UNIVERSITY OF PATRAS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ PYTHON

ΨΥΧΟΓΙΑΝΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ: ΔΗΜΗΤΡΙΟΣ ΠΑΠΑΔΟΠΟΥΛΟΣ

ΠΑΤΡΑ-2021

Πρόλογος

Καθημερινά η ανάλυση δεδομένων χρησιμοποιείται από όλο και περισσότερες εταιρίες 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.

Σκοπός της παρούσας πτυχιακής εργασίας είναι να εξεταστεί ο τρόπος λειτουργίας της Python και να κατανοηθεί η γενική δομή της. Επίσης, να αποδειχθεί ότι είναι ισάξια ίσως και καλύτερη από άλλες γλώσσες προγραμματισμού οι οποίες χρησιμοποιούνται για την διαχείριση και την ανάλυση των δεδομένων σήμερα.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον καθηγητή και εισηγητή μου κ. Παπαδόπουλο Δημήτριο που με τις υποδείξεις του βοήθησε σημαντικά στην υλοποίηση αυτής της εργασίας.

Περίληψη

Η πτυχιακή εργασία ξεκινάει με μια αναφορά στον Προγραμματισμό έτσι ώστε να κατανοηθεί η χρησιμότητα του στην επίλυση επιστημονικών και μηχανικών προβλημάτων. Αναφέρεται επιγραμματικά ο τρόπος που λειτουργεί καθώς και τα εργαλεία του.

Στο δεύτερο κεφάλαιο γίνεται μία εισαγωγή στη γλώσσα της Python, παραθέτοντας επιγραμματικά τη δράση της, τις βιβλιοθήκες της καθώς και τους τομείς στους οποίους χρησιμοποιείται. Επίσης γίνεται μια αναφορά στα πλεονεκτήματα και τα μειονεκτήματά της.

Στο τρίτο κεφάλαιο, παρουσιάζονται οι βασικές λειτουργίες της, τα χαρακτηριστικά της και η δομή της με την βοήθεια του Anaconda. Παρουσιάζονται ορισμοί και παραδείγματα σε όλα τα στάδιά της, τα οποία είναι χρήσιμα για την κατανόησή της.

Τέλος καταλήγουμε σε κάποια συμπεράσματα που αποδεικνύουν την χρησιμότητά της, τα οφέλη και την ευκολία της όταν χρησιμοποιείται τόσο από επιστήμονες όσο και από αρχάριους.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

1. Προγραμματισμός

1.1 Εισαγωγή στον Προγραμματισμό

Ο υπολογιστής είναι μια μηχανή που επεξεργάζεται δεδομένα και αντιμετωπίζει υπολογιστικά προβλήματα. Για να γίνει αυτό, θα πρέπει να του δώσουμε εντολές που μπορεί να κατανοήσει οι οποίες έχουν την μορφή προγραμμάτων. Οι εντολές αυτές φτιάχνονται από τους προγραμματιστές σε τεχνητές γλώσσες οι οποίες ονομάζονται γλώσσες προγραμματισμού.

Ο υπολογιστής είναι πλέον αναπόσπαστο κομμάτι της καθημερινής ανθρώπινης δραστηριότητας και στην ουσία μας βοηθάει στην επίλυση προβλημάτων. Ως πρόβλημα θεωρούμε οποιοδήποτε ζήτημα χρειάζεται λύση, οποιαδήποτε κατάσταση μας απασχολεί και πρέπει να αντιμετωπιστεί. Καθημερινά αντιμετωπίζουμε διάφορα είδη προβλημάτων, που κάποια μπορούμε να λύσουμε εύκολα, ενώ κάποια άλλα που είναι πιο σύνθετα δυσκολευόμαστε ή και μερικές φορές αδυνατούμε να βρούμε κάποια λύση. Η χρήση του υπολογιστή είναι σημαντικότερη στις περιπτώσεις κατά τις οποίες τα προβλήματα που καλούμαστε να αντιμετωπίσουμε απαρτίζονται από πολλά δεδομένα και ζητούμενα, ή η μέθοδος επίλυσης μπορεί είναι πολύπλοκη ή να επαναλαμβάνεται πολλές φορές.

Ένα πλεονέκτημα του υπολογιστή είναι οτι μας παρέχει μεγάλο αποθηκευτικό χώρο δεδομένων, μπορεί να εκτελεί υπολογισμούς και να επεξεργάζεται δεδομένα ταχύτερα από έναν άνθρωπο. Έχει τη δυνατότητα να εκτελεί μια λογική σειρά εντολών με συνέπεια, πειθαρχία και για όσες φορές μπορεί να χρειαστεί. Οι εντολές εισάγονται στον υπολογιστή με τη μορφή προγραμμάτων.

Ο υπολογιστής για να αντιμετωπίσει τα υπολογιστικά προβλήματα απαιτεί λογικές σκέψεις και μαθηματικές πράξεις. Έτσι εισάγουμε σ’αυτόν τα απαραίτητα δεδομένα χρησιμοποιώντας διάφορες συσκευές εισόδου, τα οποία δεδομένα επεξεργάζεται ανάλογα με το ποιες εντολές του έχουμε δώσει. Όταν επεξεργαστεί

τα δεδομένα μας παρέχει χρήσιμες πληροφορίες και απαντήσεις οι οποίες βοηθούν στην επίλυση των προβλημάτων μας.

Αλγόριθμος ονομάζεται η ακριβής και συγκεκριμένη σειρά βημάτων που απαιτείται για την επίλυση ενός προβλήματος. Για παράδειγμα τα βήματα που κάνουμε σε μια μαγειρική συνταγή ή για να λύσουμε ένα μαθηματικό πρόβλημα δομούν έναν αλγόριθμο. Ακολουθώντας τα βήματα ενός αλγορίθμου, στο τέλος θα πρέπει να βγαίνει ένα αποτέλεσμα. Επίσης, ένας αλγόριθμος θα πρέπει κάποτε να τελειώνει. Ο αλγόριθμος μπορεί να δημιουργηθεί είτε με χρήση της φυσικής γλώσσας, με διαγραμματικές τεχνικές είτε με κωδικοποίηση σε κάποια ειδική γλώσσα. Ένα πρόγραμμα αποτελεί έναν αλγόριθμο φτιαγμένο με τέτοιο τρόπο που να μπορεί να κατανοεί ο υπολογιστής και περιέχει οδηγίες που μπορούν να τον κατευθύνουν με κάθε λεπτομέρεια στο να εκτελέσει μια συγκεκριμένη εργασία και να επιλύσει ένα πρόβλημα.

1.2 Τι είναι Προγραμματισμός

Προγραμματισμός (Programming) ονομάζεται το σύνολο των διαδικασιών που πρέπει να ακολουθήσουμε για τη δημιουργία ενός προγράμματος, με την μορφή αλγορίθμων, για να μπορεί να εκτέλεσει εργασίες ή να επιλύσει κάποιο πρόβλημα μέσω του υπολογιστή. Επίσης περιλαμβάνει τον έλεγχο του προγράμματος για την ορθότητά του καθώς και την επαλήθευση του σκοπού για τον οποίο δημιουργήθηκε.

Ο προγραμματισμός είναι σημαντικό εργαλείο στη σημερινή εποχή, επειδή μπορεί να επιλύσει προβλήματα διαφόρων τύπων. Χρησιμοποιείται για την επίλυση επιστημονικών και μηχανικών προβλημάτων, με τη χρήση αριθμητικών και υπολογιστικών μεθόδων.

1.3 Γλώσσες Προγραμματισμού

Γλώσσα προγραμματισμού (programming language) ονομάζουμε όλες εκείνες τις εντολές και τους κανόνες που αποτελούν τη δομή και το νόημα μίας γλώσσας.

Η γλώσσα προγραμματισμού είναι η γλώσσα που μπορεί να καταλάβει ο υπολογιστής. Ο ηλεκτρονικός υπολογιστής χρησιμοποιεί μόνο τη γλώσσα μηχανής (machine language), έτσι δημιουργήθηκαν οι γλώσσες προγραμματισμού οι οποίες μετατρέπουν τον κώδικα σε γλώσσα μηχανής. Αρχικά, αναπτύχθηκε η γλώσσα πρώτης γενιάς (1GL - First Generation programming language). Στη συνέχεια, αναπτύχθηκε η δεύτερη γενιά γλωσσών προγραμματισμού (2GL) και τέλος ακολούθησαν οι γλώσσες τρίτης γενιάς (3GL) και τέταρτης γενιάς (4GL).

Η σημασίολογία και η σύνταξη μιας γλώσσας προγραμματισμού θα πρέπει να είναι αυστηρά ορισμένη. Η σύνταξη καθορίζεται από σύμβολα που αποτελούν «νόμιμες» εντολές ενός προγράμματος με τη μορφή συγκεκριμένης γλώσσας προγραμματισμού και η σημασιολογία ορίζει τις υπολογιστικές διαδικασίες που υλοποιεί το πρόγραμμα.

Οι άνθρωποι θα πρέπει να κατανοούν τις γλώσσες προγραμματισμού για να μπορέσουν να φτιάξουν με αυτές έναν αλγόριθμο, αλλά και οι υπολογιστές θα πρέπει να τις κατανοούν για να μπορούν να εκτελέσουν τις εντολές των προγραμμάτων. Αν χρησιμοποιούσαμε τη γλώσσα που μιλάμε, τότε αυτή θα την κατανοούσε μόνο ο άνθρωπος. Αν χρησιμοποιούσαμε τη γλώσσα μηχανής (με ακολουθία δυαδικών ψηφίων 0,1), τότε αυτή θα την κατανοούσε μόνο ο υπολογιστής. Η λύση για το παραπάνω πρόβλημα είναι να δημιουργηθούν γλώσσες προγραμματισμού που θα περιέχουν συμβολισμούς κατανοητούς από τον άνθρωπο και ειδικά μεταφραστικά προγράμματα τα οποία θα μετατρέπουν τις εντολές σε γλώσσα που μπορεί να καταλάβει ο υπολογιστής.

Η προσπάθεια του να γίνει ευκολότερη η συγγραφή, η διόρθωση και η συντήρηση προγραμμάτων είχε ως αποτέλεσμα στη δημιουργία των γλωσσών υψηλού επιπέδου. Οι γλώσσες υψηλού επιπέδου μοιάζουν με τη γλώσσα που χρησιμοποιεί ο άνθρωπος δηλαδή έχουν δικό τους αλφάβητο, λεξιλόγιο και συντακτικό. Κάποιες από τις πιο γνωστές γλώσσες είναι η C, C++, C#, Java, PHP,

Javascript, Perl, Lisp, Basic, Ruby και η Python. Ανάλογα με το είδος του προγράμματος το οποίο θέλουμε να αναπτύξουμε διαλέγουμε και την κατάλληλη γλώσσα προγραμματισμού.

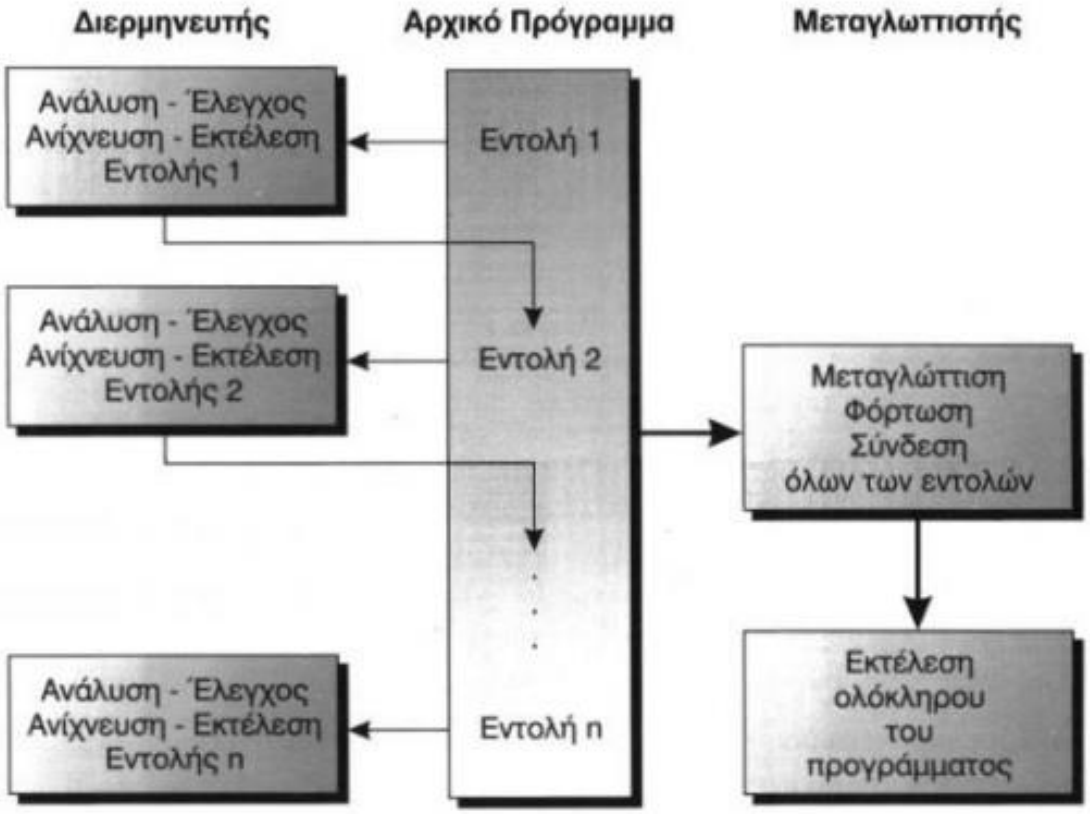
1.4 Κατηγορίες Γλωσσών Προγραμματισμού

Υπάρχουν δύο είδη γλωσσών προγραμματισμού, οι γλώσσες χαμηλού και υψηλού επιπέδου.

- Η γλώσσα υψηλού επιπέδου (high-level programming language) αποτελείται από εντολές που μπορεί να τις κατανοήσει πιο εύκολα ο χρήστης, λόγω του ότι πλησιάζουν τη φυσική γλώσσα. Επίσης, το πρόγραμμα μπορεί να επεξεργαστεί και να εκτελεστεί και σε διαφορετικό υπολογιστή από αυτόν που δημιουργήθηκε.

- Ενώ, η γλώσσα χαμηλού επιπέδου (low-level programming language) ονομάζεται και γλώσσα μηχανής, επειδή δεν χρειάζεται μεταγλωττιστή ή διερμηνευτή και το πρόγραμμα μπορεί να εκτελεστεί όπως είναι.

Για να μετατρέψουμε μια γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής υπάρχουν δύο τρόποι, με διερμηνευτή (interpreter) και με μεταγλωττιστή (compiler). Ο διερμηνευτής μετατρέπει και εκτελεί γραμμή προς γραμμή τον κώδικα σε γλώσσα μηχανής και ο μεταγλωττιστής μετατρέπει ολόκληρο το πρόγραμμα σε πρόγραμμα μηχανής και έπειτα το εκτελεί. Όμως, μετά τη μεταγλώττιση το πρόγραμμα μπορεί να χρησιμοποιηθεί, χωρίς να χρειάζεται πάλι μετατροπή.



Εικόνα 1. Διαδικασία Μετάφρασης και Εκτέλεσης ενός Προγράμματος

(Πηγή: "Ανάπτυξη εφαρμογών σε Προγραμματιστικό Περιβάλλον", βιβλίο Γ' τάξης Ενιαίου Λυκείου)

Η Python είναι υψηλού επιπέδου γλώσσα προγραμματισμού και ανήκει στις διερμηνευόμενες γλώσσες (interpreted programming languages), όπου τα προγράμματα εκτελούνται με τη βοήθεια διερμηνέα.

1.5 Εργαλεία Προγραμματισμού

Οι προγραμματιστές χρησιμοποιούν ένα ολοκληρωμένο λογισμικό το οποίο αποτελείται από τα εξής εργαλεία:

1 ένας συντάκτης κειμένων (editor) με την βοήθεια του οποίου γράφεται το αρχικό πρόγραμμα. Αυτό το πρόγραμμα ονομάζεται πηγαίο πρόγραμμα ή κώδικας (source code).

2 ένα μεταφραστικό πρόγραμμα, μεταγλωττιστή ή διερμηνευτή, το οποίο μετατρέπει το πηγαίο πρόγραμμα σε αντικείμενο πρόγραμμα ή κώδικα (object code). Το μεταφραστικό πρόγραμμα είναι αυτό που κάνει τον έλεγχο αν το πηγαίο πρόγραμμα έχει συντακτικά λάθη (λάθη στο αλφάβητο, στο λεξιλόγιο, στο συντακτικό). Εάν βρεθούν λάθη, εμφανίζει κατάλληλα διαγνωστικά μηνύματα και αν δεν υπάρχουν λάθη μόνο τότε δημιουργείται το αντικείμενο πρόγραμμα. Το αντικείμενο πρόγραμμα είναι σε γλώσσα μηχανής, αλλά ο υπολογιστής δεν μπορεί ακόμα να το εκτελέσει και πρέπει πρώτα να περάσει από κάποιες άλλες διαδικασίες. Ο μεταγλωττιστής είναι αυτός που ελέγχει όλο το πρόγραμμα συνολικά και ο διερμηνευτής αυτός που ελέγχει μία εντολή κάθε φορά, την εκτελεί και έπειτα πηγαίνει την επόμενη.

3 ένα ειδικό πρόγραμμα που ονομάζεται συνδέτης (linker), το οποίο πολλές φορές είναι ο σύνδεσμος μεταξύ του αντικείμενου προγράμματος ή του συνόλου αντικειμένων προγραμάτων με τα έτοιμα υποπρογράμματα της βιβλιοθήκης της γλώσσας προγραμματισμού ή του προγραμματιστή. Το τελικό πρόγραμμα το οποίο φτιάχνεται είναι το εκτελέσιμο πρόγραμμα ή κώδικας (executable code), έχει μετατραπεί σε γλώσσα μηχανής και μπορεί να εκτελεστεί άμεσα από τον υπολογιστή.

4 εργαλεία εντοπισμού λαθών (debuggers) τα οποία δίνουν τη δυνατότητα στον προγραμματιστή να παρακολουθεί τι ακριβώς συμβαίνει κατά την εκτέλεση ενός προγράμματος, για να μπορεί βρει και να διορθώσει τυχόν λάθη.

ΚΕΦΑΛΑΙΟ 2

2. Εισαγωγή στην Python

2.1 Ιστορική Αναδρομή

Η Python δημιουργήθηκε το 1989 από τον Ολλανδό Guido van Rossum στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI), του οποίου βασική έρευνα θεωρείται το πεδίο των Μαθηματικών και της Επιστήμης Υπολογιστών. Βασική πηγή έμπνευσης της δημιουργίας της Python είναι η γλώσσας προγραμματισμού ABC. Στην αρχή την χρησιμοποίησαν ως γλώσσα σεναρίων για το κατανεμημένο λειτουργικό σύστημα Amoeba. Όπως αναφέρει ο ίδιος ο Guido van Rossum, η πιο καινοτόμα συνεισφορά του στην επιτυχία της Python ήταν το ότι αυτή είχε τη δυνατότητα να επεκτείνεται εύκολα. Το μοντέλο της Python σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι ευέλικτα επεκτάσιμο, να παρέχει ενσωματωμένα στοιχεία (εντολές, τύπους αντικειμένων, κ.λπ.) αλλά και να δίνει τη δυνατότητα στους προγραμματιστές να προσθέτουν τα δικά τους στοιχεία ανάλογα με τις ανάγκες τους και το σύστημα που χρησιμοποιούν. Η πρώτη έκδοση κυκλοφόρησε το 1991. Η ονομασία της προέρχεται από μια κωμική σειρά της δεκαετίας του 70 το «Monty Python's Flying Circus» του BBC της Μεγάλης Βρετανίας. Ο Guido van Rossum το όνομα που έψαχνε για τη νέα γλώσσα προγραμματισμού ήθελε να είναι μικρό, μοναδικό και ελαφρώς μυστήριο. Από αυτό συμπεραίνουμε ότι δεν υπάρχει καμία απολύτως συσχέτιση με ερπετά ή το γνωστό φίδι πύθωνας.

Η Python έχει ραγδαία εξέλιξη με τις νέες εκδόσεισ και με βασικό εργαλείο τα Python Enhancement Proposals ("PEPs"), τα οποία είναι απαραίτητα στο να δίνουν γενικές πληροφορίες, οδηγίες, προτάσεις και περιγραφές για τυχόν νέα χαρακτηριστικά της γλώσσας. Μέχρι στιγμής υπάρχουν δύο σειρές εκδόσεων της Python, η σειρά 2.χ με πιο πρόσφατη έκδοση παραγωγής την 2.7.9 και η νεότερη σειρά 3.χ με πιο πρόσφατη έκδοση παραγωγής την 3.4.3. Και στις δυο σειρές εκδόσεων, η Python είναι στο το μεγαλύτερο μέρος της η ίδια, αλλά με διαφορές σε πολλές λεπτομέρειες, κυρίως στο πως λειτουργούν τα ενσωματωμένα

αντικείμενα όπως τα λεξικά και οι συμβολοσειρές. Επίσης, έχουν αφαιρεθεί κάποια χαρακτηριστικά στη νεότερη έκδοσή της και έχει επανεξεταστεί η πρότυπη βιβλιοθήκη της. Πολλές εφαρμογές άλλων κατασκευαστών είναι φτιαγμένες με βάση τις εκδόσεις της σειράς 2.χ.

2.2 Δράση

Η Python είναι από τις πιο διαδεδομένες γλώσσες προγραμματισμού σήμερα. Η Python αποτελεί έναν συμβιβασμό μεταξύ της R, η οποία ασχολείται σε μεγάλο βαθμό με την ανάλυση και την απεικόνιση δεδομένων, και της Java, η οποία αποτελεί την βάση πολλών εφαρμογών μεγάλης κλίμακας. Μεταξύ άλλων, υποστηρίζει αντικειμενοστραφή προγραμματισμό, δομημένο προγραμματισμό και λειτουργικά πρότυπα προγραμματισμού. Η Python μπορεί να πραγματοποιήσει κάθε εργασία από να βγάλει δεδομένα έως το να κατασκευάσει ιστότοπους που να μπορούν να λειτουργούν ενσωματωμένα συστήματα, όπου όλα είναι σε μια ενοποιημένη γλώσσα.

Για παράδειγμα, στο ForecastWatch, η Python βοήθησε έναν αναλυτή να μαζέψει τις προβλέψεις από άλλους ιστότοπους και να δημιουργήσει έναν μηχανισμό συσσωμάτωσης για την κατάρτιση των δεδομένων και έναν κώδικα ιστότοπου για την εμφάνιση των αποτελεσμάτων. Αρχικά χρησιμοποίησαν την PHP για να φτιάξει τον ιστότοπο μέχρι που η εταιρεία κατάλαβε ότι μπορούσε πιο εύκολα να ασχοληθεί μόνο με μια ενιαία γλώσσα.

Το Facebook, σύμφωνα με άρθρο του περιοδικού Fast Company για το 2014, επέλεξε την Python για την ανάλυση δεδομένων, επειδή ήδη χρησιμοποιούσε σε άλλα μέρη της εταιρείας. Λόγω της πολυχρησιμότητάς της, η Python είναι πολύ αγαπητή από τους προγραμματιστές.

Η Python θεωρείται σχετικά εύκολη και οι πολλές αποκλειστικές αναλυτικές βιβλιοθήκες που διαθέτει σήμερα έχει ως αποτέλεσμα, να μπορούν να βρίσκουν οι επιστήμονες πακέτα δεδομένων σε κάθε τομέα σχεδόν, τα οποία είναι ήδη προσαρμοσμένα στις ανάγκες τους και διαθέσιμα για λήψη.

Την Python συνήθως επιλέγουν προγραμματιστές οι οποίοι θέλουν να εφαρμόσουν στατιστικές τεχνικές ή ανάλυση δεδομένων στο έργο τους και επιστήμονες δεδομένων οι οποίοι ασχολούνται με εφαρμογές ιστού. Η Python θεωρείται ως κατάλληλη για την ολοκλήρωση εξελιγμένων μοντέλων και μηχανισμών πρόβλεψης που συνδέονται άμεσα με τα συστήματα παραγωγής λόγω των βιβλιοθηκών μηχανικής μάθησης σε συνδυασμό με την ευελιξία.

2.3 Πλεονεκτήματα και Μειονεκτήματα της Python

Η Python θεωρείται γλώσσα προγραμματισμού γενικής χρήσης, απλή και μπορείς να την μάθεις εύκολα,είναι αποδοτική, παραγωγική και επεκτάσιμη. Μπορούν να την χρησιμοποιήσουν τόσο αρχάριοι όσο και έμπειροι προγραμματιστές. Μπορεί να χρησιμοποιηθεί ακόμα και για εκπαιδευτικούς σκοπούς αλλά και για την δημιουργία ολοκληρωμένων εφαρμογών.

Η Python αποτελείται από αποδοτικές δομές δεδομένων υψηλού επιπέδου. Είναι διερμηνευόμενη γλώσσα προγραμματισμού και χρησιμοποιώντας την μπορούν να δημιουργηθούν σενάρια εντολών και να αναπτυχθούν γρήγορα ολοκληρωμένες εφαρμογές σε διάφορα είδη όπως στη διαχείριση συστήματος υπολογιστή, την ανάπτυξη εφαρμογών Διαδικτύου, την επεξεργασία αρχείων κειμένου, σε επιστημονικές εφαρμογές, εκπαίδευση, ανάπτυξη παιχνιδιών, κ.λπ. και στις περισσότερες πλατφόρμες υλικού υπολογιστών και Λειτουργικών Συστημάτων όπως Windows, Unix, Linux, Mac OS X, κ.λπ. Παρέχει πάρα πολλές έτοιμες βιβλιοθήκες που μπορούν να χρησιμοποιηθούν εύκολα και άμεσα. Οι βιβλιοθήκες έχουν επεκτάσεις γραμμένες σε C ή C++. Τα προγράμματα σε Python μπορεί κανείς να τα αναγνώσει εύκολα, γράφονται και διατηρούνται γρηγορότερα συγκριτικά με άλλες γλώσσες προγραμματισμού όπως οι C, C++ και Java. Ο κώδικας μπορεί να μπει σε ομάδες με αρθρώματα (modules) και πακέτα (packages).

Είναι λογισμικό ανοικτού κώδικα ανάλογα με τους προγραμματιστές οι οποίοι εργάζονται για την ανάπτυξή της και η διαχείριση γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation (PSF)

(www.python.org/psf). Το PSF έχει ως στόχο την ανάδειξη, την προστασία και την προαγωγή της Python, καθώς και την ανάπτυξη μιας ποικίλης και διεθνούς κλίμακας κοινότητας προγραμματιστών της Python. Η κοινότητα αυτή καθημερινά αυξάνεται και η Python βρίσκεται στις πρώτες θέσεις της λίστας των γνωστότερων γλωσσών προγραμματισμού. Επίσης, την χρησιμοποιούν μεγάλες εταιρείες και οργανισμοί του χώρου της Πληροφορικής και όχι μόνο.

Πέρα από τα πολλά θετικά χαρακτηριστικά της Python, πρέπει να αναφέρουμε και τα μειονεκτήματά της.

1 Τα προγραμμάτα της Python δεν εκτελούνται πάντα τόσο γρήγορα όσο στις μεταγλωττιζόμενες (compiled) γλώσσες όπως η C και η C++. Αυτό συμβαίνει γιατί ένα πρόγραμμα σε Python δεν μετατρέπεται σε δυαδικό κώδικα μηχανής που εκτελείται άμεσα από τον επεξεργαστή του υπολογιστή. Το μειονέκτημα αυτό είναι σημαντικό γιατί πολλές φορές μας ενδιαφέρει περισσότερο ο χρόνος που έχουμε για την εκτέλεση μιας εφαρμογής σε Python.

2 Επίσης, η Python δεν μπορεί να μας βοηθήσει στο να δημιουργήσουμε, για παράδειγμα, ένα νέο Λειτουργικό Σύστημα.

2.4 Βιβλιοθήκες της Python

Η μεγάλη γκάμα διαθέσιμων βιβλιοθηκών έχει ως αποτέλεσμα την επιτυχία της Python όπως συμβαίνει και με άλλες γλώσσες προγραμματισμού: υπάρχουν γύρω στις 72.000 από αυτές και αυξάνονται συνεχώς. Η βιβλιοθήκη της Python αποτελείται από εργαλεία για κάθε είδους προγραμματισμό και για την επίλυση προβλημάτων χωρίς να χρειάζεται να επιλεχθεί κάποια από τις ανταγωνιστικές. Η Python περιέχει ένα δωρεάν λογισμικό ανοικτού κώδικα με αποτέλεσμα να μπορεί οποιοσδήποτε να φτιάξει ένα πακέτο βιβλιοθήκης για να επεκτείνει τη λειτουργικότητά του. Στη συνέχεια παρουσιάζονται επιγραμματικά οι βασικές βιβλιοθήκες της:

- ***Numpy:***

  Είναι η βάση πάνω στην οποία δημιουργούνται όλα τα εργαλεία υψηλότερου επιπέδου για την επιστημονική Python. Κάποιες από τις λειτουργίες που παρέχει:

  1 N- Dimensional array: έναν πίνακα πολλών διαστάσεων με γρήγορη και αποτελεσματική μνήμη που περιέχει αριθμητικές πράξεις.

  2 Έχει τη δυνατότητα να εφαρμόσει τυποποιημένες μαθηματικές πράξεις σε πίνακες ολόκληρων δεδομένων χωρίς να δημιουργηθούν βρόχοι.

  3 Εύκολα μπορούν να μεταφερθούν δεδομένα σε εξωτερικές βιβλιοθήκες γραμμένες σε γλώσσα χαμηλού επιπέδου και επίσης από εξωτερικές βιβλιοθήκες να επιστρέφουν δεδομένα σε Python ως Numpy arrays.

Η Numpy δεν παρέχει λειτουργικότητα ανάλυσης δεδομένων υψηλού επιπέδου. Έχοντας κατανοήσει τις συστοιχίες Numpy και τις υπολογιστικές προσανατολισμένες με συστοιχίες, θα βοηθήσει να χρησιμοποιηθούν πολύ πιο αποτελεσματικά εργαλεία όπως η Pandas.

- ***Pandas:***

  Η Pandas αποτελεί την βιβλιοθήκη ανάλυσης δεδομένων της Python, η οποία χρησιμοποιείται για όλα. Η Pandas δίνει στους αναλυτές σχεδόν όλα τα κοινά εργαλεία για τα δεδομένα. Αυτό σημαίνει ότι η βασική εκκαθάριση και κάποια προηγμένη χειραγώγηση έχουν την δυνατότητα να εκτελεστούν πάνω σε ισχυρά πλαίσια δεδομένων της Pandas. Η Pandas έχει χτιστεί πάνω στη Numpy, που αυτή ήταν μία από τις βιβλιοθήκες που δημιουργήθηκαν πρώτα για να φτάσει στην επιτυχία η επιστήμη δεδομένων της Python. Οι λειτουργίες Numpy εκτίθενται σε Pandas για προηγμένη αριθμητική ανάλυση. Περιέχει υψηλού επιπέδου δομές δεδομένων και εργαλεία που έχουν δημιουργηθεί για την γρήγορη και εύκολη ανάλυση δεδομένων. Χρησιμοποιώντας την Pandas, είναι ευκολότερο να χειριστούν τα δεδομένα τα οποία λείπουν. Η Pandas είναι αυτό το εργαλείο που χρειάζεται να γίνει στα δεδομένα munging.

- ***Scipy***:

  Η βιβλιοθήκη Scipy για να λειτουργήσει, χρειάζεται τη Numpy, η οποία παρέχει εύκολο και γρήγορο χειρισμό πίνακα N-διαστάσεων (N- Dimensional array). Η βιβλιοθήκη Scipy έχει δημιουργηθεί για να χρησιμοποιεί τους αριθμητικούς πίνακες Numpy και έχει πολλές αποτελεσματικές αριθμητικές διαδικασίες, όπως διαδικασίες για αριθμητική ενσωμάτωση και βελτιστοποίηση. Η Scipy έχει ενότητες βελτιστοποίησης, γραμμικής άλγεβρας, ολοκλήρωσης και άλλων κοινών καθηκόντων στην επιστήμη των δεδομένων.

- ***Scikit-learn:***

  Η Scikit-learn αποτελεί μια βιβλιοθήκη Python για μηχανική μάθηση (Machine Learning) που έχει σαν βάση της τη Numpy και τη Scipy. Η Scikit-learn βοηθά στο να εφαρμοστούν άμεσα γνωστοί αλγόριθμοι στο σύνολο δεδομένων. Η Scikit-learn περιέχει εργαλεία για αρκετές εργασίες εκμάθησης μηχανών και εξαγωγής δεδομένων όπως η ομαδοποίηση, η ταξινόμηση, η παλινδρόμηση κ.λπ.

- ***Ipython:***

  Η IPython κάνει πιο λειτουργικό τον διερμηνέα της Python με ένα κελί που περιέχει ενδοσκόπηση, ολοκλήρωση καρτελών, εμπλουτισμένα μέσα, σύνταξη κελιού, και ανάκτηση ιστορικού εντολών. Αποτελεί επίσης και τον ενσωματωμένο διερμηνέα για τα προγράμματα, που μπορεί να είναι πολύ χρήσιμος στο να εντοπιστούν σφάλματα.

- ***Matplotlib:***

  Η Matlplotlib είναι μια βιβλιοθήκη Python για απεικόνιση. Η Matplotlib παρέχει τη δυνατότητα δημιουργίας εύκολων γραφημάτων γραμμών, ιστόγραμμα, διάγραμμα πίτας και άλλα επαγγελματικά στοιχεία. Υποστηρίζει διαφορετικά GUI back ends σε όλα τα λειτουργικά συστήματα και μπορεί να δημιουργήσει γραφικά σε κοινές μορφές διανύσματος και γραφικών όπως PDF, BMP,SVG, PNG, JPG, GIF κλπ.

Τέλος, η Python έχει κι άλλες γνωστες βιβλιοθήκες όπως η Theano και η TensorFlow που χρησιμοποιούνται για μηχανική μάθηση, η Scrapy, η NLTK και η

Pattern που χρησιμοποιούνται για την δημιουργία δεδομένων και διαχείριση της γλώσσας και η Seaborn, η Bokeh, η Basemap και η NetworkX που χρησιμοποιούνται για διαγράμματα και απεικονίσεις.

2.5 Τομείς που χρησιμοποιείται η Python

- Google
- Yahoo (Χάρτες)
- YouTube
- NASA (Πρόβλεψη καιρού)
- ABN – Amro Bank (security)
- Bit Torrent Client
- Maya – Blender (3D graphics)
- Πανεπιστήμια (University of California)
- Civilization IV – Battlefield 2 (Games)
- CIA

ΚΕΦΑΛΑΙΟ 3

3. Χαρακτηριστικά της Python

3.1 Εγκατάσταση της Python

Για την ανάλυση δεδομένων με την Python, θα χρησιμοποιηθεί το Anaconda το οποίο είναι κατάλληλο για εφαρμογές που σχετίζονται με την επιστήμη δεδομένων και την μηχανική μάθηση γιατί μπορεί να επεξεργαστεί δεδομένα ευρείας κλίμακας, παρέχει προγνωστική ανάλυση και επιστημονική πληροφορική. Το Anaconda αποτελείται από όλα σχεδόν τα εργαλεία που χρειάζονται, περιέχει τη γλώσσα πυρήνα Python, ένα βελτιωμένο REPL περιβάλλον που ονομάζεται Jupyter, αριθμητικές υπολογιστικές βιβλιοθήκες (NumPy, Pandas), σχεδιαστικές βιβλιοθήκες (Matplotlib, Seaborn), στατιστικές και βιβλιοθήκες μηχανικής μάθησης (Scikit-learn ,SciPy, Statsmodels).

Η Python δεν είναι προεγκατεστημένη με όλα εκείνα τα εργαλεία που μπορεί να χρειαστούν, ακόμη και όταν χρησιμοποιηθεί κατά την εγκατάστασή της το Anaconda. Όταν χρειάζεται ιδιαίτερες λειτουργίες, τότε μπορούν να προσαρμοστούν ανάλογα πακέτα. Για να εγκατασταθούν τα πακέτα που χρειάζονται, χρησιμοποιείται το conda (package manager).

Σε πολλά διαφορετικά λειτουργικά συστήματα και περιβάλλοντα μπορεί να χρησιμοποιηθεί και η διαδικασία εγκατάστασης είναι απλή:

- Windows:

1 Μπαίνουμε στον ιστότοπο της Python: www.python.org

2 Επιλέγουμε τον σύνδεσμο Downloads και κατεβάζουμε το πιο πρόσφατο αρχείο εγκατάστασης της σειράς 3.χ για Windows. Έχουμε την επιλογή να διαλέξουμε μεταξύ 32-bit και 64-bit εγκατάστασης.

3 Κάνουμε εκτέλεση στο αρχείο της εγκατάστασης

- Linux:

Η Python μπορεί να είναι ήδη εγκατεστημένη, αλλά μάλλον δεν θα είναι πρόσφατη έκδοση. Για να εγκαταστήσουμε τη νεότερη έκδοση, μπορούμε είτε να κατεβάσουμε τον πηγαίο κώδικα από τον ιστότοπο της Python και να τον μεταγλωττίσουμε ή να την εγκαταστήσουμε χρησιμοποιώντας τον διαχειριστή πακέτων λογισμικού (package manager) που συνοδεύει το λειτουργικό μας σύστημα.

- Mac OS X:

Και εδώ μπορεί να υπάρχει ήδη προεγκατεστημένη μια έκδοση της Python. Το πιθανότερο όμως είναι να πρόκειται για μια παλαιότερη, οπότε θα χρειαστεί να κατεβάσουμε τη νεότερη έκδοση η οποία βρίσκεται στον ιστότοπο της Python: www.python.org/downloads/mac-osx

3.2 Βασικά στοιχεία

Τα βασικότερα στοιχεία της είναι τα εξής:

- ***Αναγνωριστικά***

Αναγνωριστική ονομάζουμε μία μεταβλητή ή μία συνάρτηση ή μία κλάση ή ένα αντικείμενο (object). Μπορεί να αποτελείται από οποιαδήποτε γράμματα, αριθμούς, καθώς και το χαρακτήρα κάτω παύλας (_). Όμως, δεν γίνεται να αρχίζει από αριθμό και δεν επιτρέπονται χαρακτήρες όπως @, $ και λοιπά. Επίσης, μπορεί να γίνει διακριτή η διαφορά μεταξύ πεζών και κεφαλαίων γραμμάτων.

- ***Λέξεις – κλειδιά***

Οι λέξεις-κλειδιά αποτελούν λέξεις οι οποίες δεν μπορούν χρησιμοποιηθούν ως μεταβλητή ή άλλο αναγνωριστικό όνομα. Οι λέξεις αυτές αποτελούνται από πεζά γράμματα και είναι οι εξής: and, as, assert, break, continue, class, def, del, else, elif, except, exec, finally, from, for, global, if, is, import in, lambda, not, or, pass, print, , return, raise, try, with, while, , yield.

- ***Εσοχή***

Στην Python δεν υπάρχει εντολή για το κλείσιμο και μία ομάδα εντολών υποδηλώνεται με τη χρήση εσοχών. Η χρήση εσοχών είναι αναγκαστική έτσι ώστε να μπορεί ξεχωρίζει η κάθε ομάδα. Μεταβλητός είναι ο χώρος της εσοχής, αλλά πρέπει να περιέχουν το ίδιο περιθώριο εσοχής όλες οι εντολές.

- ***Εντολή help***

Για την σύνταξη και τη λειτουργία κάποιας εντολής χρησιμοποιούμε την εντολή help. Σύνταξη: help(όνομα_εντολής).

- ***Δηλώσεις πολλών γραμμών***

Κάθε δήλωση συνήθως τελειώνει όταν πηγαίνει σε νέα γραμμή. Ωστόσο, με τη χρήση του (\ ) μπορούμε να συνεχίσουμε την εντολή σε παραπάνω από μία γραμμή. Δηλαδή με τη χρήση του (\ ) δηλώνουμε ότι η γραμμή θα συνεχιστεί.

- ***Σχόλια***

Το σύμβολο της δίεσης (#) χρησιμοποιούμε για να προσθέσουμε σχόλια (comments) στον κώδικα. Μετά τη δίεση, όλοι οι χαρακτήρες μέχρι και

το τέλος της γραμμής, θεωρούνται σχόλια και αγνοούνται. Προσθέτοντας στην αρχή κάθε γραμμής το σύμβολο της δίεσης, εισάγουμε πολλές γραμμέσ με σχόλια.

- ***Εντολή print***

Για να εμφανίσουμε δεδομένα στην οθόνη, ο πιο απλός τρόπος είναι η χρήση της εντολής print(). Επομένως για να εμφανιστούν τα αποτελέσματα στην οθόνη χρησιμοποιούμε αυτή την εντολή. Ο τρόπος σύνταξής της είναι, print(), ανάμεσα στις παρενθέσεις και με τη χρήση εισαγωγικών μπορούμε να εισάγουμε μία συμβολοσειρά ή μία μεταβλητή. Για παράδειγμα:

```
name="Helen"

print("Her name is ", name)

Her name is Helen
```

- ***Εντολή input***

Για να εισάγουμε τιμή από το πληκτρολόγιο, κατά την εκτέλεση του κώδικα, χρησιμοποιούμε η εντολή input. Πρωτα πληκτρολογούμε την εντολή input και έπειτα μία συμβολοσειρά για εμφάνιση. Στη συνέχεια, το πρόγραμμα εμφανίζει τη συμβολοσειρά και περιμένει να του εισάγουμε την τιμή. Αφού πληκτρολογήσουμε την τιμή και πατήσουμε το enter, τότε το πρόγραμμα προχωράει στην εκτέλεση της επόμενης εντολής. Για παράδειγμα:

```
age=input("What's your age?") #Η εντολή που εκτελούμε.

What's your age? 19 #Εμφάνιση συμβολοσειράς και εισαγωγή τιμής από το χρήστη.
```

```
print("Age is", age) #Εμφάνιση της καταχώρησης.

Age is 19
```

- ***Πολλαπλές δηλώσεις σε μία γραμμή***

Χρησιμοποιώντας το ερωτηματικό (;) σε μία γραμμή, μπορούμε να κάνουμε περισσότερες από μία δηλώσεις. Επιτρέπονται στην ίδια γραμμή πολλαπλές δηλώσεις μόνο αν δεν αφορούν δηλώσεις νέας ομάδας κώδικα.

- ***Πολλαπλές ομάδες δηλώσεων***

Οι δηλώσεις οι οποίες δημιουργούν μια ενιαία ομάδα κώδικα, είναι πιο σύνθετες καταστάσεις όπως class, def, if, και while. Οι δηλώσεις αυτές ξεκινούν στην πρώτη γραμμή με τη λέξη-κλειδί και στο τέλος της γραμμής μπαίνει το σύμβολο της άνω κάτω τελείας (:) και έπειτα συνεχίζουν οι δηλώσεις που συνθέτουν την ομάδα κώδικα.

- ***Μεταβλητές***

Μεταβλητή είναι ένα χαρακτηριστικό που περιέχει ένα αντικείμενο. Το όνομα μπορεί να αποτελείται από οποιονδήποτε χαρακτήρα, εφόσον ο πρώτος χαρακτήρας είναι γράμμα και το όνομα να δεν είναι λέξη-κλειδί. Ορίζοντας μια μεταβλητή, η αποθήκευση τιμής χρησιμοποιεί κάποιο χώρο της μνήμης και έτσι καταλαμβάνει θέσεις μνήμης. Ο διερμηνέας κατατάσσει τη μνήμη ανάλογα με τον τύπο δεδομένων μίας μεταβλητής.

1 Τύποι μεταβλητών:

Είναι συγκεκριμένοι οι τύποι μεταβλητών που υποστηρίζει κάθε γλώσσα. Αναθέτοντας διαφορετικούς τύπους δεδομένων για τις μεταβλητές υπάρχει η δυνατότητα να αποθηκευτούν δεκαδικοί (decimals), ακέραιοι (integers) ή χαρακτήρες (characters). Στον ακόλουθο πίνακα αναφέρονται οι πιο συνηθισμένοι τύποι της Python:

Τύποι Μεταβλητών	Χρήση
Boolean	Για εκφράσεις αληθείς ή ψευδείς
Float	Για πραγματικούς αριθμούς (αριθμοί κινητής υποδιαστολής)
Integer (int)	Για ακέραιους αριθμούς (θετικοί ή αρνητικοί αριθμοί, αριθμοί χωρις υποδιαστολή)
String (str)	Για συμβολοσειρές (ή αλφαριθμητικά)

Πίνακας 1. Οι τύποι μεταβλητών και η χρήση τους

Επίσης σε κάποιες από της εκδόσεις της Python μπορεί να συναντήσουμε και έναν άλλο τύπο μεταβλητής, τον τύπο long, ο οποίος χρησιμοποιείται για πολύ μεγάλους ακέραιους αριθμούς. Βέβαια στις τελευταίες εκδόσεις της έχουν ως τύπο μεταβλητής τον integer ακόμα και οι μεγάλοι αριθμοί. Εφόσον δοθεί η αξία, δεν χρειάζεται να ορίσουμε το είδος της μεταβλητής. Με την καταχώρηση τιμής σε μία μεταβλητή γίνεται αυτόματα η δήλωση για να κρατηθεί χώρος μνήμης. Για παράδειγμα, a=4, η μεταβλητή a δηλώθηκε ως integer. Για να ελέγξουμε τον τύπο της μεταβλητής, αρκεί να εκτελέσουμε την εντολή type (όνομα_μεταβλητής). Για τη θέση μνήμης μιας μεταβλητής χρησιμοποιούμε τη συνάρτηση id(όνομα_μεταβλητής). Τέλος, χρησιμοποιώντας τους τελεστές "is" και "is not" μπορούμε να συγκρίνουμε τις θέσεις μνήμης δύο μεταβλητών. Τότε ο τελεστής "is" επιστρέφει:

- με True αν οι μεταβλητές εμφανίζουν στις ίδιες θέσεις μνήμης και
- με False αν δεν εμφανίζουν τις ίδιες θέσεις μνήμης.

Ενώ, ο τελεστής "is not" επιστρέφει:

- με False αν οι μεταβλητές εμφανίζουν στις ίδιες θέσεις μνήμης και
- με True αν δεν εμφανίζοουν τις ίδιες θέσεις μνήμης.

2 *Εύρος μεταβλητών:*

Τα είδη μεταβλητών είναι δύο, υπάρχουν οι τοπικές και οι καθολικές μεταβλητές. Η καθολική (global) μεταβλητή μπορεί να χρησιμοιποιηθεί σε κάθε εύρος (scope), δηλαδή έχει τη δυνατότητα να χρησιμοποιηθεί από οπουδήποτε μέσα στον κώδικα. Είναι η μεταβλητή που δεν δηλώνεται μέσα σε κάποια συνάρτηση αλλα στον κύριο κώδικα. Ενώ η τοπική (local) μεταβλητή μπορεί να χρησιμοποιηθεί μόνο στο τοπικό εύρος. Τοπική είναι η μεταβλητή που ορίζεται μέσα σε μία συνάρτηση. 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 υπερισχύει. Επιπλέον, κάθε λειτουργία έχει το δικό της τοπικό χώρο ονομάτων (namespace).

- ***Μετατροπή τύπου δεδομένων***

Υπάρχουν περιπτώσεις που ίσως χρειαστεί να γίνουν μετατροπές στους υπάρχοντες τύπους δεδομένων. Η μετατροπή αυτή πραγματοποιείται, αν χρησιμοποιήσουμε ως συνάρτηση το όνομα του τύπου δεδομένων. Οι συναρτήσεις αυτές δηλώνουν μία νέα μεταβλητή η οποία είναι αυτή που αντιπροσωπεύει την τιμή μετά τη μετατροπή.

Συνάρτηση	**Περιγραφή**
chr (x)	Μετατρέπει τον 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 μια 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 έναν μονό χαρακτήρα στην ακέραιη τιμή του.
repr (x)	Μετατρέπει τον x σε συμβολοσειρά έκφρασης.
set (s)	Μετατρέπει το s σε ένα σύνολο.
str (x)	Μετατρέπει τον x σε μία αναπαράσταση συμβολοσειράς.
tuple (s)	Μετατρέπει το s σε μία πλειάδα.
unichr (x)	Μετατρέπει έναν ακέραιο αριθμό σε ένα χαρακτήρα unicode.

Πίνακας 2.Περιγραφή συναρτήσεων

- ***Σταθερές***

  Οι ορισμένες μαθηματικές σταθερές (constants) της Python είναι:

  - e, η βάση του φυσικού λογαρίθμου logex και
  - pi, η σταθερά π=3.1415….

- ***Τελεστές***

Τελεστής ονομάζεται το σύμβολο το οποίο αναπαριστά μία πράξη. Οι τύποι τελεστών που υποστηρίζει η Python είναι οι ακόλουθοι:

1) Αριθμητικούς
2) τελεστές σύγκρισης
3) σύνθετους τελεστές, καθώς και άλλους τελεστές.

1) Αριθμητικοί τελεστές

Αριθμητικοί ονομάζονται εκείνοι οι τελεστές που αντικαθιστούν τις βασικές πράξεις. Για παράδειγμα:

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

Πίνακας 3. Λειτουργία αριθμητικών συντελεστών

2) Συγκριτικοί τελεστές

Συγκριτικοί ονομάζονται εκείνοι οι τελεστές οι οποίοι συγκρίνουν χαρακτήρες και επιστρέφουν ως αποτέλεσμα: με True, αν είναι αληθής, ή με False, αν είναι ψευδής. Οι τελεστές είναι οι εξής:

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

Πίνακας 4.Λειτουργία συγκριτικών τελεστών

3) Σύνθετοι τελεστές

Σύνθετοι ονομάζονται οι τελεστές που συνδυάζουν δύο τελεστές και χρησιμοποιούνται για τη συντόμευση μίας πράξης.

Σύντομη πράξη	**Η πράξη αναλυτικά**
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

4) Δυαδικός τελεστής

Ο δυαδικός τελεστής (bitwise operator) λειτουργεί χρησιμοποιόντας bits και η εκτέλεση της λειτουργίας γίνεται σε κάθε ένα bit (bit by bit). Δηλαδή αρχικά οι αριθμοί μετατρέπονται σε δυαδική μορφή, έπειτα εκτελούνται οι πράξεις και στη συνέχεια εμφανίζεται το αποτέλεσμα στο δεκαδικό σύστημα. Οι τελεστές που υποστηρίζει η Python είναι:

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

Πίνακας 6. Λειτουργία δυαδικών τελεστών

5) Τελεστές συμβολοσειρών

Κάποιοι τελεστές ξεκινούν με το σύμβολο επί τοις εκατό (%). Αυτοί οι τελεστές χρησιμοποιούνται για να εμφανίσουν ένα αντικείμενο σε μία συμβολοσειρά και εισάγονται στην εντολή printf(). Οι τελεστές αυτοί είναι:

Τελεστής	**Περιγραφή**
%c	Απεικονίζεται ο χαρακτήρας
%d	Απεικονίζεται (signed) ο δεκαδικός ακέραιος αριθμός (και αρνητικό αριθμό).

%E	Εκθετικός συμβολισμός (κεφαλαίο γράμμα)
%e	Εκθετικός συμβολισμός (πεζό γράμμα)
%f	Απεικονίζεται ο πραγματικός αριθμός κινητής υποδιαστολής
%g	Μικρότερος από το %f και το %e.
%G	Μικρότερος από το %Φ και το %E.
%i	Απεικονίζεται (signed) ο δεκαδικός ακέραιος αριθμός (και αρνητικό αριθμό)
%o	Απεικονίζεται ο οκταδικός ακέραιος αριθμός.
%s	Μετατρέπεται η συμβολοσειρά μέσω της συνάρτησης (str) πριν τη διαμόρφωση.
%u	Απεικονίζεται (unsigned) ο δεκαδικός ακέραιος αριθμός (αλλά όχι αρνητικό αριθμό).
%x	Απεικονίζεται ο δεκαεξαδικός ακέραιος αριθμός (πεζό γράμμα).
%X	Απεικονίζεται ο δεκαεξαδικός ακέραιος αριθμός (κεφαλαίο γράμμα).

Πίνακας 7.Περιγραφή τελεστών συμβολοσειρών

6) Προτεραιότητα τελεστών

Η προτεραιότητα των τελεστών, από τελεστές με υψηλότερη προτεραιότητα προς τους τελεστές με χαμηλότερη προτεραιότητα είναι οι εξής:

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

Πίνακας 8.Προτεραιότητα τελεστών

3.3 Δομές ελέγχου και επανάληψης

3.3.1 Δομή ελέγχου if

Όταν εκτελέσουμε μια ακολουθία εντολών και υπάρχουν όλες εκείνες οι προϋποθέσεις που χρειάζεται μια συγκεκριμένη συνθήκη, τότε χρησιμοποιείται η δομή if κι έπειτα η συνθήκη που ελέγχεται. Όταν η συνθήκη είναι αληθής, τότε θα εκτελεστούν όλες οι εντολές που περιέχονται στην if, ειδάλλως το πρόγραμμα θα συνεχίσει από το τέλος της if. Σε περίπτωση που υπάρχουν και άλλες επιλογές τότε χρησιμοποιείται το if...else ή το if...elif...else. Η δομή είναι η εξής:

```
if συνθήκη
then εντολές
else
    εντολές
end
```

Και

```
if συνθήκη 1
then   εντολές
elif συνθήκη2
    εντολές
else
    εντολές
end
```

Παράδειγμα:

```
x = 1;
if x >5
then disp('x<5')
else
disp('x<5')
end
x<5
```

3.3.2 Δομές επανάληψης

Βρόγχος (επανάληψης) είναι μια ακολουθία από εντολές που μπορούν να εκτελεστούν πολλές διαδοχικές φορές, παρόλο που δηλώνονται μία φορά. Ο κώδικας που βρίσκεται μέσα στον βρόγχο (σώμα του βρόγχου) εκτελείται για συγκεκριμένες επαναλήψεις ή για όσο ισχύει μια συνθήκη. Έτσι, χωρίζονται στους βρόχους for και while.

1. *Βρόγχος for*

   Ο βρόγχος for εκτελείται για συγκεκριμένο πλήθος φορών. Οι εντολές εκτελούνται εφόσον ικανοποιείται η συνθήκη. Αν παραλειφθεί το βήμα, εννοείται ότι ισούται με 1. Αυτό μεταφράζεται σε:

```
for δείκτης = αρχική τιμή: βήμα: τελική τιμή
    εντολές
end
```

Για παράδειγμα:

```
 >> d=0;
>> for i=1:2:10
d = d + 1;
end
>> disp(d) 5
```

Σε κάποια προγράμματα είναι απαραίτητη η εισαγωγή βρόχου μέσα σε άλλο βρόχο και αυτοί ονομάζονται πολλαπλοί ή ένθετοι βρόχοι (nested loops).

2. *Βρόγχος while*

Ο βρόγχος while λειτουργεί συνεχώς όσο πληρείται μια συγκεκριμένη συνθήκη και οι εντολές εκτελούνται εφόσον η συνθήκη είναι αληθής.:

```
while συνθήκη:
    εντολές
end
```

Για παράδειγμα:

```
>> n=5;
while n>0
str='Hi!';
n=n-1;
disp(str)
end
Hi!
Hi!
```

Hi!

Hi!

Hi!

3.3.3 Η δήλωση break

Η δήλωση break χρησιμοποιείται για την εξαγωγή του προγράμματος από έναν βρόγχο μόλις συναντήσει αυτή την δήλωση. Πρέπει να πληρείται μια προϋπόθεση για να μπορεί να εκτελεστεί και για αυτό το λόγο χρησιμοποιείται έπειτα από μια δομή if η οποία καθορίζει πότε αυτή εκτελείται.

3.3.4 Η δήλωση with

Η with μπορεί να έχει διάφορες χρήσεις. Χρησιμοποιείται για να ανοίξει ένα αρχείο και να ελέγξει το επιτυχές άνοιγμα αυτού του αρχείου. Επιπλέον εξασφαλίζει την κατάλληλη δέσμευση και αποδέσμευση των πόρων από ένα μέρος μπλοκ κώδικα όταν κλείνει, χωρίς να παίζει ρόλο το αν έχουν συμβεί ή όχι εξαιρέσεις. Αρκετά συχνή χρήση της with είναι όταν κάποια πληροφορία διαβάζεται από ένα αρχείο και έπειτα γράφεται σε ένα άλλο, αφού πρώτα έχει γίνει επεξεργασία στο αρχείο από το οποίο διαβάζεται. Για παράδειγμα:

```
with open (‘filename, ’mode’) as f1:
    εντολές
```

Με την παραπάνω δήλωση επιτυγχάνεται το κλείσιμο του αρχείου αυτόματα ύστερα από το ένθετο μπλοκ κώδικα. Αυτό είναι και το πλεονέκτημα της δήλωσης αυτής, το γεγονός ότι το αρχείο θα κλείσει άσχετα με το πόσες εντολές έπονται. Σε διαφορετική περίπτωση, αυτό επιτυγχάνεται με την εκτέλεση της εντολής close().

Με τη δήλωση with εξασφαλίζεται η σωστή δέσμευση και αποδέσμευση των πόρων, για το λόγο αυτό θεωρείται πολύ χρήσιμη η εντολή.

3.4 Συναρτήσεις

Η συνάρτηση χρησιμοποιείται σε οποιοδήποτε σημείο κώδικα και εκτελείται παραπάνω από μία φορά. Μία συνάρτηση για να λειτουργεί πρέπει να ακολουθεί τους εξής κανόνες :

1. Πρέπει να αρχίζει με τη λέξη-κλειδί και να ακολουθείται από το όνομά της, παρενθέσεις και το σύμβολο της άνω και κάτω τελείας. Ανάμεσα στις παρενθέσεις καθορίζονται τυχόν παράμετροι εισόδου.

2. Ανάμεσα στις παρενθέσεις καθορίζονται τυχόν παράμετροι εισόδου.

3. Στο μπλοκ κώδικα της συνάρτησης, που ξεκινάει μετά την άνω και κάτω τελεία, πρέπει να δημιουργείται εσοχή.

4. Η δήλωση επιστροφής (return) χρησιμοποιείται για να επιστρέψει επιχειρήματα (arguments) και είναι προαιρετική. Η δήλωση return χωρίς επιχειρήματα είναι ίδια με την δήλωση return None. Για παράδειγμα:

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

Η εκτέλεση της συνάρτησης μπορεί να γίνει είτε από το κύριο πρόγραμμα είτε από κάποια άλλη συνάρτηση όπου χρησιμοποιείται το όνομα της συνάρτησης μόνο του ή διαφορετικά με παραμέτρους. Για παράδειγμα:

```
όνομα_συνάρτησης()

ή

μεταβλητή = όνομα_συνάρτησης()
```

3.4.1 Ανώνυμες συναρτήσεις

Ανώνυμη συνάρτηση (anonymous function) καλείται μία συνάρτηση που δεν έχει δηλωθεί με τον πρότυπο τρόπο, δηλαδή χρησιμοποιώντας τη λέξη-κλειδί def. Μία ανώνυμη συνάρτηση διαθέτει τα εξής χαρακτηριστικά:

- Δημιουργείται με τη λέξη-κλειδί lambda και παρέχει τη δυνατότητα ορισμού οποιουδήποτε αριθμού επιχειρημάτων εισόδου, όμως μπορεί να επιστρέψει ένα μόνο αποτέλεσμα με τη μορφή έκφρασης.

- Δεν μπορεί να περιέχει εντολές ή περισσότερες από μία έκφραση.

- Δεν μπορεί να επιτευχθεί απευθείας κλήση, διότι απαιτεί τη δήλωση μίας έκφρασης.

- Παρέχει το δικό της τοπικό χώρο ονομάτων (namespace). Οι μεταβλητές στις οποίες διαθέτει πρόσβαση είναι οι παγκόσμιες μεταβλητές κι εκείνες που ορίζονται στη λίστα των παραμέτρων της.

Η δομή της είναι η εξής:

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

Παράδειγμα:

```
>>> lambda x, v: x+v

<function < lambda> at 0x0000005935DF01E0>
```

```
>>> sum=lambda x, v: x+v

>>> print("The sum is .", sum(25, 10))

The sum is 35.
```

3.5 Βασικές δομές

Βασικές δομές είναι εκείνα τα αντικείμενα που περιέχουν άλλα αντικείμενα. Οι βασικότερες δομές της Python είναι οι εξής: τα Αλφαριθμητικά (String), οι Λίστες (Lists), τα Σύνολα (Sets), Πλειάδα (Tuple) και Λεξικό (Dictionary).

3.5.1 Αλφαριθμητικά (Strings)

Τα Αλφαριθμητικά είναι μία ακολουθία χαρακτήρων. Αποτελείται από διάφορους χαρακτήρες και δεν υπάρχει περιορισμός στον αριθμό χαρακτήρων που θα χρησιμοποιηθούν. Για να οριστούν θα πρέπει να βρίσκονται ανάμεσα σε μονούς (' ') ή διπλούς (" ") αποστρόφους. Ωστόσο, μπορούμε να χρησιμοποιήσουμε και τριπλά εισαγωγικά (''' ''' ή """ """), τα οποία χρησιμοποιούνται για τον ορισμό συμβολοσειράς που καλύπτει παραπάνω από μία γραμμή.

Τα Strings υποστηρίζουν πολλούς μεθόδους όπως:

Μέθοδος	**Περιγραφή**
astring.capitalize()	η οποία εμφανίζει τη συμβολοσειρά με κεφαλαίο το πρώτο γράμμα.
astring.count('x')	η οποία υπολογίζει πόσες φορές βρίσκεται το x μέσα στη συμβολοσειρά.
astring.find('x')	η οποία βρίσκει τη θέση του χαρακτήρα

	x.
astring.lower()	η οποία εμφανίζει την συμβολοσειρά με πεζούς χαρακτήρες, χωρίς να την αποθηκεύει.
len(astring)	την η οποία εμφανίζει το μέγεθος της συμβολοσειράς.
astring.replace('a', 'b')	η οποία αντικαθιστά τους χαρακτήρες a και b.
astring.strip('a')	η οποία εμφανίζει τη συμβολοσειρά έχοντας αφαιρέσει το a από την αρχή και το τέλος της συμβολοσειράς.
astring.title	η οποία εμφανίζει αντίγραφο της συμβολοσειράς με κάθε λέξη να ξεκινά με κεφαλαίο γράμμα.
astring.upper()	η οποία εμφανίζει τη συμβολοσειρά με κεφαλαίους χαρακτήρες χωρίς να την αποθηκεύει.

Πίνακας 9. Μέθοδοι Strings

Μπορούμε, επίσης, να καθορίσουμε τον αριθμό των χαρακτήρων που επιθυμούμε, χρησιμοποιώντας αγκύλες, [ ], ή [:]. Το ευρετήριο ξεκινάει από το 0 για την αρχή της συμβολοσειράς, αν θέλουμε να ξεκινήσουμε από το τέλος χρησιμοποιούμε το -1.

Για την εμφάνιση ενός μέρους της συμβολοσειράς, η εντολή που θα εισάγουμε θα είναι ως εξής: print(όνομα_συμβολοσειράς[αρχή:τέλος])

Με το σύμβολο συν (+) μπορούμε να ενώσουμε δύο συμβολοσειρές.

Για παράδειγμα:

```
>>> astring="Hello "
>>> print(astring + "World")
Hello World
```

Το σύμβολο του αστερίσκου (*) μπορούμε να το χρησιμοποιήσουμε για την επανάληψη μίας συμβολοσειράς.

Για παράδειγμα:

```
>>> print(astring * 2)
Hello Hello
```

Μπορούμε να χρησιμοποιήσουμε τους τελεστές in και not in για να διαπιστώσουμε αν κάποιος χαρακτήρας ή συμβολοσειρά υπάρχει ή δεν υπάρχει, αντίστοιχα, στη συμβολοσειρά μας. Η απάντηση που θα λάβουμε είναι True, για αληθής, ή False, για ψευδής. Για παράδειγμα:

```
>>> astr="Hello World!"
>>> bstr="lo"
>>> bstr in astr
True
```

3.5.2 Λίστες

Η Λίστα αποτελείται από διάφορους τύπους δεδομένων και για αυτό χαρακτηρίζεται ως δυναμική. Μπορεί να περιέχει αγκύλες, [ ], και τα στοιχεία της χωρίζονται με κόμμα. Τα στοιχεία της έχουν συγκεκριμένη σειρά και για να προβούμε σε αυτά χρησιμοποιούμε το id του στοιχείου που βρίσκεται μέσα στην αγκύλη (list[0], list[1], ..). Επίσης αν ο δείκτης είναι αρνητική τιμή τότε ξεκινάει να μετράει από το τέλος της λίστας. Για παράδειγμα:

```
>>>list1=[4, 8, 'Hellen', 89, 'Petter' ]
>>>print ("the third element of the list is", list1[2])
the third element of the list is 'Hellen'
```

Κάποιες από τις συναρτήσεις που αποτελούνται οι Λίστες είναι οι εξής :

Συνάρτηση	**Περιγραφή**
cmp(list1,list2)	η οποία συγκρίνει δύο λίστες.
len(list1)	η οποία επιστρέφει το μέγεθος της λίστας.
max(list1)	η οποία επιστρέφει τη μέγιστη τιμή της λίστας.
min(list1)	η οποία επιστρέφει την ελάχιστη τιμή της λίστας.
list (s)	Η οποία μετατρέπει μια πλειάδα σε λίστα.

Πίνακας 10. Συναρτήσεις Λίστας

Και κάποιες από τις μεθόδους τους είναι οι ακόλουθες:

Μέθοδος	**Περιγραφή**
alist.uppend()	Προσθέτει στοιχεία στο τέλος της λίστας.
alist.clear()	Αφαιρεί όλα τα στοιχεία από τη λίστα. Ισοδυναμεί με : del a[:].
Alistcopy()	Δημιουργεί αντίγραφο στο τέλος της λίστας.
alist.count(“x”)	Εμφανίζει τον αριθμό των εμφανίσεων του x στη λίστα.
alist.extend(blist)	Προσθέτει το περιεχόμενο της blist στην alist.
alist.index(“x”)	Εμφανίζει το δείκτη του x στη λίστα.
alist.insert(id, ”x”)	Προσθέτει το στοιχείο x στη θέση id.
alist.pop()	Εμφανίζει το τελευταίο στοιχείο (ή το στοιχείο της θέσης που θα εισάγουμε) και στη συνέχεια το διαγράφει από τη λίστα.
alist.remove(“x’)	Διαγράφει το πρώτο στοιχείο x από τη λίστα.
alist.reverse()	Αντιστρέφει τα στοιχεία της λίστας.
alist.sort()	Ταξινομεί τη λίστα κατά αύξουσα σειρά.

Πίνακας 11. Μέθοδοι Λίστας

Τα σύμβολα συν (+) και αστερίσκος (*) ισχύουν και για τις λίστες, ως τελεστής συνένωσης και τελεστής επανάληψης αντίστοιχα. Για να διαγράψουμε στοιχεία από μία λίστα γίνεται με τη δήλωση del όταν γνωρίζουμε ακριβώς ποιο στοιχείο είναι. Αλλιώς, αν δεν γνωρίζουμε ποιο στοιχείο είναι χρησιμοποιούμε τη μέθοδο remove(). Για παράδειγμα:

```
>>>list1=[4, 8, 'Hellen', 89, 'Petter' ]
>>>print ("the third element of the list is", list1[2])
the third element of the list is 'Hellen'
>>>del list1[2]
>>>print ("The list is",list1)
The list is    [4, 8, 89, 'Petter']
>>>
```

3.5.3 Σύνολα (Sets)

Τα σύνολα χρησιμοποιούνται για να ομαδοποιηθούν πολλά αντικείμενα κι έπειτα να εφαρμοστούν πράξεις όπως το να ενωθούν με αποδοτικό τρόπο. Είναι αυτά που εξασφαλίζουν πως κάθε στοιχείο, αν βρίσκεται σε παραπάνω από ένα σύνολα, τελικά θα βρεθεί μόνο μια φορά στο τελικό αποτέλεσμα. Βρίσκεται μέσα σε άγκιστρο, { }, ή καθορίζεται με τη συνάρτηση set και βρίσκεται μέσα σε αγκύλες και παρενθέσεις, ([ ]). Τα στοιχεία του συνόλου πρέπει να βρίσκονται ανάμεσα σε μονούς (' ') ή διπλούς (" ") αποστρόφους. Για παράδειγμα:

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

Οι λειτουργίες που υποστηρίζουν τα σύνολα είναι οι εξής :

Λειτουργία	**Περιγραφή**
asset.add(“x”)	Για την προσθήκη στοιχείου x σε ένα σύνολο.
asset.copy()	Για αντίγραφο του συνόλου asset.
asset.remove(“x”)	Για την αφαίρεση στοιχείου x από ένα σύνολο.
len(aset)	Για την εμφάνιση το μέγεθος του συνόλου.
“x” in aset	Για να ελέγχει αν το x υπάρχει στο σύνολο asset.
“x” not in asset	Για να ελέγχει αν το x δεν υπάρχει στο σύνολο asset.

Πίνακας 12. Λειτουργίες Συνόλων

Για παράδειγμα:

```
>>> print('The length of first set is ', len(set1))
The length of first set is 3
>>> 'orange' is set1
False
>>> set1.add('orange')
>>> print(set1)
{'yellow', 'pink', 'green', 'orange'}
>>> set1.remove('pink')
```

```
>>> print(set1)

{'yellow', 'green', 'orange'}
```

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)	Έλεγχος αν όλα τα στοιχεία του συνόλου a υπάρχουν στο σύνολο b.
a >= b	a.issuperset(b)	Έλεγχος αν όλα τα στοιχεία του συνόλου b υπάρχουν στο σύνολο a.

Πίνακας 13. Πράξεις συνόλων

Για παράδειγμα:

```
>>>set3= set1|set2

>>>print ("the elements of third set are", set 3)

The elements of third set are {'red', 'purple', 'brown', 'pink', 'orange', 'blue'}

>>>set1<=set3

True
```

3.5.4 Πλειάδα

Η Πλειάδα είναι μια σειρά από τιμές, οι οποίες περικλείονται από παρενθέσεις (). Περιέχει αντικείμενα τα οποία συσχετίζονται μεταξύ τους, είναι σταθερή, δηλαδή δεν μπορεί να αλλάξει ούτε στοιχεία, ούτε μέγεθος και έχει τη δυνατότητα να περιέχει οποιοδήποτε τύπο δεδομένων. Χρησιμεύει για να επιστραφούν πολλές τιμές σε συναρτήσεις. Πλειάδα και λίστα έχουν της εξής διαφορές:

1. η πλειάδα βρίσκεται μέσα σε παρενθέσεις, ενώ η λίστα βρίσκεται μέσα σε αγκύλες

2. Η πλειάδα σε αντίθεση με την λίστα δεν μπορεί να αλλάξει τα στοιχεία και το μέγεθός της.

Θα μπορούσαμε να πούμε πως η πλειάδα θεωρείται ως λίστα μόνο για ανάγνωση (readonly). Δεν γίνεται να αφαιρεθούν στοιχεία πλειάδας. Για να διαγραφεί ολόκληρη η πλειάδα χρησιμοποιείται η εντολή del.

Παράδειγμα:

>>>tuple1=(‘aplles’, 789, ‘door’)

>>>print (“the elements of tuple are”, tuple1)

The elements of tuple are (‘aplles’, 789, ‘door’)

>>>del tuple1

>>>print (tuple1)

Traceback (most recent call last) :

File “<pyshell#3>”, line 1, in <module>

print (tuple1)

NameError: name ‘tuple1’ is not defined

>>>

Ακολουθούν οι συναρτήσεις που μπορούν να εφαρμοστούν σε μία πλειάδα:

Συνάρτηση	**Περιγραφή**
cmp(t1, t2)	Σύγκριση δύο πλειάδων.
len(t1)	Εμφάνιση μεγέθους πλειάδας
max(t1)	Εμφάνιση το μέγιστο στοιχείο της πλειάδας.
min(t1)	Εμφάνιση το ελάχιστο στοιχείο της πλειάδας.
tuple(l1)	Μετατροπή μίας λίστας σε πλειάδα

Πίνακας 14. Εισαγωγή Πλειάδας

3.5.5 Λεξικό

Το λεξικό αποτελείται από ζευγάρια κλειδιών (keys) και αξιών (values). Η αξία μπορεί να είναι οποιουδήποτε τύπου και αντιστοιχεί σε ένα κλειδί. Μπορεί να είναι αλφαριθμητικό, αριθμός ή Πλειάδα, ενώ οι αντίστοιχες τιμές μπορούν να αποτελούν οποιοδήποτε αντικείμενο της Python. Η σειρά των στοιχείων δεν είναι σταθερή, δηλαδή είναι μη ταξινομημένα, επαναληπτικά και μεταβλητά. Όμως αυτό δεν αποτελεί πρόβλημα, διότι η αναζήτηση επιτυγχάνεται με την τιμή του κλειδιού.

Το Λεξικό βρίσκεται ανάμεσα σε άγκιστρα, { }. Για την εισαγωγή τιμής ή το δικαίωμα πρόσβασης σε μία τιμή χρησιμοποιούνται οι αγκύλες, [ ]. Για τη διαγραφή στοιχείων λεξικού η όλου του λεξικού χρησιμοποιείται η εντολή del.

Για παράδειγμα:

```
>>>dict1={‘Name’:’Anna’, ‘age’:24, ‘Work’:’Teacher’}

>>>print (“The elements of dictionary are”, dict1)

The elements of dictionary are {‘Work’:’Teacher’ ‘Name’:’Anna’, ‘age’:24}

>>>del dict1 [‘Age’]

>>>print (“The elements of dictionary are”, dict1)

The elements of dictionary are {‘Work’:’Teacher’ ‘Name’:’Anna’}

>>>del dict1

>>>print (dict1)

Traceback (most recent call last) :

    File “<pyshell#5>”, line 1, in <module>

      print (dict1)

NameError: name ‘dict1’ is not defined

>>>
```

Τέλος, υπάρχουν οι ακόλουθοι μέθοδοι λεξικού :

Μέθοδος	**Περιγραφή**
adict.clear()	Για την διαγραφή όλων των στοιχείων του λεξικού.
adict = dict.copy()	Για την αντιγραφή του λεξικού.

del adict["x"]	Για την διαγραφή του ζευγαριού x
adict = dict.fromkeys(aset)	Για τη δημιουργία ενός νέου λεξικού με κλειδιά τις τιμές του συνόλου aset και αξίες None (εκτός κι αν οριστεί κατά τη δημιουργία του λεξικού),
adict.items()	Για την εμφάνιση μιας λίστας από κλειδί και αξία.
adict.keys()	Για την εμφάνιση των κλειδιών του λεξικού.
len(adict)	Για την εμφάνιση του μεγέθους του λεξικού.
adict.update(bdict)	Για την πρόσθεση του λεξικού bdict στο λεξικό addict
adict.values()	Για την εμφάνιση των τιμών του λεξικού.

Πίνακας 15. Μέθοδοι Λεξικών

3.6 Κλάση

Κλάση είναι η δομή που ενσωματώνει αντικείμενα (objects) που διαθέτουν χαρακτηριστικά (attributes) και μεθόδους (methods). Η κλάση αποτελείται από ορισμούς αντικειμένων και μεθόδων. Είναι χρήσιμη, διότι ο κώδικας μπορεί να επαναχρησιμοποιηθεί, επιπλέον όταν θέλουμε να τροποποιήσουμε κάτι αλλάζουμε μόνο το σημείο αυτό.

Η σύνταξή της έχει ως εξής, γράφουμε τη λέξη-κλειδί class και το όνομά της και το σύμβολο της άνω και κάτω τελείας (:). Στη συνέχεια, ορίζουμε δεδομένα και τυχόν μεθόδους για τα αντικείμενα της κλάσης.

Η δομή είναι η εξής:

```
Class όνομα_κλάσης:

    αντικείμενα και μέθοδοι
```

3.6.1 Αντικείμενο κλάσης

Η δημιουργία ενός αντικειμένου ονομάζεται δημιουργία στιγμιότυπου (Instance). Για την εκχώρηση τιμής σε ένα στιγμιότυπο χρησιμοποιούμε το συμβολισμό με τελεία. Τα περιεχόμενα ενός αντικειμένου θα πρέπει να προσεγγιστούν ως μεταβλητές αντικειμένου με τη χρήση τελείας.

3.6.2 Μέθοδος

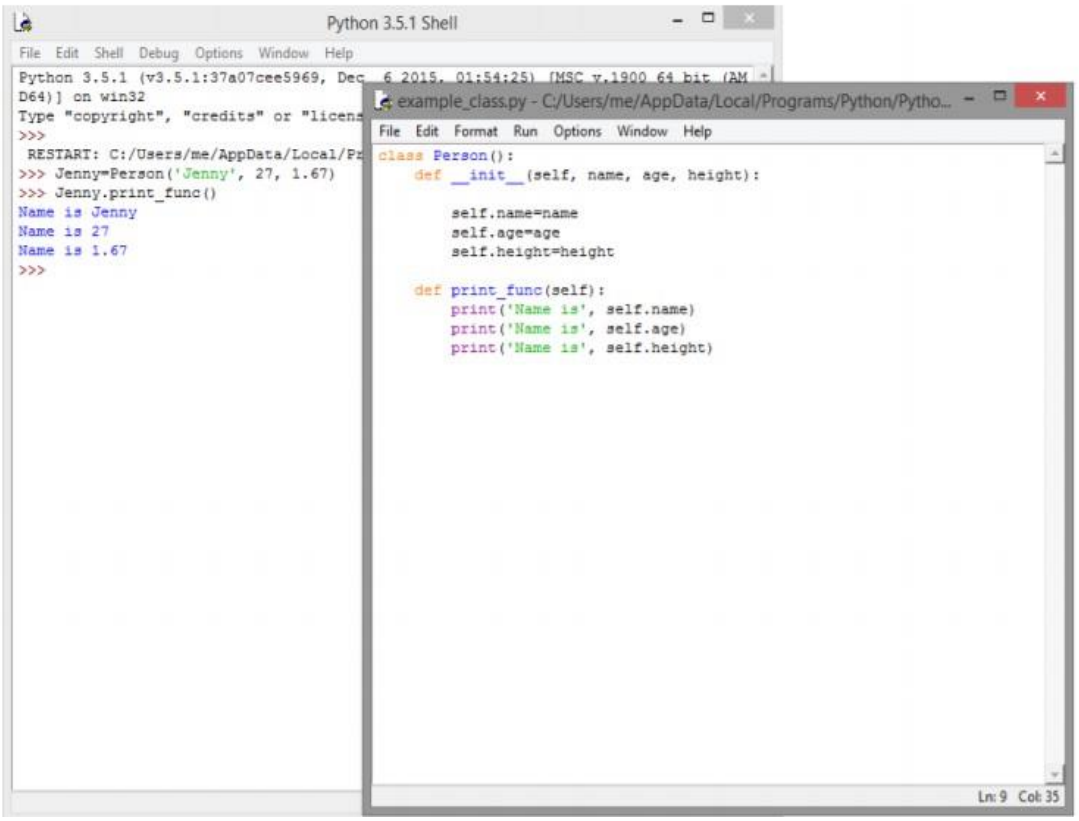
Για την περιγραφή των διαφόρων συμπεριφορών ενός αντικειμένου, χρησιμοποιείται η μέθοδος. Καθορίζεται με τη λέξη κλειδί def η οποία περιέχεται στο σώμα μίας κλάσης. Ως πρώτο όρισμα, κάθε μέθοδος μίας κλάσης πρέπει να έχει την παράμετρο self που αναφέρεται στο αντικείμενο της κλάσης. Μία μέθοδος έχει τη δυνατότητα 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
```

3.6.3 Παράμετρος self

Η μέθοδος μίας κλάσης πρέπει να έχει ένα επιπλέον όρισμα, το οποίο πρέπει να προστεθεί στην αρχή της λίστας με τις παραμέτρους. Η παράμετρος που αναφέρεται είναι η μεταβλητή self. Κατά την κλήση μίας μεθόδου δε χρειάζεται να ορίσουμε τιμή στο όρισμα self, διότι γίνεται αυτόματα. Το όνομα self δεν είναι υποχρεωτικό, είναι το όμως το επικρατέστερο, καθώς και αναγνωρίσιμο. Για να αναφερθούμε σε ένα χαρακτηριστικό μίας κλάσης ή σε μία μέθοδο, πρέπει να χρησιμοποιήσουμε το πρόθεμα self. (με τελεία).



Εικόνα 2. Παράδειγμα Κλάσης

3.6.4 Ενσωματωμένα χαρακτηριστικά

Κάθε κλάση διαθέτει κάποια ενσωματωμένα χαρακτηριστικά, στα οποία αποκτά πρόσβαση με τη χρήση της τελείας. Τα χαρακτηριστικά αυτά είναι:

- __dict__, Είναι ένα λεξικό το οποίο περιέχει το όνομα και την τιμή κάθε χαρακτηριστικού της κλάσης. Για την εμφάνισή τους πληκτρολογούμε το όνομα του αντικειμένου, τελεία και __dict__ .

- __doc__, Είναι οι συμβολοσειρές τεκμηρίωσης της Python. Ουσιαστικά, είναι η λειτουργία που κάνει η εντολή help(), μόνο που η εντολή help τα εμφανίζει με πιο ομοιόμορφο τρόπο.

- __name__, Εμφανίζει το όνομα της κλάσης.

- __module__, Όνομα ενότητας στο οποίο καθορίζεται η κλάση. Αυτό το χαρακτηριστικό είναι __main__σε διαδραστική λειτουργία.

- __bases__, Μία πλειάδα που περιέχει τις βασικές κλάσεις με τη σειρά που συνέβησαν στη βασική λίστα κλάσης.

3.6.5 Ειδικοί μέθοδοι

Υπάρχουν κάποιες μέθοδοι που υλοποιούνται μέσω των κλάσεων όπως η:

- Μέθοδος __init__(). Προέρχεται από τη λέξη initialization και ονομάζεται κλάση κατασκευαστής ή μέθοδος αρχικοποίησης. Η μέθοδος αυτή, λοιπόν, καλείται για την αρχικοποίηση ενός αντικειμένου. Η μέθοδος δεν καλείται ρητά. Χρησιμοποιείται όταν δημιουργείται ένα νέο αντικείμενο, αναφέροντας τα ορίσματα εντός των παρενθέσεων. Συντάσσεται ως εξής, __init__(self, ...).

- Μέθοδος __del__(). Καλείται για την καταστροφή ενός αντικειμένου και ονομάζεται καταστροφέας (destructor). Αντίστοιχα με τη μέθοδο __init__, πρέπει να καλεστεί όταν μία υπερκλάση την υλοποιεί . Χρησιμοποιείται ως εξής, __del__(self).

- Μέθοδος __repr__(). Δημιουργεί την "επίσημη" αναπαράσταση αλφαριθμητικού ενός αντικειμένου. Η σύνταξή της είναι __repr__(self).

- Μέθοδος __str__(). Ορίζει τον τρόπο εμφάνισης ενός αντικειμένου. Όταν θέλουμε να εμφανίσουμε ένα αντικείμενο μέσω της εντολής print, τότε εμφανίζεται σύμφωνα με τον ορισμό της μεθόδου __str__(self).

Μπορούν να υλοποιηθούν οι ακόλουθες μέθοδοι σύγκρισης:

Μέθοδος	**Περιγραφή**
__eq__(self, other)	ίσο (==)
__ne__(self, other)	άνισο (!=)
__lt__(self, other)	μικρότερο (<)
__gt__(self, other)	μεγαλύτερο (>)
__le__(self, other)	μικρότερο ή ίσο (<=)
__ge__(self, other)	μεγαλύτερο ή ίσο (>=)

Πίνακας 16. Μέθοδοι Σύγκρισης

Επίσης, μπορούν να πραγματοποιηθούν μέθοδοι οι οποίες αντιστοιχούν σε μαθηματικές πράξεις:

Μέθοδος	**Περιγραφή**
__add__(self, other)	πρόσθεση (+)
__sub__(self, other)	αφαίρεση (-)
__mul__(self, other)	πολλαπλασιασμό (*)
__truediv__(self, other)	διαίρεση (/)
__floordiv__(self, other)	διαίρεση ακεραίων (//)
__mod__(self, other)	υπόλοιπο (%)

Πίνακας 17. Μέθοδοι Μαθηματικών Πράξεων

3.6.6 Κληρονομικότητα

Ακόμα ένα χαρακτηριστικό μίας κλάσης είναι η κληρονομικότητα (inheritance). Κληρονομικότητα είναι ένας από τους τρόπους επαναχρησιμοποίησης του κώδικα. Για την ακρίβεια, είναι η δυνατότητα δημιουργίας μίας νέας κλάσης, που θα είναι η τροποποιημένη έκδοση μίας ήδη υπάρχουσας. Η νέα κλάση κληρονομεί τις ιδιότητες και τις μεθόδους της ήδη υπάρχουσας και μπορεί να τις χρησιμοποιήσει αυτούσια ή τροποποιημένα. Η υπάρχουσα κλάση ονομάζεται βασική κλάση (base class), ή αλλιώς υπερκλάση και η νέα κλάση, που κληρονομεί τη βασική κλάση, ονομάζεται παραγόμενη κλάση (derived class), ή υποκλάση.

Ο ορισμός μίας νέας κλάσης είναι ίδιος με τον ορισμό μίας απλής κλάσης, μόνο που θα πρέπει να δηλώσουμε ως όρισμα το όνομα της κλάσης από την οποία θα

κληρονομεί. Μπορούμε εκ νέου να ορίσουμε τον κατασκευαστή __init__, για να ορίσουμε τυχόν νέα χαρακτηριστικά.

3.7 Εξαιρέσεις

Όταν διαταράσσεται η κανονική ροή ενός προγράμματος κατά την εκτέλεσή του τότε δημιουργείται μία εξαίρεση (exception). Όταν το πρόγραμμα βρεί μια κατάσταση που δεν μπορεί να αντιμετωπίσει, τότε εμφανίζεται κάποιο σφάλμα (error). Όταν παρουσιάζεται μία εξαίρεση τότε θα πρέπει να αντιμετωπιστεί άμεσα, διαφορετικά το πρόγραμμα θα τερματιστεί και θα κλείσει. Υπάρχει περίπτωση να εμφανιστεί μία εξαίρεση κατά την προσπάθεια εκτέλεσης του κώδικα, ακόμα και αν μία δήλωση είναι συντακτικά σωστή. Υπάρχουν αρκετά είδη σφαλμάτων:

1. Τα συντακτικά λάθη (syntax errors) ή σφάλματα ανάλυσης, όπου κατά την εκμάθηση της γλώσσας είναι το πιο συχνό σφάλμα που εμφανίζεται. Παρακάτω ακολουθεί παράδειγμα με συντακτικό σφάλμα μη έγκυρης σύνταξης:

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

2. Στο επόμενο παράδειγμα παρουσιάζεται σφάλμα στο όνομα της εντολής, όπου δίνονται πληροφορίες για τη γραμμή που βρίσκεται το λάθος καθώς επισημαίνεται και όνομα που φαίνεται ως μη ορισμένο.

```
print(“Hi!”)
Traceback (most recent call last):
File “<pyshell#0>”, line 1, in <module>
print(“Hi!”)
NameError: name ‘Print’ is not defined
```

Στην τελευταία γραμμή του μηνύματος παρουσιάζονται πληροφορίες για το σφάλμα, καθώς εμφανίζεται και ο τύπος του σφάλματος ως μέρος του μηνύματος. Τέλος, υπάρχουν και άλλες παρόμοιες εξαιρέσεις όπως οι EnvironmentError, EOFError, ImportError, MemoryError, SystemExit, ValueError. Η εξαίρεση αποτελείται από κάποια χαρακτηριστικά τα οποία είναι συμαντικά για το χειρισμό των σφαλμάτων.

3.7.1 Χειρισμός εξαιρέσεων

Όταν αλλάξει η φυσιολογική ροή του προγράμματος τότε προκύπτει ο χειρισμός (handling) των εξαιρέσεων. Πρέπει, δηλαδή, να χειριστούμε με ειδικές συνθήκες τα σφάλματα. Ο τρόπος με τον οποίο λειτουργούν οι εξαιρέσεις είναι ο εξής:

1. Αρχικά αν αλλάξει η συνηθισμένη ροή του προγράμματος τότε θα δημιουργηθεί μία εξαίρεση.

2. Τότε θα πρέπει να αναγνωριστεί η κλάση της εξαίρεσης, η οποία εμφανιζεται στην τελευταία γραμμή του μηνύματος λάθους (για παράδειγμα: SyntaxError, TypeError και λοιπά).

3. Στη συνέχεια θα πρέπει να εφαρμοστεί ο ειδικός κώδικας εξαίρεσης για τη συγκεκριμένη περίπτωση.

4. Αν δεν αντιμετωπιστεί η εξαίρεση η οποία έχει δημιουργηθεί, τότε θα σταματήσει η εκτέλεση του προγράμματος. Οι εξαιρέσεις οι οποίες δεν έχουν αντιμετωπιστεί, εμφανίζονται στο traceback που αυτό είναι υπέθυνο για την ενημέρωση όσων έχουν συμβεί.

Η δήλωση try-except πραγματοποιείται ακολουθώντας τα εξής βήματα :

- Αρχικά θα πρέπει να ξεκινάει με τη λέξη try, άνω και κάτω τελεία (:) κι έπειτα το μπλοκ κώδικα.

- Στη συνέχεια, θα πρέπει να περιέχει τη λέξη except, το όνομα της εξαίρεσης, άνω και κάτω τελεία (:) και να συνοδεύεται από ένα μπλοκ κώδικα το οποίο χειρίζεται το σφάλμα.

Για τον καθορισμό των διαφόρων εξαιρέσεων, μία δήλωση try μπορεί να περιέχει παραπάνω από μία δήλωση except. Από τις εξαιρέσεις αυτές μόνο ένας χειριστής θα μπορεί να εκτελεστεί. Ο χειριστής ασχολείται μόνο με εξαιρέσεις που προκύπτουν στην αντίστοιχη δήλωση try και όχι σε άλλες δηλώσεις try.

Μπορεί να παραλειφθεί το όνομα εξαίρεσης στην τελευταία δήλωση except και αυτή να χρησιμοποιηθεί ως εξαίρεση "μπαλαντέρ". Όμως, χρειάζεται πολύ προσοχή η χρήση της δήλωσης αυτής. Θα μπορούσε να εμφανιστεί στην περίπτωση ενός μηνύματος λάθους. Επιπλέον, περιέχει έναν προαιρετικό όρο else ο οποίος όταν υπάρχει θα πρέπει να ακολουθεί όλες τις except δηλώσεις. Η δήλωση try-except έχει την εξής δομή:

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

Μπορούμε να συναντήσουμε διάφορα είδη εξαιρέσεων, όπως επίσης, μπορούμε να φτιάξουμε δικές μας εξαιρέσεις. Ακολουθεί μία λίστα με ορισμένες από τις διαθέσιμες εξαιρέσεις:

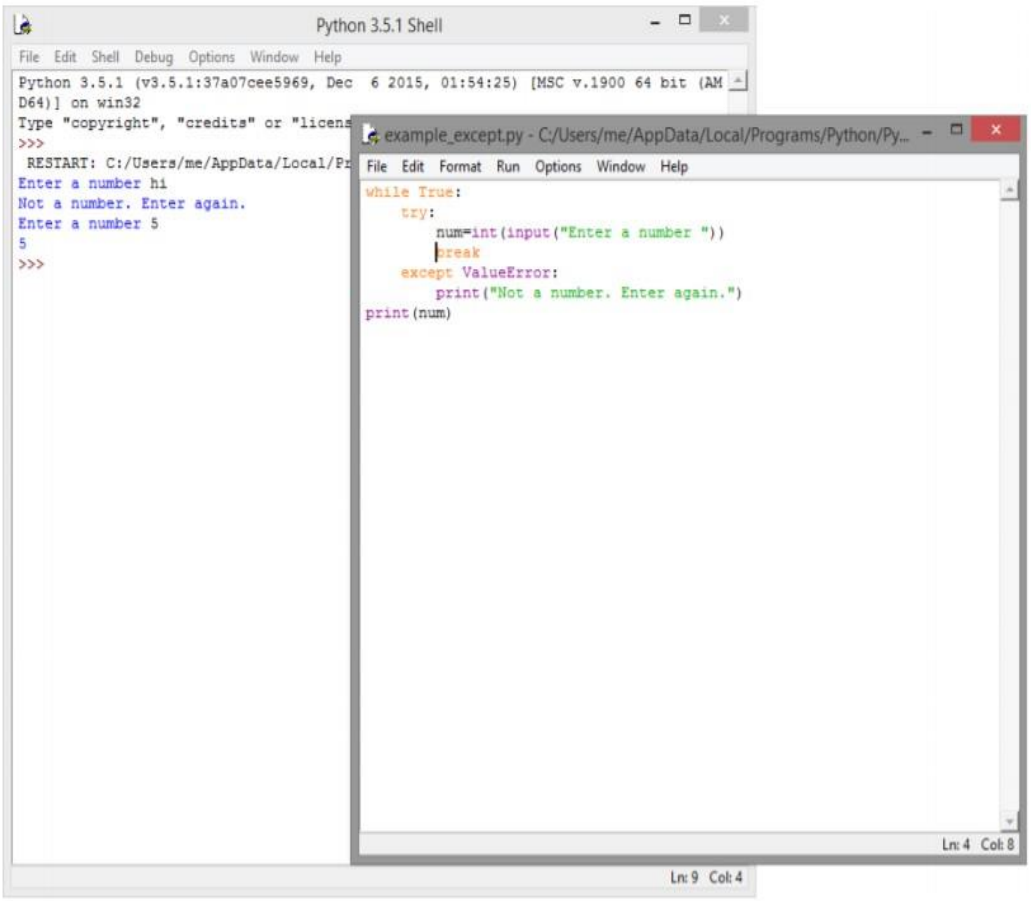
Εξαίρεση	**Περιγραφή**
Exception	Η βασική κλάση για όλες τις εξαιρέσεις. Είτε ενσωματωμένες εξαιρέσεις, είτε ορισμένες από το χρήστη προέρχονται από αυτή την κλάση.
ArithmeticError	Βασική κλάση για τις ενσωματωμένες εξαιρέσεις για διάφορα σφάλματα αριθμητικών υπολογισμών.
AssertionError	Εμφανίζεται όταν μία δήλωση assert αποτύχει.
AttributeError	Στην περίπτωση που μία αναφορά χαρακτηριστικού ή εκχώρηση αποτυγχάνει. (Όταν ένα χαρακτηριστικό δεν υποστηρίζει αναφορές ή εκχωρήσεις χαρακτηριστικού το σφάλμα αποδίδεται σε TypeError.)
EnvironmentError	Βασική κλάση για όλες τις εξαιρέσεις που συμβαίνουν έξω από το περιβάλλον της Python.
EOFError	Όταν η συνάρτηση input δεν έχει διαβάσει δεδομένα και επιτυγχάνεται το τέλος του αρχείου (EOF).
FloatingPointError	Όταν αποτύχει ένας υπολογισμός κινητής υποδιαστολής.
ImportError	Όταν μία δήλωση import αποτύχει να βρει τον ορισμό αρθρώματος.
IndexError	Όταν ο δείκτης δεν βρίσκεται σε κάποια ακολουθία.
KeyError	Όταν το καθορισμένο κλειδί δεν βρίσκεται στο σύνολο

	των υπαρχόντων κλειδιών του λεξικού.
KeyboardInterrupt	Όταν διακοπεί η εκτέλεση του προγράμματος από το χρήστη (συνήθως πατώντας το συνδυασμό των πλήκτρων Ctrl + C).
MemoryError	Θέτεται όταν μία λειτουργία "ξεμείνει" από μνήμη.
NameError	Όταν ένα αναγνωριστικό δεν βρεθεί στο τοπικό ή καθολικό χώρο ονομάτων.
OverflowError	Όταν ένα αποτέλεσμα υπερβαίνει το μέγιστο όριο ενός αριθμητικού τύπου.
RuntimeError	Όταν εντοπίζεται ένα σφάλμα που δεν εμπίπτει σε κάποια από τις άλλες κατηγορίες.
StopIteration	Όταν κατά τη μέθοδο next() ο δείκτης επανάληψης δεν οδηγήσει σε κάποιο αντικείμενο.
SyntaxError	Όταν παρουσιάζεται ένα συντακτικό λάθος.
SystemError	Όταν ο διερμηνέας βρίσκει ένα εσωτερικό σφάλμα. Μία συμβολοσειρά δείχνει ποιο είναι το σφάλμα, σε χαμηλού επιπέδου όρους (low-level terms).
SystemExit	Όταν ο διερμηνέας κλείσει με τη χρήση της συνάρτησης sys.exit().
TypeError	Όταν μία λειτουργία ή συνάρτηση εφαρμοστεί σε ένα αντικείμενο ακατάλληλου τύπου. Η συμβολοσειρά δίνει λεπτομέρειες σχετικά με τον ασυνδύαστο τύπο.
ValueError	Όταν μία ενσωματωμένη λειτουργία ή συνάρτηση λάβει έναν έγκυρο τύπο επιχειρήματος, αλλά μία ακατάλληλη τιμή (και η κατάσταση δεν περιγράφεται από κάποια πιο ακριβής εξαίρεση, όπως IndexError).

ZeroDivisionError	Όταν ο διαιρέτης σε μία διαίρεση είναι 0.

Πίνακας 18. Διαθέσιμοι Τύποι Εξαιρέσεων

Ένα παράδειγμα εξαίρεσης είναι το εξής:



Εικόνα 3. Παράδειγμα Εξαίρεσης

3.7.2 Δήλωση assert

Χρησιμοποιούμε την δήλωση assert όταν θέλουμε να δηλώσουμε ότι κάτι είναι αληθές. Είναι, ουσιαστικά, ένας έλεγχος που ενεργοποιείται ή απενεργοποιείται μετά από ένα τμήμα κώδικα. Θα μπορούσαμε να παρομοιάσουμε έναν ισχυρισμό (assertion) με μία δήλωση raise-if-not. Μια έκφραση η οποία ελέγχεται κι αν ο ισχυρισμός αποτύχει, τότε έχει ως αποτέλεσμα μία εξαίρεση AssertionError. Το επιχείρημα AssertionError μπορεί να αντιμετωπιστεί όπως κάθε άλλη εξαίρεση,

δηλαδή με το να χρησιμοποιήσουμε τη δήλωση try-except. Αλλά αν δεν αντιμετωπιστεί, όπως και οι υπόλοιπες εξαιρέσεις, θα τερματίσει το πρόγραμμα και θα παράγει ένα traceback. Αυτή η λειτουργία χρησιμοποιείται συχνά από τους προγραμματιστές. Συνήθως, τοποθετούν ισχυρισμούς κατά την έναρξη μίας συνάρτησης για να ελέγξουν αν μία εισαγωγή είναι έγκυρη, και ύστερα από μία κλήση συνάρτησης για να ελέγξουν αν η έξοδος είναι έγκυρη.

3.7.3 Δήλωση except χωρίς εξαιρέσεις

Μπορεί να υπάρξει δήλωση except χωρίς καθορισμένες εξαιρέσεις. Διότι συμπεριλαμβάνει όλες τις εξαιρέσεις χωρίς να αναγνωρίζεται η αιτία του προβλήματος που ενδέχεται να προκύψει. Ωστόσο, δεν θεωρείται καλή πρακτική προγραμματισμού, επειδή συμπεριλαμβάνει όλες τις εξαιρέσεις. Η δομή του είναι η ακόλουθη:

```
try:
    λειτουργίες
except:
    εντολές
else:
    εντολές
```

3.7.4 Δήλωση except με πολλαπλές εξαιρέσεις

Μία δήλωση except μπορεί να χειριστεί πολλαπλές εξαιρέσεις ως μία εντός παρενθέσεων. Η δομή της είναι η εξής:

```
try:
    λειτουργίες
```

except (όνομα_εξαίρεσης1, όνομα_εξαίρεσης2,..):

εντολές

else:

εντολές

3.7.5 Δήλωση try-finally

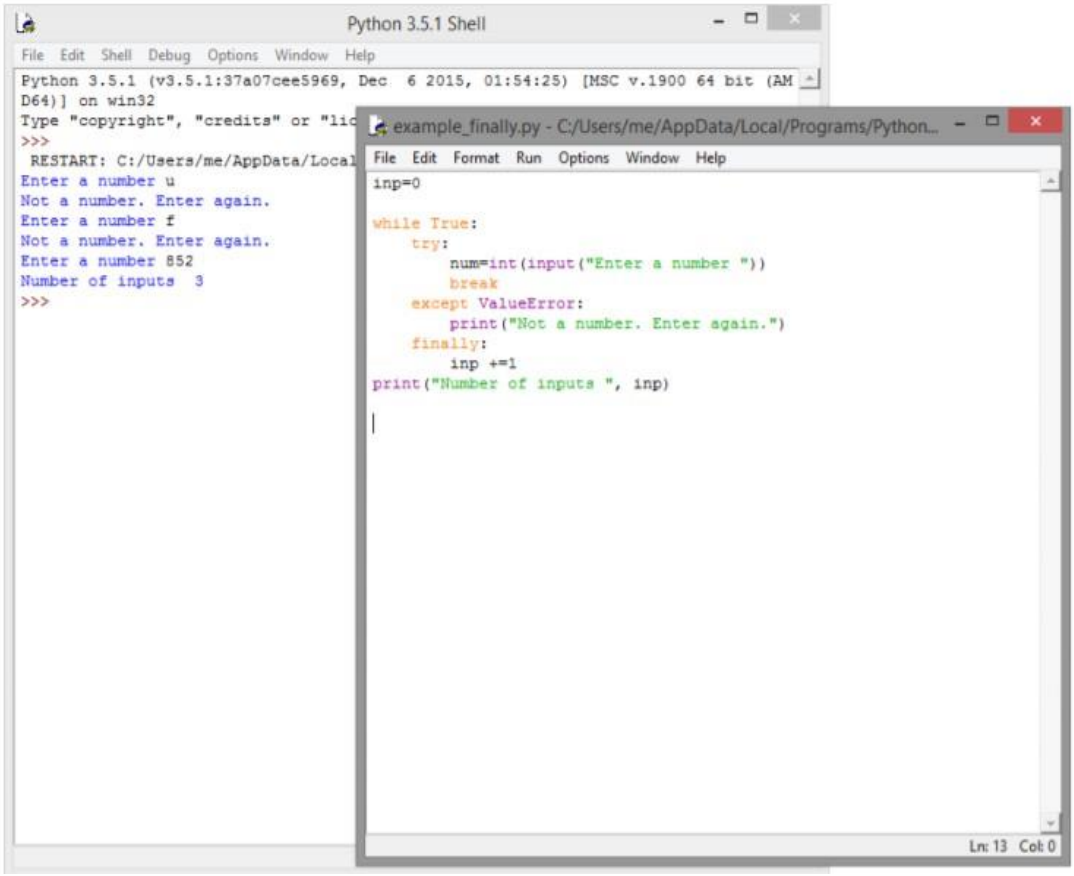
Υπάρχει η δυνατότητα να χρησιμοποιήσουμε μία δήλωση try μαζί με μία δήλωση finally και το μπλοκ κώδικα της καθεμίας. Η δήλωση finally είναι ένα μέρος που θέτουμε οποιοδήποτε τμήμα κώδικα πρέπει οπωσδήποτε να εκτελεστεί, είτε η δήλωση try θέσει μία εξαίρεση είτε όχι. Ωστόσο, η δήλωση try μπορεί να παρέχει μία ή παραπάνω δηλώσεις except, ή μία δήλωση finally, αλλά δεν είναι δυνατόν να περιέχει και τις δύο δηλώσεις. Επιπλέον, δεν μπορούμε να χρησιμοποιήσουμε μία δήλωση else μαζί με μία δήλωση finally. Ακολουθεί η δομή μίας δήλωσης try-finally:

try:

λειτουργίες

finally:

εντολές



Εικόνα 4. Παράδειγμα εξαίρεσης με finally

3.7.6 Επιχείρημα εξαίρεσης

Μπορεί να προστεθεί στη δήλωση της εξαίρεσης ένα επιχείρημα (argument), το οποίο να περιλαμβάνει μία τιμή με πρόσθετες πληροφορίες σχετικά με την εξαίρεση. Η μεταβλητή μπορεί να λάβει μία μόνο τιμή ή πολλαπλές τιμές με τη μορφή μιας πλειάδας. Τα περιεχόμενα του επιχειρήματος διαφοροποιούνται ανάλογα με την εξαίρεση. Αν γράψουμε κώδικα που χειρίζεται μία μόνο εξαίρεση, μπορούμε στη συνέχεια να προσθέσουμε μία μεταβλητή Argument που να ακολουθεί το όνομα της εξαίρεσης στη δήλωσή της. Εάν υπάρχουν πολλαπλές εξαιρέσεις, μπορούμε να προσθέσουμε τη μεταβλητή στο τέλος της σειράς με τα ονόματα εξαιρέσεων. Μπορούμε να συντάξουμε το επιχείρημα (argument) στη δήλωση μίας εξαίρεσης (except) ως εξής:

```
try:
    λειτουργίες
except επιχείρημα, Argument:
    εντολές
```

3.7.7 Ανάδειξη εξαιρέσεων

Η δήλωση raise μπορεί να αναγκάσει μία συγκεκριμένη εξαίρεση να συμβεί. Όπως για παράδειγμα:

```
>>> raise NameError('Hi')
Traceback (most recent call last):
File "", line 1, in <module>
raise NameError('Hi')
NameError: Hi
```

Η δήλωση μας δείχνει το όνομα του σφάλματος εξαίρεσης και το αντικείμενο της εξαίρεσης που πρόκειται να συμβεί. Η εξαίρεση που θα αναδειχθεί πρέπει να προέρχεται από μία κλάση εξαίρεσης, δηλαδή είτε άμεσα είτε έμμεσα να είναι παραγωγή της κλάσης exception. Η ανάδειξη (raising) εξαιρέσεων συμβαίνει όταν χρησιμοποιήσουμε την εντολή raise, στη συνέχεια δώσουμε την ονομασία του σφάλματος εξαίρεσης, καθώς και το αντικείμενο της εξαίρεσης που πρόκειται να συμβεί. Για παράδειγμα:

```
>>> try:
    raise NameError('Hi')
```

```
    ...
except NameError:
    print('It became an exception.')
    raise ...
    ...
```

3.7.8 Εξαιρέσεις ορισμένες από το χρήστη

Με την Python έχουμε τη δυνατότητα να δημιουργήσουμε τις δικές μας εξαιρέσεις από τη δημιουργία μιας νέας κλάση εξαίρεσης. Οι εξαιρέσεις πρέπει τυπικά να προέρχονται από την κλάση εξαίρεσης, είτε άμεσα είτε έμμεσα. Η κλάση εξαίρεσης μπορεί να δημιουργήσει οτιδήποτε μπορεί να δημιουργήσει και οποιαδήποτε άλλη κλάση. Όμως, συνήθως διατηρείται απλή προσφέροντας διάφορα χαρακτηριστικά τα οποία δίνουν πληροφορίες για την εξαίρεση. Μία συνηθισμένη τακτική, όταν δημιουργούμε ένα άρθρωμα που μπορεί να αναδείξει αρκετά ξεχωριστά λάθη, είναι η δημιουργία μίας βασικής κλάσης για εξαιρέσεις που ορίζεται από το συγκεκριμένο άρθρωμα.

3.8 Αρχείο

Η Python περιέχει βασικές συναρτήσεις και μεθόδους για την επεξεργασία αρχείων. Τα αρχεία αποθηκεύουν και ανακτούν πληροφορίες οι οποίες δεν έχουν να κάνουν μόνο με κάποιο συγκεκριμένο πρόγραμμα που τρέχει. Στην ενότητα αυτή παρουσιάζονται αυτές οι λειτουργίες.

3.8.1 Άνοιγμα, ανάγνωση και εγγραφή αρχείου

Αρχικά, πριν κάνουμε οποιαδήποτε λειτουργία σε ένα αρχείο, θα πρέπει να το ανοίξουμε ώστε να ενημερωθεί το λειτουργικό ότι πρόκειται να χρησιμοποιηθεί. Για να ανοίξει το αρχείο χρησιμοποιούμε τη συνάρτηση open(), όπου ανάμεσα στις παρενθέσεις εισάγουμε το όνομα του αρχείου. Η συνάρτηση επιστρέφει έναν αριθμό τον οποίο χρησιμοποιούμε για την προσπέλαση του αρχείου και για άλλες λειτουργίες. Η εντολή open() μπορεί να υποστηρίξει κι άλλες λειτουργίες, τις οποίες μπορούμε να δηλώσουμε κατά την πληκτρολόγηση της εντολής. Οι λειτουργίες είναι ανάγνωση, εγγραφή και άλλες. Η δομή της εντολής έχει ως εξής: open(όνομα_αρχείου, λειτουργία)

Οι λειτουργίες που υλοποιούνται μέσω της open() φαίνονται στον παρακάτω πίνακα:

Λειτουργία	**Περιγραφή**
'r'	Ανάγνωση αρχείου.
'w'	Εγγραφή αρχείου (διαγραφή πιθανών προηγούμενων εγγράφων).
'x'	Δημιουργία νέου αρχείου και άνοιγμα για εγγραφή.
'a'	Προσθήκη εγγράφων (χωρίς τη διαγραφή προηγούμενων).
'b'	Αρχείο σε δυαδική μορφή (binary mode).
'+'	Προσθήκη εγγραφών στο τέλος ενός αρχείου.
'r+'	Άνοιγμα αρχείου για ανάγνωση και εγγραφή, ο δείκτης του αρχείου τοποθετείται στην αρχή του αρχείου.

'w+'	Άνοιγμα αρχείου για ανάγνωση και εγγραφή. Αν το αρχείο υπάρχει, τότε αντικαθιστάται. Αν δεν υπάρχει, τότε δημιουργείται για ανάγνωση και εγγραφή.
'a+'	Άνοιγμα αρχείου για ανάγνωση και προσθήκη εγγραφών, ο δείκτης τοποθετείται στο τέλος του αρχείου. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται για ανάγνωση και εγγραφή.

Πίνακας 19. Λειτουργίες της εντολής open ()

Μόνο έναν από τους τρόπος ανοίγματος 'r', 'w', 'a', και λοιπά μπορούμε να επιλέξουμε. Αν δεν επιλεχθεί κάποιος, τότε από προεπιλογή, ανοίγει το αρχείο μόνο για ανάγνωση.

Όταν ολοκληρωθούν οι λειτουργίες που πρέπει να γίνουν στο αρχείο, το αρχείο θα πρέπει να κλείσει ώστε να απελευθερωθούν κάποιοι πόροι του συστήματος. Αυτό επιτυγχάνεται με την εντολή close(). Υπάρχει η δυνατότητα να ενημερωθούμε για κάποιες πληροφορίες σχετικές με το αρχείο, οι εντολές φαίνονται στον επόμενο πίνακα:

Ιδιότητα	**Περιγραφή**
fid.clozed	Εμφανίζει True ή False, αν το αρχείο έχει κλείσει.
fid.mode	Εμφανίζει τη λειτουργία πρόσβασης του αρχείου.
fid.name	Εμφανίζει το όνομα του αρχείου.

Πίνακας 20. Πληροφορίες σχετικές με το αρχείο

Οι μέθοδοι που υποστηρίζει η Python σχετικά με τα αρχεία, εμφανίζονται στον πίνακα που ακολουθεί:

Μέθοδος	Περιγραφή
fid.cloze()	Κλείσιμο ενός αρχείου.
fid.fileno()	Επιστέφει τον ακέραιο περιγραφέα αρχείου που χρησιμοποιείται από την εφαρμογή για λειτουργίες I/O (εισόδων, εξόδων) από το λειτουργικό σύστημα.
fid.next()	Επιστρέφει την επόμενη γραμμή από το αρχείο κάθε φορά που καλείται.
fid.readline(size)	Διαβάζει μια γραμμή από το αρχείο.
fid.seek(offset)	Ορίζει το δείκτη θέσης του αρχείου.
fid.tell()	Επιστρέφει την τρέχουσα θέση του αρχείου.
Fid.truncate(size)	Περικόπτει το μέγεθος του αρχείου. Αν έχει οριστεί η προαιρετική παράμετρος size, τότε περικόπτει το αρχείο ως το πολύ αυτό το μέγεθος.
fid.wrtelines(seq)	Γράφει μια ακολουθία από συμβολοσειρές στο αρχείο.

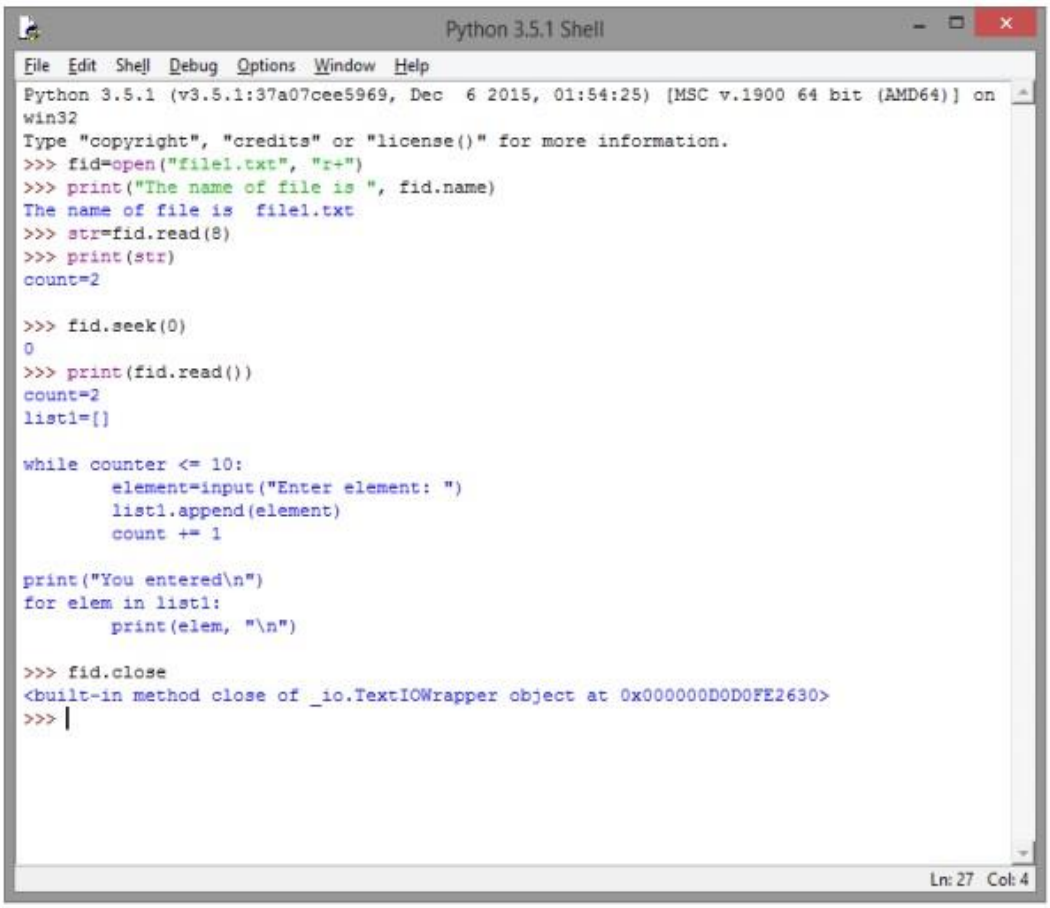
Πίνακας 21.Μέθοδοι Αρχείου

3.8.2 Διάβασμα από αρχείο

Η βασική συνάρτηση για την ανάγνωση ενός αρχείου είναι η εντολή read(). Αν η εντολή χρησιμοποιηθεί χωρίς όρισμα θα διαβάσει όλο το αρχείο, διαφορετικά μπορεί να οριστεί ο αριθμός των bytes του αρχείου που θα πρέπει να διαβαστούν.

Υπάρχει και η εντολή readlines() για την ανάγνωση ενός αρχείου. Η διαφορά της εντολής read() με την readlines() είναι ότι¨:

- η read() διαβάζει όλο το αρχείο και το επιστρέφει σε μία μεταβλητ
- η readlines() διαβάζει τις γραμμές μία-μία και τις επιστρέφει σε μία λίστα.



Εικόνα 5. Παράδειγμα ανάγνωσης από ένα αρχείο.

3.8.3 Εγγραφή σε αρχείο

Η συνάρτηση write() χρησιμοποιείται για να κάνουμε εγγραφή ενός αρχείου. Η εντολή καταγράφει μία συμβολοσειρά σε ένα αρχείο και επιστρέφει το σύνολο των χαρακτήρων. Η εντολή αυτή δεν προσθέτει το χαρακτήρα νέας γραμμής ('\n') στο τέλος της συμβολοσειράς, θα πρέπει εμείς να τον προσθέσουμε.

3.8.4 Θέση αρχείου

Για να βρούμε την τρέχουσα θέση του δείκτη ανάγνωσης ή εγγραφής αρχείου μέσα στο αρχείο χρησιμοποιούμε την εντολή tell(). Η σύνταξη της εντολής είναι η εξής: όνομα_αντικειμένου.tell(). Για παράδειγμα:

```
>>> tst=open("test1.txt", "r+")

 >>> str = tst.read(30)
```

```
>>> print("Read String is : ", str)

Read String is : num1=int(input("an integer: ")

>>> pos=tst.tell()

>>> print("Current file position : ", pos)

Current file position : 30

 >>> tst.close()
```

3.8.5 Άρθρωμα os στην επεξεργασία αρχείου

Το άρθρωμα os περιέχει μεθόδους για την επεξεργασία αρχείου, όπως μετονομασία ενός αρχείου, ή διαγραφή ενός αρχείου. Για να χρησιμοποιήσουμε αυτές τις μεθόδους πρώτα πρέπει να εισαχθεί το άρθρωμα και έπειτα να γίνει η κλήση των συναρτήσεων.

Για να μετονομαστεί ένα αρχείο χρησιμοποιούμε τη μέθοδο rename(). Η μέθοδος rename() δέχεται δύο ορίσματα, το τρέχων όνομα και το νέο όνομα ενός αρχείου. Η δομή του κώδικα είναι η εξής:

```
>>> import os

>>> os.rename("test1.txt", "exmpl.txt")
```

Για να διαγραφεί ένα αρχείο χρησιμοποιείται η μέθοδος remove(). Η μέθοδος δέχεται ως όρισμα το όνομα του αρχείου που θέλουμε να διαγράψουμε. Η δομή του κώδικα έχει ως εξής:

```
>>> import os

>>> os.remove("exmpl.txt")
```

3.8.6 Άρθρωμα os στην επεξεργασία καταλόγου

Η Python μας δίνει τη δυνατότητα χειρισμού καταλόγων. Το άρθρωμα os περιέχει διάφορες μεθόδους για το χειρισμό καταλόγων. Μπορούμε να

δημιουργήσουμε, να τροποποιήσουμε και να διαγράψουμε κατάλογους χρησιμοποιώντας το αρθρώματος os.

Η μέθοδος mkdir() δημιουργεί καταλόγους στον τρέχοντα κατάλογο. Ως όρισμα της μεθόδου πρέπει να περιλαμβάνεται το όνομα του καταλόγου που θα δημιουργηθεί. Η σύνταξη του κώδικα είναι η εξής:

```
>>> import os
>>> os.mkdir("newdir")
```

Η τροποποίηση ενός καταλόγου γίνεται με τη μέθοδο chdir(). Η μέθοδος αλλάζει τον τρέχοντα κατάλογο. Ως όρισμα εισάγουμε το όνομα που θέλουμε για τον τρέχοντα κατάλογο. Η δομή της είναι:

```
>>> import os
>>> os.chdir("/home/dir2")
```

Η μέθοδος getcwd() μας δείχνει τον τρέχοντα κατάλογο εργασίας. Η δομή του κώδικα είναι η ακόλουθη:

```
>>> import os
>>> os.getcwd()
```

Η μέθοδος rmdir() χρησιμοποιείται για να διαγράψουμε έναν κατάλογο. Ως όρισμα στη μέθοδο rmdir() εισάγεται ο κατάλογος που θέλουμε να διαγράψουμε.

```
>>> import os
>>> os.rmdir("newdir")
```

Πριν δοκιμάσουμε να διαγράψουμε ένα κατάλογο πρέπει να έχουμε αφαιρέσει όλα τα περιεχόμενά του. Διαφορετικά, θα εμφανιστεί το ακόλουθο μήνυμα λάθους.

```
Traceback (most recent call last):
 File "<pyshell#5>", line 1, in <module>
os.rmdir("newdir")
OSError: [WinError 145] Ο κατάλογος δεν είναι κενός: 'newdir'
```

ΚΕΦΑΛΑΙΟ 4

4.1 Συμπεράσματα

Όποιος θέλει να γνωρίσει τον κόσμο της επιστήμης των δεδομένων, πρέπει πρώτα να εξοικειωθεί με μια γλώσσα προγραμματισμού. Μαθαίνοντας κάποιος τις διαδικασίες και τις τεχνικές μίας γλώσσας θα έχει μία δυνατή βάση στην ανάλυση των δεδομένων, η οποία είναι απαραίτητη προτού εισχωρήσει βαθύτερα σ΄αυτήν. Αποκτώντας αυτή τη βάση, θα είναι ευκολότερη η μετάβαση σε άλλες γλώσσες με αρχικό πάντα γνώμονα την πρώτη γλώσσα προγραμματισμού. Υπάρχουν πολλές γλώσσες προγραμματισμού που μπορεί να διαλέξει, αλλά η Python είναι αδιαμφισβήτητα η δυνατότερη βάση για τον κόσμου της ανάλυσης δεδομένων.

Η Python αποτελεί γλώσσα με πολλές δυνατότητες και οφέλη. Γι’αυτό το λόγο όλο και περισσότεροι άνθρωποι την έχουν ως πρώτη επιλογή για την δουλειά τους.

Αρχικά, είναι μία γλώσσα ανοιχτού κώδικα, εύκολη στην εκμάθηση και στη χρήση, με ευανάγνωστο κώδικα. Την Python μπορούν να χρησιμοποιήσουν τόσο αρχάριοι όσο και επιστήμονες που το αντικείμενό τους είναι τα δεδομένα και θέλουν εξελίξουν τις ικανότητές τους. Βέβαια η εφαρμογή των διαφόρων τεχνικών χρησιμοποιώντας ένα νέο εργαλείο μπορεί να αποδειχθεί δύσκολη, έχοντας υπόψην ότι η ανάλυση των δεδομένων περιέχει διάφορες τεχνικές πρόβλεψης για τις οποίες μπορούν να χρησιμοποιηθούν πολλά διαφορετικά εργαλεία διεξαγωγής δεδομένων. Το πόσο απλή είναι η Python μπορεί να χρησιμεύσει είτε σ’αυτόν που είναι έμπειρος ερευνητής δεδομένων ή αναλυτής, ένας μηχανικός λογισμικού που αρχίζει να δουλεύει πιο εντατικά στη μηχανική μάθηση, ή ακόμα και σ’έναν παντελώς αρχάριο.

Επίσης, μπορεί να ανταποκριθεί και στην κατασκευή εργαλείων ανάλυσης, ειδικά όταν σχετίζεται με τη δημιουργία μιας υπηρεσίας ιστού, στο να επιτραπεί σε άλλους να βρουν αποκλίσεις στα σύνολα δεδομένων τους. Τέλος, αν και υπάρχουν γλώσσες που έχουν καλύτερη οπτικοποίηση δεδομένων, και η Python μπορεί να παρέχει μια αρκετά μεγάλη γκάμα.

Η Python έχει μεγάλο εύρος βιβλιοθηκών, για την ανάπτυξη ιστού, χρησιμοποιώντας την ανάπτυξη παιχνιδιών, μέχρι τη μηχανική μάθηση και την οπτικοποίηση δεδομένων, όπως αναφέρθηκε και παραπάνω, αλλά και πολλά πακέτα τα οποία βοηθούν στο να γίνει η εκμάθησή της πιο εύκολη.

Παρότι τα πλεονεκτήματά της είναι αρκετά, είναι αργή σε σύγκριση με άλλες γλώσσες προγραμματισμού, όπως για παράδειγμα η C. Επίσης, το πακέτο για την εγκατάστασή της δεν είναι τόσο τακτοποιημένο με αποτέλεσμα, ο προγραμματιστής να μπαίνει στη διαδικασία να ελέγχει για το αν περιλαμβάνονται όλα τα απαραίτητα αρχεία.

Κλείνοντας, η Python είναι σίγουρα ένα ισχυρό και ευέλικτο εργαλείο, που επιλέγουν ακόμα και εταιρείες όπως η Google, το Facebook ή η Microsoft. Επιτρέπει να γίνονται πολλά μέσα σε λιγότερο χρόνο, είναι εύκολή και ευπροσάρμοστη γλώσσα, που σημαίνει ότι έχει τη δυνατότητα να ενσωματωθεί σε οποιαδήποτε ροή εργασίας, κάνοντάς την χρήσιμη με τα εργαλεία που ήδη χρησιμοποιούνται ή ενδέχεται να χρησιμοποιηθούν πιο μετά. Όλ'αυτά είναι που την καθιστούν είτε πρώτη επιλογή είτε δεύτερη, σε μια αγορά όπου η διαφοροποίηση γίνεται το κλειδί για ανάπτυξη.

4.2 Παραδείγματα ασκήσεων με την χρήση της Python

Άσκηση 1:

Να δημιουργηθεί πρόγραμμα στη γλώσσα προγραμματισμού Python, το οποίο να παίρνει από τον χρήστη μια φράση μα χαρακτήρες του αγγλικού αλφαβήτου και να εμφανίζει στην οθόνη την φράση, διατηρώντας μόνο τους αγγλικούς κεφαλαίους χαρακτήρες.

Λύση:

```
s=input("Type:")

t=[]
```

```
for i in range (len(s)):
    if(  (s[i]>='A' and s[i]< ='Z' or (s[i]< =' '):
        t.append(s[i])
print (''.join(t))
```

Άσκηση 2:

Έχοντας την λογική συνάρτηση F = ((A XOR B) AND (C XOR D)), να δημιουργηθεί πρόγραμμα στην γλώσσα προγραμματισμού Python, που να διαβάζει τις τιμές των μεταβλητών A, B, C, D και θα εμφανίζει στην οθόνη την έξοδο F.

Λύση:

```
a = int(input("Type value of A (0 -1): "))
b = int(input("Type value of B (0 -1): "))
c = int(input("Type value of C (0 -1): "))
d = int(input("Type value of D (0 -1): "))
print ((a^b) & (c|d))
```

Άσκηση 3:

Να δημιουργηθεί συνάρτηση σε γλώσσα python που δέχεται ως όρισμα ένα θετικό ακέραιο αριθμό και επιστρέφει την αξία του στο οκταδικό.

Λύση:

```
decimal = int(input("Enter a decimal integer: "))
if decimal == 0:
    print (0)
else:
     bstring = ''
     while decimal > 0:
         remainder = decimal%8
```

```
        decimal = decimal//8
        bstring = str(remainder) + bstring
    print (bstring)
```

Άσκηση 4:

Να δημιουργηθεί πρόγραμμα σε γλώσσα python, το οποίο θα διαβάζει από το πληκτρολόγιο τα στοιχεία ενός πίνακα διαστάσεων 10 x 10 και αν ο πίνακας είναι αντισυμμετρικός θα εμφανίζει αντίστοιχο μήνυμα.

Λύση:

```
N=10
arr = [[0]*(N) for i in range(N)]
asym = 1
for i in range(N):
    for j in range(N):
        arr[i][j]=float(input("Type value:"))
for i in range(N):
   for j in range(i+1,N):
      if (arr[i][j] != -arr[j][i]):
         asym = 0
         break
if (asym == 0):
   print ("The matrix is not antisymmetric")
else:
   print("The matrix is antisymmetric ")
```

Άσκηση 5:

Να σημιουργηθεί μια συνάρτηση two_ranking με δυο παραμέτρους:

(1) candidates, που είναι μια λίστα υποψηφίων να εκλεγούν π.χ. ['cand1','cand2','cand3','cand4'], και

(2) voters, που είναι ένα λεξικό με κλειδιά κάποιους ψηφοφόρους (voters) και τιμές μια πλειάδα (tuple) αποτελούμενη από 2 υποψήφιους κατά σειρά προτίμησης π.χ. {'voter1':('cand1','cand2'), 'voter2':('cand3','cand4'), 'voter3':('cand2','cand3')}.

Η συνάρτηση two_ranking, για κάθε ψηφοφόρο, θα δίνει 2 ψήφους στον υποψήφιο που ο ψηφοφόρος αυτός τον επιλέγει πρώτο και 1 ψήφο στον υποψήφιο που τον επιλέγει δεύτερο. Στο τέλος, η συνάρτηση two_ranking εκτυπώνει υποψήφιους μαζί με τους συνολικούς ψήφους που πήραν από την ψηφοφορία.

Λύση:

```
def two_ranking(candidates,voters):
   results={}
   for key,val in voters.items():
       u=2
       for candi in val:
 if candi not in results:
     results[candi]=u
else:
   results[candi]+=u
u-=1
   result={}
   for i,v in results.items():
       if v not in result:
result[v]=[i]
       else:
result[v].append(i)
   for i in sorted(result.keys(),reverse=True):
       for j in result[i]:
           print ('O',j, 'phre',results[j],'psifous')
```

Άσκηση 6:

Να δημιουργηθεί μια συνάρτηση στη γλώσσα προγραμματισμού python που παίρνει ως είσοδο έναν θετικό ακέραιο αριθμό N και τυπώνει όλους τους ακέραιους διαιρέτες του N (συμπεριλαμβανομένων του 1 και του ίδιου του αριθμού) και επιστρέφει το πλήθος τους. Για παράδειγμα, για N = 14, η συνάρτηση θα τυπώνει τους αριθμούς 1, 2, 7, 14 και θα επιστρέφει την τιμή 4.

Λύση:

```
def dieretes(N):
  c=0
  for i in range(1,N+1):
    if (N%i==0):
      print (i,end=' ')
      c+=1
  return(c)
```

Άσκηση 7:

Να δημιουργηθεί κλάση Fraction σε γλώσσα προγραμματισμού python η οποία θα παίρνει ως είσοδο τον αριθμητή και τον παρονομαστή του κλάσματος και κατά την αρχικοποίησή της θα τυπώνει το κλάσμα απλοποιημένο.

Λύση:

```
class Fraction:
    def gcd(self, n, d):
        while d:
 n, d = d, n%d
        return n
    def __init__(self, n, d):
        self.num = int(n / self.gcd(abs(n), abs(d)))
        self.denom = int(d / self.gcd(abs(n), abs(d)))
        if self.denom < 0:
```

```
self.denom = abs(self.denom)
self.num = -1*self.num
        elif self.denom == 0:
raise ZeroDivisionError
        if (self.denom!=1):
print self.num,"/",self.denom
        else:
           print self.num
```

Άσκηση 8:

Να δημιουργηθεί μια συνάρτηση vathmologia με παράμετρο εισόδου τη λίστα agwnes, η οποία να επιστρέφει ένα λεξικό με κλειδιά τις ομάδες της λίστας agwnes και τιμές την βαθμολογία τους. Δίνεται ότι για κάθε νίκη μια ομάδα (όταν δηλ. έχει πετύχει περισσότερα γκολ από την αντίπαλη ομάδα) κερδίζει 2 βαθμούς για κάθε ισοπαλία 1 βαθμό (όταν δηλ. έχει πετύχει ίσο πλήθος γκολ σε σχέση με την αντίπαλη ομάδα) ενώ δεν κερδίζει κανένα βαθμό σε περίπτωση ήττας. Διευκρινίζεται ότι τα αποτελέσματα των αγώνων δίνονται με τη μορφή agwnes = ['a-b:4-1', 'a-c:1-3', 'b-c:1-1', 'c-d:2-3', 'da:3-1'], όπου a, b, c, d είναι ομάδες.

Λύση:

```
def bathmologia(agwnes):
   scores = {}
   for i in agwnes:
        tl=i.split(":")
        tl1=tl[0].split("-")
        if tl1[0] not in scores:
scores[tl1[0]]=0
        if tl1[1] not in scores:
scores[tl1[1]]=0
        tl2=tl[1].split("-")
        if tl2[0] > tl2[1]:
```

```
scores[tl1[0]] += 2
        elif tl2[0] < tl2[1]:
scores[tl1[1]] += 2
        else:
scores[tl1[0]] += 1
scores[tl1[1]] += 1
    return scores
```

Βιβλιογραφία

1. "Ανάπτυξη εφαρμογών σε Προγραμματιστικό Περιβάλλον", βιβλίο μαθητή Γ' τάξης Τεχνολογικής Κατεύθυνσης Ενιαίου Λυκείου, Οργανισμός Εκδόσεων Διδακτικών Βιβλίων, Αθήνα
2. T. Gaddis, Ξεκινώντας με την Python.
3. John Guttag (2015). Υπολογισμοί και Προγραμματισμός Με Την Python. Μετάφραση: Παναγιώτης Καναβός, Επιμέλεια: Γεώργιος Μανής, Εκδόσεις Κλειδάριθμος.
4. Ellis Horowitz (1993). Βασικές Αρχές Γλωσσών Προγραμματισμού. 2η έκδοση, Εκδόσεις Κλειδάριθμος.
5. M. Harrison, Learning the Pandas Library: Python Tools for Data Munging, Analysis, and Visualization.
6. Cody Jackson (2011). Learning to Program Using Python. Ηλεκτρονικό βιβλίο, ελεύθερα διαθέσιμο.
7. W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.

1. http://aggelid.mysch.gr/pythonbook/INTRODUCTION_TO_COMPUTER_PROGRAMMING_WITH_PYTHON.pdf
2. https://qz.com/1063071/thegreat-r-versus-python-for-data-science-debate/.
3. https://www.python.org/.
4. https://www.dataquest.io/blog/pandas-python-tutorial/.
5. http://python.org.gr
6. https://www.datasciencegraduateprograms.com/python/.
7. https://codeburst.io/overview-of-python-data-visualization-tools-e32e1f716d10.
8. https://el.wikipedia.org/wiki/Python
9. https://en.wikipedia.org/wiki/Python_(programming_language)
10. https://www.learnpython.org/
11. https://slideplayer.gr/slide/1943931/
12. http://www.data-analysis-inpython.org/.
13. https://pandas.pydata.org/.
14. https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm

15. https://chrisalbon.com/python/data_wrangling/pandas_dataframe_importing_csv/.
16. https://pandas.pydata.org/.