεν μπορούν να προσομοιωθούν αποτελεσματικά σε κλασικούς υπολογιστές.

Οι κβαντικοί αλγόριθμοι μπορούν να ταξινομηθούν και να ομαδοποιηθούν σε σχέση με τις βασικές τεχνικές που χρησιμοποιούνται από τον αλγόριθμο και τον τύπο του προβλήματος που έχουμε.

Οι πιο γνωστοί αλγόριθμοι είναι ο αλγόριθμος του Shor για παραγοντοποίηση και ο αλγόριθμος του Grover για την αναζήτηση μιας μη δομημένης βάσης δεδομένων ή μιας μη ταξινομημένης λίστας. Ο πρώτος κβαντικός αλγόριθμος που λύνει ένα πρόβλημα πιο αποτελεσματικά από τον πιο γνωστό κλασικό αλγόριθμο είναι ο κβαντικός αλγόριθμος του Bernstein-Vazirani. Άλλοι κβαντικοί αλγόριθμοι είναι ο αλγόριθμος των Deutsch και του Jozsa καθώς και ο αλγόριθμος του Simon οι οποίοι λύνουν ένα πρόβλημα μαύρου κουτιού και άλλοι πολλοί.

Εμείς σε αυτή την ενότητα, θα ασχοληθούμε και θα εφαρμόσουμε τον αλγόριθμο Bernstein – Vazirani, εισάγοντας αρχικά το πρόβλημα, την κλασική του λύση καθώς και τον κβαντικό αλγόριθμο για την επίλυσή του. Στη συνέχεια με τη βοήθεια του Qiskit πραγματοποιούμε τον κβαντικό αλγόριθμο «τρέχοντας» σε προσομοιωτή ή σε κάποια συσκευή.

6.3 Γιατί Bernstein - Vazirani

Θα χρησιμοποιήσουμε τον αλγόριθμο Bernstein - Vazirani, γιατί αυτός ο κβαντικός αλγόριθμος δεν περιέχει συναρτήσεις, με αποτέλεσμα η όλη διαδικασία να είναι πιο απλή σε σύγκριση με τους υπόλοιπους αλγόριθμους, πολύ γρήγορη και πιο εύχρηστη. Σκοπός αυτού του αλγόριθμου είναι αναζήτηση μιας «κρυφής» ακολουθίας bit(bit-sequence), που αποτελείται για παράδειγμα από (n) bit. Έτσι σε αντίθεση με ένα κλασικό υπολογιστή που χρειάζεται (n) αναζητήσεις για την εύρεσή μιας συνάρτησης (s), με τη χρήση ενός κβαντικού υπολογισμού απαιτείται μία και μόνο αναζήτηση. Συνεπώς, ο αλγόριθμος Bernstein – Vazinari αντί να διακρίνει δύο διαφορετικές κατηγορίες συναρτήσεων, προσπαθεί να ανακαλύψει μια συμβολοσειρά κωδικοποιημένη σε μία συνάρτηση και για αυτό το λόγο αποτελεί τον καταλληλότερο αλγόριθμο ο οποίος παρέχει την καλύτερη υπολογιστική πολυπλοκότητα.

6.4 Κβαντικός Αλγόριθμος Bernstein - Vazirani

Ο αλγόριθμος Bernstein - Vazirani είναι ένας κβαντικός αλγόριθμος που δημιουργήθηκε αρχικά από τον Ethan Bernstein και τον Umesh Vazirani το 1992. Ο αλγόριθμος αυτός είναι μια επέκταση του προηγούμενο αλγορίθμου Deutsch - Jozsa σε μια γενίκευση για την εύρεση μιας κρυφής συμβολοσειράς bit, δηλαδή προσπαθεί να μάθει μια συμβολοσειρά κωδικοποιημένη σε μια συνάρτηση. Έδειξε ότι σε σχέση με τον αλγόριθμο Deutsch - Jozsa πλεονεκτεί στη χρήση ενός κβαντικού υπολογιστή ως υπολογιστικού εργαλείου για πιο σύνθετα προβλήματα.

6.5 Ορισμός προβλήματος Bernstein - Vazirani

Θεωρούμε μια Boolean συνάρτηση την $f: \{0,1\}^n \to \{0,1\}$ δηλαδή είναι 0 ή 1, με $f(x) = s \cdot x (modulo 2) = (s_1 * x_1 \otimes s_2 * x_2 \otimes \dots \otimes s_n * x_n)\ (modulo 2)$, για μια άγνωστη συμβολοσειρά $s \in \{0,1\}^n$.

Σκοπός είναι να προσδιοριστεί η άγνωστη συμβολοσειρά s.

Η πιο κλασική και αποτελεσματική μέθοδος για την εύρεση μιας μυστικής συμβολοσειράς είναι η αξιολόγηση της συνάρτησης (n) φορές με τιμές εισόδου (input) $x = 2^i$ με $i \in \{0,1, \dots, n-1\}$ και είναι:

$$f(1000 \dots 0_n) = S_1$$
$$f(0100 \dots 0_n) = S_2$$
$$\vdots$$
$$f(0010 \dots 1_n) = S_n$$

Έτσι, για την κλασική λύση που χρειάζεται τουλάχιστον (n) ερωτήματα της συνάρτησης (s), με τη χρήση κβαντικού υπολογισμού απαιτείται ένα και μόνο ερώτημα.

Ο κβαντικό αλγόριθμος είναι ως εξής:

Εφαρμόζουμε έναν μετασχηματισμό Hadamard στο n στην κατάσταση qubit

$\left| 0 \right\rangle^{\otimes n}$ και έχουμε:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^{n-1}} \left| \mathrm{x} \right\rangle$$

Στη συνέχεια, εφαρμόζουμε τον χρησμό U_f που μετασχηματίζει το $\left| x \right\rangle \to (-1)^{f(x)} \left| x \right\rangle$ και μπορεί να προσομοιωθεί μέσω του μετασχηματισμού:

$$\left| b \right\rangle \left| x \right\rangle \to \left| b \otimes \mathrm{f(x)} \right\rangle \left| x \right\rangle.$$

Εφαρμόζοντας τον χρησμό σε $\frac{|x\rangle-|1\rangle}{\sqrt{2}}\ |x\rangle$ (προσθήκη mod two), μετατρέποντας την υπέρθεση σε:

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^{n-1}}(-1)^{f(x)}\,|\mathrm{x}\rangle$$

Ακολουθεί μετασχηματισμός πύλης Hadamard σε κάθε qubit και για $s_i = 1$ η κατάστασή του μετατρέπεται από $|-\rangle$ σε $|1\rangle$ και παρομοίως για $s_i = 0$ η κατάσταση του μετατρέπεται από $|+\rangle$ σε $|0\rangle$.

Έτσι, ο αλγόριθμος για $H^{\otimes n}$ είναι ο μετασχηματισμός Hadamard σε (n) qubits και αναπαρίσταται ως εξής:

$$|0\rangle^n \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)+xy}\,|x\rangle \xrightarrow{H^{\otimes n}}$$

$\frac{1}{2^n}\sum_{x,y\in\{0,1\}^n}(-1)^{f(x)+xy}\ |y\rangle = |s\rangle$.

Είναι $|s\rangle$ για ένα συγκεκριμένο y και έτσι:

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)+x\,y} = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{x(ss\otimes\,\mathrm{y})} = 1$$

Ισχύει s ⊗ y = 0 μόνο όταν a = y, δηλαδή το μόνο μηδενικό πλάτος είναι ενεργοποιημένο |s⟩ και η μέτρηση της εξόδου του κυκλώματος στην υπολογιστική βάση αποδίδει τη μυστική συμβολοσειρά s.

Ας δούμε μια εφαρμογή του αλγόριθμου Bernstein – Vazirani.

Εφαρμογή 5

Έχουμε μια συνάρτηση f ως μια λειτουργία ενός «μαύρου κουτιού» (black box), δηλαδή ένα σύστημα το οποίο χωρίς να γνωρίζουμε την εσωτερική του λειτουργία, είναι δυνατό να προσδιοριστεί το σύστημα από την εικόνα των εισόδων (inputs) και των εξόδων (outputs). Η f που παίρνει τιμές στην είσοδό (input) του μια σειρά από x bit και επιστρέφει είτε 0 ή 1.

Έστω η συνάρτηση $f\colon\{0,1\}^n \to \{0,1\}$ δηλαδή είναι 0 ή 1, (input: n-bits, output: 1 bit) με $f(x) = s\cdot x\ modulo 2 = (s_1\cdot x_1 \otimes\ s_2\cdot x_2 \otimes \ldots \otimes s_n\cdot x_n)$ για μια άγνωστη συμβολοσειρά s $\{0,1\}^n$.

Σκοπός είναι να προσδιοριστεί η κρυφή bit-sequence, δηλαδή η άγνωστη συμβολοσειρά s.

Αν για παράδειγμα θεωρήσουμε ότι n = 3 και s = 010, τροφοδοτώντας το σύστημα με input x και αν $x_1 = 100,\ x_2 = 001,\ x_n = 110$ τότε έχουμε:

$$x_1 \cdot s = \big(1 \cdot 1 + 0 \cdot 0 + 0 \cdot 0\big) modulo 2 = 1$$

$$x_2 \cdot s = \big(1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1\big) modulo 2 = 0$$

$$x_3 \cdot s = \big(1 \cdot 1 + 1 \cdot 1 + 0 \cdot 0\big) modulo 2 = 0$$

Επομένως, μας εκφράζει πλήρως τη λειτουργία ενός «μαύρου κουτιού» των αντίστοιχων διανυσμάτων των modulo2.

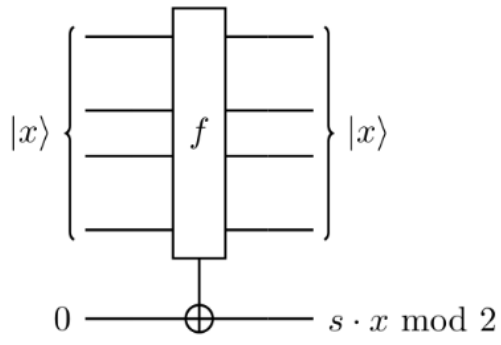
Για κάθε μια είσοδο (input) παίρνουμε ως έξοδο το n-οστό στοιχείο της συμβολοσειράς s και ύστερα από n (input), καθορίζονται όλα τα $s_1, s_2, \dots, s_n$ και επομένως είναι δυνατή η επανάκτηση της συμβολοσειράς s, η οποία η τροφοδότηση πραγματοποιείται μόνο με μια είσοδο (input).

Έτσι, μας δίνει στην είσοδο (input) x και επομένως, η κρυφή συμβολοσειρά s (bit), μιας ήδη γνωστής για παράδειγμα 3-bits συνάρτηση, είναι δυνατό να ανταποκριθεί στην ιδιότητα των Bernstein – Vazirani, $f(x) = s \cdot x$ ή διαφορετικά να αποκαλυφθεί τροφοδοτώντας το σύστημα με την ακολουθία των παρακάτω x εισόδων (inputs) προσδιορίζοντας την κρυφή bit-sequence s:

x	f(x)		x	f(x)
000	0		100	0
001	0		101	0
010	1		110	1
011	1		111	1

Σχήμα: 3 – Bernstein – Vazirani: 3bits συνάρτηση

Παρακάτω φαίνεται ένα κύκλωμα που υλοποιεί τη συνάρτηση:



Σχήμα: 4 – Κβαντικό κύκλωμα με υλοποίηση αλγορίθμου Bernstein – Vazirani

• Η κλασική λύση

Η συνάρτηση:

$$f(x) = s \cdot x (modulo 2)$$

μας δίνει στην είσοδο (input) x και επομένως, η κρυφή συμβολοσειρά s (bit-sequence) είναι δυνατό να αποκαλυφθεί τροφοδοτώντας το σύστημα με την ακολουθία των διαφορετικών εισόδων (inputs):

Είσοδος (x)
100 . . . 0
010 . . . 0
001 . . . 0
000 . . . 1

Κάθε τροφοδότηση μας δίνει ένα διαφορετικό s (bits) .

Έτσι, για να βρούμε την κρυφή συμβολοσειρά (bit-sequence) αρκεί να θέσουμε συγκεκριμένα ερωτήματα στη συνάρτηση με διαφορετικές τιμές στην είσοδο.

Για παράδειγμα ως ερώτημα, με αρχική κατάσταση '100…0' ως πρώτο ερώτημα και πρώτο bit είναι 1, τότε η συνάρτηση μας δίνει 1, διαφορετικά αν είναι 0 τότε μας δίνει 0. Παρομοίως, για το δεύτερο bit και ως ερώτημα έχουμε την κατάσταση '010…0' κατά το οποίο ζητάμε ένα bit ανά αναφορά ώστε το ερώτημα να γίνεται πιο απλό. Συγκεκριμένα, αν αρχικά έχουμε '0' το οποίο το συγκρίνουμε με την έξοδο της συνάρτησης και αν είναι ίσα, τότε δεν κάνουμε κάτι, αλλιώς το τροποποιούμε σε '0' ή '1'.

- $x = 100 \cdots 0 \Rightarrow f(x) = s_1$
- $x = 010 \cdots 0 \Rightarrow f(x) = s_2$
- …
- $x = 000 \cdots 1 \Rightarrow f(x) = s_n$

Συνεπώς, για κάθε μια είσοδο (input) παίρνουμε ως έξοδο το n-οστό στοιχείο της συμβολοσειράς s και ύστερα από n (input), καθορίζονται όλα τα $s_1, s_1, \cdots, s_n$ και επομένως είναι δυνατή η επανάκτηση της συμβολοσειράς s, η οποία η τροφοδότηση πραγματοποιείται μόνο με μια είσοδο (input).

Παρακάτω φαίνεται το πρόγραμμα των Bernstein – Vazirani στην κλασική προσέγγιση:

Είναι:

```
1.  s = input('Secret binary number =');
2.  n = len(s)  #number of digits
3.  s = list(s)  #change the type of s list
4.  t = []  #empty list
```

```
5.  for i in range(n):  #initialize t->'0000...'
6.      t.append('o')
7.  for i in range(n):
8.      if t[i] != s[i]:
9.          t[i] = '1'
10. for i in range(n):
11.     print(t[i], end = '  ')
```

• Η κβαντική λύση

Το σπουδαίο με τους κβαντικούς υπολογιστές είναι η υπολογιστική τους ικανότητα για την εύρεση της μυστικής συμβολοσειράς της ακολουθίας bit με ένα και μόνο βήμα.

Ο αλγόριθμος Bernstein – Vazirani αποτελείται από τέσσερα βήματα:

1. Εφαρμογή της πύλης Hadamard σε κάθε qubit.
2. Εφαρμογή του συγκεκριμένου μετασχηματισμού (στο συγκεκριμένο παράδειγμα είναι ο μετασχηματισμός του μαύρου κουτιού) είναι Q_f που υλοποιεί τη συνάρτηση f.
3. Εφαρμογής της πύλης Hadamard σε κάθε qubit.
4. Μέτρηση κάθε qubit.

Εφαρμογή 6

Θεωρούμε μια μυστική συμβολοσειρά τριών bits (n = 3), για παράδειγμα έχουμε $s = \{0,1,1\}$.

Ο αλγόριθμος των Bernstein – Vazirani αποτελείται από τα παρακάτω τέσσερα βήματα:

➢ 1ο <u>Βήμα του αλγόριθμου:</u>
Αρχικά δημιουργούμε έναν καταχωρητή. Επειδή το μέγεθος της κρυφής συμβολοσειράς αποτελείται από τρεις αριθμούς, ο καταχωρητής που θα δημιουργήσουμε αποτελείται από 3 qubits, με αρχική κατάσταση 0 και έχουμε:

• $|000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}1\\0\end{bmatrix}\\0\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} =$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

- 2ο <u>Βήμα του αλγόριθμου:</u>

Εφαρμόζουμε μια πύλη Hadamard σε κάθε qubits (τρία) που έχουμε:

$$H^{\otimes 3} = H \otimes H \otimes H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} =$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \Rightarrow$$

$$H^{\otimes 3} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix}.$$

Άρα,

$$H^{\otimes 3}\left|000\right\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow$$

$$H^{\otimes 3} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow$$

$H^{\otimes 3} = \frac{1}{\sqrt{2}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + 111\rangle)$
⇒

$H^{\otimes 3} = \frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} |x\rangle$.

- 3ο Βήμα του αλγόριθμου:

Εφαρμόζουμε κάποιου σχετικά με το ερώτημα της συγκεκριμένης συνάρτησης του προβλήματος (oracle), στο κβαντικό κύκλωμα και γράφεται ως εξής:

$\frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} |x\rangle \Rightarrow \frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} (-1)^{f(x)} |\mathrm{x}\rangle$.

Σύμφωνα με την ιδιότητα του αλγόριθμο των Bernstein – Vazirani και για $f(x) = x \cdot s$ και η παραπάνω σχέση γίνεται:

$\frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} (-1)^{f(x)} |\mathrm{x}\rangle = \frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} (-1)^{x\cdot s} |\mathrm{x}\rangle$ και είναι:

$$\frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} (-1)^{x\cdot s} |\mathrm{x}\rangle = \frac{1}{2\sqrt{2}}$$

$\left((-1)^{000\ \ 011} |000\rangle + (-1)^{001\ \ 011} |001\rangle + \cdots + (-1)^{001\ \ 011} |111\rangle\right)$=

$$= \frac{1}{2\sqrt{2}} = (|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle - |110\rangle + 111\rangle) =$$

$$= \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}.$$

- 4ο Βήμα του αλγόριθμου:

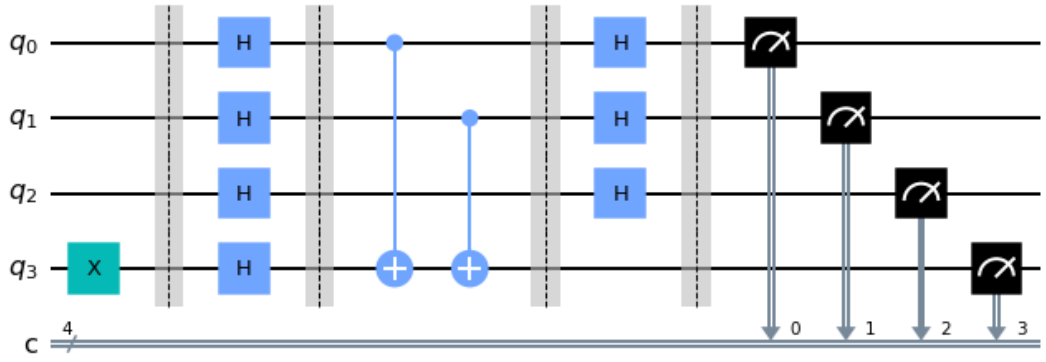
Εφαρμόζουμε παρομοίως μια πύλη Hadamard σε κάθε qubits (τρία) που έχουμε:

$H^{\otimes 3} \frac{1}{2\sqrt{2}} \sum_{x\in\{0,1\}^3} (-1)^{x\cdot s} |\mathrm{x}\rangle =$

$$= \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix} \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} =$$

$$= \left(\frac{1}{2\sqrt{2}}\right)^2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 8 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 8 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |011\rangle.$$

Παρακάτω φαίνεται η τελική μορφή του κβαντικού κυκλώματος:



Σχήμα: 5 – Κβαντικό κύκλωμα Bernstein – Vazirani για $s = \{0, 1, 1\}$

Έτσι, φαίνεται η μυστική συμβολοσειρά, δηλαδή η ακολουθία bit $s = \{0,1,1\}$ η οποία υπολογίστηκε με μία μόνο προσπάθεια. Συνεπώς, σύμφωνα με την παραπάνω εφαρμογή, αντιλαμβανόμαστε τη σπουδαιότητα των κβαντικών υπολογιστών, η οποία αναμφισβήτητα είναι εντυπωσιακά καταπληκτική.

Ας δούμε ένα παράδειγμα.

Παράδειγμα 5

Θέλουμε να βρούμε μια μυστική συμβολοσειρά η οποία αποτελείται από n bits. Ο αλγόριθμος Bernstein-Vazirani ενδείκνυται πάρα πολύ για τη συγκεκριμένη διεργασία και τα βήματα της εφαρμογής του όπως είδαμε παραπάνω, είναι:

1. Θεωρούμε έστω $s = \{s_0, s_1, s_2, \cdots, s_{n-1}\}$, με s_i να είναι ο ή 1.
2. Δημιουργία ενός κβαντικού καταχωρητή με (n) qubits και με αρχική κατάσταση $|0\rangle$.
3. Εφαρμογή της πύλης Hadamard σε όλα τα qubits.
4. Ερώτημα στο oracle.
5. Εφαρμογή της πύλης Hadamard σε κάθε qubits.
6. Μέτρηση κάθε qubit, δηλαδή του κβαντικού κυκλώματος.

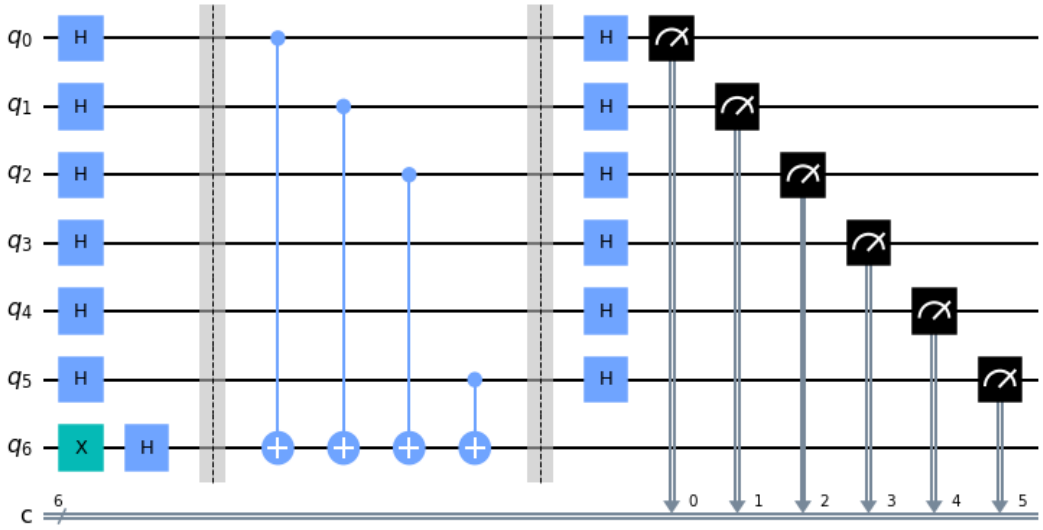
Χρησιμοποιούμε το Qiskit για να προγραμματίσουμε τον αλγόριθμο των Bernstein-Vazirani. Παρακάτω φαίνεται το πρόγραμμα που υλοποιεί τον αλγόριθμο:

```
#Bernstein-Vazirani algorithm
from qiskit import QuantumCircuit, execute, Aer, IBMQ
%matplotlib inline
from qiskit.tools.visualization import plot_histogram

secret = input('Secret number (bit nequence) =');
n = len(secret) #number of digits
#initiate the register
circuit = QuantumCircuit(n+1, n)
#Apply the Hadamard gate
circuit.h(range(n))
circuit.draw(output='mpl')

circuit.x(len(secret))
circuit.h(len(secret))
circuit.barrier()
circuit.draw(output='mpl')
#Oracle
for i, digit in enumerate(reversed(secret)):
    if digit == '1':
        circuit.cx(i, n)
circuit.barrier()
circuit.draw(output='mpl')
circuit.h(range(n))
circuit.draw(output='mpl')
circuit.measure(range(n),range(n))
circuit.draw(output='mpl')
simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend = simulator, shots=1) .result()
counts = result.get_counts()
print(counts)
```

```
Secret number (bit nequence) = 100111
```

Αν πληκτρολογήσουμε σαν κρυφή συμβολοσειρά (bit sequence) = 100111, το αποτέλεσμα που μας εμφανίζει είναι: {‘100111’ : 1}.

Αν εκτελέσουμε το παραπάνω πρόγραμμα μέχρι και την γραμμή 27 και για s = 100111, εμφανίζεται το κβαντικό κύκλωμα, το οποίο είναι:



Σχήμα: 6 – Κβαντικό κύκλωμα με υλοποίηση αλγορίθμου Bernstein – Vazirani, για s = 100111

6.6 Κβαντικά Παιχνίδια

Μια αρκετά δελεαστική εφαρμογή αυτής της σπουδαίας Τεχνολογίας, της Κβαντικής Υπολογιστικής, είναι τα κβαντικά παιχνίδια.

Ο καθηγητής πρώτης βαθμίδας του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Θεσσαλίας, κ. Σάββας Ηλίας, στο εξαιρετικό, καινοτόμο και πολύ ιδιαίτερο βιβλίο του, με τίτλο «Κβαντική Υπολογιστική» (από τη θεωρία στην πράξη), αναπτύσσει με πολύ κατανοητό και ενδιαφέρον τρόπο αρκετά κβαντικά παιχνίδια, από τα οποία εμείς θα δούμε τα τρία.

6.6.1 Παιχνίδι 1ο

Αυτό το κβαντικό παιχνίδι, το οποίο δεν είναι και τόσο τίμιο συμμετέχουν δύο παίκτες, ο Α και ο Β. Έχουμε ένα κβαντικό καταχωρητή με αρχικό qubit σε αρχική κατάσταση $|0\rangle$. Αρχικά ο παίκτης Α ξεκινά το παιχνίδι με $|0\rangle$.

• Πρώτη κίνηση:

Ο παίκτης Β, έχει τις επιλογές να δράσει με μία κβαντική πύλη Hadamard (H) ή με μια κβαντική πύλη X ή με καμία πύλη.

• Δεύτερη κίνηση:

Ο παίκτης Α, μπορεί να δράσει μόνο με μια κβαντική πύλη X ή και τίποτα.

• Τρίτη κίνηση:

Ο παίκτης Β, έχει τις επιλογές να δράσει επίσης με μια κβαντική πύλη Hadamard (H) ή X ή και καμία.

Πότε έχουμε νικητή στο παιχνίδι;

• Εάν το αποτέλεσμα είναι $|1\rangle$, τότε κερδίζει ο παίκτης Α, διαφορετικά κερδίζει ο παίκτης Β.

➢ Σκοπός του παιχνιδιού:

Και οι δύο παίκτες θα πρέπει να επιλέγουν κάθε φορά την κατάλληλη κβαντική πύλη για να είναι ο νικητής του παιχνιδιού. Όμως, το παιχνίδι όπως αναφέραμε αρχικά δεν είναι και τόσο έντιμο, εάν ο παίκτης Β γνωρίζει την επόμενη κίνησή του ανεξάρτητα από τις επιλογές του παίκτη Α.

Ας δούμε αναλυτικά, υποθέτοντας ότι ο παίκτης Β δρα με μια κβαντική πύλη Hadamard (H).

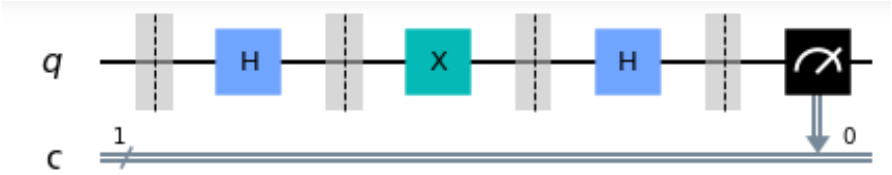
Έχουμε:

• Πρώτη κίνηση:

Ο παίκτης Β δρα με μια κβαντική πύλη Hadamard (H) πάντα (όπως έχουμε δεδομένο εξ αρχής).

• Δεύτερη κίνηση:

Ο παίκτης Α, δρα με μία κβαντική πύλη X.



Σχήμα: 7 – 1η Περίπτωση ο παίκτης Α δρα με πύλη X

Πραγματοποιούμε τους υπολογισμούς για να βρούμε το αποτέλεσμα, εφαρμόζοντας διαδοχικά τις κβαντικές πύλες όπως φαίνονται στο παραπάνω σχήμα και είναι:

• $x_0 = \mathrm{H}\,|\,0\,\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}.$

• $x_1 = \mathrm{H}\,|\,x_0\,\rangle = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\cdot\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}.$

Και στη συνέχεια:

• $x_2 = \mathrm{H}\,|\,x_1\,\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\cdot\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}2\\0\end{bmatrix} = \frac{1}{2}\cdot 2\cdot\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} = |\,0\,\rangle.$

• Τρίτη κίνηση:

Ο παίκτης Α, δεν δρα σε καμία κβαντική.

Σχήμα: 8 – 2η Περίπτωση ο παίκτης Α δεν δρα με πύλη

Έτσι, σύμφωνα με το παραπάνω σχήμα, έχουμε ως αποτέλεσμα:

• $|\,q_0\,\rangle = \mathrm{H}\,|\,0\,\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\cdot\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}.$

- $|q_1\rangle = \mathrm{H}|q_0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\cdot\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1}{2}\cdot 2\cdot\begin{bmatrix}1\\ 0\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |0\rangle.$

Συνεπώς, ο παίκτης Β κερδίζει πάντα εφαρμόζοντας απλά μια πύλη Hadamard, οποιαδήποτε κίνηση και να επιλέξει να κάνει τελικά ο παίκτης Α.

Παρακάτω φαίνεται το πρόγραμμα εκτέλεσης αυτού του κβαντικού παιχνιδιού.

Είναι:

```
from qiskit import *
circuit = QuantumCircuit(qr,cr)
circuit.barrier()
print('Ο παίκτης Β: επιλογή 0 για πύλη Hadamard, 1 για πύλη X, και 2 για καμία πύλη')
epologi = -1
while epilogi<0 or epilogi>2:
    epilogi = int(input('Παίκτη Β, επιλέγεις:'))
if epilogi == 0:
    circuit.h(0)
else:
    if epilogi == 1:
        circuit.x(0)
circuit.barrier()
print('Ο Παίκτης Α: επιλογή 0 για πύλη X, και 2 για καμία πύλη ')
epologi = -1
while epilogi<0 or epilogi>1:
    epilogi = int(input('Παίκτη Α, επιλέγεις:'))
if epilogi == 1:
    circuit.x(0)
circuit.barrier()
print('Ο παίκτης Β: επιλογή 0 για πύλη Hadamard, 1 για πύλη X, και 2 για καμία πύλη')
epologi = -1
while epilogi<0 or epilogi>2:
    epilogi = int(input('Παίκτη Β, επιλέγεις:'))
if epilogi == 0:
    circuit.h(0)
else:
    if epilogi == 1:
        circuit.x(0)
circuit.barrier()
circuit.measure(0,0)
circuit.draw(output='mpl')
aer_sim = Aer.get_backend('aer_simulator')
qobj=assemble(circuit)
result = aer_sim.run(qobj).result()
counts = result.get_counts()
a = int(list(counts.keys())[0])
if a == 0:
    print('αποτέλεσμα = ', a, ' Κερδίζει ο Β')
else:
    print('αποτέλεσμα = ', a, ' Κερδίζει ο Α')
```

6.6.2 Παιχνίδι 2ο – Κβαντική Ναυμαχία

Η ναυμαχία είναι ένα κλασικό και διαχρονικό παιχνίδι που συνδυάζει στρατηγική και ενθουσιασμό. Η κβαντική ναυμαχία αποτελεί ένα από τα πρώτα κβαντικά παιχνίδια με πολλές εκδοχές του παιχνιδιού, αλλά διαφοροποιημένο και πιο απλό από την κλασική ναυμαχία. Παρουσιάστηκε πρώτη φορά τη δεκαετία του 1930 και αργότερα το 1967 σαν επιτραπέζιο παιχνίδι από τον Milton Bradley.

Εμείς, θα δούμε μια εύκολη εκδοχή της ναυμαχίας.

Αρχικά, το παιχνίδι παίζεται σε έναν πίνακα 5 x 5 στον οποίο ο παίκτης Α τοποθετεί ένα μόνο πλοίο, χωρίς να γνωρίζει ο παίκτης Β την θέση αυτή. Ο παίκτης Β μπορεί να τοποθετήσει τρεις βόμβες.

Παρακάτω φαίνεται ο πίνακας της κβαντική ναυμαχίας με πλοίο και βόμβες.

B	≈	≈	≈	≈
≈	A	≈	≈	≈
≈	≈	≈	B	≈
≈	≈	≈	≈	≈
≈	≈	≈	≈	B

Σχήμα: 9 – Πίνακας της κβαντικής ναυμαχίας με πλοίο και βόμβες

- Σκοπός του παιχνιδιού:

Ο σκοπός της κβαντικής ναυμαχίας σε αντίθεση με την εκδοχή της κλασικής ναυμαχίας, δεν είναι να πετύχει ο παίκτης Β ακριβώς τη θέση του πλοίου που έχει τοποθετήσει ο αντίπαλός του ο παίκτης Α, αλλά θα πρέπει να σημαδεύσει εύστοχα τις βόμβες σε σχέση με την απόσταση της από το πλοίο του παίκτη Α, προκαλώντας ζημιές στο πλοίο του.

Έτσι, ο στόχος του παιχνιδιού με λίγα λόγια, είναι να ξεπεράσει το όριο των ζημιών που θα προκαλέσει ένας παίκτης στον αντίπαλό του. Εμείς θεωρούμε ότι αν ξεπεράσει το 50% σε ζημιές το πλοίο, τότε αυτό καταστρέφεται και τελικά βυθίζεται. Κερδίζει ο παίκτης Β αν βυθίσει το πλοίο του παίκτη Α, διαφορετικά κερδίζει ο παίκτης Α.

Στην κβαντική ναυμαχία, η απόσταση ενός πλοίου σχετικά με την απόσταση μίας βόμβας που θα τοποθετηθεί κοντά στο πλοίο παίζει κυρίαρχο ρόλο, αφού είναι

εφικτό να προκαλέσει ζημιές (μικρές ή μεγάλες) στο πλοίο. Ακόμα, εάν μια βόμβα τοποθετηθεί ακριβώς στο τετράγωνο του πλοίου, δεν είναι εφικτό να βυθίσει το πλοίο αυτό και μόνο.

Παρακάτω φαίνεται ο πίνακας κβαντικής ναυμαχίας, αριθμώντας τα τετράγωνα ως εξής:

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

Σχήμα: 10 – Πίνακας της κβαντικής ναυμαχίας με αρίθμηση

Η κβαντική υπολογιστική συγκεκριμένα, μας δίνει τη δυνατότητα, ανάλογα με την απόσταση του πλοίου και της βόμβας, να περιστρέψουμε το πλοίο (qubit) γύρο από τον άξονα Z κατά φ μοίρες.

Ας θεωρήσουμε τυχαία ότι το πλοίο βρίσκεται στη θέση με συντεταγμένες (2,2) και η βόμβα βρίσκεται στη θέση με συντεταγμένες (1,4). Υποθέτουμε σαν απόσταση, τη μεγαλύτερη, απόλυτη τιμή (θετικός αριθμός) της αφαίρεσης των δύο συντεταγμένων των θέσεων του πλοίου και τη βόμβας.

- Έτσι, η απόσταση του πλοίου από τη βόμβα, είναι:

Απόσταση (α) = $\max\{|2-1|, |2-4|\} = 2$.

- Η περιστροφή που θα πραγματοποιηθεί, είναι:

$$\varphi = \frac{\pi}{3 \cdot \alpha + 1} = \frac{\pi}{7} \text{ ακτίνια.}$$

Σημείωση

Η περιστροφή μεταβάλλεται ανάλογα με το μέγεθος του τετραγώνου της ναυμαχίας ή λαμβάνοντας υπόψιν κάποιους άλλους παράγοντες, όπως η γωνία φ περιστροφής που θεωρούμε κάθε φορά εμείς.

Φτάνοντας στο τέλος του παιχνιδιού και μετρώντας το κύκλωμα, βρίσκουμε το ποσοστό των 1 που έχουμε ως αποτέλεσμα. Αν το ποσοστό αυτό είναι μεγαλύτερο από 50 % (που θεωρήσαμε στην αρχή του παιχνιδιού), τότε το πλοίο βυθίζεται. Διαφορετικά, το πλοίο απειλήθηκε με τραυματισμούς να βυθιστεί, αλλά τελικά τα κατάφερε και δε βυθίστηκε.

Παρακάτω φαίνεται το πρόγραμμα της εμφάνισης του κυκλώματος της κβαντικής ναυμαχίας (για την συγκεκριμένη εκδοχή του):

```
from qiskit import *
from math import pi
import numpy as np
n = 1
qr = QuantumRegister(n)
cr = ClassicalRegister(n)
qc = QuantumCircuit(qr,cr)
print('Παίκτης Χ, τοποθέτηση πλοίου:')
Xx = int(input() )
Xy = int(input() )
print('Παίκτης Υ, τοποθέτηση 3 βομβών:')
b = np.zeros((3,2))
a = np.zeros(3)
for i in range(3):
    print('τοποθέτηση ', i, 'βόμβας:')
    b[i][0] = int(input() )
    b[i][1]= int(input() )
for i in range(3):
    a[i] = max(abs(b[i][0]-Xx), abs(b[i][1]-Xy))
    qc.ry(pi/3*a[i]+1, qr[0])
qc.measure(qr,cr)
qc.draw(output='mpl')

```

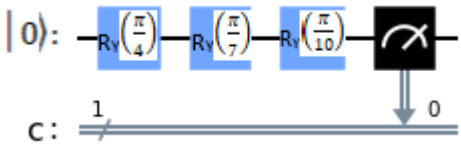
```
Παίκτης Χ, τοποθέτηση πλοίου:
```

```


from qiskit.visualization import plot_histogram
aer_sim = Aer.get_backend('aer_simulator')
qobj = assemble(qc)
result = aer_sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)

pososto_zimias = result.get_counts()['1']/1024
if pososto_zimias >= 0.75:
    print('Το πλοίο βυθίστηκε. . . ')
else:
    print('Το πλοίο βομβαρδίστηκε αλλά άντεξε!!!')
print('Ποσοστό ζημιών =', pososto_zimias*100,'%')
```

Να δούμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1, 2 και 3: Ενσωμάτωση των κατάλληλων βιβλιοθηκών.
- Γραμμές 4, 5, 6 και 7: Δημιουργία κβαντικού κυκλώματος αποτελούμενο από ένα qubit.
- Γραμμές 8, 10, 11, 12, 13, 14, 15, 16 και 17: Εισαγωγή συντεταγμένων του πλοίου και των βομβών (χωρίς να ελέγχεται η ορθότητα των απαντήσεων).
- Γραμμή 18: Για κάθε βόμβα.
- Γραμμή 19: Υπολογισμός της απόστασης της βόμβας από το πλοίο (τη μεγαλύτερη απόσταση σχετικά με τις δύο συντεταγμένες).
- Γραμμή 20: Περιστροφή του πλοίου-qubits γύρο από τον άξονα Z.
- Γραμμές 21 και 22: Μέτρηση και εμφάνιση του κβαντικού κυκλώματος.
- Γραμμές 23, 24, 25, 26, 27 και 28: Υπολογισμός μέτρηση και εμφάνιση του ραβδογράμματος του αποτελέσματος.
- Γραμμή 29: Υπολογισμός της ζημιάς που προκλήθηκε στο πλοίο (1024 θεωρούμε τον αριθμό επαναλήψεων εκτέλεσης του κυκλώματος).
- Γραμμές 30, 31, 32, 33, 34 και 35: Εμφάνιση του αποτελέσματος με πληροφορίες του πίνακα από το σχήμα: 12 και το αποτέλεσμα θα είναι:
- Το βομβαρδίστηκε, αλλά άντεξε!!! Ποσοστό ζημιών = 49.8046875 %.

Όταν «τρέξουμε» το παραπάνω πρόγραμμα, το κβαντικό κύκλωμα θα είναι:



Σχήμα: 11 – Κβαντικό κύκλωμα προγράμματος – κβαντικής ναυμαχίας

Για να μας εμφανιστεί το ραβδόγραμμα του αποτελέσματος, συνεχίσουμε το παραπάνω πρόγραμμα ως εξής:

```
23 from qiskit.visualization import plot_histogram
24 aer_sim = Aer.get_backend('aer_simulator')
25 qobj=assemble(circuit)
26 result = aer_sim.run(qobj).result()
27 counts = result.get_counts()
28 plot_histogram(counts)
```

Όταν «τρέξουμε» το πρόγραμμα μέχρι τη γραμμή 28, θα εμφανιστείς:

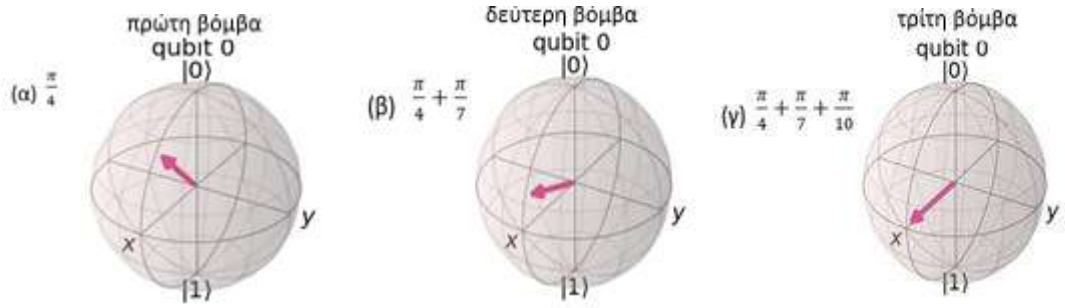
Για να υπολογίσουμε την κλίση που θα εκτελέσει το πλοίο –qubit ύστερα από κάθε βόμβα εκτελούμε το παρακάτω πρόγραμμα:

```
from qiskit.quantum_info import Statevector
from qiskit.visualization import plot_bloch_multivector
#%matplotlib inline
qc = QuantumCircuit(1)
qc.ry(pi/4,0)
state = Statevector.from_instruction(qc)
plot_bloch_multivector(state, title='πρώτη βόμβα')
```

Παρακάτω φαίνεται η εκτέλεση του προγράμματος, η οποία εμφανίζει την απεικόνιση στη σφαίρα Bloch, με τη βαθμιαία περιστροφή του διανύσματος θέσης, για παράδειγμα η γωνία περιστροφής με κλίση που θα έχει το πλοίο κατά τη διάρκεια του βομβαρδισμού να είναι $\frac{\pi}{4}$.

Σημείωση

Επαναλαμβάνουμε προσθέτοντας την αντίστοιχη γωνία περιστροφής με κλίση που θα έχει το πλοίο κατά τη διάρκεια του βομβαρδισμού κάθε φορά, $\frac{\pi}{4}+\frac{\pi}{7}$ και $\frac{\pi}{4}+\frac{\pi}{7}+\frac{\pi}{10}$.



Σχήμα: 12 – Σφαίρα Bloch κβαντικής ναυμαχίας

Αυτή η εκδοχή της κβαντικής ναυμαχίας είναι αρκετά απλή και θα μπορούσε να τροποποιηθεί σε μια πιο δύσκολη εκδοχή, όπως:

- ✓ Να έχουν τη δυνατότητα να συμμετάσχουν δύο ή περισσότερους παίκτες ως αντίπαλοι.
- ✓ Να χρησιμοποιήσουν μεγαλύτερο πίνακα ναυμαχίας 10 x 10 κλπ.
- ✓ Να υπάρχουν περισσότερα πλοία καθώς και αριθμό βομβών.
- ✓ Να τροποποιηθεί το μέγεθος της ζημιάς που προκαλεί μία βόμβα.
- ✓ Να σχεδιαστεί ένα γραφικό περιβάλλον για το παιχνίδι.

Στο διαδίκτυο αρκετοί άνθρωποι έχουν κάνει προσωπική χρήση διάφορες μορφές, σκέψεις, πλάνα και εφαρμογές σχετικών πτυχών του παιχνιδιού της κβαντικής ναυμαχίας όπου μπορεί κανείς να δελεαστεί και να συνθέσει.

6.6.3 Παιχνίδι 3ο – Κβαντική Τρίλιζα

Το τρίτο και πιο περίπλοκο κβαντικό παιχνίδι, είναι η κβαντική τρίλιζα που σίγουρα διαφέρει από την παραδοσιακή τρίλιζα.

- Σκοπός του παιχνιδιού:

Ο σχηματισμός τριών τετραγώνων (τριάδες), οριζόντια, κάθετα και διαγώνια, στα προσχεδιασμένα τετράγωνα, με το συμβολισμό του ίδιου παίκτη, Χ ή Ο, χωρίς ο αντίπαλος παίκτης να καταφέρει να το εμποδίσει αυτό.

Παρακάτω, φαίνεται στο πινακάκι ένα παιχνίδι της τρίλιζας και συγκεκριμένα νικητής είναι ο παίκτης Χ.

X	O	X
O	X	O
O	X	X

Σχήμα: 13 – Τρίλιζα

Αριθμούμε το πινακάκι της κβαντικής τρίλιζας ως εξής:

0	1	2
3	4	5
6	7	8

Σχήμα: 14 – Κβαντική τρίλιζα – αρίθμηση τετραγώνων

Στην κβαντική τρίλιζα, ο κάθε παίκτης μπορεί να τοποθετήσει στο πινακάκι της τρίλιζας Χ ή Ο ανάλογα από το τι έχει τοποθετήσει ο αντίπαλός του παίκτης. Όμως αντίθετα με την κλασική τρίλιζα, ένας παίκτης μπορεί ακόμη να διαλέγει δύο ταυτόχρονα κουτάκια από το πινακάκι της κβαντικής τρίλιζας και να τα τοποθετεί σε κβαντική διεμπλοκή $\left(\left|\Phi^{+}\right\rangle\right)$.

Είναι:

$$\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}\left(\left|00\right\rangle + \left|11\right\rangle\right).$$

Στη συνέχεια, όταν ο πίνακας της τρίλιζας συμπληρωθεί, εκείνη τη στιγμή τα τετράγωνα που είναι σε κβαντική κατάσταση θα καταστραφούν και σε σχέση με το αποτέλεσμα τοποθετείται ξανά ένα Ο ή Χ , ελέγχοντας το παιχνίδι και έτσι τελικά εμφανίζεται ο νικητής του παιχνιδιού. Σε αυτό το σημείο απαιτείται προσοχή για τις κινήσεις των παικτών. Να επισημάνουμε, ότι όταν δύο τετραγωνάκια βρίσκονται σε διεμπλοκή το ίδιο σύμβολο Ο ή Χ, αφού δεν είναι δυνατό να γνωρίζει κάποιος παίκτης από πριν το αποτέλεσμα και σε αυτή την περίπτωση θα εμφανιστεί Χ και Χ ή Ο κα Ο.

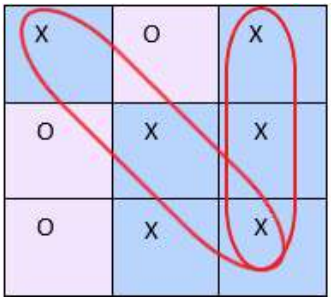
Θεωρούμε ότι ύστερα από ένα παιχνίδι ο πίνακας της κβαντικής τρίλιζας θα είναι ως εξής:

ΔΧ	ΔΟ	Χ
ΔΟ	ΔΟ	ΔΟ
Ο	Χ	ΔΧ

Σχήμα: 15 – Πίνακας κβαντικής τρίλιζας ύστερα από ένα παιχνίδι

Συγκεκριμένα, ο παίκτης Χ έθεσε σε διεμπλοκή τα τετράγωνα 0 και 8 ενώ ο παίκτης Ο έθεσε τα τετράγωνα 1, 4, 3 και 5.

Ύστερα από την κατάρρευση (ο πίνακας γεμίζει και τα τετράγωνα που είναι σε διεμπλοκή καταρρέουν) και τη μέτρηση του κυκλώματος, ο πίνακας τροποποιείται όπως φαίνεται παρακάτω.



Σχήμα: 16 – Πίνακας τρίλιζας ύστερα από την υλοποίηση του κβαντικού κυκλώματος

Επομένως ο νικητής του συγκεκριμένου παιχνιδιού είναι ο παίκτης X και με διπλή νίκη (κύρια διαγώνιο και τρίτη στήλη κάθετα, όπως φαίνεται στο παραπάνω σχήμα).

Θεωρούμε ότι για τον παίκτη X ισχύει $|0\rangle$ ενώ για τον παίκτη O έχουμε $|1\rangle$ και το πρόγραμμα εμφάνισης, για το συγκεκριμένο παιχνίδι της κβαντικής τρίλιζας (τοποθεσίας συμβόλων O ή X των παικτών), φαίνεται παρακάτω:

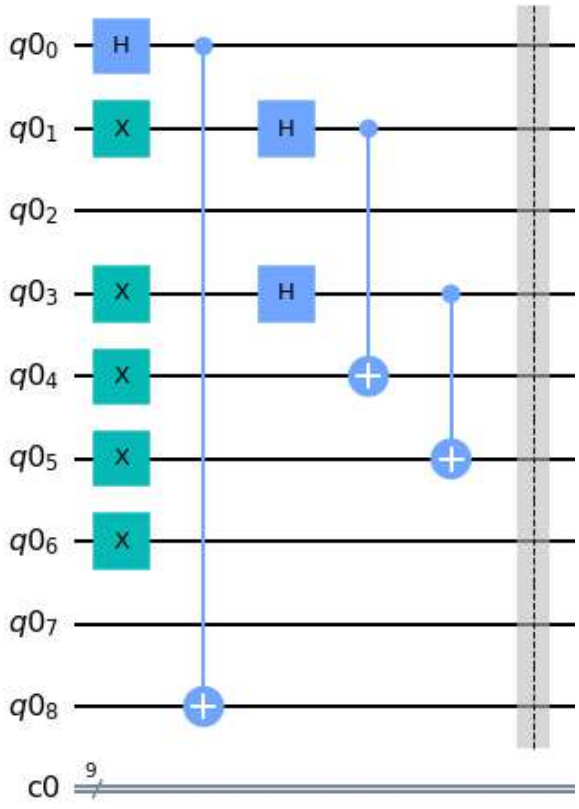
```
1  paiktis_X = [0, 8, 7, 2]
2  paiktis_O = [1, 4, 3, 5, 6]
3  diemploki = [0, 8, 1, 4, 3, 5]
4  from qiskit import *
5  from qiskit.visualization import plot_histogram
6  qr = QuantumRegister(9)
7  cr = ClassicalRegister(9)
8  triliza = QuantumCircuit(qr,cr)
9  for i in range(len(paiktis_O)):
10     triliza.x(paiktis_O[i])
11 #triliza.barrier()
12 for i in range(0, len(diemploki), 2):
13     triliza.h(diemploki[i])
14     triliza.cx(diemploki[i], diemploki[i+1])
15 triliza.barrier()
16 triliza.measure(qr,cr)
17 triliza.draw(output='mpl')
18 job = execute(triliza, BasicAer.get_backend('qasm_simulator'), shots = 1)
19 result = job.result()
20 a = list(result.get_counts().keys() )
21 temp = list(a[0])
22 telikos_pinakas = temp[::-1]
```

Να δούμε αναλυτικά το παραπάνω πρόγραμμα (ανά γραμμή):

- Γραμμές 1 και 2: Δήλωση στον πίνακα της τρίλιζας σχετικά με την προτίμηση των παικτών και αποθήκευσή τους στις αντίστοιχες λίστες.
- Γραμμή 3: Επιπλέον δηλώσεις των παικτών όταν θέτουν σε διεμπλοκή δύο τετράγωνα αποθηκεύονται σε διαφορετική λίστα.
- Γραμμές 4 και 5: Εισαγωγή των κατάλληλων βιβλιοθηκών.

- Γραμμές 6, 7 και 8: Δημιουργία κβαντικού κυκλώματος με 9 qubits και 9 κλασικά bits (ο πίνακας της τρίλιζας αποτελείται από 3 x 3 = 9 τετράγωνα).
- Γραμμές 9 και 10: Εφαρμογή πύλης X στα τετράγωνα του παίκτη O για να μετατραπεί σε |1⟩.
- Γραμμές 12,13 και 14: Εφαρμογή κβαντικής διεμπλοκής.
- Γραμμή 20: Αποθήκευση των αποτελεσμάτων.
- Γραμμή 21: Επιλογή του πρώτο αποτελέσματος που θα εμφανίσει το πρόγραμμα (το πρόγραμμα δίνει όλα τα πιθανά αποτελέσματα).
- Γραμμή 22: Αντιστροφή του αποτελέσματος (τα αποτελέσματα εμφανίζονται με αντίστροφη σειρά και με αυτό τον τρόπο τα εμφανίζουμε στη σωστή σειρά).

Όταν «τρέξουμε» το παραπάνω πρόγραμμα, το κβαντικό κύκλωμα του συγκεκριμένου παιχνιδιού θα είναι:



Σχήμα: 17 – Κβαντικό κύκλωμα προγράμματος – κβαντικής ναυμαχίας

Το συγκεκριμένο παιχνίδι της κβαντικής τρίλιζας αποτελεί μια απλοποιημένη εκδοχή από τις διάφορες μορφές που είναι δυνατό να υλοποιηθούν.

Στο διαδίκτυο έχουν συλληφθεί και «ανεβάσει» πάρα πολλοί άνθρωποι, διάφορες μορφές, σκέψεις, σχέδια καθώς και εκτελέσεις πολλών εκδοχών του παιχνιδιού της κβαντικής τρίλιζας όπου μπορεί κάποιος να εμπνευστεί και να τις αντλήσει.

ΑΣΚΗΣΕΙΣ ΓΙΑ ΛΥΣΗ

1) Να εξηγήσετε γιατί χρησιμοποιούμε τον αλγόριθμο των Bernstein - Vazirani σε ένα κβαντικό παιχνίδι.

2) Εφαρμόστε μια παραλλαγή στο παράδειγμα του πρώτου κβαντικού παιχνιδιού αυτού του κεφαλαίου. Συγκεκριμένα, εάν ο παίκτης Β επιδράσει με μια πύλη Χ στην πρώτη κίνηση, παρατηρώντας την ενδιάμεση κίνηση του παίκτη Α, ποια θα πρέπει να είναι η τελευταία του κίνηση για να κερδίσει.

3) Να γράψετε πρόγραμμα που να προσομοιώνει το παρακάτω παιχνίδι. Έχουμε το παιχνίδι πέτρα - ψαλίδι - χαρτί. Στο παιχνίδι συμμετάσχουν ο παίκτης Α και ένας κβαντικός υπολογιστής. Οι τρεις διαφορετικές κβαντικές καταστάσεις υποθέτουμε ότι είναι πέτρα=00, ψαλίδι=01 και χαρτί=10. Δεδομένο είναι ότι η πέτρα κερδίζει το ψαλίδι, το ψαλίδι κερδίζει το χαρτί και το χαρτί κερδίζει την πέτρα. Ο παίκτης επιλέγει μία από τις τρεις αυτές κβαντικές καταστάσεις και δημιουργούμε έναν κβαντικό καταχωρητή των 2 qubits θέτοντάς τα σε κατάσταση υπέρθεσης χρησιμοποιώντας την πύλη Hadamard. Στην συνέχεια, υπολογίζουμε το αποτέλεσμα του κυκλώματος. Όταν εκτελέσουμε το πρόγραμμά μας, αν το αποτέλεσμά είναι μια μη αποδεκτή κατάσταση, δηλαδή είναι 11, τότε σε αυτή την περίπτωση «τρέχουμε» πάλι το πρόγραμμά μας ώστε να μας πάρουμε μια αποδεκτή κατάσταση (00 ή 01 ή 10). Τέλος, συγκρίνουμε το αποτέλεσμα με την αρχική επιλογή του παίκτη. Το πρόγραμμα επαναλαμβάνεται πολλές φορές και όποιος από τους παίκτες συγκεντρώσει πρώτος 10 νίκες θα είναι και ο νικητής του παιχνιδιού.

4) Να γράψετε πρόγραμμα που να προσομοιώνει ένα δικό σας κβαντικό παιχνίδι στρατηγικής χρησιμοποιώντας τον αλγόριθμο των Bernstein - Vazirani.

Συμπεράσματα

Οι κβαντικοί υπολογιστές και γενικότερα οι κβαντικές τεχνολογίες αποτελούν ένα καινοτόμο και συναρπαστικό επιστημονικό πεδίο έρευνας, το οποίο υπόσχονται σπουδαίες προοπτικές διερεύνησης.

Η επιστήμη των κβαντικών συστημάτων τα τελευταία χρόνια έχει εξελιχθεί ραγδαία βελτιώνοντας την ικανότητα ελέγχου των κβαντικών φαινομένων που συντελούν τη βάση του κβαντικού υπολογισμού.

Έτσι, τα πολύτιμα αυτά εργαλεία πραγματοποιούν συγχρόνως πολλούς υπολογισμούς ελαττώνοντας σημαντικά τον χρόνο επεξεργασίας της πληροφορίας, παρέχοντας τη δυνατότητα να επιταχύνουν τις υπολογιστικές εργασίες σε δευτερόλεπτα.

Λόγω αυτού του τεχνολογικού άλματος η κβαντική μηχανική οδήγησε σε επανάσταση στην επιστήμη της Φυσικής, της ιατρικής, της ασφάλειας πληροφοριών, της τεχνητής νοημοσύνης και όχι μόνο, με αποτέλεσμα να υπάρχει πλέον στις μέρες μας μεγάλη διαθεσιμότητα αυτών των εργαλείων που μέχρι πριν από κάποια χρόνια δεν μπορούσε κανείς να το φανταστεί.

Ωστόσο, η κβαντική πληροφορική βρίσκεται σε πρώιμο στάδιο και δεν είναι ξεκάθαρο σε πόσο χρονικό διάστημα αυτές οι μηχανές θα είναι εφικτό να «βγουν» από τα ερευνητικά εργαστήρια και να εφαρμοστούν εύκολα στην καθημερινότητα των ανθρώπων καθώς απαιτείται σημαντική δουλειά ακόμη πριν την κατασκευή ενός κβαντικού υπολογιστή ώστε να προσφέρει πρακτική χρησιμότητα.

Η κβαντική υπολογιστική αξιοποιεί το φαινόμενο της κβαντικής διεμπλοκής που θεωρείται ως η μοναδική πηγή της κβαντικής μηχανικής και είναι ζωτικής σημασίας, αφού τη χρησιμοποιούμε στους κβαντικούς υπολογιστές για να πραγματοποιήσουμε κβαντικούς υπολογισμούς καθώς και να αναπτύξουμε αλγόριθμους. Υποστηρίζει σημαντικές εφαρμογές που σχετίζονται με τα συστήματα της κβαντικής κρυπτογραφίας, της οποίας η εφαρμογή έχει πολλές πιθανότητες να πραγματοποιηθεί μελλοντικά.

Με βάση τους κβαντικούς αλγόριθμους, οι οποίοι παρόλο που δεν μπορούν να «τρέξουν» σε κλασικούς υπολογιστές μπορούν να προσομοιωθούν καθώς υπάρχει η δυνατότητα εφαρμογής διάφορων κβαντικών εφαρμογών, όπως τα κβαντικά παιχνίδια τα οποία στο πλαίσιο του παιχνιδιού, τηρούν κβαντικούς κανόνες.

Γενικά, οι κβαντικοί υπολογιστές με τον ερχομό τους αποτελούν μια τρομερή τεχνολογική επανάσταση και είναι ιδανικοί για να προσομοιώνουν κβαντικά φαινόμενα. Παρόλο όμως που μπορούν να επιλύσουν τεράστια προβλήματα πολυπλοκότητας με καταπληκτική ταχύτητα τα οποία είναι πρακτικά αδύνατον να λυθούν από τους σημερινούς "κλασικούς υπολογιστές", δεν είναι πιο γρήγοροι σε όλα συγκριτικά με τους κλασικούς υπολογιστές και αυτό οφείλεται στη διαχείριση των qubits. Έτσι, οι κβαντικοί υπολογιστές δεν είναι δυνατό να αντικατασταθούν από

τους κλασικούς υπολογιστές αφού για κάποιες συγκεκριμένες εργασίες οι κλασικοί υπολογιστές είναι καταλληλότεροι, αλλά να δουλεύουν παράλληλα με αυτούς, για την καλύτερη και αποτελεσματικότερη τεχνολογική και επιστημονική εξέλιξη.

Έτσι λοιπόν, η κβαντική υπολογιστική προϋποθέτει την κατάλληλη εκπαίδευση ανάπτυξης δεξιοτήτων καθώς και έργα οικοδόμηση της κατανόησης για το πώς η κβαντική υπολογιστική μπορεί να βοηθήσει ραγδαία στην επίλυση πολλών τεχνολογικών, επιστημονικών και επιχειρηματικών προβλημάτων.

Όπως είναι κατανοητό, η ένταξή της θεωρείται ουσιαστική στη Δευτεροβάθμια Εκπαίδευση, η οποία θα οδηγήσει στην εξέλιξη της δομής των εκπαιδευτικών αναλυτικών προγραμμάτων και θα αποτελέσει το κατάλληλο υπόβαθρο προς το καλύτερο.

Συνεπώς, ο κβαντικός υπολογισμός και οι κβαντικές τεχνολογίες υπόσχονται να διευρύνουν τα όρια της επιστημονικής γνώσης της ανθρωπότητας και είναι σχεδόν βέβαιο ότι θα οδηγήσουν σε νέες ενδιαφέρουσες και συναρπαστικές επιστημονικές ανακαλύψεις, συνδέοντας τη σύγχρονη επιστημονική έρευνα με την εκπαίδευση ενημερώνοντας τους μαθητές, αυριανούς πολίτες και ίσως επιστήμονες, για τις εξελίξεις του τεχνολογικού πολιτισμού.

Ολοκληρώνοντας, είναι αναμφίβολο ότι η κβαντική υπολογιστική και ευρύτερα η κβαντική τεχνολογία, θα συμβάλει καθοριστικά στη δημιουργία ενός νέου «μαγικού κόσμου» με υψηλές προοπτικές εξέλιξης που υπόσχεται να αλλάξει το μέλλον συντριπτικά.

Βιβλιογραφία

[1] Σάββας Κ. Ηλίας, & Σαμπάνη Ε. Μαρία, (2022). Κβαντική Υπολογιστική: από τη θεωρία στην Πράξη. Λάρισα: Εκδόσεις Τζιόλα.

[2] Καραφυλλίδης, Ιωάννης (2015). Κβαντική υπολογιστική. Αθήνα, Αττικής, Ελλάδα: Εκδόσεις Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα.

[3] Καραφυλλίδης, Ιωάννης (2005). Κβαντικοί υπολογιστές: Βασικές Έννοιες, Αθήνα: Εκδόσεις Κλειδάριθμος.

[4] Andrew Davies and Patrick Kennedy (2017). Special Report: Quantum Computing: From Little Things, Εκδότης: Australian Strategic Policy Institute. Ανακτήθηκε από:http://www.jstor.com/stable/resrep16820.6

[5] Wikipedia, "Information", url:https://en.wikipedia.org/wiki/Information

[6] Wikipedia, "Πληροφορία", url:https://el.wikipedia.org/wiki/%CE%A0%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%AF%CE%B1

[7] Wikipedia, "Κβαντική Μηχανική", url:https://el.wikipedia.org/wiki/%CE%9A%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%AE_%CE%BC%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE

[8] Wikipedia, "Quantum Mechanics", url:https://en.wikipedia.org/wiki/Quantum_mechanics

[9] Wikipedia, "Quantum Computer", url:https://wikipredia.net/el/Quantum_computer , https://bahasa.wiki/el/Quantum_computer

[10] Wikipedia, "Quantum Computing", url: https://en.wikipedia.org/wiki/Quantum_computing

[11] Wikipedia, "Qubit", url:https://en.wikipedia.org/wiki/Qubit

[12] Wikipedia, "Quantum Algorithm", url:https://bahasa.wiki/el/Quantum_algorithm

[13] Wikipedia, "Moore' s Law", url: https://en.wikipedia.org/wiki/Moore%27s_law

[14] Wikipedia, "Κβαντική Διεμπλοκή", url:https://el.wikipedia.org/wiki/%CE%9A%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%AE_%CE%B4%CE%B9%CE%B5%CE%BC%CF%80%CE%BB%CE%BF%CE%BA%CE%AE

[15] Wikipedia, "Κβαντική Υπέρθεση", url:https://el.wikipedia.org/wiki/%CE%9A%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%AE_%CF%85%CF%80%CE%AD%CF%81%CE%B8%CE%B5%CF%83%CE%B7

[16] Wikipedia, " Quantum Supremacy",
url: https://en.wikipedia.org/wiki/Quantum_supremacy

[17] Wikipedia, "Quantum Teleportation",
url: https://en.wikipedia.org/wiki/Quantum_teleportation

[18] Επιστήμη: Τεχνολογία: Κβαντική υπεροχή-Τι είναι και τι σημαίνει το επίτευγμα της Google για την Πληροφορική [apeiro.gr.]. Ανακτήθηκε από:https://www.apeiro.gr/epistimi/tehnologia/kbantiki-yperohi-ti-einai-kai-ti-simainei-epiteygma-tis-google-gia-tin

[19] Θεωρίες Φυσικής: Η Αξία της πληροφορίας (2016, Ιούνιος 22) [physics4u.]. Ανακτήθηκε από:http://physics4u.gr/blog/2016/06/22/%CE%B7-%CE%B1%CE%BE%CE%AF%CE%B1-%CF%84%CE%B7%CF%82-%CF%80%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%AF%CE%B1%CF%82/

[20] Φυσικοί και Φυσική από το Διαδίκτυο: Κβαντική Θεωρία: Πώς λειτουργεί ένας κβαντικός υπολογιστής (2021, Αύγουστος 18) [physicsgg]. Ανακτήθηκε από:https://physicsgg.me/2021/08/18/%CF%80%CF%8E%CF%82-%CE%BB%CE%B5%CE%B9%CF%84%CE%BF%CF%85%CF%81%CE%B3%CE%B5%CE%AF-%CE%AD%CE%BD%CE%B1%CF%82-%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CF%8C%CF%82-%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3/

[21] Φυσικοί και Φυσική από το Διαδίκτυο: Κβαντική Θεωρία: Οι μελλοντικές εφαρμογές των κβαντικών υπολογιστών είναι απρόβλεπτες (2022, Μάρτιος 26) [physicsgg]. Ανακτήθηκε από:https://physicsgg.me/2022/03/26/%CE%BF%CE%B9-%CE%BC%CE%B5%CE%BB%CE%BB%CE%BF%CE%BD%CF%84%CE%B9%CE%BA%CE%AD%CF%82-%CE%B5%CF%86%CE%B1%CF%81%CE%BC%CE%BF%CE%B3%CE%AD%CF%82-%CF%84%CF%89%CE%BD-%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA/

[22] Luke Lango. (2022, Νοέμβριος 8). InvestorPlace. Οι κβαντικοί υπολογιστές θα είναι μεγαλύτεροι από την ανακάλυψη της φωτιάς. Ανακτήθηκε από:https://investorplace.com/hypergrowthinvesting/2022/08/quantum-computing-could-solve-the-worlds-energy-crisis/?fbclid=IwAR2FGVQS1V-DgdnWEtvFo6FWdb_kPlFr7ZksA9rAc8DzUg6_Z0lawT8oBZY

[23] Chistos Elpidis. (2021, Νοέμβριος 15). Techgear. Ο ισχυρότερος κβαντικός υπολογιστής στον κόσμο (127 qubits). Ανακτήθηκε από:https://www.techgear.gr/ibm-eagle-o-ischyroteros-kvantikos-epexergastis-ston-kosmo-127-qubits-31837

[24] Ο πρώτος μεγάλος κβαντικός επεξεργαστής φτιαγμένος μόνο από φως λέιζερ (2019, Οκτώβριος 21) [Skai.gr.]. Ανάκτηση από:https://www.skai.gr/news/technology/o-protos-megalos-kvantikos-epeksergastis-ftiagmenos-mono-apo-fos-leizer

[25] Η Google κατασκεύασε κβαντικό υπολογιστή (2019, Οκτώβριος 23) [Skai.gr.]. Ανακτήθηκε από:https://www.skai.gr/news/technology/i-google-kataskeyase-kvantiko-ypologisti

[26] Ο κβαντικός υπολογιστής της Google που δεν μοιάζει με κανέναν άλλο στον κόσμο (2019, Σεπτέμβριος 24) [Zougla.gr.]. Ανακτήθηκε από:https://www.zougla.gr/technology/article/sycamore-o-kvantikos-ipologistis-tis-gogle-pou-den-miazi-me-kanenan-alon-ston-kosmo

[27] Κβαντική πληροφορική - Είναι η απάντηση στην κλασσική πληροφορική (2020, Ιούνιος 18) [kavosnews.gr]. Ανακτήθηκε από:https://kavosnews.gr/%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CE%AE-%CF%80%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CE%BA%CE%AE-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%B7-%CE%B1%CF%80%CE%AC%CE%BD%CF%84%CE%B7%CF%83/

[28] Μανουσέλης Σπύρος. (2019, Οκτώβριος 5). Το μυστήριο του κβαντικού υπολογιστή. Η εφημερίδα των Συντακτών - Ανεξάρτητη Συνεταιριστική εφημερίδα - efsyn. Ανακτήθηκε από:https://www.efsyn.gr/epistimi/mihanes-toy-noy/213482_mystirio-toy-kbantikoy-ypologisti

[29] Μανουσέλης Σπύρος. (2022, Φεβρουάριος 26). Από τους ψηφιακούς στους κβαντικούς υπολογιστές. Η εφημερίδα των Συντακτών - Ανεξάρτητη Συνεταιριστική εφημερίδα - efsyn. Ανακτήθηκε από:https://www.efsyn.gr/epistimi/mihanes-toy-noy/333817_apo-toys-psifiakoys-stoys-kbantikoys-ypologistes

[30] Καρουζάκης Γιώργος. (2022, Μάιος 26). Κβαντική Θεωρία: Ο χάρτης των κβαντικών υπολογιστών. Physicsgg: Φυσικοί και Φυσική από το Διαδίκτυο. Ανακτήθηκε από:https://physicsgg.me/2022/05/26/%CE%BF-%CF%87%CE%AC%CF%81%CF%84%CE%B7%CF%82-%CF%84%CF%89%CE%BD-%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CF%8E%CE%BD-%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CF%8E%CE%BD/

[31] Τι είναι ο κβαντικός υπολογιστής και πώς θα αλλάξει την καθημερινότητά μας (2018, Φεβρουάριος 17) [newsbeast.gr]. Ανακτήθηκε από:https://www.newsbeast.gr/weekend/arthro/3306993/ti-ine-o-kvantikos-ipologistis-ke-pos-tha-allaxi-tin-kathimerinotita-mas

[32] Τι είναι και πώς λειτουργεί ο κβαντικός υπολογιστής (2020, Ιούλιος 14) [zimzamphysics.gr]. Ανακτήθηκε από:http://www.zimzamphysics.gr/2020/07/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%BA%CE%B1%CE%B9-%CF%80%CF%8E%CF%82-%CE%BB%CE%B5%CE%B9%CF%84%CE%BF%CF%85%CF%81%CE%B3%CE%B5%CE%AF-%CE%BF-%CE%BA%CE%B2%CE%B1%CE%BD%CF%84%CE%B9%CE%BA%CF%8C/

[33] Τεχνητή νοημοσύνη – κβαντικοί υπολογιστές: Δύο «δαίμονες» που δεν πρέπει να συνεργαστούν ανεξέλεγκτα (2022, Αύγουστος 25). Η Ναυτεμπορική. Ανακτήθηκε από:https://m.naftemporiki.gr/story/1897698?fbclid=IwAR3O6eEHhJkr5HyX5ivp0kaOY5ojeKQIiRP_NUMNu1tMeTJaOfp8610UcBk

[34] IBM anuncia sus ultimos avances en computacion cuantica en el CES 2020 (2020, Ιανουάριος 8) [Esemanal.mx.]. Ανακτήθηκε

από:https://esemanal.mx/2020/01/ibm-anuncia-sus-ultimos-avances-en-computacion-cuantica-en-el-ces-2020/

[35] Παλιούρης Γιάννης. (2019, Ιανουάριος 9). Ο πρώτος κβαντικός υπολογιστής για εμπορική χρήση είναι γεγονός. [Liberal.gr.] Ανακτήθηκε από:https://www.liberal.gr/news/o-protos-kbantikos-upologistis-gia-emporiki-chrisi-einai-gegonos/235710

[36] Κβαντική διεμπλοκή: Νέο επίτευγμα αλλάζει τον τρόπο λειτουργίας του Internet (2020, Φεβρουάριος) [Hellas-now.com.]. Ανακτήθηκε από:https://hellas-now.com/kvantiki-diemploki-neo-epiteygma-allazei-ton-tropo-leitoyrgias/?fbclid=IwAR0G7SRrZi5GXm8LjfSkOdecLeoSuNGVQwKYS738SFbwY4V3UwC1ksTVU9A

[37] Sparkes Matthew. (2022, Αύγουστος 18). Google' s quantum supremacy challenged by ordinary computers, for now. Newscientist.com. Ανακτήθηκε από:https://www.newscientist.com/article/2333837-googles-quantum-supremacy-challenged-by-ordinary-computers-for-now/?fbclid=IwAR1lEtYXQpZvh9Cy6IUSPIUpNoCywFQVCZBGFLFiF6zWDjx0UM96UXszIDc

[38] Τσινάβος Γιάννης. (2022, Απρίλιος 21). «Τι δουλειά έχει» η Hyundai με την κβαντική μηχανική [4troxoi.gr.]. Ανακτήθηκε από:https://www.4troxoi.gr/epikairotita/kosmos/hyundai-ionq/?fbclid=IwAR28k02fgRPkBvmPT9tORKGXS6N0azrob9sk07FE8cuJqs0N5yzyHkjqX6w

[39] Aws.amazon.com. Ανακτήθηκε από:https://aws.amazon.com/braket/

[40] Το IonQ αξίζει το άλμα; (2021, Μάιος 5) [BitcoinEthereumNews.com]. Ανακτήθηκε από:https://el.bitcoinethereumnews.com/economy/is-ionq-worth-the-leap/?fbclid=IwAR02gs8K2SMLZObMPvaYRblpWgKcLNgWqSoF6eGwnZXAxiga_e1NR4Vlco0

[40] Neto Renato. (2021, Νοέμβριος 15). Επεξήγηση κβαντικών υπολογιστών για αρχάριους. HashDork.com. Ανακτήθηκε από:https://hashdork.com/el/quantum-computing/?fbclid=IwAR1Cgd_YdrZ5T2soKYv4JI8yVbaiFVOaTWTv-3N6E--Bciu6xO3Xnz2FBOo

[41] Wikipedia, "No-Cloning Theorem", url:https://en.wikipedia.org/wiki/No-cloning_theorem

[42] Άλλο ένα βήμα για το κβαντικό Διαδίκτυο (2022, Μάιος 25). Η Ναυτεμπορική. Ανακτήθηκε από:https://www.naftemporiki.gr/story/1866297/allo-ena-bima-gia-to-kbantiko-diadiktuo

[43] Δρ. Νικολαΐδης Χρήστου (2003). Σημειώσεις Μαθήματος Γραμμικής Άλγεβρας. Πανεπιστήμιο Θεσσαλίας: Τμήμα Πολιτικών Μηχανικών. Ανακτήθηκε από: http://users.uoa.gr/~atzanis/MathGeo/%CD%E9%EA%EF%EB%E1%FA%E4%E7%F2_%C3%F1%E1%EC%EC%E9%EA%DE_%C1%EB%E3%E5%E2%F1%E1/linear1.pdf

[44] Φωτόδεντρο - Διαδραστικά Σχολικά Βιβλία: ebooks.edu.gr. Μαθηματικά Γ ' Λυκείου: Θετικών Σπουδών & Σπουδών Υγείας και Σπουδών Οικονομίας & Πληροφορικής): Βιβλίο μαθητή:1 Πίνακες – Γραμμικά Συστήματα. Ανακτήθηκε

από:http://ebooks.edu.gr/ebooks/v/html/8547/2732/Mathimatika-G-Lykeiou-ThSp_html-apli/indexA1_1.html

[45] Υπουργείο Παιδείας και Θρησκευμάτων, Ινστιτούτο Εκπαιδευτικής Πολιτικής (2020). Σημειώσεις Στοιχεία Πιθανοτήτων και Στατιστικής: Γ' Λυκείου: Ομάδας Προσανατολισμού Ανθρωπιστικών Σπουδών. Ινστιτούτο Τεχνολογίας Υπολογιστών και Εκδόσεων ΔΙΟΦΑΝΤΟΣ. Ανακτήθηκε από: http://ebooks.edu.gr/ebooks/v/pdf/8547/5311/22-0269-01_Stoicheia-Pithanotiton-kai-Statistikis_G-Lykeiou-AnthrSp_Vivlio-Mathiti/

[46] Τσαγκαράκη Μαργαρίτα (2022). Στοιχεία Πιθανοτήτων και Στατιστικής: Γ ' Γενικού Λυκείου: Ομάδας Προσανατολισμού Ανθρωπιστικών Σπουδών. Ηράκλειο Κρήτης: Εκδόσεις taexeiola. Ανακτήθηκε από: https://www.taexeiola.gr/mathimatika-kai-stoixeia-statistikhs-g-lykeiou-lysari/

[47] Σγαρμπάς, Κ. (2015, Μάρτιος 18). Ανακτήθηκε από:https://eclass.upatras.gr/modules/document/file.php/EE742/Open%20Courses/

[48] Wikipedia, "Kronecker", url:https://en.wikipedia.org/wiki/Kronecker_product

[49] Wikipedia, "Complex Number", url: https://en.wikipedia.org/wiki/Complex_number

[50] Wikipedia, "Python.org", url:https://www.python.org

[51] Wikipedia, "Python", url:https://el.wikipedia.org/wiki/Python ,

[52] Wikipedia, "History of Python", url: https://en.wikipedia.org/wiki/History_of_Python , https://www.geeksforgeeks.org/history-of-python/

[53] Wikipedia, "Python Tutorial W3schols", url: https://www.w3schools.com/python/

[54] Wikipedia, "Project Jupyter", url: https://en.wikipedia.org/wiki/Project_Jupyter

[55] Wikipedia, "Jupyter Notebook", url:https://jupyter.org/

[56] Wikipedia, "Qiskit", url:https://en.wikipedia.org/wiki/Qiskit

[57] Wikipedia, "Qiskit Learn", url: https://qiskit.org/textbook/what-is-quantum.html

[58] Wikipedia, "Qiskit Tutorial", url: https://qiskit.org/documentation/intro_tutorial1.html

[59] Wikipedia, "Programming", url:https://en.wikipedia.org/wiki/Computer_programming

[60] Wikipedia, "Quantum Cryptography",
url:https://en.wikipedia.org/wiki/Quantum_cryptography