ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ PYTHON»

ΚΩΣΤΟΠΟΥΛΟΥ ΣΤΕΡΓΙΑΝΗ

Επιβλέπων: Καθηγητής Γεώργιος Σταμούλης

Ιούνιος 2018

Στον μπαμπά μου

Υπεβλήθη για την εκπλήρωση μέρους των απαιτήσεων για την απόκτηση του Προπτυχιακού Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω από καρδιάς τον καθηγητή μου κ. Σταμούλη Γεώργιο για την πολύτιμη βοήθειά του στη διεκπεραίωση της πτυχιακής εργασίας, αλλά και την βοήθειά του καθ'ολη την διάρκεια των σπουδών μου και ειδικά τον τελευταίο χρόνο, πρακτικά και ψυχολογικά. Έπειτα, τον καθηγητή μου κ. Ευμορφόπουλο Νέστωρ και για την δική του πολύτιμη βοήθεια.

Το σύντροφο μου, που συνέβαλε με όποιο τρόπο μπορούσε, είτε πρακτικό τρόπο είτε ψυχολογικό, για την ολοκλήρωση της πτυχιακής εργασίας.

Τους φίλους μου που ήταν δίπλα μου όλα αυτά τα χρόνια και με στήριζαν.

Τέλος, το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένειά μου, που με στήριζε οικονομικά και ψυχολογικά και παρότι οι συνθήκες δεν ήταν οι ιδανικές, δεν σταμάτησαν λεπτό να πιστεύουν σε μένα.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια έχει γίνει μία έκρηξη στα δεδομένα που έχουν δημιουργηθεί και στο πόσο αυτά επηρεάζουν την αγορά εργασίας. Κάθε μέρα όλο και περισσότερες εταιρίες χρησιμοποιούν την ανάλυση δεδομένων για την επίτευξη των στόχων τους με αποτέλεσμα αυτή η επιστήμη να έχει γίνει αναγκαία, και τα εργαλεία που μπορούν να χρησιμοποιηθούν σε αυτή, απολύτως βασικά.

Στη παρούσα πτυχιακή εργασία αναλύεται η γλώσσα προγραμματισμού Python και η χρήση της στο τομέα της ανάλυσης των δεδομένων. Αρχικά, γίνεται μία αναφορά στις βασικές λειτουργίες της γλώσσας, στα χαρακτηριστικά της και στη δομή της. Παραθέτονται ορισμοί και παραδείγματα χρήσιμα για την κατανόησή της.

Στη συνέχεια, αναλύονται δύο βασικές της βιβλιοθήκες οι οποίες χρησιμοποιούνται στην ανάλυση των δεδομένων, η NymPy και η Pandas. Με τη βοήθεια του Jupiter Notebook, γίνεται ανάλυση των χαρακτηριστικών τους και των μεθόδων τους και παρουσιάζονται κάποια παραδείγματα τα οποία αποσκοπούν στην κατανόηση των βιβλιοθηκών και στην ανάδειξη της ευκολίας στη χρήση της. Επίσης γίνεται και μία μικρή αναφορά στη Matplotlib, που μαζί με τις άλλες δύο συνθέτουν τις βασικές γνώσεις που χρειάζεται κάποιος που θέλει να κάνει ανάλυση δεδομένων με την Python.

Επιπροσθέτως, γίνεται και μία σύγκριση με την γλώσσα προγραμματισμού R μιας και είναι ο άλλος γνωστός τρόπος για να γίνει ανάλυση και παρουσιάζονται τα συμπεράσματα της συνολικής εργασίας.

Σκοπός της παρούσας πτυχιακής εργασίας είναι να δοθούν κάποιες κατευθυντήριες γραμμές για την Python και τον τρόπο χρήσης γενικά, αλλά και ειδικά στην ανάλυση των δεδομένων. Να κατανοηθούν οι βασικές αρχές της λειτουργίας του NumPy και Pandas, όπως και η γενική δομή της Python. Τέλος, να δειχθεί ότι είναι ισάξια ή και καλύτερη από άλλους τρόπους που χρησιμοποιούνται για την διαχείριση και την ανάλυση των δεδομένων σήμερα.

ABSTRACT

Over the past few years there has been an explosion of created data and how they affect the market field. Every day, more and more companies use data analysis to achieve their goals, making this science and the tools that can be used in it, absolutely nessecery.

In this thesis the Python programming language is analysed and its use in the field of data analysis. At first, a reference is made to the basic functions of the language, its characteristics and its structure. There are some definitions and examples useful for understanding it.

Then two basic libraries are analyzed, which are used in the data analysis, NymPy and Pandas. With the help of Jupiter Notebook, an analysis of their features and methods is presented, and some examples are presented which aim to understand libraries and to show the ease of use. There is also a small reference to Matplotlib, which along with the other two synthesizes the basic knowledge that is required in order someone to do data analysis with Python.

In addition, a comparison with the programming language R is made since it is the other known way to analyze and then the conclusions of the overall thesis are presented.

The purpose of this thesis is to provide some guidelines for Python and how to use it in general and especially in data analysis. To understand the NumPy and Pandas libraries, as also the basic structure of Python. Finally, it is concluded that Python is equal or better than other ways used to manage and analyze data today.

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας πινάκων

ΚΕΦΑΛΑΙΟ 1:
ΕΙΣΑΓΩΓΗ

1.1 Ιστορική αναδρομή

Η γλώσσα προγραμματισμού Python δημιουργήθηκε το 1990 από τον Γκουίντο βαν Ρόσσουμ (Guido van Rossum) στο Centrum Wiskunde & Informatica (CWI) ως μία γλώσσα σεναρίων, με κύριο στόχο την ευκολία χρήσης της, ενώ διακρίνεται για τις πολλές βιβλιοθήκες της που την καθιστούν γρήγορη στην εκμάθησή της. Το όνομα είναι ιδιοκτησία του Monty Python, το οποίο ο δημιουργός επέλεξε για να δείξει ότι η Python πρέπει να είναι διασκεδαστική στη χρήση.

1.2 Python και ανάλυση δεδομένων

Η Python είναι μια γλώσσα από τις πλέον διαδεδομένες γλώσσες προγραμματισμού σήμερα, που γιορτάζεται για την αποδοτικότητα και την αναγνωσιμότητα του κώδικα. Ως γλώσσα προγραμματισμού για την επιστήμη των δεδομένων, η Python αντιπροσωπεύει έναν συμβιβασμό μεταξύ της R, η οποία επικεντρώνεται σε μεγάλο βαθμό στην ανάλυση και την απεικόνιση δεδομένων, και της Java, η οποία αποτελεί τη ραχοκοκαλιά πολλών εφαρμογών μεγάλης κλίμακας. Υποστηρίζει αντικειμενοστραφή προγραμματισμό, δομημένο προγραμματισμό και λειτουργικά πρότυπα προγραμματισμού, μεταξύ άλλων. Υπάρχει ένα αστείο στην κοινότητα της Python ότι "η Python είναι γενικά η δεύτερη καλύτερη γλώσσα για όλα". Η Python μπορεί να χειριστεί κάθε εργασία από την εξόρυξη δεδομένων έως την κατασκευή ιστότοπων για τη λειτουργία ενσωματωμένων συστημάτων, όλα σε μια ενοποιημένη γλώσσα.

Στο ForecastWatch, για παράδειγμα, η Python χρησιμοποιήθηκε για να γράψει έναν αναλυτή για να συλλέξει τις προβλέψεις από άλλους ιστότοπους, έναν μηχανισμό συσσωμάτωσης για την κατάρτιση των δεδομένων και τον κώδικα ιστότοπου για την εμφάνιση των αποτελεσμάτων. Η PHP αρχικά χρησιμοποιήθηκε για να χτίσει τον ιστότοπο μέχρι που η εταιρεία συνειδητοποίησε ότι ήταν ευκολότερο να ασχοληθεί μόνο με μια ενιαία γλώσσα.

Το Facebook, σύμφωνα με άρθρο του περιοδικού Fast Company για το 2014, επέλεξε να χρησιμοποιήσει την Python για ανάλυση δεδομένων, επειδή ήδη χρησιμοποιήθηκε τόσο ευρέως σε άλλα μέρη της εταιρείας. Για το λόγο αυτό και για άλλους, η Python είναι πολύ αγαπητή από τους προγραμματιστές.

Η αναγνωσιμότητα και η απλότητα της Python την καθιστούν σχετικά εύκολη και ο αριθμός των αποκλειστικών αναλυτικών βιβλιοθηκών που διατίθενται σήμερα σημαίνει ότι οι επιστήμονες δεδομένων σχεδόν σε κάθε τομέα θα βρουν πακέτα ήδη προσαρμοσμένα στις ανάγκες τους διαθέσιμα για λήψη.

Λόγω της επεκτασιμότητας της Python και της φύσης γενικού σκοπού της, καθώς η δημοτικότητά του εξερράγη, κάποιος θα την χρησιμοποιούσε τελικά για την ανάλυση δεδομένων. Η Python είναι συχνά η επιλογή για προγραμματιστές που πρέπει να εφαρμόσουν

στατιστικές τεχνικές ή ανάλυση δεδομένων στο έργο τους ή για επιστήμονες δεδομένων των οποίων τα καθήκοντα πρέπει να ενσωματωθούν με εφαρμογές ιστού ή περιβάλλοντα παραγωγής. Ο συνδυασμός βιβλιοθηκών μηχανικής μάθησης και ευελιξίας καθιστά την Python μοναδικά κατάλληλη για την ανάπτυξη εξελιγμένων μοντέλων και μηχανισμών πρόβλεψης που συνδέονται άμεσα με τα συστήματα παραγωγής.

1.3 Βιβλιοθήκες της Python

Όπως συμβαίνει με πολλές άλλες γλώσσες προγραμματισμού, οι διαθέσιμες βιβλιοθήκες οδηγούν στην επιτυχία της Python: περίπου 72.000 από αυτές στο Python Package Index (PyPI) και αυξάνονται συνεχώς. Με την Python, η συνηθισμένη βιβλιοθήκη έχει δημιουργηθεί με εργαλεία για κάθε είδους προγραμματισμό και για την επίλυση προβλημάτων χωρίς να χρειαστεί να επιλεχθεί κάποια μεταξύ ανταγωνιστικών βιβλιοθηκών λειτουργίας.

Η Python έχει ένα δωρεάν λογισμικό ανοικτού κώδικα και συνεπώς μπορεί κανείς να γράψει ένα πακέτο βιβλιοθήκης για να επεκτείνει τη λειτουργικότητά του.

Στη συνέχεια παρουσιάζονται επιγραμματικά οι βασικές βιβλιοθήκες της:

1.3.1 Numpy

Είναι το θεμέλιο πάνω στο οποίο κατασκευάζονται όλα τα εργαλεία υψηλότερου επιπέδου για την επιστημονική Python. Εδώ είναι μερικές από τις λειτουργίες που παρέχει:

1. N- Dimensional array, ένα πολυδιάστατο πίνακα με γρήγορη και αποδοτική μνήμη που παρέχει αριθμητικές πράξεις.
2. Μπορούν να εφαρμοστούν τυποποιημένες μαθηματικές πράξεις σε πίνακες ολόκληρων δεδομένων χωρίς να γραφτούν βρόχοι.
3. Είναι πολύ εύκολο να μεταφερθούν δεδομένα σε εξωτερικές βιβλιοθήκες γραμμένες σε γλώσσα χαμηλού επιπέδου και επίσης για εξωτερικές βιβλιοθήκες να επιστρέφουν δεδομένα σε Python ως Numpy arrays.

Η NumPy δεν παρέχει λειτουργικότητα ανάλυσης δεδομένων υψηλού επιπέδου. Έχοντας κατανόηση των συστοιχιών NumPy και των υπολογιστικών προσανατολισμένων με συστοιχίες, θα βοηθήσει να χρησιμοποιηθούν πολύ πιο αποτελεσματικά εργαλεία όπως η Pandas.

1.3.2 Pandas

Η Pandas είναι η βιβλιοθήκη ανάλυσης δεδομένων της Python, που χρησιμοποιείται για όλα, από την εισαγωγή δεδομένων από υπολογιστικά φύλλα του Excel σε σύνολα επεξεργασίας για ανάλυση χρονοσειρών. Η Pandas τοποθετεί σχεδόν όλα τα κοινά εργαλεία

για τα δεδομένα στα χέρια των αναλυτών. Αυτό σημαίνει ότι η βασική εκκαθάριση και κάποια προηγμένη χειραγώγηση μπορούν να εκτελεστούν με ισχυρά πλαίσια δεδομένων της Pandas. Η Pandas είναι χτισμένη πάνω στη NumPy, μία από τις πρώτες βιβλιοθήκες πίσω από την ιστορία επιτυχίας της επιστήμης δεδομένων της Python. Οι λειτουργίες NumPy εκτίθενται σε Pandas για προηγμένη αριθμητική ανάλυση. Περιέχει δομές δεδομένων υψηλού επιπέδου και εργαλεία που έχουν σχεδιαστεί για να κάνουν την ανάλυση δεδομένων γρήγορη και εύκολη. Χρησιμοποιώντας την Pandas, είναι ευκολότερο να χειριστούν τα δεδομένα τα οποία λείπουν. Συγχωνεύει άλλες σχεσιακές λειτουργίες που βρίσκονται σε δημοφιλείς βάσεις δεδομένων (για παράδειγμα SQLbased). Η Pandas είναι το καλύτερο εργαλείο για να γίνει στα δεδομένα munging.

1.3.3 Scipy

Η βιβλιοθήκη SciPy εξαρτάται από τη NumPy, το οποίο παρέχει εύκολο και γρήγορο χειρισμό πίνακα N-διαστάσεων(N- Dimensional array). Η βιβλιοθήκη SciPy είναι κατασκευασμένη για να λειτουργεί με τους αριθμητικούς πίνακες NumPy και παρέχει πολλές φιλικές προς το χρήστη και αποτελεσματικές αριθμητικές ρουτίνες, όπως ρουτίνες για αριθμητική ενσωμάτωση και βελτιστοποίηση. Η SciPy διαθέτει ενότητες βελτιστοποίησης, γραμμικής άλγεβρας, ολοκλήρωσης και άλλων κοινών καθηκόντων στην επιστήμη των δεδομένων.

1.3.4 Scikit-learn

Η Scikit-learn είναι μια βιβλιοθήκη Python για μηχανική μάθηση(Machine Learning) που βασίζεται στη NumPy και τη SciPy. Η Scikit-learn βοηθά στην ταχεία εφαρμογή γνωστών αλγορίθμων στο σύνολο δεδομένων. Η Scikit-learn περιλαμβάνει εργαλεία για πολλές τυπικές εργασίες εκμάθησης μηχανών και εξόρυξης δεδομένων (όπως ομαδοποίηση, ταξινόμηση, παλινδρόμηση κ.λπ.).

1.3.5 IPython

Η IPython επεκτείνει τη λειτουργικότητα του διαδραστικού διερμηνέα της Python με ένα αλληλεπιδραστικό κελί που προσθέτει ενδοσκόπηση, εμπλουτισμένα μέσα, σύνταξη κελιού, ολοκλήρωση καρτελών και ανάκτηση ιστορικού εντολών. Λειτουργεί επίσης ως ενσωματωμένος διερμηνέας για τα προγράμματα, ο οποίος μπορεί να είναι πραγματικά χρήσιμος για τον εντοπισμό σφαλμάτων.

1.3.6 Matplotlib

Η Matlplotlib είναι μια βιβλιοθήκη Python για απεικόνιση(visualization). Η Matplotlib επιτρέπει να γίνονται εύκολα γραφήματα γραμμών, διάγραμμα πίτας, ιστόγραμμα και άλλα επαγγελματικά στοιχεία. Όταν χρησιμοποιείται μέσα στην IPython, η Matplotlib έχει διαδραστικά χαρακτηριστικά όπως το ζουμ και το panning. Υποστηρίζει διαφορετικά GUI back ends σε όλα τα λειτουργικά συστήματα και μπορεί επίσης να εξάγει γραφικά σε κοινές μορφές διανύσματος και γραφικών: PDF, SVG, JPG, PNG, BMP, GIF κλπ.

Τέλος, υπάρχουν και άλλες γνωστές βιβλιοθήκες της Python όπως η Theano και η TensorFlow για μηχανική μάθηση, η Scrapy, η NLTK και η Pattern για εξόρυξη δεδομένων και επεξεργασία της γλώσσας και η Seaborn, η Bokeh, η Basemap και η NetworkX για κατασκευή διαγράμματος και απεικόνιση.

ΚΕΦΑΛΑΙΟ 2:
ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ PYTHON

2.1 Εγκατάσταση της Python

Για την ανάλυση δεδομένων με την χρήση Python, θα χρησιμοποιηθεί το Anaconda που είναι μια δωρεάν και ανοιχτού κώδικα κατανομή της Python και της R, για εφαρμογές σχετικές με την επιστήμη δεδομένων και την μηχανική μάθηση (επεξεργασία δεδομένων ευρείας κλίμακας, προγνωστική ανάλυση, επιστημονική πληροφορική). Το Anaconda τοποθετεί σχεδόν όλα τα εργαλεία που θα χρειαστεί σε ένα μικρό πακέτο: τη γλώσσα πυρήνα Python, ένα βελτιωμένο REPL περιβάλλον που ονομάζεται Jupyter, αριθμητικές υπολογιστικές βιβλιοθήκες (NumPy, Pandas), σχεδιαστικές βιβλιοθήκες (Seaborn, Matplotlib), στατιστικές και μηχανικής μάθησης βιβλιοθήκες (SciPy, Scikit-learn, Statsmodels).

Συγκεκριμένα, επιλέχθηκε η έκδοση 3.6 (γενικά 3.x) αλλά και οι δύο εκδόσεις μπορούν να χρησιμοποιηθούν γιατί οι διαφορές είναι μικρές.

Η γλώσσα Python δεν έρχεται προεγκατεστημένη με όλα τα χαρακτηριστικά που μπορεί να χρειαστούν, ακόμη και όταν εγκατασταθεί χρησιμοποιώντας το Anaconda. Όταν χρειάζεται ιδιαίτερη λειτουργικότητα, μπορούν να αναζητηθούν πακέτα Python. Για να εγκατασταθούν τα πακέτα χρησιμοποιείται το **conda** (package manager).

Για παράδειγμα, για την εγκατάσταση του Jupyter Notebook, το οποίο είναι ένα εργαλείο για τη διεξαγωγή διαδραστικών εργασιών επιστήμης δεδομένων στον περιηγητή, χρησιμοποιείται η ακόλουθη εντολή:

```
conda update jupyter
```

Άλλα παραδείγματα είναι τα εξής:

```
conda install seaborn
conda install scipy
```

Αν και το conda είναι ο διαχειριστής πακέτων για Anaconda, μπορεί και το pip να χρησιμοποιηθεί εναλλακτικά. Ορισμένα πακέτα δεν θα είναι διαθέσιμα από το conda ή το Anaconda.org. Όταν συμβεί αυτό, μπορεί να χρησιμοποιηθεί το pip για να εγκαταστήσει τα πακέτα.

```
pip install -U --user seaborn
```

2.2 Βασικά στοιχεία

2.2.1 Αναγνωριστικά

Αναγνωριστικό είναι το όνομα που προσδιορίζει είτε μία μεταβλητή, είτε μία συνάρτηση, είτε μία κλάση είτε ένα αντικείμενο (object). Μπορεί να περιέχει οποιαδήποτε γράμματα, αριθμούς, καθώς και το χαρακτήρα κάτω παύλας (_). Όμως, δεν μπορεί να ξεκινάει από αριθμό και δεν επιτρέπονται άλλοι ειδικοί χαρακτήρες (@, $ και λοιπά). Επίσης, ισχύει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων.

2.2.2 Λέξεις-Κλειδιά

Οι λέξεις-κελιδιά είναι δεσμευμένες λέξεις που δεν μπορούν χρησιμοποιηθούν ως μεταβλητή ή άλλο αναγνωριστικό όνομα. Οι λέξεις αυτές περιέχουν πεζά γράμματα και είναι οι εξής:

and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield.

2.2.3 Εσοχή

Στην Python δεν υπάρχει εντολή κλεισίματος, ένα μπλοκ εντολών υποδηλώνεται με τη χρήση εσοχών. Η χρήση εσοχών είναι υποχρεωτική, ώστε να ξεχωρίζει το κάθε μπλοκ. Ο χώρος της εσοχής είναι μεταβλητός, αλλά όλες οι εντολές κάθε μπλοκ πρέπει να περιέχουν το ίδιο περιθώριο εσοχής.

2.2.4 Εντολή help

Χρησιμοποιείται την εντολή help για τη λειτουργία και τη σύνταξη κάποιας εντολής. Σύνταξη: help(όνομα_εντολής).

2.2.5 Δηλώσεις πολλών γραμμών

Συνήθως, κάθε δήλωση τελειώνει καθώς μεταβαίνει σε νέα γραμμή. Ωστόσο, υπάρχει η δυνατότητα συνέχισης της εντολής σε παραπάνω από μία γραμμή, με τη χρήση του (\ ). Με τη χρήση του υποδηλώνεται ότι η γραμμή θα συνεχιστεί.

2.2.6 Σχόλια

Για την προσθήκη σχολίων (comments) στον κώδικα χρησιμοποιείται το σύμβολο της δίεσης. Όλοι οι χαρακτήρες μετά τη δίεση και μέχρι το τέλος της γραμμής θεωρούνται σχόλια και αγνοούνται. Για την εισαγωγή πολλών γραμμών με σχόλια, θα πρέπει στην αρχή κάθε γραμμής να προστίθεται το σύμβολο της δίεσης.

2.2.7 Εντολή print

Ο πιο απλός τρόπος για την εμφάνιση δεδομένων στην οθόνη είναι η χρήση της εντολής print(). Χρησιμοποιείται η εντολή για την εμφάνιση αποτελεσμάτων στην οθόνη. Η σύνταξή της είναι, print(), ανάμεσα στις παρενθέσεις και με τη χρήση εισαγωγικών μπορεί να εισαχθεί μία συμβολοσειρά ή μία μεταβλητή.

Ακολουθεί ένα παράδειγμα,

```
name="John"
print("His name is ", name)
 His name is John
```

2.2.8 Εντολή input

Για την εισαγωγή τιμής από το πληκτρολόγιο, κατά την εκτέλεση του κώδικα, χρησιμοποιείται η εντολή input. Πληκτρολογείται η εντολή input και μία συμβολοσειρα για εμφάνιση και στη συνέχεια, το πρόγραμμα εμφανίζει τη συμβολοσειρά και αναμένει για την εισαγωγή τιμής από το χρήστη. Αφού πληκτρολογηθεί η τιμή και πατηθεί το enter, το πρόγραμμα συνεχίζει στην εκτέλεση της επόμενης εντολής.

Για παράδειγμα,

```
name=input("What's your name?") #Η εντολή που εκτελούμε.
What's your name? Nick #Εμφάνιση συμβολοσειράς και εισαγωγή τιμής από το χρήστη.
print("Name is", name) #Εμφάνιση της καταχώρησης.
Name is Nick
```

2.2.9 Πολλαπλές δηλώσεις σε μία γραμμή

Μπορεί να γίνει παραπάνω από μία δήλωση σε μία γραμμή, με τη χρήση του ερωτηματικού (;). Επιτρέπονται οι πολλαπλές δηλώσεις στην ίδια γραμμή, αρκεί να μην αφορούν δηλώσεις νέου μπλοκ κώδικα.

2.2.10 Πολλαπλές ομάδες δηλώσεων

Υπάρχουν δηλώσεις οι οποίες δημιουργούν ένα ενιαίο μπλοκ κώδικα, δηλαδή οι σύνθετες καταστάσεις, όπως if, while, def και class. Οι δηλώσεις αυτές αρχίζουν, στην πρώτη γραμμή, με τη λέξη-κλειδί και στο τέλος της γραμμής υπάρχει το σύμβολο της άνω κάτω τελείας (:) και στη συνέχεια υπάρχουν δηλώσεις που συνθέτουν το μπλοκ κώδικα.

2.2.11 Μεταβλητές

Μεταβλητή είναι ένα αναγνωριστικό που περιέχει ένα αντικείμενο. Το όνομα μπορεί να περιέχει οποιουσδήποτε χαρακτήρες, αρκεί ο πρώτος χαρακτήρας να είναι γράμμα και το

όνομα να μην αποτελεί λέξη-κλειδί. Η αποθήκευση τιμής καταλαμβάνει θέσεις μνήμης, δηλαδή, με τον ορισμό μίας μεταβλητής χρησιμοποιεί κάποιο χώρο της μνήμης. Ο διερμηνέας κατανέμει τη μνήμη με βάση τον τύπο δεδομένων μίας μεταβλητής.

2.2.11.1 *Τύποι μεταβλητών*

Η κάθε γλώσσα υποστηρίζει συγκεκριμένους τύπους μεταβλητών. Με την ανάθεση διαφορετικών τύπων δεδομένων για τις μεταβλητές μπορούν να αποθηκευτούν ακέραιοι (integers), δεκαδικοί (decimals), ή χαρακτήρες (characters). Οι πιο συνηθισμένοι τύποι, στη γλώσσα Python, αναφέρονται στον ακόλουθο πίνακα

Τύποι μεταβλητών	**Χρήση**
Integer (int)	Ακέραιοι αριθμοί (θετικοί ή αρνητικοί αριθμοί, χωρίς υποδιαστολή)
Float	Πραγματικοί αριθμοί (αριθμοί κινητής υποδιαστολής)
String (str)	Συμβολοσειρές (ή αλφαριθμητικά)
Boolean	Εκφράσεις αληθείς ή ψευδείς

Πίνακας 1 Οι τύποι μεταβλητών και η χρήση τους

Σε άλλες εκδόσεις (εκδόσεις 2.x) της Python υπάρχει και ο τύπος μεταβλητής long, ο οποίος ισχύει για πολύ μεγάλους ακέραιους αριθμούς. Στις πρόσφατες εκδόσεις ακόμα και οι μεγάλοι αριθμοί έχουν ως τύπο μεταβλητής integer. Δεν είναι απαραίτητο να οριστεί το είδος της μεταβλητής, αρκεί μόνο να δοθεί η αξία. Η δήλωση για την κράτηση χώρου μνήμης γίνεται αυτόματα με την καταχώρηση τιμής σε μία μεταβλητή. Για παράδειγμα, a=2, η μεταβλητή a δηλώθηκε ως integer. Για να ελεχθεί ο τύπος της μεταβλητής, αρκεί να εκτελεστεί η εντολή type(όνομα_μεταβλητής). Για τη θέση μνήμης μιας μεταβλητής, χρησιμοποιείται η συνάρτηση id(όνομα_μεταβλητής). Τέλος, μπορούν να συγκριθούν οι θέσεις μνήμης δύο μεταβλητών χρησιμοποιώντας τους τελεστές "is" και "is not". Ο τελεστής "is" επιστρέφει True αν οι μεταβλητές δείχνουν στις ίδιες θέσεις μνήμης και False αν δεν δείχνουν. Ενώ, ο τελεστής "is not" επιστρέφει False αν οι μεταβλητές δείχνουν στις ίδιες θέσεις μνήμης και True αν δεν δείχνουν.

2.2.11.2 *Εύρος μεταβλητών*

Υπάρχουν δύο είδη μεταβλητών, οι τοπικές και οι καθολικές μεταβλητές. Μία καθολική (global) μεταβλητή είναι προσβάσιμη σε κάθε εύρος (scope), δηλαδή μπορεί να χρησιμοποιηθεί από οπουδήποτε μέσα στον κώδικα. Είναι η μεταβλητή που δηλώνεται στον κύριο κώδικα και όχι μέσα σε κάποια συνάρτηση. Ενώ μία τοπική (local) μεταβλητή είναι προσβάσιμη μόνο στο τοπικό εύρος. Μία μεταβλητή που ορίζεται μέσα σε μία συνάρτηση, είναι τοπική μεταβλητή. Δηλαδή, μπορεί να χρησιμοποιηθεί μόνο στη συνάρτηση στην οποία έχει δηλωθεί. Αν υπάρχει μία τοπική και μία καθολική μεταβλητή με το ίδιο όνομα, τότε η τοπική μεταβλητή επικρατεί έναντι της τοπικής. Επιπλέον, κάθε λειτουργία διαθέτει το δικό της τοπικό χώρο ονομάτων (namespace).

2.2.12 Μετατροπή τύπου δεδομένων

Κάποιες φορές μπορεί να χρειαστεί να εφαρμοστούν μετατροπές των υπαρχόντων τύπων δεδομένων. Η μετατροπή αυτή γίνεται χρησιμοποιώντας το όνομα του τύπου δεδομένων ως συνάρτηση. Οι συναρτήσεις αυτές επιστρέφουν μία νέα μεταβλητή που αντιπροσωπεύει την τιμή μετά τη μετατροπή αυτή.

Συνάρτηση	**Περιγραφή**
chr(x)	Μετατρέπει έναν ακέραιο αριθμό σε χαρακτήρα.
complex(real, imag)	Δημιουργεί έναν σύνθετο αριθμό.
dict(d)	Δημιουργεί ένα λεξικό. Το d πρέπει να είναι μία ακολουθία από κλειδί και αξία πλειάδας.
eval(str)	Αξιολογεί μία συμβολοσειρά και επιστρέφει ένα αντικείμενο.
float(x)	Μετατρέπει το x σε έναν αριθμό κινητής υποδιαστολής.
hex(x)	Μετατρέπει έναν ακέραιο αριθμό σε μία δεκαεξαδική συμβολοσειρά.
int(x)	Μετατρέπει το x σε ακέραιο αριθμό.
list(s)	Μετατρέπει το s σε μία λίστα.
oct(x)	Μετατρέπει έναν ακέραιο αριθμό σε μία οκταδική συμβολοσειρά.
ord(x)	Μετατρέπει ένα μονό χαρακτήρα στην ακέραιη τιμή του.
repr(x)	Μετατρέπει το x σε συμβολοσειρά έκφρασης.
set(s)	Μετατρέπει το s σε ένα σύνολο.
str(x)	Μετατρέπει το x σε μία αναπαράσταση συμβολοσειράς.
tuple(s)	Μετατρέπει το s σε μία πλειάδα.
unichr(x)	Μετατρέπει έναν ακέραιο αριθμό σε ένα unicode χαρακτήρα.

Πίνακας 2 Περιγραφή συναρτήσεων

2.2.13 Σταθερές

Οι μαθηματικές σταθερές (constants) που είναι ορισμένες στην Python είναι:
pi, η σταθερά π=3.1415.. και e, η βάση του φυσικού λογαρίθμου logex.

2.2.14 Τελεστές

Τελεστής είναι ένα σύμβολο που αναπαριστά μία πράξη. Η Python υποστηρίζει τους ακόλουθους τύπους τελεστών: αριθμητικούς, τελεστές σύγκρισης, σύνθετους τελεστές, καθώς και άλλους τελεστές.

2.2.14.1 Αριθμητικοί τελεστές

Αριθμητικοί είναι οι τελεστές που αναπαριστούν τις βασικές πράξεις, όπως φαίνεται και στον παρακάτω πίνακα.

Τελεστής	Λειτουργία
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
%	Υπόλοιπο διαίρεσης
**	Ύψωση σε δύναμη
//	Ακέραιο μέρος πηλίκου μίας διαίρεσης

Πίνακας 3 Λειτουργία αριθμητικών τελεστών

2.2.14.2 *Συγκριτικοί τελεστές*

Συγκριτικοί είναι οι τελεστές που χρησιμοποιούνται για τη σύγκριση χαρακτήρων και επιστρέφουν ως αποτέλεσμα True, αν είναι αληθής, ή False, αν είναι ψευδής. Οι τελεστές παρουσιάζονται στον επόμενο πίνακα.

Τελεστές	Λειτουργία
==	Ισότητα
!=	Ανισότητα
<=	Μικρότερο ή ίσο
>=	Μεγαλύτερο ή ίσο
>	Μεγαλύτερο
<	Μικρότερο

Πίνακας 4 Λειτουργία συγκριτικών τελεστών

2.2.14.3 *Σύνθετοι τελεστές*

Σύνθετοι χαρακτηρίζονται οι τελεστές που αποτελούν συνδυασμός δύο τελεστών. Χρησιμεύουν στη συντόμευση μίας πράξης.

Σύντομη πράξη	Αναλυτική πράξη
a+=b	a=a+b
a-=b	a=a-b
a*=b	a=a*b
a/=b	a=a/b

Πίνακας 5 Ανάλυση σύνθετων τελεστών

2.2.14.4 *Δυαδικός τελεστής*

Δυαδικός τελεστής (bitwise operator) λειτουργεί σε bits και εκτελεί τη λειτουργία σε κάθε ένα bit (bit by bit). Ουσιαστικά μετατρέπονται οι αριθμοί σε δυαδική μορφή εκτελούνται οι

πράξεις και στη συνέχεια, δίνεται το αποτέλεσμα στο δεκαδικό σύστημα. Οι υποστηριζόμενοι τελεστές είναι οι ακόλουθοι.

Τελεστής	**Λειτουργία**
&	Δυαδικό AND
|	Δυαδικό OR
^	Δυαδικό XOR
~	Δυαδικό συμπλήρωμα
<<	Δυαδική κύλιση αριστερά
>>	Δυαδική κύλιση δεξιά

Πίνακας 6 Λειτουργία δυαδικών τελεστών

2.2.14.5 *Τελεστές συμβολοσειρών*

Υπάρχουν τελεστές που ξεκινούν με το σύμβολο επί τοις εκατό (%) και χρησιμοποιούνται για την εμφάνιση ενός αντικειμένου σε μία συμβολοσειρά. Οι τελεστές αυτοί εισάγονται στην εντολή printf(). Ακολουθεί ένας πίνακας με τους τελεστές αυτούς.

Τελεστής	**Περιγραφή**
%c	Αναπαράσταση χαρακτήρα
%d	Αναπαράσταση (signed) δεκαδικού ακέραιου αριθμού (και αρνητικό αριθμό).
%e	Εκθετικός συμβολισμός (πεζό γράμμα)
%E	Εκθετικός συμβολισμός (κεφαλαίο γράμμα)
%f	Αναπαράσταση πραγματικού αριθμού κινητής υποδιαστολής
%g	Μικρότερος από το %f και το %e.
%G	Μικρότερος από το %F και το %E.
%i	Αναπαράσταση (signed) δεκαδικού ακέραιου αριθμού (και αρνητικό αριθμό)
%o	Αναπαράσταση οκταδικού ακέραιου αριθμού.
%s	Μετατροπή συμβολοσειράς μέσω της συνάρτησης str() πριν τη διαμόρφωση.
%u	Αναπαράσταση (unsigned) δεκαδικού ακέραιου αριθμού (αλλά όχι αρνητικό αριθμό).
%x	Αναπαράσταση δεκαεξαδικού ακέραιου αριθμού (πεζό γράμμα).
%X	Αναπαράσταση δεκαεξαδικού ακέραιου αριθμού (κεφαλαίο γράμμα).

Πίνακας 7 Περιγραφή τελεστών συμβολοσειρών

2.2.14.6 *Προτεραιότητα τελεστών*

Στον παρακάτω πίνακα θα παρουσιαστεί η προτεραιότητα των τελεστών, από τους τελεστές με υψηλότερη προτεραιότητα προς τους τελεστές με χαμηλότερη προτεραιότητα.

Τελεστής	Περιγραφή
**	Ύψωση σε δύναμη
~	Συμπλήρωμα
*, /, %, //	Πολλαπλασιασμός, Διαίρεση, Υπόλοιπο διαίρεσης, Ακέραιο μέρος πηλίκου μίας διαίρεσης
+, -	Πρόσθεση, Αφαίρεση
<<, >>	Δυαδική κύλιση αριστερά, Δυαδική κύλιση δεξιά
&	Δυαδικό AND
^, |	Δυαδικό XOR, Δυαδικό OR
>=, <=	Μεγαλύτερο ή ίσο, Μικρότερο ή ίσο
==, !=	Ισότητα, Ανισότητα

Πίνακας 8 Προτεραιότητα τελεστών

2.3 Δομές ελέγχου και επανάληψης

2.3.1 Δομή Ελέγχου if

Αν επιθυμείται η εκτέλεση μιας ακολουθίας εντολών εφόσον πληρείται μια συγκεκριμένη συνθήκη, τότε χρησιμοποιείται η δομή if και στην συνέχεια η συνθήκη η οποία ελέγχεται. Αν η συνθήκη είναι αληθής, τότε το σύνολο των εντολών που περιέχονται στην if αν θα εκτελεστούν, αλλιώς το πρόγραμμα θα συνεχίσει από το τέλος της if. Αν υπάρχουν και άλλες επιλογές χρησιμοποιείται το if...else ή το if...elif...else.
Η δομή ειναι :

```
if (συνθήκη):
        εντολές
else:
        εντολές
```

Και :

```
if (συνθήκη):
        εντολές
elif (συνθήκη):
        εντολές
else:
        εντολές
```

2.3.2 Δομές επανάληψης

Ένας βρόγχος (επανάληψης) είναι μια ακολουθία εντολών οι οποίες δηλώνονται μια φορά, αλλά μπορούν να εκτελεστούν πολλές διαδοχικές φορές. Ο κώδικας μέσα στον βρόγχο, γνωστό και ως σώμα του βρόγχου, εκτελείται για έναν καθορισμό αριθμό επαναλήψεων, ή για όσο ισχύει μια συνθήκη. Έτσι, διαχωρίζονται σε βρόχους for και while.

2.3.2.1 *Βρόχος for*

Ο βρόγχος for εκτελούνται για συγκεκριμένο πλήθος φορών. Στη συνθήκη της σύνταξής του είτε χρησιμοποιείται η συνάρτηση range() είτε κάποιο string ή λίστα. Στη συνάρτηση range(), όσο αυτή επιστρέφει ένα καινούργιο αντικείμενο, οι εντολές στο σώμα του βρόγχου συνεχίζουν να εκτελούνται. Αυτό μεταφράζεται σε:

```
for i in range ( ... ) :
      εντολές
```

Στην άλλη περίπτωση, όσο το i είναι μέσα στη συνθήκη, οι εντολές συνεχίζουν να εκτελούνται, άρα:

```
for i in συνθήκη :
      εντολές
```

2.3.2.2 *Βρόχος while*

Αντίθετα με τους βρόγχους for, ο βρόγχος while τρέχει συνεχώς όσο πληρείται μια συγκεκριμένη συνθήκη:

```
while συνθήκη
      εντολές
```

2.3.3 *Η δήλωση break*

Η δήλωση break χρησιμοποιείται για να βγει το πρόγραμμα από έναν βρόγχο μόλις την συναντήσει. Για να εκτελεστεί πρέπει να πληρείται μια προϋπόθεση, και για αυτό συνοδεύεται από μια δομή if που καθορίζει πότε εκτελείται. Για παράδειγμα,

```
for letter in 'Python':
   if letter == 'h':
      break
   print 'Current Letter :', letter
```

2.3.4 *Η δήλωση with*

Στην έκδοση 3 της Python, η λέξη with αποτελεί λέξη κλειδί, και μπορεί να έχει διάφορες χρήσεις. Η κύρια χρήση της είναι η απόκτηση και η απελευθέρωση πόρων. Με τη with ανοίγει ένα αρχείο και χρησιμοποιείται για τον έλεγχο του επιτυχούς ανοίγματος αυτού του αρχείου. Η with εξασφαλίζει την κατάλληλη δέσμευση και αποδέσμευση των πόρων από ένα μέρος μπλοκ κώδικα όταν κλείνει ανεξάρτητα από το αν έχουν συμβεί ή όχι εξαιρέσεις. Η with μπορεί να χρησιμοποιηθεί με την εμφώλευση των δυο with δηλώσεων. Μια χρήση

που βρίσκει αρκετά συχνά η συγκεκριμένη λειτουργικότητα είναι όταν διαβάζεται κάποια πληροφορία από ένα αρχείο και γράφεται σε ένα άλλο, αφού έχει γίνει κάποια επεξεργασία στο αρχείο από το οποίο διαβάζεται. Για παράδειγμα η with

```
with open('output.txt', 'w') as f:
    f.write('Hi there!')
```

2.4 Συναρτήσεις

Η συνάρτηση είναι ένα μέρος ομαδοποιημένου και επαναχρησιμοποιήσιμου κώδικα που επιτελεί σε κάποια λειτουργία. Μπορεί να τη χρησιμοποιηθεί σε οποιοδήποτε σημείο του υπόλοιπου κώδικα και καλείται παραπάνω από μία φορά. Μία συνάρτηση πρέπει να ακολουθεί τους παρακάτω κανόνες :

1. Η συνάρτηση πρέπει να ξεκινάει με τη λέξη-κλειδί def, ακολουθούμενη από το όνομά της, παρενθέσεις και το σύμβολο της άνω και κάτω τελείας. Ανάμεσα στις παρενθέσεις μπορούν να οριστούν τυχόν παράμετροι εισόδου.
2. Η δήλωση επιστροφής (return) είναι προαιρετική. Χρησιμοποιείται για να επιστρέψει επιχειρήματα (arguments) στον καλούντα. Μία δήλωση return χωρίς επιχειρήματα είναι ίδια με μία δήλωση return None.

Η σύνταξή της έχει ως εξής:

```
def όνομα_συνάρτησης():
εντολές
```

Για την κλήση της συνάρτησης, η οποία μπορεί να γίνει είτε από το κύριο πρόγραμμα είτε από κάποια άλλη συνάρτηση, χρησιμοποιείται το όνομα της συνάρτησης μόνο του ή αλλιώς με παράμετρους.

2.4.1 Ανώνυμες Συναρτήσεις

Η ανώνυμη συνάρτηση χρησιμοποιείται για να οριστεί και να περαστεί σαν όρισμα κάπου αλλού. Αντί της λέξης def χρησιμοποιείται η λέξη lambda, το όνομα της συνάρτησης και οι παρενθέσεις από τα ορίσματα παραλείπονται και επιστρέφει την τιμή χωρίς την χρήση του return.

```
lambda x1, x2, ..: έκφραση, όπου x1, x2 ορίσματα εισόδου
```

2.5 Βασικές δομές

Οι βασικές δομές της Python είναι : τα String (Αλφαριθμητικά), οι Λίστες (Lists), τα Σύνολα (Sets), Πλειάδα (Tuple) και Λεξικό (Dictionary).

2.5.1 Αλφαριθμητικά (Strings)

Το String είναι μία ακολουθία χαρακτήρων. Μπορούν να περιέχουν διάφορους χαρακτήρες και δεν περιορίζονται στον αριθμό χαρακτήρων που θα χρησιμοποιηθούν. Για να οριστούν θα πρέπει να περιέχονται σε μονούς (' ') ή διπλούς (" ") αποστρόφους. Για παράδειγμα,

```
myString = ' This is a string. '
```

Τέλος, είναι αμετάβλητα άρα δεν μπορεί να αλλάξει η τιμή τους μετά την δημιουργία τους. Τα Strings υποστηρίζουν πολλούς μεθόδους όπως

1. την myString.capitalize(), η οποία εμφανίζει το String με κεφαλαίο γράμμα,
2. την myString.count('x'), η οποία μετράει πόσες φορές βρίσκεται το x μέσα στο String,
3. την myString.find('x'), η οποία βρίσκει την θέση του x μέσα στο String,
4. την myString.replace('a', 'b'), η οποία αντικαθιστά τους χαρατήρες a και b,
5. την len(string), η οποία εμφανίζει το μέγεθος του String και
6. την myString.title, η οποία εμφανίζει αντίγραφο του με κάθε λέξη να ξεκινά με κεφαλαίο.

2.5.2 Λίστες

Η Λίστα είναι ένας σύνθετος τύπος δεδομένων. Μπορεί να περιέχει διάφορους τύπους δεδομένων και για αυτό οι λίστες χαρακτηρίζονται ως δυναμικές. Περικλείεται από αγκύλες, [ ], και τα στοιχεία της διαχωρίζονται με κόμμα. Τα στοιχεία της έχουν προκαθορισμένη σειρά και για πρόσβαση σε αυτά χρησιμοποιείται το id του στοιχείου στην αγκύλη (list[0], list[1], ..). Τέλος, αν ο δείκτης έχει αρνητική τιμή σημαίνει ότι ξεκινάει το μέτρημα από το τέλος της λίστας. Για παράδειγμα μία λίστα,

```
myList = [5, 9, 'Mary', 98, 'Nick']
print ("The third element of the list is ", myList[2])
The third element of the list is Mary
```

Αντίστοιχα με τα Strings, οι Λίστες έχουν συναρτήσεις και μεθόδους κάποιες από τις οποίες είναι οι εξής :

1. cmp(myList1, myList2), η οποία συγκρίνει δύο λίστες,
2. len(myList), η οποία επιστρέφει το μέγεθος της λίστας,
3. max(myLlist) και min(myList), που επιστρέφουν το μέγιστο και το ελάχιστο της λίστας αντίστοιχα,
4. myList.append(), που προσθέτει στοιχεία στο τέλος της λίστας,
5. myList.clear(), η οποία αφαιρεί όλα τα στοιχεία από τη λίστα και ισοδυναμεί με del a[:],
6. myList.copy(), που αντιγράφει την λίστα,

7. myList.count("x"), η οποία μετράει την εμφάνιση των x,
8. myList.remove("x"), η οποία αφαιρεί το πρώτο x που θα βρει από την λίστα,
9. myList.reverse(), η οποία αντιστρέφει τα στοιχεία της λίστας,
10. myList.extend(blist), η οποία προσθέτει το περιεχόμενο της blist στην myList,
11. myList.index("x"), η οποία βρίσκει τον δείκτη x στη λίστα και
12. myList.insert(id, "x"), η οποία προσθέτει το στοιχείο x στη θέση id στη λίστα.

2.5.3 Σύνολα (Sets)

Τα σύνολα διευκολύνουν την ομαδοποίηση πολλών αντικειμένων και την εφαρμογή στην συνέχεια πράξεων όπως η ένωση τους με αποδοτικό τρόπο, εξασφαλίζοντας πως κάθε στοιχείο, αν περιέχεται σε πάνω από ένα σύνολα, τελικά θα βρεθεί μόνο μια φορά στο τελικό αποτέλεσμα. Πρόκειται για μια δομή δεδομένων που αναπαριστάται ως ένα μη διατεταγμένο σύνολο μοναδικών στοιχείων.

Περικλείεται από άγκιστρο, { }, ή ορίζεται με τη συνάρτηση set και περικλείεται από αγκύλες και παρενθέσεις, ([ ]). Τα στοιχεία του συνόλου πρέπει να περιέχονται σε μονούς ('') ή διπλούς (" ") αποστρόφους.

Για παράδειγμα,

```
set1={'green', 'yellow', 'pink'}
set2=set(['purple', 'red', 'blue'])
print('The sets are ', set1, set2)
The sets are {'yellow', 'pink', 'green'} {'purple', 'blue',
'red'}
```

Τα σύνολα υποστηρίζουν διάφορες λειτουργίες κάποιες από τις οποίες είναι οι εξής :

1. len(mySet), η οποία εμφανίζει το μέγεθος του συνόλου,
2. "x" in mySet, που ελέγχει αν το x υπάρχει στο σύνολο mySet,
3. "x" not in aset, που ελέγχει αν το x δεν υπάρχει στο σύνολο mySet,
4. mySet.copy(), η οποία δημιουργεί ένα αντίγραφο του mySet,
5. mySet.add("x"), που χρησιμοποιείται για την προσθήκη στοιχείου x σε ένα σύνολο,
6. mySet.remove("x"), που χρησιμοποιείται για την αφαίρεση στοιχείου x από ένα σύνολο.

Τέλος, υπάρχουν και οι πράξεις συνόλου οι οποίες είναι :

Πράξη	**Συνάρτηση**	**Περιγραφή**
a | b	a.union(b)	Ένωση δύο συνόλων.
a & b	a.intersection(b)	Τομή δύο συνόλων.
a - b	a.difference(b)	Διαφορά των συνόλων.
a ^ b	a.symmetric_difference(b)	Συμμετρική διαφορά των δύο συνόλων.
a <= b	a.issubset(b)	Ελέγχει αν όλα τα στοιχεία του συνόλου a υπάρχουν στο σύνολο b.
a >= b	a.issuperset(b)	Ελέγχει αν όλα τα στοιχεία του συνόλου b υπάρχουν στο σύνολο a.

Πίνακας 9 Πράξεις συνόλων

Για παράδειγμα,

```
set3=set1|set2
 print("The elements of third set are ", set3)
 The elements of third set are {'purple', 'orange', 'green',
'red', 'yellow', 'blue'}
 set1<=set3
 True
```

2.5.4 Πλειάδα

Η Πλειάδα περιέχει αντικείμενα που συσχετίζονται το ένα με το άλλο, δεν μπορεί να αλλάξει ούτε μέγεθος ούτε στοιχεία και μπορεί να περιέχει οποιοδήποτε τύπο δεδομένων. Χρησιμοποιείται για την επιστροφή πολλών τιμών σε συναρτήσεις. Η πλειάδα με τη λίστα έχουν της εξής διαφορές:

1. η πλειάδα περικλείεται από παρενθέσεις, ενώ η λίστα από αγκύλες
2. τα στοιχεία και το μέγεθος της πλειάδας δεν μπορούν να αλλάξουν όπως της λίστας.

Η πλειάδα μπορεί να θεωρηθεί ως λίστα μόνο για ανάγνωση (readonly). Δεν γίνεται αφαίρεση στοιχείων πλειάδας. Για τη διαγραφή ολόκληρης της πλειάδας χρησιμοποιείται η εντολή del.

Τέλος, η πλειάδα υποστηρίζει αρκετές συναρτήσεις όπως τις γνωστές cmp(t1, t2), len(t1), max(t1), min(t1) και tuple(l1), στην οποία μία λίστα μετατρέπεται σε πλειάδα.

2.5.5 Λεξικό

Τα λεξικά είναι δομές που μπορούν να περιέχουν πολλούς τύπους δεδομένων και διανέμονται με ζεύγη κλειδιών-τιμών: για κάθε (μοναδικό) κλειδί, το λεξικό εξάγει μία τιμή. Τα κλειδιά μπορούν να είναι Strings, αριθμοί ή Πλειάδες, ενώ οι αντίστοιχες τιμές μπορούν να είναι οποιοδήποτε αντικείμενο της Python. Τα λεξικά είναι μη ταξινομημένα, επαναληπτικά και μεταβλητά. Το Λεξικό περικλείεται από άγκιστρα, { }. Για την εκχώρηση τιμής, ή την απόκτηση πρόσβασης σε μία τιμή χρησιμοποιούνται οι αγκύλες, [ ]. Η διαγραφή στοιχείων λεξικού η όλου του λεξικού επιτυγχάνεται με την εντολή del.

Για παράδειγμα,

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print ("dict['Name']: ", dict['Name'])
print ("dict['Age']: ", dict['Age'])
```

Τέλος, υπάρχουν οι ακόλουθοι μέθοδοι λεξικού :

1. myDict.clear(), που διαγράφει όλα τα στοιχεία του λεξικού,
2. myDict = dict.copy(), η οποία αντιγράφει το λεξικό,
3. del myDict["x"], που διαγράφει το ζευγάρι x,
4. myDdict = dict.fromkeys(mySet), η οποία δημιουργεί ένα νέο λεξικό με κλειδιά τις τιμές του συνόλου mySet και αξίες None (εκτός κι αν οριστεί κατά τη δημιουργία του λεξικού),
5. myDict.items(), που εμφανίζει μία λίστα από κλειδί και αξία,
6. myDict.keys(), η οποία εμφανίζει τα κλειδιά του λεξικού,
7. len(myDict), που εμφανίζει το μέγεθος του λεξικού,
8. myDict.update(bict1), η οποία προσθέτει στο λεξικό myDict το λεξικό bict1,
9. myDict.values(), που εμφανίζει τις τιμές του λεξικού.

2.6 Κλάση

Ο αντικειμενοστραφής προγραμματισμός αποτελεί μια τεχνική προγραμματισμού η οποία βασίζεται στα αντικείμενα και στις αλληλεπιδράσεις που μπορεί να έχουν μεταξύ τους ώστε ώστε να σχεδιαστεί κάποιο πρόγραμμα. Οι κλάσεις και τα αντικείμενα είναι κύριο μέρος του. Η Κλάση καθορίζει τα βασικά χαρακτηριστικά (attributes) και συμπεριφορές (methods) των αντικειμένων που περιγράφονται από αυτή. Το αντικείμενο είναι ένα συγκεκριμένο στιγμιότυπο της κλάσης. Τα αντικείμενα της ίδιας κλάσης μπορούν να αποδεχθούν τα ίδια μηνύματα. Τέλος, υποστηρίζεται η κληρονομικότητα στην οποία οταν μια κλάση αποτελεί εξειδίκευση μιας άλλης, και έτσι έχει όλα τα χαρακτηριστικά που περιγράφουν την κλάση από την οποία κληρονομεί, καθώς και κάποια ιδιαίτερα που περιγράφουν ειδικά την συγκεκριμένη ομάδα αντικειμένων.

Η σύνταξη είναι η εξής:

```
class όνομα_κλάσης:
    αντικείμενα και μέθοδοι
```

2.6.1 Μέθοδος

Η μέθοδος περιγράφει τις διάφορες συμπεριφορές ενός αντικειμένου. Ο ορισμός μίας μεθόδου γίνεται με τη λέξη κλειδί def που περιέχεται στο σώμα μίας κλάσης. Κάθε μέθοδος μίας κλάσης πρέπει να περιέχει ως πρώτο όρισμα την παράμετρο self, η οποία αναφέρεται στο αντικείμενο της κλάσης. Μία μέθοδος μπορεί να κληθεί μόνο πάνω σε κάποιο αντικείμενο.

Για παράδειγμα,

```
class Dog:
    kind = 'canine' # μεταβλητή της κλάσης που μοιράζεται όλα τα στιγμιότυπα
    def __init__(self, name):
        self.name = name # μεταβλητή της κλάσης μοναδική σε κάθε στιγμιότυπο
```

2.6.2 Ειδικοί Μέθοδοι

Υπάρχουν κάποιες μέθοδοι που υλοποιούνται μέσω των κλάσεων όπως η

1. __init__(), η οποία καλείται για την αρχικοποίηση ενός αντικειμένου
2. __del__(), η οποία καλείται για την καταστροφή ενός αντικειμένου.

2.7 Εξαιρέσεις

Μία εξαίρεση (exception) προκύπτει όταν κατά την εκτέλεση ενός προγράμματος διαταράσσεται η κανονική ροή του προγράμματος. Όταν το πρόγραμμα συναντά μια κατάσταση που δεν μπορεί να αντιμετωπίσει, με αποτέλεσμα να εμφανίζεται κάποιο σφάλμα (error). Όταν εμφανίζεται μία εξαίρεση θα πρέπει να χειριστεί αμέσως, αλλιώς το πρόγραμμα τερματίζει και κλείνει. Μπορεί να εμφανιστεί μία εξαίρεση κατά την προσπάθεια εκτέλεσης του κώδικα ακόμα και στην περίπτωση που μία δήλωση είναι συντακτικά σωστή.

Υπάρχουν πολλά είδη σφαλμάτων όπως:

1. Τα συντακτικά λάθη (syntax errors) ή σφάλματα ανάλυσης είναι το πιο συχνό σφάλμα που εμφανίζεται κατά την εκμάθηση της γλώσσας. Το παρακάτω, εμφανίζει συντακτικό σφάλμα μη έγκυρης σύνταξης. Για παράδειγμα,

```
for x in
SyntaxError: invalid syntax
```

2. Παρακάτω εμφανίζεται σφάλμα στο όνομα της εντολής. Δίνοντας πληροφορίες για τη γραμμή στην οποία βρίσκεται το λάθος καθώς και όνομα που φαίνεται ως μη ορισμένο.

```
print("Hello!")
Traceback (most recent call last):
File "<pyshell#0>", line 1, in <module>
print("Hello!")
NameError: name 'Print' is not defined
```

Η τελευταία γραμμή του μηνύματος εμφανίζει πληροφορίες για το σφάλμα, όπου, ως μέρος του μηνύματος, εμφανίζεται και ο τύπος του σφάλματος. Η εξαίρεση περιέχει κάποια σημαντικά χαρακτηριστικά για το χειρισμό των σφαλμάτων.

Τέλος, υπάρχουν οι EnvironmentError, EOFError, ImportError, MemoryError, SystemExit, ValueError και άλλες παρόμοιες εξαιρέσεις.

2.7.1 Χειρισμός εξαιρέσεων

Ο χειρισμός (handling) των εξαιρέσεων πρέπει να γίνει όταν αλλάξει η φυσιολογική ροή του προγράμματος. Ο γενικός μηχανισμός λειτουργίας των εξαιρέσεων είναι ο εξής.

1. Αν αλλάξει η συνηθισμένη ροή του προγράμματος τότε εγείρεται μία εξαίρεση.
2. Θα πρέπει να αναγνωριστεί η κλάση της εξαίρεσης, που αναγράφεται στην τελευταία γραμμή του μηνύματος λάθους (για παράδειγμα, SyntaxError, TypeError και λοιπά).
3. Τότε θα πρέπει να εκτελεστεί ο ειδικός κώδικας εξαίρεσης για τη συγκεκριμένη περίπτωση.
4. Αν δεν υπάρξει χειρισμός της εξαίρεσης που εγείρεται, τότε η εκτέλεση του προγράμματος θα σταματήσει. Οι εξαιρέσεις οι οποίες δεν έχουν χειριστεί, εμφανίζονται στο traceback που ενημερώνει για αυτά που έχουν συμβεί.

Η δήλωση try-except συντάσσεται με τα ακόλουθα βήματα : στην αρχή, πρέπει να περιλαμβάνει τη λέξη try, άνω και κάτω τελεία (:) και στη συνέχεια το μπλοκ κώδικα. Έπειτα, περιλαμβάνει τη λέξη except, το όνομα της εξαίρεσης, άνω και κάτω τελεία (:) και ακολουθείται από ένα μπλοκ κώδικα που χειρίζεται το σφάλμα. Μία δήλωση try μπορεί να διαθέτει παραπάνω από μία δήλωση except, για τον καθαρισμό των διαφόρων εξαιρέσεων. Από τις εξαιρέσεις αυτές θα εκτελεστεί το πολύ ένας χειριστής. Ο χειριστής χειρίζεται μόνο εξαιρέσεις που προκύπτουν στην αντίστοιχη δήλωση try και όχι σε άλλες δηλώσεις try. Στην τελευταία δήλωση except μπορεί να παραλειφθεί το όνομα εξαίρεσης και να χρησιμοποιηθεί ως εξαίρεση "μπαλαντέρ". Όμως, η χρήση της δήλωσης αυτής χρειάζεται πολύ προσοχή. Μπορεί να χρησιμοποιηθεί για την εμφάνιση ενός μηνύματος λάθους. Επιπλέον, έχει έναν προαιρετικό όρο else και όταν υπάρχει πρέπει να ακολουθεί όλες τις except δηλώσεις. Είναι χρήσιμο στην περίπτωση που υπάρχει κώδικας που πρέπει να εκτελεστεί, αν η πρόταση try δεν εγείρει μία εξαίρεση.

Η δομή της δήλωσης try-except φαίνεται παρακάτω:

```
try:
   λειτουργίες
except όνομα_εξαίρεσης1:
   εντολές
except όνομα_εξαίρεσης2:
   εντολές
.....
else:
   εντολές
```

Τέλος, υπάρχει και η δήλωση try-finally η οποία εμπεριέχει κάποιες εντολές οι οποίες πρέπει να εκτελεστούν είτε θέσει μια εξαίρεση η όχι η δήλωση try. Όμως, δεν μπορεί να υπάρχει ταυτόχρονα η δήλωση finally με την δήλωση except ή η δήλωση finally με την else. Η δομή της είναι:

```
try:
    λειτουργίες
finally:
    εντολές
```

2.8 Αρχείο

Τα αρχεία αποθηκεύουν και ανακτούν πληροφορίες οι οποίες δεν έχουν να κάνουν μόνο με κάποιο συγκεκριμένο πρόγραμμα που τρέχει.

2.8.1 Προσπέλαση

Προτού εκτελεστεί οποιαδήποτε λειτουργία πάνω σε ένα αρχείο, πρέπει να ανοίξει, έτσι ώστε να ενημερωθεί το λειτουργικό ότι πρόκειται να χρησιμοποιηθεί το αρχείο και να αναλάβει αυτό να το ανασύρει από τον σκληρό δίσκο. Για να ανοίξει ένα αρχείο, χρησιμοποιείται η συνάρτηση open() με όρισμα το όνομα του αρχείου. Η συνάρτηση επιστρέφει έναν περιγραφέα για το αρχείο, ο οποίος και χρησιμοποιείται ώστε να προσπελαστεί. Έπειτα μπορούν να εφαρμοστούν, χρησιμοποιώντας τον περιγραφέα, διάφορες λειτουργίες πάνω στο αρχείο. Οι λειτουργίες που υλοποιούνται μέσω της open() φαίνονται στον πίνακα.

Λειτουργίες	**Περιγραφή**
‘‘r’’	Ανάγνωση
‘‘w’’	Εγγραφή (Διαγραφή τυχών προηγούμενων περιεχομένων)
‘‘a’’	Προσθήκη (Διατήρηση τυχών προηγούμενων περιεχομένων)
‘‘b’’	Αρχείο δυαδικής μορφής
‘‘+’’	Προσθήκη δεδομένων στο τέλος του αρχείου ‘‘r+’’ είναι άνοιγμα αρχείου και για ανάγνωση/εγγραφή

Πίνακας 10 Λειτουργίες αρχείων

Για το κλείσιμο ενός αρχείου χρησιμοποιείται η εντολή f.close.

2.8.2 Διάβασμα και εγγραφή

Η κύρια συνάρτηση για διάβασμα από αρχείο είναι η read(). Χωρίς όρισμα θα διαβάσει μέχρι το τέλος του αρχείου, αλλιώς για τον αριθμό bytes όπως καθορίζεται από το όρισμα της. Για την ανάγνωση ενός αρχείου υπάρχει και η εντολή readlines(). Η διαφορά της εντολής read() με την readlines() είναι ότι η read() διαβάζει όλο το αρχείο και το επιστρέφει σε μία μεταβλητή, ενώ η readlines() διαβάζει τις γραμμές μία-μία και τις επιστρέφει σε μία λίστα.

Για την εγγραφή ενός αρχείου χρησιμοποιείται η συνάρτηση write(). Η εντολή γράφει μία συμβολοσειρά σε ένα αρχείο και επιστρέφει το σύνολο των χαρακτήρων. Δημιουργείται ένα αρχείο :

```
file = open("testfile.txt","w")
file.write("Hello World")
file.write("This is our new text file")
file.write("and this is another line.")
file.write("Why? Because we can.")

file.close()
```

Για παράδειγμα, διαβάζεται ως εξής και με τους δύο τρόπους :

```
file = open("testfile.text", "r")
print file.read()

file = open("testfile.txt", "r")
print file.readlines()
```

Το print του readlines θα εμφανίσει την λίστα :

```
['Hello World', 'This is our new text file', 'and this is
another line.', 'Why? Because we can.']
```

ΚΕΦΑΛΑΙΟ 3:
ΑΝΑΛΥΣΗ ΤΩΝ NumPy, Pandas ΚΑΙ Matplotlib

Στο κεφάλαιο αυτό γίνεται περαιτέρω ανάλυση των πιο γνωστών βιβλιοθηκών της Python NumPy, Pandas και μία αναφορά της Matplotlib. Για την ανάλυση του συγκεκριμένου κεφαλαίου χρησιμοποιήθηκε το Jupiter Notebook.

3.1 NumPy

Το πακέτο επέκτασης της Python για επιστημονικούς υπολογισμούς ονομάζεται NumPy και μέσω της κλάσης ndarray μας παρέχει προχωρημένες δυνατότητες χειρισμού πινάκων N-διαστάσεων.

3.1.1 Πίνακες NumPy (Arrays)

Οι πίνακες NumPy είναι πολυδιάστατες ομοιόμορφες συλλογές στοιχείων που αποθηκεύονται σε συνεχόμενα μπλοκ μνήμης και δεικτοδοτούνται μέσω μιας πλειάδας θετικών ακεραίων.

Τα βασικότερα χαρακτηριστικά του αντικειμένου ndarray είναι :

1. ndim: ο αριθμός των διαστάσεων του πίνακα. Ο αριθμός αυτός αναφέρεται στην Python ως rank
2. shape: Μια πλειάδα που καθορίζει το μέγεθος του πίνακα ως προς κάθε διάσταση. Πίνακας nxm έχει ως shape την τιμή (m,n)
3. dtype: Ο τύπος των στοιχείων του πίνακα. Εκτός των βασικών τύπων της Python το NumPy παρέχει επιπλέον τύπους όπως τα numpy.int16 και numpy.float64
4. itemsize: Το μέγεθος ενός στοιχείου του πίνακα σε bytes.

Η εισαγωγή της γίνεται με την εντολή import:

```
import numpy as np
```

3.1.1.1 *Δημιουργία Πινάκων*

Για τη δημιουργία ενός μονοδιάστατου πίνακα χρησιμοποιείται μία λίστα ή πλειάδα της Python με τη συνάρτηση array(seq) :

```
In [18]: import numpy as np

         a = np.array([1, 2, 3])   # Δημιουργία του πίνακα
         print(type(a))            # Εμφανίζει "<class 'numpy.ndarray'>"
         print(a.shape)            # Εμφανίζει "(3,)"
         print(a[0], a[1], a[2])   # Εμφανίζει "1 2 3"
         a[0] = 5                  # Αλλάζει ένα στοιχείο του πίνακα
         print(a)                  # Εμφανίζει "[5, 2, 3]"ω

         <class 'numpy.ndarray'>
         (3,)
         1 2 3
         [5 2 3]
```

Εικόνα 1 Δημιουργία πίνακα NumPy

Για την δημιουργία ενός δισδιάστατου πίνακα το όρισμά του πρέπει να είναι μία ακολουθία που τα στοιχεία της πρέπει να είναι ακολουθίες ίδιου μήκους. Για την πρόσβαση σε κάποιο στοιχείο χρησιμοποιούνται δύο δείκτες που χωρίζονται με κόμμα και δεν είναι υποχρεωτική η χρήση δευτέρου ζεύγους αγκυλών. Οι δείκτες αναφέρονται στη γραμμή και τη στήλη αντίστοιχα.

```
In [21]: b = np.array([[1,2,3],[4,5,6]])    # Δημιουργία δισδιάστατου πίνακα
         print(b.shape)                     # Εμφανίζει "(2, 3)"
         print(b[0, 0], b[0, 1], b[1, 0])   # Εμφανίζει "1 2 4"

         (2, 3)
         1 2 4
```

Εικόνα 2 Δημιουργία δισδιάστσατου πίνακα NumPy

Τέλος, υπάρχουν και άλλοι τρόποι να δημιουργηθούν πίνακες όπως οι συναρτήσεις zeros(shape) και ones(shape) δημιουργούν πίνακες με μηδενικά και μονάδες αντίστοιχα, η συνάρτηση random: random.random([size]) με τυχαίες τιμές και η συνάρτηση full(size[,fill_value]) η οποία χρησιμοποιεί τιμές ορισμένες απο το περιβάλλον.

Παραδείγματα όπως φαίνονται παρακάτω:

```
In [22]: import numpy as np

         a1 = np.zeros((2,2))   # Δημιουργεί ένα πίνακα γεμάτο μηδενικά
         print(a1)              # Εμφανίζει "[[ 0.  0.]
                               #          [ 0.  0.]]"

         a2 = np.ones((1,2))    # Δημιουργεί ένα πίνακα γεμάτο ένα
         print(a2)              # Εμφανίζει "[[ 1.  1.]]"

         a3 = np.full((2,2), 7)  # Δημιουργεί ένα πίνακα με τιμές που ορίζονται
                                 # από το περιβάλλον
         print(a3)               # Εμφανίζει "[[ 7.  7.]
                                #          [ 7.  7.]]"

         a4 = np.random.random((2,2))  # Δημιουργεί ένα πίνακα γεμάτο τυχαίες
                                       # τιμες
         print(a4)           # Μπορεί να εμφανίζει "[[ 0.91940165 0.08143945]
                                #                  [ 0.68744145  0.87236685]]"

         [[0. 0.]
          [0. 0.]]
         [[1. 1.]]
         [[7 7]
          [7 7]]
         [[0.40078806 0.34177429]
          [0.83064816 0.36451996]]
```

Εικόνα 3 Διαφορετικοί τρόποι δημιουργίας πινάκων NumPy

3.1.1.2 *Μέθοδοι πινάκων*

Στη NumPy επιτρέπεται η συνένωση δύο πινάκων με την μέθοδο concatenate.

```
a1 = array([1,2,3])
a2 = array([4,5,6])
a = concatenate((a1,a2))
```

Σε πολυδιάστατους πίνακες , οι πίνακες πρέπει να έχουν ίδιες διαστάσεις, Η συνένωση γίνεται κατά την πρώτη διάσταση, εκτός αν καθοριστεί κάποια άλλη με τη βοήθεια του ορίσματος axis:

```
a1 = array([[1,2],[3,4]])
a2 = array([[5,6],[7,8]])
a = concatenate((a1,a2),axis=1)
```

Τέλος, υπάρχουν οι μεθόδοι :

1. copy(), η οποία επιστρέφει ένα ακριβές αντίγραφο του πίνακα,
2. astype(), η οποία επιστρέφει ένα αντίγραφο του πίνακα στο οποίο έχει αλλάξει ο τύπος των στοιχείων του,
3. resize(), η οποία αλλάζει τις διαστάσεις και το μέγεθος του πίνακα. Τα νέα στοιχεία έχουν αρχική τιμή 0,

4. reshape(newshape), η οποία δίνει τη δυνατότητα να αλλάξουν οι διαστάσεις ενός πίνακα, χωρίς να πειραχτούν τα δεδομένα του.

3.1.1.3 *Τύποι δεδομένων*

Κάθε πίνακας είναι είναι ένα σύνολο στοιχείων ίδιου τύπου. Το NumPy παρέχει ένα μεγάλο σύνολο αριθμητικών τύπων δεδομένων που μπορούν να χρησιμοποιηθούν για να δημιουργηθούν πίνακες. Ο NumPy προσπαθεί να μαντέψει έναν τύπο δεδομένων κατά τη δημιουργία ενός πίνακα, αλλά οι λειτουργίες που κατασκευάζουν συστοιχίες συνήθως περιλαμβάνουν επίσης ένα προαιρετικό όρισμα για να καθοριστεί ρητά ο τύπο δεδομένων.

Για παράδειγμα,

```
In [24]: import numpy as np
         x = np.array([1, 2])   # Το NumPy διλέγει τον τύπο δεδομένων
         print(x.dtype)         # Εμφανίζει "int64"

         x = np.array([1.0, 2.0])   # Το NymPy διαλέγει τον τύπο δεδομένων
         print(x.dtype)             # Εμφανίζει "float64"

         x = np.array([1, 2], dtype=np.int64)   # Δίνεται συγκεκριμένος τύπος
         print(x.dtype)                         # Εμφανίζει "int64"

         int32
         float64
         int64
```

Εικόνα 4 Τύποι δεδομένων στοιχείων του πίνακα

3.1.1.4 *Δείκτες (Indexing) πινάκων*

Η NumPy προσφέρει πολλούς τρόπους που μπορούν να μπουν δείκτες στους πίνακες, όπως τον τεμαχισμό (Slicing).

Το Slicing είναι παρόμοιο με τις λίστες της Python, οι πίνακες μπορούν να τεμαχιστούν. Αφού οι πίνακες μπορούν να είναι πολυδιάστατοι, πρέπει να διευκρινιστεί ένα slice για κάθε διάσταση του πίνακα. Πιο συγκεκριμένα, η χρήση του [:] υποδεικνύει ότι χρησιμοποιούνται όλες οι γραμμές ή οι στήλες αντίστοιχα ενώ η χρήση [x:y] ότι χρησιμοποιούνται οι γραμμές μέχρι το x, με αυτό να εξαιρείται, και οι στήλες μέχρι το y με αυτό να εξαιρείται.
Στο παρακάτω κομμάτι κώδικα δίνεται ένα παράδειγμα στη χρήση του Slicing:

```
In [25]: # Δημιουργείται ο ακόλουθος πίνακας με μέγεθος (3, 4)
         # [[ 1  2  3  4]
         #  [ 5  6  7  8]
         #  [ 9 10 11 12]]
         a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

         # Παράδειγμα της χρήσης του slicing στον πίνακα a.

         a1 = a[0, :] # Εμφανίζει την πρώτη γραμμή του a [1, 2, 3, 4]
         a2 = a[:, 0] # Εμφανίζει την πρώτη στήλη του a [1, 5, 9]
         a3 = a[:, :2] # Εμφανίζει ακριβώς τις στήλες 0 και 1 χωρίς την 2
         a4 = a[:, 2:] # Εμφανίζει τις στήλες από την 2 και μετά
                       # συμπεριλαμβάνοντάς την

         # Χρησιμοποιείται το slicing για να δημιουργηθεί ένας υποπίνακας που
         # αποτελείται από τις 2 πρώτες γραμμές και τις στήλες 1 και 2. Η γραμμή 2  # και στήλη 3 δεν συμπεριλαμβάνονται στον πίνακα b.
         # [[2 3]
         #  [6 7]]
         b = a[:2, 1:3]
```

Εικόνα 5 Παράδειγμα του Slicing

Τέλος, ένα slice του πίνακα είναι μια εικόνα στα ίδια δεδομένα, άρα τροποποιώντας το θα τροποποιηθεί και ο αρχικός πίνακας.

```
print(a[0, 1])   # Εμφανίζει "2"
b[0, 0] = 77     # b[0, 0] είναι το ίδιο με το στοιχείο a[0, 1]
print(a[0, 1])   # Εμφανίζει "77"
```

```
2
77
```

Εικόνα 6 Τροποποίηση μέσω slice του αρχικού πινακα

Ένα slice είναι πολύ βολικό και το γεγονός ότι μπορεί να δημιουργηθεί ως προβολή το καθιστά αποτελεσματικό. Αλλά μερικές λειτουργίες δεν μπορούν πραγματικά να γίνουν με slice. Κάνοντας indexing σε ένα πίνακα ακεραίων ή σε ένα πίνακα boolean δημιουργείται ένα αντίγραφο του πίνακα και όχι μία προβολή όπως φαίνεται παραπάνω.
Στο παρακάτω κομμάτι κώδικα δίνεται παράδειγμα για τον πίνακα ακεραίων.

```
In [27]: a = np.array([[1,2], [3, 4], [5, 6]])

         # Indexing σε πίνακα ακεραίων .
         # Ο πίνακας που επιστρέφει θα έχει μέγεθος (3,) και θα εμφανίζει
         print(a[[0, 1, 2], [0, 1, 0]])  # "[1 4 5]"

         # Για την κατανόηση του παραδείγματος χρησιμοποιείται η παρακάτω εντολή
         #που είναι ισάξια
         print(np.array([a[0, 0], a[1, 1], a[2, 0]]))  # Το οποίο εμφανίζει
                                                       # ακριβώς "[1 4 5]"

         # Μπορεί να χρησιμοποιηθεί ξανά το ίδιο στοιχείο από τον αρχικό πίνακα
         print(a[[0, 0], [1, 1]])  # Εμφανίζει "[2 2]"

         # Είναι ισάξιο με το εξής
         print(np.array([a[0, 1], a[0, 1]]))  # Το οποίο εμφανίζει "[2 2]"

         [1 4 5]
         [1 4 5]
         [2 2]
         [2 2]
```

Εικόνα 7 Indexing σε πίνακα ακεραίων

Επίσης, με τον πίνακα ακεραίων υπάρχει η επιλογή ή η αλλαγή κάποιου στοιχείου.

```
In [28]: a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])

         # Δημιουργείται ένας πίνακας με την επιλογή στοιχείων από τον a
         b = np.array([0, 2, 0, 1])

         # Διαλέγει τα στοιχεία από κάθε σειρά του a με βάση τον b
         print(a[np.arange(4), b])  # Εμφανίζει "[ 1  6  7 11]"

         # Προσθέτει σε κάθε στοιχείο το 10
         a[np.arange(4), b] += 10

         print(a)  # Εμφανίζει "   [[11,  2,  3],
                   #                [ 4,  5, 16],
                   #                [17,  8,  9],
                   #                [10, 21, 12]] "

         [ 1  6  7 11]
         [[11  2  3]
          [ 4  5 16]
          [17  8  9]
          [10 21 12]]
```

Εικόνα 8 Επιλογή ή αλλαγή στοιχείου σε πίνακα ακεραίων

Στη συνέχεια δίνεται το παράδειγμα για τον πίνακα boolean.

```
In [30]: a = np.array([[1,2], [3, 4], [5, 6]])

         bool_idx = (a > 2)   # Βρίσκει τα στοιχεία του a που είναι > 2
                              # αυτο επιστρέφει ένα πίνακα από Booleans με το ίδιο
                              # σχήμα του a, όπου κάθε θέση του bool_idx δείχνει
                              # εάν το στοιχείο σε αυτή τη θέση του a είναι > 2.

         print(bool_idx)      # Εμφανίζει "[[False False]
                              #            [ True  True]
                              #            [ True  True]]"

         # Χρησιμοποιείται ένας πίνακας boolean για να κατασκευαστεί μία σειρά
         # που αποτελείται απο τα στοιχεία των πραγματικών τιμών του bool_idx
         print(a[bool_idx])  # Εμφανίζει "[3 4 5 6]"

         # Το παραπάνω πραγματοποιείται και με τον εξής τρόπο
         print(a[a > 2])     # Εμφανίζει "[3 4 5 6]"

         [[False False]
          [ True  True]
          [ True  True]]
         [3 4 5 6]
         [3 4 5 6]
```

Εικόνα 9 Παραδείγματα για πίνακα boolean

3.1.1.5 *Βασικές μαθηματικές συναρτήσεις*

Η βιβλιοθήκη NumPy υποστηρίζει τις μαθηματικές πράξεις και παρέχει πάνω από 60 μαθηματικές συναρτήσεις (universal functions) για τον υπολογισμό ποσοτήτων των στοιχείων ενός πίνακα.

```
In [31]: x = np.array([[1,2],[3,4]], dtype=np.float64)
         y = np.array([[5,6],[7,8]], dtype=np.float64)

         # Πρόσθεση
         print(x + y)
         print(np.add(x, y))

         # Αφαίρεση
         print(x - y)
         print(np.subtract(x, y))

         # Πολλαπλασιασμός
         print(x * y)
         print(np.multiply(x, y))

         # Διαίρεση
         print(x / y)
         print(np.divide(x, y))

[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

Εικόνα 10 Βασικές μαθηματικές συναρτήσεις στη NumPy

Τέλος, υπάρχει και η συνάρτηση dot που χρησιμοποιείται για διάφορους υπολογισμούς, ανάλογα με τα ορίσματα με τα οποία καλείται. Για μονοδιάστατα διανύσματα υπολογίζει το εσωτερικό τους γινόμενο ενώ για πίνακες 2 διαστάσεων εκτελεί πολλαπλασιασμό πινάκων. Η dot για πολυδιάστατους πίνακες υπολογίζει το εσωτερικό γινόμενο της τελευταίας διάστασης του πρώτου πίνακα με τη δεύτερη διάσταση από το τέλος του δεύτερου πίνακα.

3.1.1.6 *Μητρώο (Matrix)*

Η NumPy προσφέρει πολλούς τρόπους για την αναμόρφωση ή την διαχείριση δεδομένων των πινάκων και ένας από αυτούς είναι το matrix, που είναι μια υποκλάση της ndarray. Το matrix είναι ένας εξειδικευμένος δισδιάστατος πίνακας που διατηρεί τη φύση του στη διάρκεια των λειτουργιών. Έχει ορισμένους ειδικούς χειριστές όπως το τελεστή * που χρησιμοποιείται για τον πολλαπλασιασμό πινάκων και το ** για την ύψωση σε δύναμη.

Τέλος, χρησιμοποιεί το σύμβολο T για τον ανάστροφο πίνακα, το H για το συζυγή ανάστροφο και το I για τον αντίστροφο πίνακα.

```
a = np.matrix('1 2; 3 4')
print(a) # Εμφανίζει "[[1 2]
                       [3 4]] "
```

3.1.1.7 *Broadcasting*

Το Broadcasting είναι ένας μηχανισμός που επιτρέπει στο NumPy να δουλεύει με πίνακες διαφορετικών σχημάτων κατά τη διάρκεια της εκτέλεσης αριθμητικών πράξεων. Συνήθως, υπάρχει ένας μικρός πίνακας και ένας μεγαλύτερος, και πρέπει ο μικρός να χρησιμοποιηθεί πολλές φορές για να εκτελεστεί μία λειτουργία στον μεγαλύτερο πίνακα.

Για την πρόσθεση ενός διανύσματος σε κάθε σειρά ενός matrix μπορεί να χρησιμοποιηθεί ο τρόπος :

```
In [33]:
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))   # Δημιουργία 4 αντιγράφων του v
print(vv)                 # Εμφανίζει "[[1 0 1]
                          #           [1 0 1]
                          #           [1 0 1]
                          #           [1 0 1]]"
y = x + vv
print(y)  # Εμφανίζει "[[ 2  2  4
          #           [ 5  5  7]
          #           [ 8  8 10]
          #           [11 11 13]]"

[[1 0 1]
 [1 0 1]
 [1 0 1]
 [1 0 1]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

Εικόνα 11 Πρόσθεση διανύσματος με matrix

Το broadcasting επιτρέπει να γίνει η πράξη χωρίς να δημιουργηθούν τα αντίγραφα του v.

```
In [34]: x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
         v = np.array([1, 0, 1])
         y = x + v  # Πρόσθεση με broadcasting
         print(y)

         [[ 2  2  4]
          [ 5  5  7]
          [ 8  8 10]
          [11 11 13]]
```

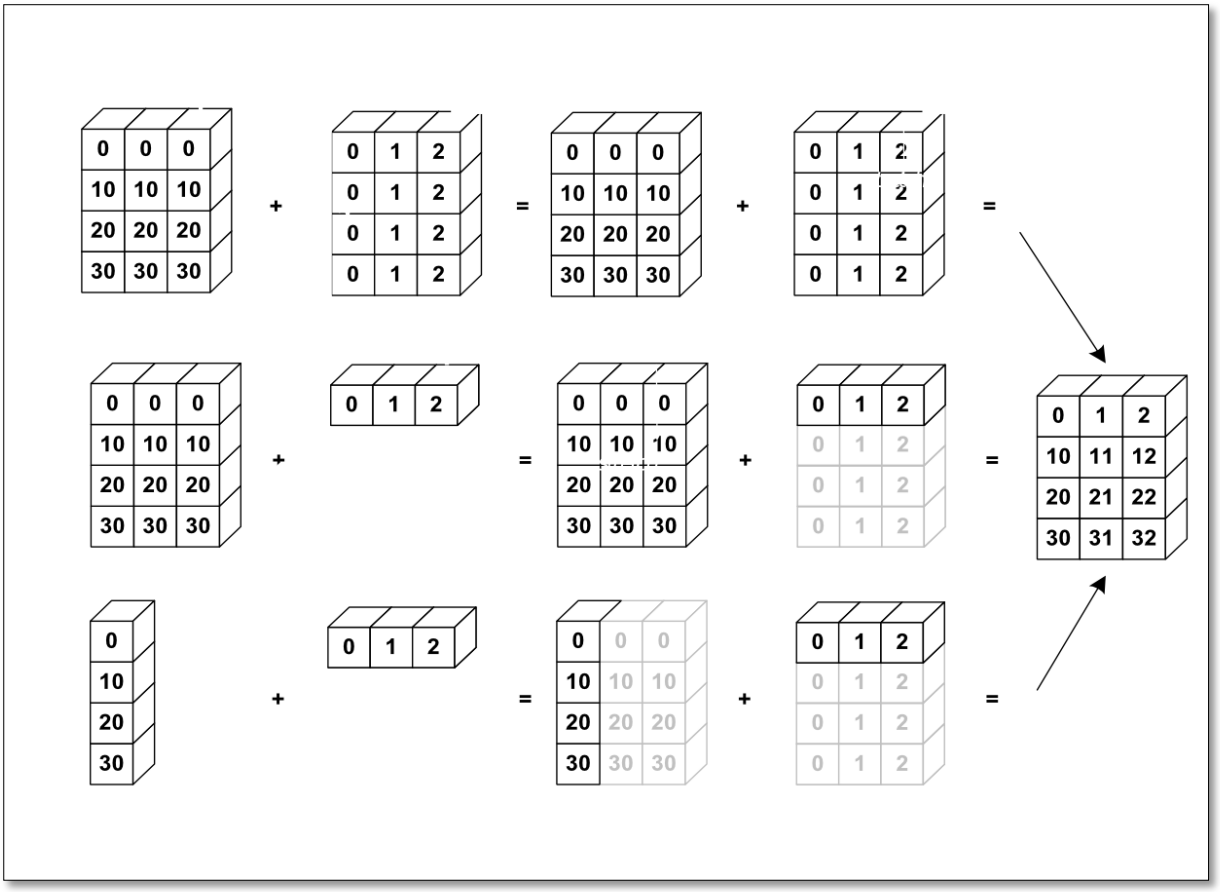
Εικόνα 12 Πρόσθεση με broadcasting

Οι κανόνες για να γίνει broadcasting ανάμεσα σε δύο πίνακες είναι οι εξής :

1. Αν οι διαστάσεις είναι ίδιες τότε συνεχίζει κανονικά
2. Σε οποιαδήποτε διάσταση όπου μια διάταξη είχε μέγεθος 1 και ο άλλος πίνακας είχε μέγεθος μεγαλύτερο από 1, η πρώτη σειρά συμπεριφέρεται σαν να αντιγράφηκε κατά μήκος αυτής της διάστασης
3. Όταν ένας πίνακας δεν έχει άλλες διαστάσεις, χρησιμοποιείται το 1 στην σύγκριση μέχρι να τελειώσουν οι διαστάσεις του δεύτερου.

Ακολουθούν κάποια παραδείγματα με τα μεγέθη του A και του B και το αποτέλεσμα της ένωσης, όπως και η εικόνα 3.1 :

```
A      (2d array):  5 x 4
B      (1d array):      1
Result (2d array):  5 x 4


A      (2d array):  5 x 4
B      (1d array):      4
Result (2d array):  5 x 4

A      (3d array):  15 x 3 x 5
B      (3d array):  15 x 1 x 5
Result (3d array):  15 x 3 x 5

A      (3d array):  15 x 3 x 5
B      (2d array):       3 x 5
Result (3d array):  15 x 3 x 5

A      (3d array):  15 x 3 x 5
B      (2d array):       3 x 1
Result (3d array):  15 x 3 x 5
```



Εικόνα 13 Παράδειγμα του broadcasting

3.2 Pandas

Συνήθως λέγεται ότι το 80% της ανάλυσης δεδομένων ξοδεύεται στην προετοιμασία και τον καθαρισμό των δεδομένων. Το Pandas είναι το καλύτερο εργαλείο για την χειραγώγηση των δεδομένων και τον καθαρισμό τους.

3.2.1 DataFrame και Series

Το Pandas έχει δύο τύπους δεδομένων :

1. το Series, το οποίο είναι ένας μονοδιάστατος πίνακας που φέρει ετικέτα, ικανός να έχει οποιοδήποτε τύπο δεδομένων (ακέραιους, συμβολοσειρές, αντικείμενα Python, κ.τ.λ.). Οι ετικέτες του axis αναφέρονται συλλογικά ως δείκτες. Η βασική μέθοδος δημιουργίας μιας σειράς είναι:

```
s = pd.Series(data, index=index)
```

2. το DataFrame, το οποίο είναι μια δισδιάστατη δομή δεδομένων που φέρει ετικέτα με στήλες διαφορετικών τύπων. Είναι γενικά το πιο συχνά χρησιμοποιούμενο αντικείμενο pandas. Το DataFrame υποστηρίζει δεδομένα τα οποία είτε είναι άλλα DataFrame, είτε Series, είτε NumPy ndarray, που μπορεί να καταγραφεί η να δομηθεί, είτε δισδιάστατο ndarray, είτε dictionaries από μονοδιάστατους ndarray, από λίστες, dictionaries από Series. Εκτός από τα δεδομένα, μπορεί επίσης να καθοριστούν ο index και τα ονόματα των columns(στηλών) για το DataFrame. Ο index, αφενός, δείχνει τη διαφορά στις σειρές, ενώ τα ονόματα των columns υποδεικνύουν τη διαφορά στις στήλες. Εάν δεν έχουν περαστεί, θα κατασκευαστούν από τα δεδομένα εισόδου με βάση τους κανόνες κοινής λογικής.

Εδώ, θα αναλυθεί ο τύπος δεδομένων DataFrame μιας και είναι ο πιο συνηθισμένος στη χρήση.

3.2.2 Δημιουργία DataFrame

Αρχικά, ένα DataFrame μπορεί να δημιουργηθεί μέσω του dict αποτελούμενο από Series. Ο index που θα χρησιμοποιηθεί θα είναι η ένωση των index του Series. Εάν δεν περαστούν columns τότε αυτές θα είναι τα κλειδια του dict των Series.

```
In [58]: d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
              'two' : pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

         datafr = pd.DataFrame(d)

         datafr
```

Out[58]:

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

Εικόνα 14 Δημιουργία DataFrame από Series' dictionary

Έπειτα, μπορεί να δημιουργηθεί από ένα dict από ndarrays ή και από λίστες. Τα ndarrays πρέπει να έχουν το ίδιο μήκος. Εάν περάσει ένας index, πρέπει σαφώς να έχει το ίδιο μήκος με τους πίνακες. Αν δεν περάσει τότε το αποτέλεσμα θα είναι το range(n), όπου n είναι το μήκος του πίνακα.

```
In [59]: d = {'one' : [1., 2., 3., 4.],
              'two' : [4., 3., 2., 1.]}

         pd.DataFrame(d)
```

Out[59]:

	one	two
0	1.0	4.0
1	2.0	3.0
2	3.0	2.0
3	4.0	1.0

```
In [70]: pd.DataFrame(d, index=['a', 'b', 'c', 'd'])
```

Out[70]:

	one	two
a	1.0	4.0
b	2.0	3.0
c	3.0	2.0
d	4.0	1.0

Εικόνα 15 Δημιουργία DataFrame από ndarrays' dictionary

Ή από NumPy ndarray.

```
In [67]: data = [(1,2.,'Hello'), (2,3.,"World")]
         pd.DataFrame(data)
```

Out[67]:

	0	1	2
0	1	2.0	Hello
1	2	3.0	World

```
In [68]: pd.DataFrame(data, index=['first', 'second'])
```

Out[68]:

	0	1	2
first	1	2.0	Hello
second	2	3.0	World

```
In [69]: pd.DataFrame(data, columns=['C', 'A', 'B'])
```

Out[69]:

	C	A	B
0	1	2.0	Hello
1	2	3.0	World

Εικόνα 16 Δημιουργία DataFrame από NumPy ndarray

Καταλήγοντας, ο πιο απλός τρόπος είναι μέσω της πληκτρολόγησης των δεδομένων σε Python με το χέρι, η οποία προφανώς λειτουργεί μόνο για μικροσκοπικά σύνολα δεδομένων.

```
In [9]: new_dataframe = pd.DataFrame ( {
                    "column_1" : [1, 2, 3, 4, 5],
                    "another_column" : ['th', 'col', 'ha', 'sti', 'ins'],
                    "float_column" : [0.1, 0.5, 33, 48, 42.588],
                    "binary_solo" : [True, False, True, True, False] } )

In [10]: new_dataframe
```

Out[10]:

	another_column	binary_solo	column_1	float_column
0	th	True	1	0.100
1	col	False	2	0.500
2	ha	True	3	33.000
3	sti	True	4	48.000
4	ins	False	5	42.588

Εικόνα 17 Δημιουργία DataFrame με το χέρι

3.2.3 CVS αρχεία

Η δημιουργία αρχείων DataFrames από αρχεία CSV (με κόμμα διαχωρισμένα) γίνεται εξαιρετικά απλή με τη λειτουργία read_csv () στα Pandas, αφού είναι γνωστή η διαδρομή προς το αρχείο. Ένα αρχείο CSV είναι ένα αρχείο κειμένου που περιέχει δεδομένα σε μορφή πίνακα, όπου οι στήλες διαχωρίζονται με το χαρακτήρα ',' και οι γραμμές βρίσκονται σε ξεχωριστές γραμμές.

Εάν τα δεδομένα είναι σε κάποια άλλη μορφή, όπως μια βάση δεδομένων SQL ή ένα αρχείο Excel (XLS / XLSX), μπορούν να χρησιμοποιηθούν άλλες συναρτήσεις που

διαβάζονται από αυτές τις πηγές σε DataFrames, δηλαδή read_xlsx, read_sql. Ωστόσο, για απλότητα, μερικές φορές η εξαγωγή δεδομένων απευθείας στο CSV και η χρήση τους, είναι προτιμότερη.

Σε αυτό το παράδειγμα, φορτώνονται τα δεδομένα παραγωγής της Global Food από ένα αρχείο CSV που έχει ληφθεί από την ιστοσελίδα του διαγωνισμού Data Science, Kaggle. Τα δεδομένα ανοίγονται πρώτα σε Excel για μια προεπισκόπηση:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	Y1961	Y1962	Y1963
2	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	1928	1904	1666
3	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183	183	182
4	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76	76	76
5	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237	237	237
6	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210	210	214
7	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	403	403	410
8	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	17	18	19
9	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	0	0	0
10	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	111	97	103
11	AF	2	Afghanistan	2536	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	45	45	45
12	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	0	0	0
13	AF	2	Afghanistan	2542	Sugar (Raw Equivalent)	5142	Food	1000 tonnes	33.94	67.71	45	41	43
14	AF	2	Afghanistan	2543	Sweeteners, Other	5142	Food	1000 tonnes	33.94	67.71	0	0	0
15	AF	2	Afghanistan	2745	Honey	5142	Food	1000 tonnes	33.94	67.71	2	2	2
16	AF	2	Afghanistan	2549	Pulses, Other and products	5521	Feed	1000 tonnes	33.94	67.71	1	1	1
17	AF	2	Afghanistan	2549	Pulses, Other and products	5142	Food	1000 tonnes	33.94	67.71	15	16	17
18	AF	2	Afghanistan	2551	Nuts and products	5142	Food	1000 tonnes	33.94	67.71	2	3	1
19	AF	2	Afghanistan	2560	Coconuts - Incl Copra	5142	Food	1000 tonnes	33.94	67.71	0	0	0
20	AF	2	Afghanistan	2561	Sesame seed	5142	Food	1000 tonnes	33.94	67.71	10	10	10
21	AF	2	Afghanistan	2563	Olives (including preserved)	5142	Food	1000 tonnes	33.94	67.71	0	0	0
22	AF	2	Afghanistan	2571	Soyabean Oil	5142	Food	1000 tonnes	33.94	67.71	0	0	0
23	AF	2	Afghanistan	2572	Groundnut Oil	5142	Food	1000 tonnes	33.94	67.71	0	0	0
24	AF	2	Afghanistan	2573	Sunflowerseed Oil	5142	Food	1000 tonnes	33.94	67.71	3	3	3
25	AF	2	Afghanistan	2574	Rape and Mustard Oil	5142	Food	1000 tonnes	33.94	67.71	0	0	0
26	AF	2	Afghanistan	2575	Cottonseed Oil	5142	Food	1000 tonnes	33.94	67.71	4	6	9
27	AF	2	Afghanistan	2577	Palm Oil	5142	Food	1000 tonnes	33.94	67.71	0	0	0
28	AF	2	Afghanistan	2579	Sesameseed Oil	5142	Food	1000 tonnes	33.94	67.71	2	2	2

Εικόνα 18 Τα δείγματα δεδομένων για αυτήν την ανάρτηση αποτελούνται από πληροφορίες για την παγκόσμια παραγωγή τροφίμων από το 1961 έως το 2013. Εδώ το αρχείο CSV εξετάζεται στο Microsoft Excel.

Τα δείγματα δεδομένων περιέχουν 21.478 σειρές δεδομένων, με κάθε σειρά να αντιστοιχεί σε μια πηγή τροφής από μια συγκεκριμένη χώρα. Οι πρώτες 10 στήλες αντιπροσωπεύουν πληροφορίες σχετικά με τη χώρα δειγματοληψίας και τον τύπο τροφίμων / ζωοτροφών και οι υπόλοιπες στήλες αντιπροσωπεύουν την παραγωγή τροφίμων για κάθε έτος από το 1963 έως το 2013 (63 στήλες συνολικά).

3.2.3.1 *Διάβασμα και έξοδος αρχείου CSV*

Για να διαβαστεί το αρχείο διαβάζεται το όνομα με το οποίο το αρχείο είναι αποθηκευμένο στον υπολογιστή και χρησιμοποιείται το encoding που είναι ο τρόπος για να γίνει η ερμηνεία των bytes.

```
df = pd.read_csv("FAO+database.csv", encoding='latin1')
```

Χρησιμοποιώντας το Jupiter Notebook, η έξοδος είναι το όνομα του DataFrame. Η εκτύπωση είναι ένας βολικός τρόπος για την προεπισκόπηση των δεδομένων που έχουν

φορτωθεί, επιβεβαιώνεται ότι τα ονόματα των στηλών έχουν εισαχθεί σωστά, ότι οι μορφές δεδομένων είναι όπως αναμένεται και αν υπάρχουν τιμές που λείπουν οπουδήποτε.

```
In [39]: import pandas as pd
         df=pd.read_csv("FAO+database.csv", encoding='latin1')

         df
```

Out[39]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
5	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	...	231.0	67.0	82.0	67.0	69.0	71.0
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0

Εικόνα 19 Διάβασμα αρχείου CSV

Παρατηρείται ότι το Pandas εμφανίζει μόνο 20 στήλες από προεπιλογή για δεδομένα πλαισίων δεδομένων και μόνο 60 περίπου σειρές, περικόπτοντας το μεσαίο τμήμα.
Για να αλλάξουν οι σειρές ή οι στήλες που εμφανίζονται χρησιμοποιείται η :

```
In [51]: pd.set_option("display.max_rows",999) # Αλλάζει τις σειρές σε 999

         pd.get_option("display.max_rows") # Εμφανίζει τις σειρές

Out[51]: 999
```

Εικόνα 20 Εντολές set_option και get_option για αλλαγή στον αριθμό εμφάνισης και την εμφάνιση στηλών

3.2.4 Μέθοδοι shape και ndim

Η εντολή shape δίνει πληροφορίες σχετικά με το μέγεθος του συνόλου δεδομένων - το shape επιστρέφει μια πλειάδα με τον αριθμό των γραμμών και τον αριθμό των στηλών για τα δεδομένα στο DataFrame. Μια άλλη περιγραφική ιδιότητα είναι το ndim που δίνει τον αριθμό των διαστάσεων στα δεδομένα, συνήθως 2.

```
In [48]: df.shape
Out[48]: (21477, 63)
```

```
In [50]: df.ndim
Out[50]: 2
```

Εικόνα 21 Μέθοδοι shape και ndim

Τα δεδομένα παραγωγής τροφίμων περιέχουν 21.477 σειρές, το καθένα με 63 στήλες όπως φαίνεται από την έξοδο του .shape. Υπάρχουν δύο διαστάσεις - δηλ. DataFrame 2D με ύψος και πλάτος. Εάν τα δεδομένα είχαν μόνο μία στήλη, το ndim θα επέστρεφε 1. Τα σύνολα δεδομένων με περισσότερες από δύο διαστάσεις στο Pandas ονομάζονταν Πάνελ, αλλά αυτές οι μορφές έχουν καταργηθεί. Η συνιστώμενη προσέγγιση για πολυδιάστατα (> 2) δεδομένα είναι η χρήση της βιβλιοθήκης Python Xarray.

3.2.5 Μέθοδοι head και tail

Η μέθοδος DataFrame.head() στο Pandas, από προεπιλογή, δείχνει τις 5 πρώτες σειρές δεδομένων στο DataFrame. Το αντίθετο είναι το DataFrame.tail(), το οποίο σας δίνει τις τελευταίες 5 σειρές.

Δίνοντας έναν αριθμό τα Pandas θα εκτυπώσουν τον καθορισμένο αριθμό γραμμών όπως φαίνεται στο παρακάτω παράδειγμα. Το head() και το tail() αποτελούν βασικά τμήματα των λειτουργιών Python Pandas για διερεύνηση των συνόλων δεδομένων.

```
In [53]: df.head() #Εμφανίζει τις 5 πρώτες σειρές
```

Out[53]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	315.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	60.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	178.0

5 rows × 63 columns

```
In [55]: df.tail() #Εμφανίζει τις τελευταίες 5 σειρες
```

Out[55]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2(
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0	41
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	1
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	2
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 63 columns

Εικόνα 22 Εμφανίζει τις πρώτες και τις τελευταίες 5 σειρές του DataFrame

```
In [56]: df.head(20) #Εμφανίζει τις πρώτες 20 σειρές
```

Out[56]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y201
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0	4605.
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0	442.
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0	315.
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0	60.
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0	178.
5	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	...	231.0	67.0	82.0	67.0	69.0	71.0	82.
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0	14.
7	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	...	2.0	1.0	1.0	0.0	0.0	0.0	0.
8	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	...	276.0	294.0	294.0	260.0	242.0	250.0	192.
9	AF	2	Afghanistan	2536	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	...	50.0	29.0	61.0	65.0	54.0	114.0	83.
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0	0.

Εικόνα 23 Εμφανίζει, μετά από επιλογή, τις πρώτες 20 σειρές του DataFrame

3.2.6 Τύποι δεδομένων του DataFrame

Πολλά DataFrames έχουν διαφορετικούς τύπους δεδομένων, δηλαδή μερικές στήλες είναι αριθμοί, μερικές είναι συμβολοσειρές και μερικές είναι ημερομηνίες κλπ. Εσωτερικά, τα αρχεία CSV δεν περιέχουν πληροφορίες για τους τύπους δεδομένων που περιέχονται σε κάθε στήλη, όλα τα δεδομένα είναι μόνο χαρακτήρες. Το Pandas συνάγει τους τύπους δεδομένων κατά τη φόρτωση των δεδομένων, π.χ. εάν μια στήλη περιέχει μόνο αριθμούς, οι pandas θα ορίσουν τον τύπο δεδομένων αυτής της στήλης σε αριθμητικό.
Για τον τύπο κάθε στήλης χρησιμοποιείται το df.dtypes.

```
In [57]: df.dtypes #Τύποι κάθε στήλης
Out[57]: Area Abbreviation      object
         Area Code               int64
         Area                   object
         Item Code               int64
         Item                   object
         Element Code            int64
         Element                object
         Unit                   object
         latitude              float64
         longitude             float64
         Y1961                 float64
         Y1962                 float64
         Y1963                 float64
         Y1964                 float64
         Y1965                 float64
         Y1966                 float64
         Y1967                 float64
         Y1968                 float64
         Y1969                 float64
         Y1970                 float64
         Y1971                 float64
         Y1972                 float64
```

Εικόνα 24 Τύποι δεδομένων του DataFrame

Σε ορισμένες περιπτώσεις, η αυτοματοποιημένη εξαγωγή δεδομένων τύπου μπορεί να δώσει απροσδόκητα αποτελέσματα. Οι συμβολοσειρές φορτώνονται ως τύποι δεδομένων αντικειμένων, επειδή τεχνικά, το DataFrame διατηρεί έναν δείκτη στα δεδομένα του string αλλού στη μνήμη.

Για να αλλάξει τον τύπο δεδομένων μιας συγκεκριμένης στήλης, χρησιμοποιείται η συνάρτηση .astype(). Για παράδειγμα, για τη στήλη Item Code ως sting :

```
data['Item Code'].astype(str)
```

3.2.7 Περιγράφοντας τα δεδομένα με την describe

Η describe δημιουργεί περιγραφικά στατιστικά στοιχεία που συνοψίζουν την κεντρική τάση, τη διασπορά και το σχήμα της κατανομής του συνόλου δεδομένων, εξαιρουμένων των τιμών NaN. Αναλύει τόσο αριθμητικές σειρές όσο και σειρές αντικειμένων, καθώς και σύνολα στηλών DataFrame με μεικτούς τύπους δεδομένων. Η έξοδος θα ποικίλει ανάλογα με τα παρεχόμενα. Για τις αριθμητικές στήλες, η describe επιστρέφει τα βασικά στατιστικά στοιχεία: τον αριθμό των τιμών, τον μέσο όρο, την τυπική απόκλιση, το ελάχιστο, το μέγιστο και το 25ο, το 50ο και το 75ο ποσοτικό για τα δεδομένα μιας στήλης. Για τις στήλες συμβολοσειρών, η describe επιστρέφει τον αριθμό των τιμών, τον αριθμό των μοναδικών καταχωρήσεων, την πιο συχνά εμφανιζόμενη τιμή ('top') και τον αριθμό των φορών που εμφανίζεται η κορυφαία τιμή ('freq').

Για παράδειγμα,

```
In [5]: df['Y2013'].describe() #Η describe περιγράφει μία στήλη από αριθμούς, την Y2013

Out[5]: count     21477.000000
        mean        575.557480
        std        6218.379479
        min        -246.000000
        25%           0.000000
        50%           8.000000
        75%          90.000000
        max      489299.000000
        Name: Y2013, dtype: float64

In [6]: df['Area'].describe() #Η describe περιγράφει μία στήλη από συμβολοσειρές (strings), την Area

Out[6]: count     21477
        unique      174
        top       Spain
        freq        150
        Name: Area, dtype: object
```

Εικόνα 25 Η describe περιγράφει στατιστικά στοιχεία μιας στήλης με αριθμούς και μιας με strings

Αν η describe καλείται σε ολόκληρο το DataFrame, επιστρέφονται τα στατιστικά μόνο για τις στήλες με αριθμητικούς τύπους δεδομένων και στη μορφή DataFrame.

```
In [8]: df.describe()
```

Out[8]:

	Area Code	Item Code	Element Code	latitude	longitude	Y1961	Y1962	Y1963	Y1964	Y1965	...
count	21477.000000	21477.000000	21477.000000	21477.000000	21477.000000	17938.000000	17938.000000	17938.000000	17938.000000	17938.000000	...
mean	125.449411	2694.211529	5211.687154	20.450613	15.794445	195.262069	200.782250	205.464600	209.925577	217.556751	...
std	72.868149	148.973406	146.820079	24.628336	66.012104	1864.124336	1884.265591	1861.174739	1862.000116	2014.934333	...
min	1.000000	2511.000000	5142.000000	-40.900000	-172.100000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	63.000000	2561.000000	5142.000000	6.430000	-11.780000	0.000000	0.000000	0.000000	0.000000	0.000000	...
50%	120.000000	2640.000000	5142.000000	20.590000	19.150000	1.000000	1.000000	1.000000	1.000000	1.000000	...
75%	188.000000	2782.000000	5142.000000	41.150000	46.870000	21.000000	22.000000	23.000000	24.000000	25.000000	...
max	276.000000	2961.000000	5521.000000	64.960000	179.410000	112227.000000	109130.000000	106356.000000	104234.000000	119378.000000	...

8 rows × 58 columns

Εικόνα 26 Επιστροφή στατιστικών στοιχείων με αριθμητικούς τύπους δεδομένων

Με την χρήση της επόμενης εντολής επιστρέφονται τα στατιστικά από όλες τις στήλες.

```
df.descibe(include='all')
```

```
In [7]: df.describe(include='all')
```

Out[7]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2(
count	21356	21477.000000	21477	21477.000000	21477	21477.000000	21477	21477	21477.000000	21477.000000	...	21128.000000	21128.000(
unique	173	NaN	174	NaN	115	NaN	2	1	NaN	NaN	...	NaN	N
top	ES	NaN	Spain	NaN	Milk - Excluding Butter	NaN	Food	1000 tonnes	NaN	NaN	...	NaN	N
freq	150	NaN	150	NaN	558	NaN	17528	21477	NaN	NaN	...	NaN	N
mean	NaN	125.449411	NaN	2694.211529	NaN	5211.687154	NaN	NaN	20.450613	15.794445	...	486.690742	493.153
std	NaN	72.868149	NaN	148.973406	NaN	146.820079	NaN	NaN	24.628336	66.012104	...	5001.782008	5100.057(
min	NaN	1.000000	NaN	2511.000000	NaN	5142.000000	NaN	NaN	-40.900000	-172.100000	...	0.000000	0.000(
25%	NaN	63.000000	NaN	2561.000000	NaN	5142.000000	NaN	NaN	6.430000	-11.780000	...	0.000000	0.000(
50%	NaN	120.000000	NaN	2640.000000	NaN	5142.000000	NaN	NaN	20.590000	19.150000	...	6.000000	6.000(
75%	NaN	188.000000	NaN	2782.000000	NaN	5142.000000	NaN	NaN	41.150000	46.870000	...	75.000000	77.000(
max	NaN	276.000000	NaN	2961.000000	NaN	5521.000000	NaN	NaN	64.960000	179.410000	...	360767.000000	373694.000(

11 rows × 63 columns

Εικόνα 27 Περιγραφή στατιστικών στοιχείων όλων των στηλών του πίνακα

3.2.8 Συλλογή και διαχείριση δεδομένων

3.2.8.1 Επιλογή στήλης

Υπάρχουν τρεις κύριες μέθοδοι επιλογής στηλών σε pandas:

1. χρησιμοποιώντας το όνομα της στήλης, π.χ. data.column_name,
2. χρησιμοποιώντας το όνομα της στήλης ως συμβολοσειρά, π.χ. data ['column_name'],
3. χρησιμοποιώντας αριθμητικό indexing και τον επιλογέα data.iloc[:, column_number]. Το "iloc" σε pandas χρησιμοποιείται για την επιλογή σειρών και στηλών ανά αριθμό, με τη σειρά που εμφανίζονται στο πλαίσιο δεδομένων. Δηλαδή, κάθε σειρά έχει έναν αριθμό γραμμής από το 0 έως το σύνολο των σειρών και το iloc επιτρέπει επιλογές με βάση αυτούς τους αριθμούς. Το ίδιο ισχύει και για τις στήλες.

Για παράδειγμα και με την χρήση του head() για την εμφάνιση των 5 πρώτων.

```
In [21]: df.Area.head()
Out[21]: 0    Afghanistan
         1    Afghanistan
         2    Afghanistan
         3    Afghanistan
         4    Afghanistan
         Name: Area, dtype: object

In [22]: df['Area'].head()
Out[22]: 0    Afghanistan
         1    Afghanistan
         2    Afghanistan
         3    Afghanistan
         4    Afghanistan
         Name: Area, dtype: object
```

```
In [24]: df.iloc[:,2].head()
Out[24]: 0    Afghanistan
         1    Afghanistan
         2    Afghanistan
         3    Afghanistan
         4    Afghanistan
         Name: Area, dtype: object

In [26]: df.iloc[:,2:4].head() # Εμφάνιση των στηλών 3 και 4
Out[26]:
```

	Area	Item Code
0	Afghanistan	2511
1	Afghanistan	2805
2	Afghanistan	2513
3	Afghanistan	2513
4	Afghanistan	2514

Εικόνα 28 Κλήση στηλών με τους 3 διαφορετικούς τρόπους

Όταν επιλέγεται μια στήλη χρησιμοποιώντας οποιαδήποτε από αυτές τις μεθοδολογίες, ένα pandas.Series είναι ο τύπος δεδομένων που προκύπτει. Μια σειρά pandas είναι ένα μονοδιάστατο σύνολο δεδομένων. Είναι χρήσιμες οι βασικές λειτουργίες που μπορούν να εκτελεστούν σε αυτές τις σειρές δεδομένων, συμπεριλαμβανομένων της άθροισης (.sum()), του μέσου όρου (.mean()), της μέτρησης (.count()), της επιλογής του μεσαίου στοιχείου (.median()), και της αντικατάστασης των τιμών που λείπουν (.fillna (new_value)).

```
In [28]: #Γίνεται χρήση της στήλης 'Y2007'
         [df['Y2007'].sum(),    #Άθροισμα από τις τιμές της στήλης
          df['Y2007'].mean(),   #Μέσος όρος των τιμών της στήλης
          df['Y2007'].median(), #Μέσο των τιμών της στήλης
          df['Y2007'].nunique(), #Αριθμός με μοναδική είσοδο
          df['Y2007'].min(),    #Το μικρότερο των τιμών της στήλης
          df['Y2007'].max()]    #Το μεγαλύτερο των τιμών της στήλης

Out[28]: [10867788.0, 508.48210358863986, 7.0, 1994, 0.0, 402975.0]
```

Εικόνα 29 Βασικές λειτουργίες που εκτελούνται σε ένα Series

3.2.8.2 *Επιλογή σειράς*

Οι γραμμές σε ένα DataFrame επιλέγονται, συνήθως, χρησιμοποιώντας τις μεθόδους επιλογής iloc / loc, ή χρησιμοποιώντας λογικούς επιλογείς (επιλέγοντας με βάση την τιμή μιας άλλης στήλης ή μεταβλητής).

Οι βασικές μέθοδοι είναι:

1. αριθμητική επιλογή σειράς χρησιμοποιώντας το iloc, π.χ. data.iloc [0:10,:] - επιλογή των πρώτων 10 σειρών,
2. επιλογή της σειράς με βάση την ετικέτα χρησιμοποιώντας το loc και αυτό ισχύει μόνο αν έχει οριστεί ένας index στο πλαίσιο δεδομένων, π.χ. data.loc [44,:],
3. επιλογή σειράς χρησιμοποιώντας δηλώσεις, π.χ. [data ["Περιοχή"] == "Ιρλανδία")] - επιλέγονται οι σειρές όπου η τιμή περιοχής είναι «Ιρλανδία».

```
In [32]: df.loc[44,:]

Out[32]: Area Abbreviation            AF
         Area Code                     2
         Area                Afghanistan
         Item Code                  2655
         Item                       Wine
```

```
In [37]: df.iloc[2,:]

Out[37]: Area Abbreviation                    AF
         Area Code                             2
         Area                         Afghanistan
         Item Code                           2513
         Item                 Barley and products
```

```
In [36]: df[df['Area']=='Ireland']
```

Out[36]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y201
9533	IE	104	Ireland	2511	Wheat and products	5521	Feed	1000 tonnes	53.41	-8.24	...	968.0	976.0	902.0	685.0	1063.0	804.0	783.
9534	IE	104	Ireland	2511	Wheat and products	5142	Food	1000 tonnes	53.41	-8.24	...	395.0	423.0	501.0	449.0	470.0	493.0	512.
9535	IE	104	Ireland	2805	Rice (Milled Equivalent)	5521	Feed	1000 tonnes	53.41	-8.24	...	3.0	3.0	3.0	4.0	5.0	4.0	4.
9536	IE	104	Ireland	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	53.41	-8.24	...	11.0	6.0	6.0	9.0	14.0	15.0	16.
9537	IE	104	Ireland	2513	Barley and products	5521	Feed	1000 tonnes	53.41	-8.24	...	993.0	908.0	1047.0	904.0	1242.0	1290.0	1283.

Εικόνα 30 Κλήση γραμμών με τους 3 διαφορετικούς τρόπους

Οι μέθοδοι επιλογής για στήλες και σειρές μπορούν να συνδοιαστούν με πολλούς τρόπους για να επιτύχει το επιθυμητό αποτέλεσμα.

```
In [38]: df.iloc[[2,4,9], [1,3,5]] #Γραμμή 3,5,10 και στήλη 2,4,6
```

Out[38]:

	Area Code	Item Code	Element Code
2	2	2513	5521
4	2	2514	5521
9	2	2536	5521

```
In [48]: df.loc[45:49, ['Y2010', 'Y2012']] #Γραμμή με όνομα 45,46,47,48,49 και στήλη με όνομα Y2010 και Y2012
```

Out[48]:

	Y2010	Y2012
45	3.0	3
46	0.0	0
47	133.0	153
48	157.0	155
49	67.0	47

```
In [47]: df.iloc[5:9, 10:15] #Από γραμμή 6 εώς 9 και από στήλη 11 εώς 15
```

Out[47]:

	Y1961	Y1962	Y1963	Y1964	Y1965
5	403.0	403.0	410.0	415.0	415.0
6	17.0	18.0	19.0	20.0	21.0
7	0.0	0.0	0.0	0.0	0.0
8	111.0	97.0	103.0	110.0	113.0

Εικόνα 31 Συνδυασμός κλήσης γραμμών και στηλών

3.2.8.3 *Boolean indexing*

Το df.loc[selection] είναι η πιο κοινή μέθοδος που χρησιμοποιείται με τα Pandas DataFrames για το boolean indexing. Με το boolean indexing περνιέται ένας πίνακας ή μια σειρά(Series) με τιμές True/False στο index.loc για να επιλεγούν οι γραμμές όπου το Series έχει True τιμές. Ένα δεύτερο επιχείρημα μπορεί να μεταβιβαστεί στο loc για να επιλεγούν συγκεκριμένες στήλες έξω από το πλαίσιο δεδομένων. Και πάλι, οι στήλες αναφέρονται με το όνομα για τον index του loc και μπορεί να είναι μια μονή συμβολοσειρά, μια λίστα με στήλες ή μια λειτουργία slice.

```
In [89]: #Επιλογή των γραμμών με το όνομα 'Ireland' και των στηλών μεταξύ του 'Item' και του 'Element'
         df.loc[(df['Area'] == 'Ireland'), 'Item':'Element']
```

Out[89]:

	Item	Element Code	Element
9533	Wheat and products	5521	Feed
9534	Wheat and products	5142	Food
9535	Rice (Milled Equivalent)	5521	Feed
9536	Rice (Milled Equivalent)	5142	Food

```
In [88]: #Το  παρακάτω δεν εμφανίζει κανένα στοιχείο αφού δεν υπάρχουν γραμμές που να συνδοιάζουν και τα δυο κομμάτια
         df.loc[(df['Area'] == 'Ireland') & ('Area Code' == 2), 'Item':'Element']
```

Out[88]:

Item	Element Code	Element

Εικόνα 32 Boolean indexing

3.2.8.4 *Διαγραφή σειρών και στηλών*

Για τη διαγραφή σειρών και στηλών από το DataFrames, το Pandas χρησιμοποιεί τη συνάρτηση "drop". Για τη διαγραφή μιας στήλης ή πολλών στηλών, χρησιμοποιείται το όνομα της στήλης και ορίζεται ο "axis" ως 1. Εναλλακτικά, όπως στο παρακάτω παράδειγμα, η παράμετρος "columns" έχει προστεθεί στο Pandas, το οποίο κόβει την ανάγκη για "axis". Η συνάρτηση drop επιστρέφει ένα νέο DataFrame, με τις στήλες να καταργούνται. Για την επεξεργασία του αρχικού DataFrame, η παράμετρος "inplace" μπορεί να οριστεί σε True και δεν υπάρχει επιστρεφόμενη τιμή.

```
In [49]: # Διαγραφή στηλών
         # Διαγραφή της στήλης 'Area' από το DataFrame
         df = df.drop("Area", axis=1)
         # Διαγραφή της στήλης 'Area' με διαφορετικό τρόπο
         df = df.drop(columns="area")
         # Διαγραφή της στήλης 'Area' με το inplace
         df.drop("Area", axis=1, inplace=True).
         # Διαγραφή πολλών στηλών 'Y2001', 'Y2002', 'Y2003'
         df = df.drop(["Y2001", "Y2002", "Y2003"], axis=1)
```

Εικόνα 33 Διαγραφή στηλών

Οι σειρές μπορούν επίσης να διαγραφούν χρησιμοποιώντας τη συνάρτηση "drop", καθορίζοντας τον axis = 0, είτε χρησιμοποιώντας κάποια ετικέτα, είτε, τέλος, χρησιμοποιώντας το iloc που προσδιορίζει τις σειρές με τον index.

```
In [55]: # Διαγραφή των σειρών με ετικέτα 0, 1, 2
         df = df.drop([0,1,2], axis=0)
         # Διαγραφή των γραμμών με την ετικέτα 'Ireland'
         # Πριν την διαγραφή, επισημαίνεται ο δείκτης 'Area'
         df = df.set_index("Area")
         df = df.drop("Ireland", axis=0).
         # Διαγραφή των πρώτων 5 γραμμών χρησιμοποιώντας το iloc
         df = df.iloc[5:,]
```

Εικόνα 34 Διαγραφή σειρών

3.2.8.5 *Μετονομασία στηλών*

Οι μετονομαζόμενες στήλες επιτυγχάνονται εύκολα σε Pandas χρησιμοποιώντας τη rename του DataFrame. Η rename είναι εύκολη στη χρήση και αρκετά ευέλικτη. Η χρήση της γίνεται με δύο τρόπους :

1. Μετονομασία με ανάθεση παλαιών ονομάτων σε νέα ονόματα χρησιμοποιώντας τη μορφή {"old_column_name": "new_column_name", ...}
2. Μετονομασία παρέχοντας μια συνάρτηση με την οποία αλλάζουν τα ονόματα των στηλών. Οι συναρτήσεις εφαρμόζονται σε κάθε όνομα στήλης.

```
In [58]: # Μετονομασία της στήλης Area σε 'place_name'
df = df.rename(columns={"Area": "place_name"})
# Με την χρήση του inplace
df.rename(columns={"Area": "place_name"}, inplace=True)
# Μετονομασία περισσότερων στηλών
df.rename(
    columns={
        "Area": "place_name",
        "Y2009": "year_2009"
    },
    inplace=True
)
# Μετονομασία στήλης με την χρήση συνάρτησης για παράδειγμα μετατροπή όλων των στηλών σε πεζά γράμματα
df.rename(columns=str.lower)
```

Out[58]:

	area abbreviation	area code	item code	item	element code	element	unit	latitude	longitude	y1961	...	y2003	y2005	y2006	y2007	y2008	year_2009
0	AF	2	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	1928.0	...	3095.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	183.0	...	460.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	76.0	...	48.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	237.0	...	144.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	210.0	...	63.0	208.0	233.0	249.0	247.0	195.0
5	AF	2	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	403.0	...	117.0	67.0	82.0	67.0	69.0	71.0
6	AF	2	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	17.0	...	16.0	21.0	11.0	19.0	21.0	18.0

Εικόνα 35 Μετονομασία στηλών

3.2.9 Ταξινόμηση στο DataFrame (Sorting)

Η μέθοδος sort_index χρησιμοποιείται για ταξινόμηση στο DataFrame. Ο axis είναι 0 ή 1 ανάλογα αν η ταξινόμηση θα γίνει σε σειρά ή στήλη αντίστοιχα.

```
In [97]: df.sort_index(axis=1, ascending=False).head(4)
```

Out[97]:

	longitude	latitude	Y2013	Y2012	Y2011	Y2010	Y2009	Y2008	Y2007	Y2006	...	Y1962	Y1961	Unit	Item Code	Item	Element Code	Element	Area Code	Ab
0	67.71	33.94	4895	4810	4711.0	4605.0	4538.0	4252.0	4164.0	3704.0	...	1904.0	1928.0	1000 tonnes	2511	Wheat and products	5142	Food	2	
1	67.71	33.94	422	425	476.0	442.0	415.0	490.0	455.0	546.0	...	183.0	183.0	1000 tonnes	2805	Rice (Milled Equivalent)	5142	Food	2	
2	67.71	33.94	360	367	203.0	315.0	379.0	230.0	263.0	262.0	...	76.0	76.0	1000 tonnes	2513	Barley and products	5521	Feed	2	
3	67.71	33.94	89	78	72.0	60.0	55.0	62.0	48.0	44.0	...	237.0	237.0	1000 tonnes	2513	Barley and products	5142	Food	2	

4 rows × 63 columns

```
In [98]: df.sort_index(axis=0, ascending=False).head(4)
```

Out[98]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2(
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0	
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0	2
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0	1

4 rows × 63 columns

Εικόνα 36 Ταξινόμηση με βάση τις στήλες ή με βάση τις γραμμές

Η ταξινόμηση μπορεί να γίνει και με μία στήλη ή περισσότερες :

```
In [99]: df.sort_values(by=['Area', 'Item'], ascending=False).head(4)
```

Out[99]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y201(
21424	ZW	181	Zimbabwe	2655	Wine	5142	Food	1000 tonnes	-19.02	29.15	...	2.0	2.0	2.0	2.0	3.0	3.0	4.(
21356	ZW	181	Zimbabwe	2511	Wheat and products	5142	Food	1000 tonnes	-19.02	29.15	...	248.0	263.0	329.0	364.0	371.0	411.0	380.(
21406	ZW	181	Zimbabwe	2605	Vegetables, Other	5142	Food	1000 tonnes	-19.02	29.15	...	144.0	149.0	186.0	162.0	114.0	156.0	181.(
21462	ZW	181	Zimbabwe	2918	Vegetables	5142	Food	1000 tonnes	-19.02	29.15	...	161.0	166.0	208.0	185.0	137.0	179.0	215.(

4 rows × 63 columns

Εικόνα 37 Ταξινόμηση για περισσότερες από μία στήλες

3.2.10 Χαμένες τιμές

Η μέθοδος dropna αφαιρεί τις χαμένες τιμές από το DataFrame.

```
In [100]: df_drop_missing = df.dropna()

In [101]: df_drop_missing
```

21471	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5521	Feed	1000 tonnes	-19.02	29.15	...	21.0	21.0	21.0	21.0	21.0	23.0
21472	ZW	181	Zimbabwe	2948	Milk - Excluding Butter	5142	Food	1000 tonnes	-19.02	29.15	...	373.0	357.0	359.0	356.0	341.0	385.0
21473	ZW	181	Zimbabwe	2960	Fish, Seafood	5521	Feed	1000 tonnes	-19.02	29.15	...	5.0	4.0	9.0	6.0	9.0	5.0
21474	ZW	181	Zimbabwe	2960	Fish, Seafood	5142	Food	1000 tonnes	-19.02	29.15	...	18.0	14.0	17.0	14.0	15.0	18.0
21475	ZW	181	Zimbabwe	2961	Aquatic Products, Other	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0
21476	ZW	181	Zimbabwe	2928	Miscellaneous	5142	Food	1000 tonnes	-19.02	29.15	...	0.0	0.0	0.0	0.0	0.0	0.0

17817 rows × 63 columns

Εικόνα 38 Αφαίρεση χαμένων τιμών από το DataFrame

Το αποτέλεσμα είναι να εμφανίζονται οι σειρές που δεν έχουν κανένα χαμένο στοιχείο. Το αντίστοιχο μπορεί να γίνει και για τις στήλες, για τις στήλες και τις σειρές μαζί ή και για συγκεκριμένα στοιχεία.

```
In [ ]: df.dropna(1) #Εμφανίζει στις στήλες που δεν έχουν χαμένα στοιχεία
        df.dropna(how=all) #Εμφανίζει τον συνδοιασμό στηλών και γραμμών
        df.dropna(subset=['Area', 'Item']) #Εμφανίζει για τις συγκεκριμένες στήλες αν υπάρχουν χαμένες τιμές
```

Εικόνα 39 Τρόποι χρήσης της dropna

Αντίστοιχα, υπάρχει η μέθοδος fillna για να γεμίζει τις τιμές NA/NaN. Τέλος, υπάρχουν οι μέθοδοι isna και notna, για τον εντοπισμό των χαμένων τιμών ή των μη-χαμένων τιμών αντίστοιχα. Επιστρέφουν True αν βρουν την τιμή που αναζητούν ή False αν δεν την βρουν.

```
In [137]: df.isna()
```

Out[137]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009	Y2010	Y2011
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False

Εικόνα 40 Τρόπος χρήσης της isna

3.2.11 Λοιπές συναρτήσεις

Τα DataFrames έχουν πολλές ακόμα συναρτήσεις, όπως την duplicated που ελέγχει αν υπάρχουν ίδιες γραμμές ή στήλες και με την χρήση της drop τις αφαιρεί, ή και την merge και την concat, οι οποίες ενώνουν και συγχωνεύουν DataFrames αντίστοιχα.

3.3 Matplotlib

Τέλος, σε αυτό το κεφάλαιο θα γίνει μια μικρή αναφορά στη Matplotlib, η οποία είναι μια βιβλιοθήκη της python που παρέχει δυνατότητες σχεδίασης γραφικών παραστάσεων υψηλής ποιότητας. Για την εγκατάστασή του

```
pip install matplotlib
```

Η pyplot είναι ένα τμήμα της Matplotlib που αποτελεί μια συλλογή εντολών με λειτουργίες παρόμοιες με αυτές του Matlab.

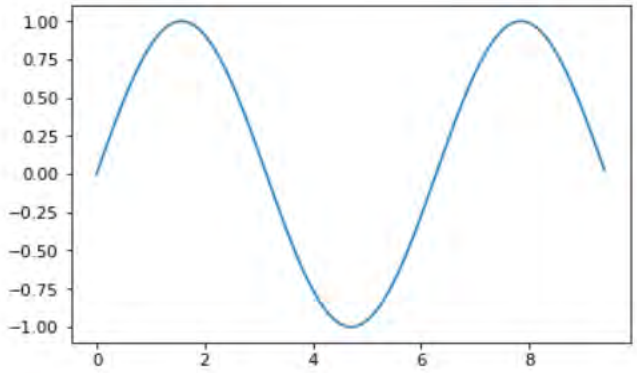
```
import matplotlib.pyplot as plt
```

Υπάρχει μια τελευταία διαμόρφωση για να ολοκληρωθεί πριν εμφανιστούν plots. Πρέπει να επισημανθεί στο jupyter να τα εμφανίσει ως εικόνες στο ίδιο το notebook. Για να γίνει αυτό χρησιμοποιείται η ακόλουθη εντολή σε ένα από τα cells notebook:

```
%matplotlib inline
```

Για παράδειγμα,

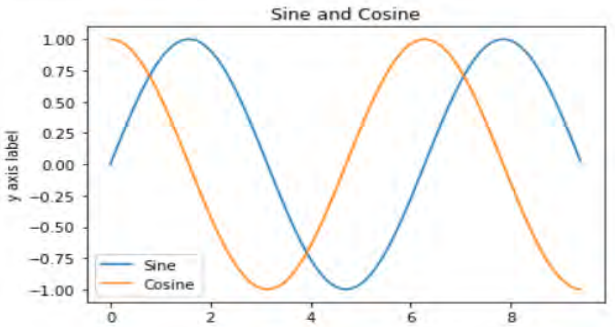
```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

        # Υπολογίζει τις συντεταγμένες x και y για σημεία σε καμπύλη ημιτονοειδούς
        x = np.arange(0, 3 * np.pi, 0.1)
        y = np.sin(x)

        # Με τη χρήση του matplotlib γίνεται το διάγραμμα
        plt.plot(x, y)
        plt.show()  # Πρέπει να καλεστεί η plt.show() για να εμφανιστεί
```



Εικόνα 41 Παράδειγμα χρήσης Matplotlib

Με λίγη επιπλέον εργασία μπορούν εύκολα να σχεδιαστούν πολλές γραμμές ταυτόχρονα και να προστεθεί μια ετικέτα legend (συνοπτική εξήγηση των συμβόλων), τίτλου και άξονα:

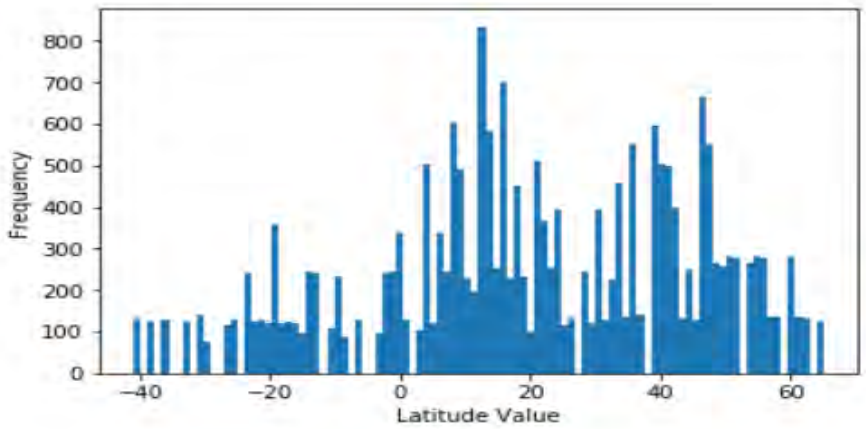
```
In [7]: import numpy as np
        import matplotlib.pyplot as plt

        # Υπολογίζει τις συντεταγμένες x και y για σημεία σε καμπύλες sine και cosine
        x = np.arange(0, 3 * np.pi, 0.1)
        y_sin = np.sin(x)
        y_cos = np.cos(x)

        # Γίνεται plot με την χρήση του matplotlib
        plt.plot(x, y_sin)
        plt.plot(x, y_cos)
        plt.xlabel('x axis label')
        plt.ylabel('y axis label')
        plt.title('Sine and Cosine')
        plt.legend(['Sine', 'Cosine'])
        plt.show()
```



Εικόνα 42 Σχεδιασμός πολλών διαφορετικών γραμμών με το Matplotlib

Με τη χρήση του παραδείγματος που χρησιμοποιήθηκε για την ανάλυση του Pandas, δημιουργείται ένα ιστόγραμμα που εμφανίζει την κατανομή των αξόνων γεωγραφικού πλάτους στο σύνολο δεδομένων.

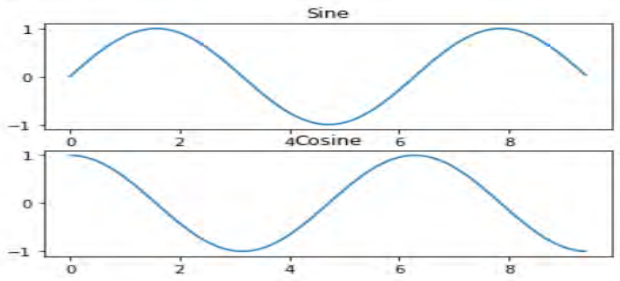
```
In [10]: df['latitude'].plot(kind='hist', bins=100)
         plt.xlabel('Latitude Value')
Out[10]: Text(0.5,0,'Latitude Value')
```



Εικόνα 43 Δημιουργία ιστογράμματος (kind= 'hist') στη βάση δεδομένων που χρησιμοποιήθηκε στην ανάλυση του Pandas

Τέλος, μπορούν να σχεδιαστούν διαφορετικά πράγματα στο ίδιο σχήμα χρησιμοποιώντας τη συνάρτηση subplot.

```
In [12]: # Υπολογίζει τις συντεταγμένες x και y για σημεία σε καμπύλες sine και cosine
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Ορίζει ένα subplot grid που έχει ύψος 2 και πλάτος 1,
# και ορίζει το πρώτο τέτοιο subplot ως ενεργό.
plt.subplot(2, 1, 1)

# Κάνει το πρώτο plot
plt.plot(x, y_sin)
plt.title('Sine')

# Ορίζει το δεύτερο subplot ως ενεργό και κάνει το δεύτερο plot
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('cosine')

# Εμφανίζει την εικόνα
plt.show()
```



Εικόνα 44 Χρήση του subplot

ΚΕΦΑΛΑΙΟ 4:
ΣΥΓΚΡΙΣΗ PYTHON – R

Η γλώσσα R, μαζί με την Python, είναι η πιο δημοφιλείς γλώσσες προγραμματισμού για την ανάλυση δεδομένων. Παρόλο που η Python συχνά επαινείται ως γλώσσα γενικού σκοπού με μια σύντομη κατανόηση, η λειτουργικότητα της R αναπτύχθηκε με γνώμονα τους στατιστικολόγους και την επιστήμη των δεδομένων, που την κάνει αυτόματα ιδανική για αυτό.

Παρόλα αυτά υπάρχουν τα θετικά και τα αρνητικά και για τις δύο. Και οι δύο είναι γλώσσες είναι ανοικτού κώδικα, έχουν μεγάλη ποικιλία προηγμένων εργαλείων, έχουν εξαιρετικά υποστηρικτικές αναπτυξιακές κοινότητες και προσφέρουν αξιόπιστες και καλές ευκαιρίες σταδιοδρομίας.

Η R έχει μια μακρά και αξιόπιστη ιστορία και μια ισχυρή κοινότητα υποστήριξης στον κλάδο των δεδομένων. Μαζί, αυτά τα γεγονότα σημαίνουν ότι κάποιος μπορεί να βασιστεί στην ηλεκτρονική υποστήριξη από άλλους στον τομέα εάν χρειαστεί βοήθεια ή έχει ερωτήσεις σχετικά με τη χρήση της γλώσσας. Επιπλέον, υπάρχουν πολλά δημοσιευμένα πακέτα, περισσότερα από 5.000 στην πραγματικότητα, τα οποία μπορεί να μεταφορτώσει για χρήση σε συνδυασμό με την R για να επεκτείνει τις δυνατότητές του σε νέα ύψη. Αυτό κάνει την R τόσο καλή για τη διεξαγωγή σύνθετης διερευνητικής ανάλυσης δεδομένων. Η R ενσωματώνεται επίσης καλά με άλλες γλώσσες υπολογιστών όπως C ++, Java και C. Δύο από τα μεγαλύτερα μειονεκτήματα του R είναι ότι η γλώσσα είναι αργή και έχει μια εξαιρετικά απότομη καμπύλη μάθησης. Επειδή η R αναπτύχθηκε εξ ολοκλήρου από τους στατιστικολόγους και τους επιστήμονες δεδομένων, έχει περισσότερο περίπλοκο και συγκεκριμένο σκοπό.

Όταν χρειάζεται να γίνει βαριά στατιστική ανάλυση ή γραφική απεικόνιση, η R είναι η καλύτερη επιλογή. Οι κοινές μαθηματικές λειτουργίες, όπως ο πολλαπλασιασμός των πινάκων, δουλεύουν κατ 'ευθείαν έξω από το κιβώτιο και η συντακτική παράσταση της γλώσσας διευκολύνει τη μετάφραση από το μαθηματικό σε κώδικα, ειδικά για κάποιον που δεν έχει καθόλου ή έχει ελάχιστο υπόβαθρο προγραμματισμού.

Από την άλλη, όπως αναφέρθηκε και παραπάνω η Python είναι μια γενική γλώσσα προγραμματισμού που μπορεί να κάνει ό, τι χρειάζεται για να κάνει munging δεδομένα, για την μηχανική δεδομένων, για να κάνει wrangling τα δεδομένα και πολλά άλλα. Είναι απλούστερο να αποκτήσει εμπειρία κάποιος από την R αν έχει μάθει προηγουμένως μια αντικειμενοστραφή γλώσσα προγραμματισμού όπως Java ή C ++.
Επιπλέον, επειδή η Python είναι αντικειμενοστραφής γλώσσα προγραμματισμού, είναι ευκολότερο να γραφτούν μεγάλης κλίμακας, διατηρήσιμο και εύρωστο κώδικα μαζί της, παρά με την R.

Παρόλο που η Python δεν διαθέτει τόσο ολοκληρωμένο σύνολο πακέτων και βιβλιοθηκών που διατίθενται στους επαγγελματίες του τομέα δεδομένων, όπως η R, ο συνδυασμός της Python με εργαλεία όπως Pandas, Numpy, Scipy, Scikit-Learn και Seaborn την κάνει να φτάνει στο επιθυμητό σημείο. Η γλώσσα γίνεται επίσης σιγά-σιγά πιο χρήσιμη για εργασίες όπως η μηχανική μάθηση.

Σε γενικές γραμμές, η επιλογή της γλώσσας που θα χρησιμοποιήσει κάποιος γίνεται αποκλειστικά απ΄αυτόν. Για την ανάλυση των δεδομένων, οι διαφορές μεταξύ του R και του Python αρχίζουν να καταρρέουν. Τα περισσότερα από τα κοινά καθήκοντα που έχουν συσχετιστεί με ένα πρόγραμμα ή με το άλλο είναι πλέον δυνατά και στα δύο. Είναι αρκετά παρόμοιες, στην πραγματικότητα, οπότε όποια και από τις δύο γλώσσες επιλεγεί αρχικά, μπορεί στην συνέχεια να γίνει και εκμάθηση της άλλης.

ΚΕΦΆΛΑΙΟ 5:
ΣΥΜΠΕΡΑΣΜΑΤΑ

Όσοι πρωτοεμφανίζονται στον κόσμο της επιστήμης των δεδομένων, είναι καλύτερο να εξοικειωθούν πρώτα με μια γλώσσα προγραμματισμού. Οι διαδικασίες και οι τεχνικές μίας γλώσσας είναι οι σημαντικότερες και κατανοώντας αυτές κάποιος θα έχει μία ισχυρή βάση στην ανάλυση των δεδομένων, πριν εντρυφήσει βαθύτερα σ΄αυτήν. Αφού όμως αποκτήσει την ισχυρή βάση η μετάβαση σε άλλες γλώσσες θα είναι πολύ εύκολη με αρχικό πάντα γνώμονα την πρώτη γλώσσα προγραμματισμού. Υπάρχουν πολλές γλώσσες προγραμματισμού που μπορεί να επιλεγούν για αυτό, αλλά η Python είναι αδιαμφισβήτητα πρόδομος για την πλειονότητα του κόσμου της ανάλυσης δεδομένων.

Η Python είναι μία γλώσσα με πολλά οφέλη και πολλές δυνατότητες. Αυτό έχει σαν αποτέλεσμα όλο και περισσότεροι άνθρωποι να την χρησιμοποιούν ως πρώτη επιλογή για την δουλειά τους.

Αρχικά, είναι μία γλώσσα ανοιχτού κώδικα, πολύ εύκολη στη χρήση και στην εκμάθηση και ο κώδικας της γενικά θεωρείται ευανάγνωστος. Η Python είναι ιδανική για αρχάριους ή για επιστήμονες που ασχολούνται με τα δεδομένα και θέλουν να φτιάξουν τις ικανότητές τους. Λαμβάνοντας υπόψη ότι η ανάλυση των δεδομένων συμπεριλαμβάνει διάφορες τεχνικές πρόβλεψης για τις οποίες μπορούν να χρησιμοποιηθούν πολλά διαφορετικά εργαλεία εξόρυξης δεδομένων, η εφαρμογή αυτών των τεχνικών χρησιμοποιώντας ένα νέο εργαλείο μπορεί να αποδειχθεί δύσκολη. Η απλότητα της Python απευθύνεται σε μια σειρά διαφορετικών ανθρώπων. Είτε κάποιος είναι έμπειρος ερευνητής δεδομένων ή αναλυτής, ένας μηχανικός λογισμικού που αρχίζει να συνεργάζεται στενότερα με τη μηχανική μάθηση, ή ακόμα και ένας πλήρης αρχάριος, είναι μια εύκολη γλώσσα προγραμματισμού για να πάρει.

Έπειτα, είναι καλύτερη για την κατασκευή εργαλείων ανάλυσης, ειδικά για όταν πρόκειται για τη δημιουργία μιας υπηρεσίας ιστού για να επιτραπεί σε άλλους να βρουν αποκλίσεις στα σύνολα δεδομένων τους. Επίσης, παρότι υπάρχουν γλώσσες που παρέχουν καλύτερη οπτικοποίηση δεδομένων, και η Python παρέχει μια μεγάλη γκάμα και για αυτό.

Η Python έχει μια τεράστια κοινότητα γύρω της, συμπεριλαμβανομένης μιας ισχυρής και αυξανόμενης παρουσίας στην κοινότητα των επιστημών δεδομένων. Έχει ένα μεγάλο εύρος βιβλιοθηκών, για την ανάπτυξη ιστού, μέσω της ανάπτυξης παιχνιδιών, μέχρι τη μηχανική μάθηση και την οπτικοποίηση δεδομένων, όπως αναφέρθηκε και προηγουμένως, αλλά και πολλά πακέτα που βοηθάνε στην πιο εύκολη βαθιά εκμάθησή της.

Παρότι τα θετικά στοιχεία υπερτερούν, η Python είναι αργή συγκριτικά με άλλες γνωστές γλώσσες προγραμματισμού, όπως η C. Επιπλέον, το πακέτο εγκατάστασης δεν είναι τόσο τακτοποιημένο με αποτέλεσμα ο προγραμματιστής να πρέπει να ελέγχει αν περιλαμβάνονται όλα τα αρχεία που είναι απαραίτητα.

Καταλήγοντας, η Python είναι σίγουρα μια πολύ καλή γλώσσα, η οποία επιλέγεται από τους καλύτερους στον κόσμο, εταιρείες όπως το Google, το Facebook ή η Microsoft. Είναι ένα ισχυρό και ευέλικτο εργαλείο που επιτρέπει να γίνονται περισσότερα μέσα σε λιγότερο χρόνο. Σε μια αγορά όπου η διαφοροποίηση γίνεται ολοένα και περισσότερο το κλειδί για

την ανάπτυξη, η Python, είτε είναι η πρώτη επιλογή είτε η δεύτερη, μπορεί να είναι μόνο καλό πράγμα. Το καλύτερο με την Python είναι το γεγονός ότι είναι ευπροσάρμοστη και εύκολη, πράγμα που σημαίνει ότι μπορεί να το ενσωματωθεί σε οποιαδήποτε ροή εργασίας, καθιστώντας την χρήσιμη με τα εργαλεία που ήδη χρησιμοποιούνται ή ενδέχεται να χρησιμοποιηθούν πιο μετά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] K. Jain, "analyticsvidhya," 16 Ιανουάριος 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/.

[2] D. Kopf, "qz," 22 Σεπτέμβριος 2017. [Online]. Available: https://qz.com/1063071/the-great-r-versus-python-for-data-science-debate/.

[3] K. Willems, "DataCamp," 12 Μάιος 2015. [Online]. Available: https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis.

[4] T. Gaddis, Ξεκινώντας με την Python.

[5] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2011.

[6] M. Harrison, Learning the Pandas Library: Python Tools for Data Munging, Analysis, and Visualization, 2016.

[7] J. VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, 2016.

[8] "python.org," [Online]. Available: https://www.python.org/.

[9] K. Oleinik, "Boston University," [Online]. Available: https://www.bu.edu/tech/files/2017/09/Python-for-Data-Analysis.pdf.

[10] A. Beck, "wis.kuleuven.be," [Online]. Available: https://wis.kuleuven.be/CHARM/images/partners/python-lecture.pdf.

[11] M. J. G. M. J. S.Shiraiwa, "Tools to export published datasets together with metadata from IDL/Python/MATLAB and πScope," 2018, Μάιος.

[12] V. Paruchuri, "www.dataquest.io," 2016. [Online]. Available: https://www.dataquest.io/blog/pandas-python-tutorial/.

[13] T. Petrou, "medium.com," 2017. [Online]. Available: https://medium.com/dunder-data/how-to-learn-pandas-108905ab4955.

[14] "www.datasciencegraduateprograms.com," [Online]. Available: https://www.datasciencegraduateprograms.com/python/.

[15] I. Bobriakov, "www.datascience.com," 6 Σεπτέμβριος 2017. [Online]. Available: https://www.datascience.com/blog/top-python-libraries-for-data-science-in-2017.

[16] E. Kirzhner, "codeburst.io," 6 Δεκέμβριος 2017. [Online]. Available: https://codeburst.io/overview-of-python-data-visualization-tools-e32e1f716d10.

[17] M. Bierly, "blog.modeanalytics.com," 8 Ιούνιος 2016. [Online]. Available: https://blog.modeanalytics.com/python-data-visualization-libraries/.

[18] "www.codecademy.com," [Online]. Available: https://www.codecademy.com/articles/install-python-data-analysis.

[19] "drivendata.github.io," [Online]. Available: https://drivendata.github.io/pydata-setup/.

[20] "www.python-course.eu," [Online]. Available: https://www.python-course.eu/matrix_arithmetic.php.

[21] "www.tutorialspoint.com," [Online]. Available:

https://www.tutorialspoint.com/numpy/index.htm.

[22] "docs.scipy.org," [Online]. Available: https://docs.scipy.org.

[23] "www.scipy-lectures.org," [Online]. Available: https:/https://www.scipy-lectures.org/.

[24] "www.data-analysis-in-python.org," [Online]. Available: http://www.data-analysis-in-python.org/.

[25] "pandas.pydata.org," [Online]. Available: https://pandas.pydata.org/.

[26] "www.tutorialspoint.com," [Online]. Available: https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm.

[27] P. Krill, "www.infoworld.com," 24 Φεβρουάριος 2015. [Online]. Available: https://www.infoworld.com/article/2887974/application-development/a-developer-s-guide-to-the-pro-s-and-con-s-of-python.html.

[28] "www.kaggle.com," [Online]. Available: https://www.kaggle.com/unitednations/global-food-agriculture-statistics.

[29] C. ALBON, "chrisalbon.com," 20 Δεκέμβριος 2017. [Online]. Available: https://chrisalbon.com/python/data_wrangling/pandas_dataframe_importing_csv/.

[30] L. Tagliaferri, "www.digitalocean.com," 10 Φεβρουάριος 2017. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-the-pandas-package-and-work-with-data-structures-in-python-3.

[31] "pandas.pydata.org," [Online]. Available: https://pandas.pydata.org/.

[32] M. C. A. K. Phuong Vo.T.H, Share Facebook Twitter Pinterest, 2017.