ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ
ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνικές Δομές Δεδομένων Προγραμματισμού σε Python

ΟΝΟΜΑΤΕΠΩΝΥΜΑ ΦΟΙΤΗΤΩΝ

ΚΟΥΤΣΟΥΚΟΣ ΑΝΤΩΝΗΣ

ΠΟΛΥΜΕΝΑΚΟΣ ΙΩΑΝΝΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΣΤΑΜΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΠΑΤΡΑ 2020

ΠΡΟΛΟΓΟΣ

Η ελκυστικότητα της γλώσσας προγραμματισμού Python σε συνδυασμό με την πρόκληση της διδασκαλίας της και τις δυνατότητες της να διαχειρίζεται δομές δεδομένων ήταν οι βασικοί λόγοι επιλογή αυτού του θέματος πτυχιακής εργασίας για την ολοκλήρωση των σπουδών μας στο ΤΕΙ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ.

Η python είναι γλώσσα προγραμματισμού υψηλού επιπέδου διερμηνευόμενη ,δυναμική και ανήκει στο ελεύθερο λογισμικό. Η δομή της πτυχιακής εργασίας χωρίζεται σε δύο μέρη. Στο πρώτο μέρος παρουσιάζουμε λίγα λόγια για την python και το πιο σημαντικό μέρος τις τεχνικές δομές δεδομένων και στο δεύτερο μέρος παρουσιάζεται η καταλληλόλητα της για εκμάθηση σε σχέση με άλλες γλώσσες προγραμματισμού και τα συμπεράσματα.

Το εκπαιδευτικό υλικό χωρίζεται σε 6 κεφάλαια όπου το μεγαλύτερο μέρος αφορά τις δομές δεδομένων στη python και αναλύοντας τες με ορισμούς και πολλά παραδείγματα κώδικα σε κάθε δομή. Στην εισαγωγή αναφέρουμε λίγα πράγματα για την python ορισμούς, λίγο από την ιστορία της το πώς αναπτύχθηκε και το πεδίο εφαρμογής της και πως μπορούμε να την εγκαταστήσουμε ,πως τρέχουμε το πρώτο μας πρόγραμμα και διάφορα προγράμματα για την εφαρμογή της. Στο πρώτο κεφάλαιο αναφέρουμε γενικούς ορισμούς για δομές δεδομένων και τις λειτουργίες τους. Ακολουθεί μια σειρά από τελεστές, μεταβλητές και τύποι δεδομένων. Στα υπόλοιπα κεφάλαια αναφέρουμε αναλυτικά τις δομές δεδομένων με παραδείγματα και κώδικες και σχήματα για βοήθεια κατανόησης, τα παραδείγματα που παρατίθενται είναι μικρά ώστε να επιτρέπουν στον εκπαιδευόμενο να επικεντρώνεται στο ζήτημα που εξηγείται κάθε φορά. Τέλος στα τελευταία κεφάλαια αναφέρουμε την αξιολόγηση καταλληλόλητας για την εκμάθηση βασικών αρχών στις δομές δεδομένων.

Θέλουμε να ευχαριστήσουμε τον κ. Στάμο Κωνσταντίνο για τη συνεργασία μας και τις συμβουλές του.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία αναπτύχθηκε εκπαιδευτικό υλικό σε μορφές μικρού κώδικα και ορισμών με σκοπό να χρησιμοποιηθεί για ένα αρχικό μάθημα αλγορίθμων και δομών δεδομένων, αντίστοιχο με τις γλώσσες που γίνονται τώρα στα τμήματα εκπαίδευσης .

Το μάθημα δεν απευθύνεται μόνο σε αναγνώστες με ιδιαίτερη γνώση στο θέμα αλλά και σε άτομα που έχουν σκοπό να μάθουν την γλώσσα και τις δομές της . Μετά από μια σύντομη εισαγωγική παρουσίαση της python (τι είναι , την ιστορία της , την εφαρμογή της , τρόπος εγκατάσταση της) παρουσιάζονται οι δομές δεδομένων της. Σε αυτά τα κεφάλαια παρουσιάζονται δομές δεδομένων που συναντάμε στη python και που μπορεί να εφαρμοστούν σε αυτήν με κώδικες και παραδείγματα που μπορούν να αναλύσουν πιο εύκολα τους ορισμούς που έχουμε δώσει και να δώσουν ένα πιο κατανοητό δείγμα των δομών δεδομένων της python.

Παρακάτω θα δούμε γιατί η python είναι κατάλληλη για εκμάθηση βασικών αρχών στις δομές δεδομένων ώστε με την μελέτη αυτών των επιχειρημάτων να κατανοήσουμε όχι μόνο οι απλοί αναγνώστες αλλά και εκπαιδευτικοί ότι η python είναι η πιο κατάλληλη για εκμάθηση αυτών σε αντίθεση με άλλες γλώσσες.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ……..1

ΠΕΡΙΛΗΨΗ ……..2

ΠΕΡΙΕΧΟΜΕΝΑ……...3

1. ΕΙΣΑΓΩΓΗ……....5
   1.1 ΟΡΙΣΜΟΣ ΤΗΣ PYTHON…… ….5
   1.2 ΙΣΤΟΡΙΑ ΤΗΣ PYTHON……....6
   1.3 ΠΕΔΙΟ ΕΦΑΡΜΟΓΗΣ ΤΗΣ PYTHON…….…7
   1.4 ΕΓΚΑΤΑΣΤΑΣΗ ΤΗΣ PYTHON…….…..8
   1.5 ΕΚΤΕΛΕΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ PYTHON……...10
   1.6 ΤΟ ΠΡΩΤΟ ΜΑΣ ΠΡΟΓΡΑΜΜΑ……12
   1.7 ΤΟ ΜΟΝΤΕΛΟ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ PYTHON……..13
2. ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΒΑΣΙΚΟΙ ΤΕΛΕΣΤΕΣ……...14
   2.1 ΤΙΜΕΣ ΚΑΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ……..……...14
   2.2 ΜΕΤΑΒΛΗΤΕΣ……..15
   2.3 ΕΚΦΡΑΣΕΙΣ ΤΕΛΕΣΤΕΣ ΚΑΙ ΣΧΟΛΙΑ……..17
   2.4 ΕΛΕΓΧΟΣ ΡΟΗΣ ΔΕΔΟΜΕΝΩΝ……...…..19
      2.4.1 ΛΟΓΙΚΗ BOOLEAN……...20
      2.4.2 ΔΟΜΗ ΕΠΙΛΟΓΗΣ IF……22
      2.4.3 ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ FOR……24
      2.4.4 ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ WHILE……26
      2.4.5 ΕΝΤΟΛΗ BREAK……27
3. ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ…… …..28
   3.1 ΟΡΙΣΜΟΣ ……..……..26
   3.2 ΛΕΙΤΟΥΡΓΙΕΣ……28
   3.3 ΠΙΝΑΚΕΣ……29
      3.3.1 ΤΑ ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΩΝ ΠΙΝΑΚΩΝ……30
      3.3.2 ΤΑ ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΩΝ ΠΙΝΑΚΩΝ ……....31
      3.3.3 ΟΙ ΛΕΙΤΟΥΡΓΙΕΣ ΤΩΝ ΠΙΝΑΚΩΝ……32
   3.4 ΣΤΟΙΒΕΣ……..33
      3.4.1 ΕΦΑΡΜΟΓΕΣ ΠΟΥ ΜΠΟΡΟΥΜΕ ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΣΟΥΜΕ ΣΤΟΙΒΕΣ……...……34

3.4.2 ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ ΣΤΟΙΒΑΣ…….……35
3.4.3 Η ΣΤΟΙΒΑ ΣΤΗΝ ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ ΟΙ ΔΙΑΦΟΡΕΣ ΜΕ ΤΙΣ ΣΩΡΟΥΣ ΚΑΙ Η ΕΦΑΡΜΟΓΗ ΤΗΣ……..35
3.5 ΟΥΡΕΣ…….……38
3.5.1 ΕΦΑΡΜΟΓΕΣ ΤΗΣ ΟΥΡΑΣ……39
3.6 ΛΙΣΤΕΣ…….…...40
3.6.1 ΣΥΝΑΡΤΗΣΕΙΣ ΣΕ ΛΙΣΤΕΣ……...43
3.6.2 ΠΡΟΣΘΕΣΗ ΣΤΟΙΧΕΙΩΝ ΣΕ ΛΙΣΤΕΣ……...45
3.6.3 ΑΦΑΙΡΕΣΗ ΣΤΟΙΧΕΙΩΝ ΣΕ ΛΙΣΤΕΣ……47
3.7 ΔΕΝΤΡΑ…… …….49
3.7.1 ΔΥΑΔΙΚΑ ΔΕΝΤΡΑ……53
3.8 ΛΕΞΙΚΑ…… …….56
3.9 ΠΛΕΙΑΔΕΣ…….…….61
3.10 ΣΥΝΟΛΑ…….……..65
3.10.1 ΤΟΜΗ ΣΥΝΟΛΩΝ……67
3.10.2 ΕΝΩΣΗ ΣΥΝΟΛΩΝ……..68
3.10.3 ΔΙΑΦΟΡΑ ΣΥΝΟΛΩΝ …….69
3.10.4 ΣΥΜΜΕΤΡΙΚΗ ΔΙΑΦΟΡΑ ΣΥΝΟΛΩΝ……..71
3.10.5 ΥΠΟΣΥΝΟΛΑ ΚΑΙ ΤΕΛΕΣΤΕΣ ΣΥΓΚΡΙΣΗΣ…….……..72
3.11 ΑΛΦΑΡΙΘΜΗΤΙΚΑ……76

4 . ΓΙΑΤΙ Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΡΥΤΗΟΝ ΕΙΝΑΙ ΚΑΤΑΛΛΗΛΗ ΓΙΑ ΕΚΜΑΘΗΣΗ ΚΑΙ ΓΙΑ ΔΙΔΑΣΚΑΛΙΑ……77
4.1 Η ΧΡΗΣΗ ΤΗΣ ΡΥΤΗΟΝ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ……..78
4.2 ΡΥΤΗΟΝ Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΤΗ ΜΕΓΑΛΥΤΕΡΗ ΖΗΤΗΣΗ……79
4.3 ΜΗΧΑΝΙΚΗ ΕΚΜΑΘΗΣΗ…….……...80
4.4 ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ…….……81

5.ΒΙΒΛΙΟΓΡΑΦΙΑ-ΑΝΑΦΟΡΕΣ……..…..83

ΕΙΣΑΓΩΓΗ

Σχήμα 1.0 Το Λογότυπο της Python

1.1 Ορισμός της Python

Η Python είναι διερμηνευόμενη (interpreted) και δυναμική γλώσσα προγραμματισμού υψηλόυ επιπέδου.Συμπεριλαμβάνεται στις γλώσσες *Προστακτικού Προγραμματισμού* (Imperative Programming) και υποστηρίζει *Διαδικαστικό (Procedural Programming)* αλλά και Αντικειμενοστρεφές προγραμματισμό (Object – Oriented Programming). Άπο τα σημαντικά πλεονεκτήματα της είναι οτι διαθέτει έτοιμες βιβλιοθήκες που μπόρουν να χρησιμοποιηθούν και επεκτασιμότητα καθώς έχουμε την δυνατότητα να προσθέσουμε ενότητες (Modules) γραμμένα σε C και C++ .Είναι απλή στην εκμάθηση της ,και αναπτύσσεται ως ανοικτού κώδικα λογισμικό.

1.2 Ιστορία της Python

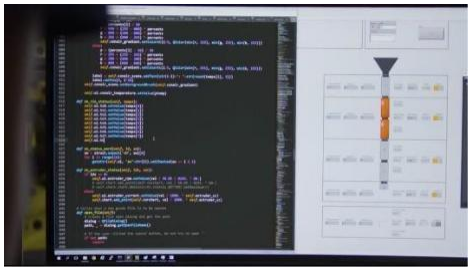
Η Python δημιοργήθηκε στις αρχές της δεκαετίας του 1990 απο τον Guido Van Rossum στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) στο Reston της Βιρτζίνια, ως διάδοχος της γλώσσας προγραμματισμού ABC . Την ονομασία της ο δημιουργός της, την εμπνεύστηκε απο την κωμική σειρά Monty Python's Flying Circus του BBC καθώς ήθελε το όνομα να είναι μικρό και μυστήριο. Αρχικός της σκοπός ήταν να είναι γλώσσα σεναρίων για το λειτουργικό σύστημα Amoeba. Το Python 1.6 ήταν η τελευταία έκδοση που κυκλοφόρησε απο το ερευνητικό κέντρο.Το 2000 ,ο Guido και η βασική ομάδα ανάπτυξης της Python μετακόμισαν στο BeOpenPythonLabs.Το Python 2.0 ήταν η πρώτη και μοναδική κυκλοφορία απο το BeOpen.com. Στην συνέχεια Εντάχθηκαν στην Digital Creations.Την δεδομένη στιγμή η σειρά εκδόσεων 2.X έχει σταματήσει να αναβαθμίζεται και η τελευταία της έκδοση είναι η 2.7.18 .Η σειρά 3.X συνεχίζεται και η τελευταία έκδοση είναι η 3.8.3. Η διαχείριση γίνεται απο τον οργανισμό Python Software Foundation (PSF) .

1.3 Πεδίο Εφαρμογής

Η ευκολία στην εκμάθηση της ,και η δυνατότητα γρήγορης ανάπτυξης ολοκληρωμένου λογισμικού στις περισσότερες πλατφόρμες την έχει κάνει μία απο τις πιο ανταγωνιστικές γλώσσες . Μπορεί να χρησιμοποιηθέι για Δημιουργία και επικοινωνία διαδικτυακών σελίδων ,βάσεων δεδομένων . Επίσης χρησιμοποιείται για ανάπτυξη λογισμικού , αλλα και σαν γλώσσα σεναρίων (scriptin language). Ένα πεδίο εφαρμογής που είναι και ο σκοπός να αναλύσει η συγκεκριμένη εργασία ,είναι και οι Δομές Δεδομένων καθώς η Python είναι ένα εργαλείο για την οργάνωση και διαχείριση της .

Ένα μεγάλο ποσοστό απο τις μεγαλύτερες εταρείες πληροφορικής ,χρησιμοποιούν στις πλατφόρμες τους την γλώσσα προγραμματισμού Python.Κάποιες απο αυτές τις εταιρείες είναι:

- Η ***Google*** η οποία την χρησιμοποιεί στα συστήματα αναζήτησης.
- Το ***Youtube*** στο διαμοιρασμό των video του.
- Το ***Dropbox*** για τον διαμοιρασμό αρχείων απο την πλευρά του server αλλα και του client.
- Το ***Netflix*** στις υπηρεσίες του για την προβολή των ταινιών του.
- Ακόμα και η ***NASA*** την χρησιμοποιεί για την δημιούργία λογισμικών σε ερευνητικά προγράμματα.

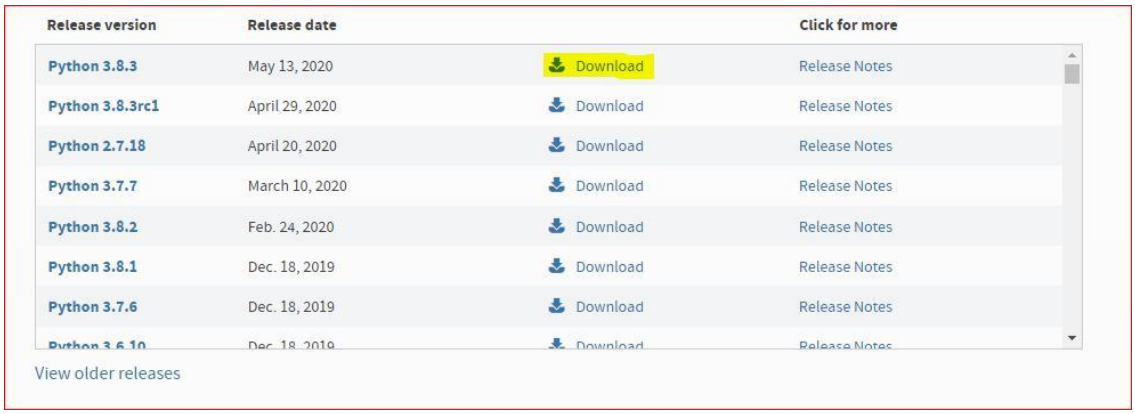


Σχήμα 1.1 Σχεδιασμός λογισμικού της NASA με γλώσσα Python

1.4 Εγκατάσταση της Python

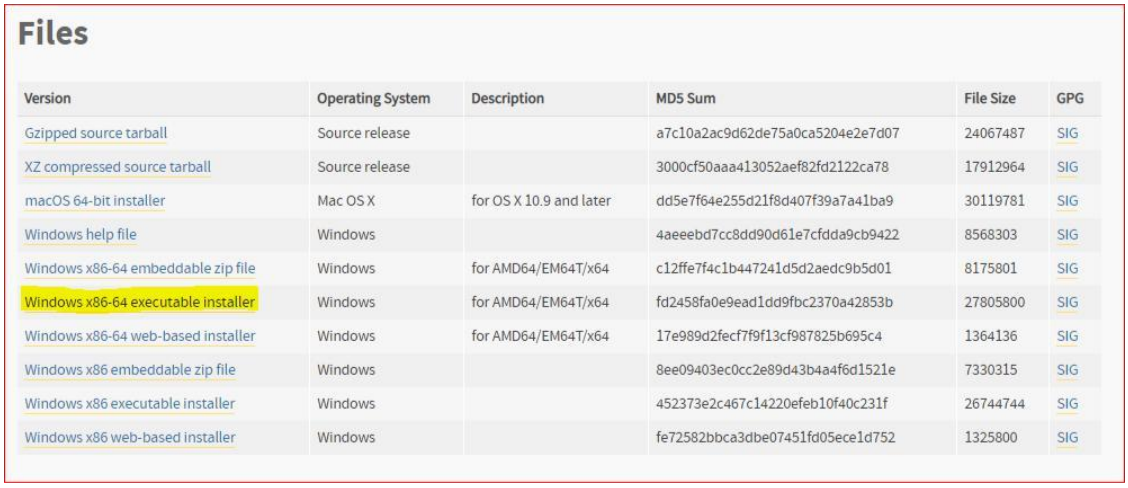
Η διαδικασία εγκατάστασης είναι Απλή :

1. Μπαίνουμε στην σελίδα της Python (https://www.python.org/downloads/)



Εικόνα 1.2 *Κατέβασμα αρχείων εγκατάστασης της Python*

2. Επιλέγουμε το σύνδεσμο Download και Κατεβάζουμε την τελευταία έκδοση .Έχουμε την δυνατότητα να διαλέξουμε μεταξύ 32 BIT ή 64 BIT εγκατάστασης. (Εικόνα 1.3)



Files

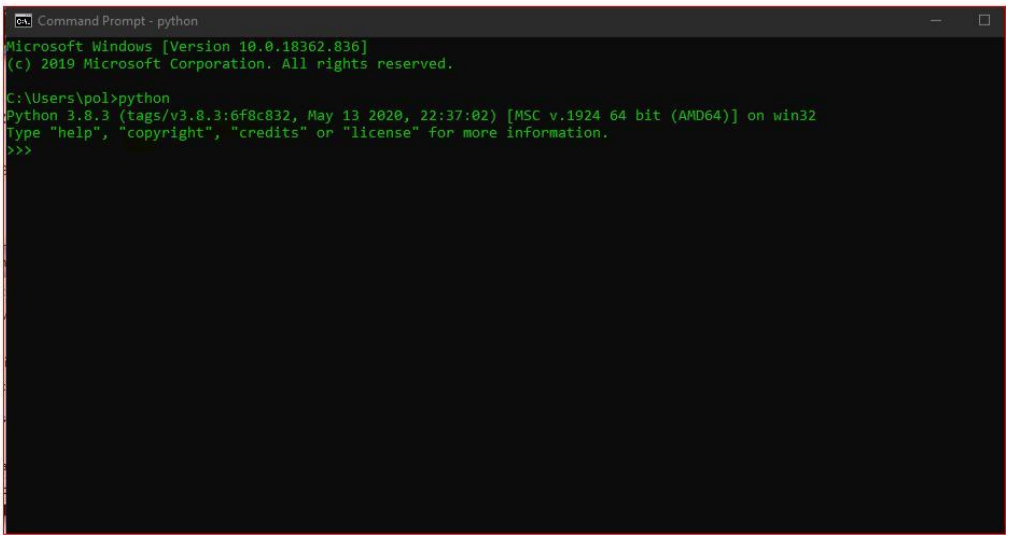
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		a7c10a2ac9d62de75a0ca5204e2e7d07	24067487	SIG
XZ compressed source tarball	Source release		3000cf50aaa413052aef82fd2122ca78	17912964	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	dd5e7f64e255d21f8d407f39a7a41ba9	30119781	SIG
Windows help file	Windows		4aeeebd7cc8dd90d61e7cfdda9cb9422	8568303	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	c12ffe7f4c1b447241d5d2aedc9b5d01	8175801	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	fd2458fa0e9ead1dd9fbc2370a42853b	27805800	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	17e989d2fecf7f9f13cf987825b695c4	1364136	SIG
Windows x86 embeddable zip file	Windows		8ee09403ec0cc2e89d43b4a4f6d1521e	7330315	SIG
Windows x86 executable installer	Windows		452373e2c467c14220efeb10f40c231f	26744744	SIG
Windows x86 web-based installer	Windows		fe72582bbca3dbe07451fd05ece1d752	1325800	SIG

Εικόνα 1.3 *Κατέβασμα αρχείων εγκατάστασης της Python 2*

3. Επιλέγουμε τη συγκεκριμένη έκδοση και την κατεβάζουμε στον υπολογιστή μας (Εικόνα 1.3)

Εικόνα 1.4 *εγκατάστασης της Python*

4. Εκτελούμε το αρχείο της εγκατάστασης και πατάμε και την επιλογή “Add Python to PATH “ (Εικόνα 1.4)



Εικόνα 1.5 *εγκατάστασης της Python 2*

5. Για να επιβεβαιώσουμε ότι έχει πραγματοποιηθεί η εγκατάσταση ,ανοίγουμε το Command Prompt και εισάγουμε την λέξη “Python”,και πέρνουμε το παραπάνω μήνυμα το οπόιο δηλώνει ότι η εγκατάσταση πραγματοποιήθηκε επιτυχώς .(Εικόνα 1.4)

1.5 Εκτέλεση προγράμματος σε Python

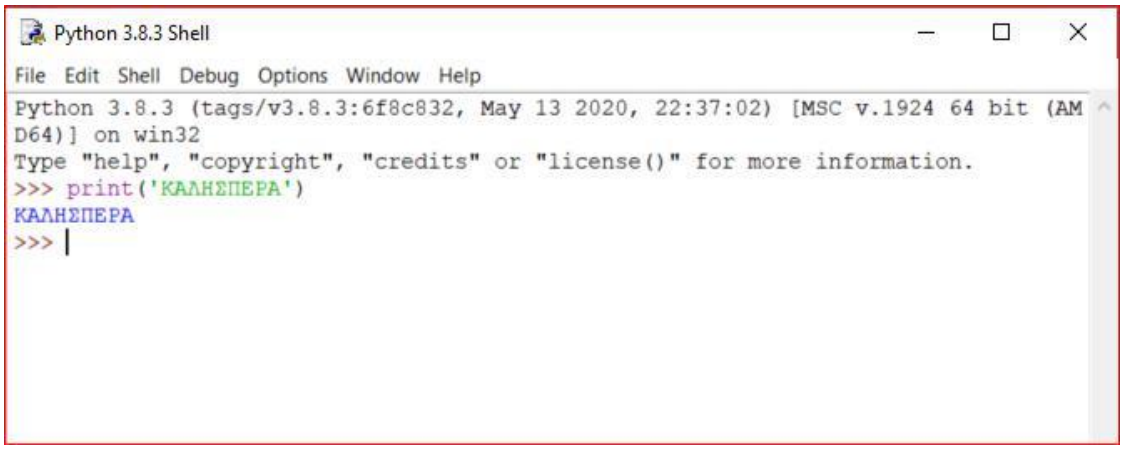
Η Python όπως είπαμε στην εισαγωγή ειναι διερμηνευόμενη γλώσσα προγραμματισμού.Περιλαμβάνει τον **διερμηνευτή (interpreter)** και **την βιβλιοθήκη (library)**. Ο διερμηνευτής διαβάζει και αναλύει το πρόγραμμα μας και εκτελεί τις εντολές .Η βιβλιοθήκη διαθέτει κώδικα ,με τη μορφή **ανθρωμάτων (modules)**,ο οποίος μπορεί να χρησιμοποιηθεί για την εκτέλεση του προγράμματος μας.

Για να γράψουμε πρόγραμμα στην Python ανοίγουμε την γραμμή εντολών και πληκτρολογούμε την εντολη Python για να εκκινήσουμε την κονσόλα του διερμηνευτή της **Python (Python Shell)**.Στη συνέχεια δίνουμε τις εντολές και

εκτελούνται απο τον διερμηνευτή με **διαδραστικό τρόπο (Interactive mode)**.(Εικόνα 1.5)

Μπαίνοντας στο προγραμματιστικό περιβάλλον της Python βλέπουμε τρία βελάκια με φορά προς τα δεξιά (>>>) τα οποία ονομάζονται **Προτροπή (Prompt)** ,εκεί γράφουμε τις εντολές μας και αφού πατήσουμε ENTER ,εκτελείται η εντολή απο τον διερμηνευτή και στην επόμενη γραμμή εμφανίζεται το αποτέλεσμα .Παρακάτω εμφανίζεται πάλι η προτροπή (>>>) και ο διαρμηνευτής περιμένει την επόμενη εντολή μας.

Η Python διαθέτει ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (IDE) στο οποίο γράφουμε και εκτελούμε τα προγράμματα μας .Το συγκεκριμένο περιβάλλον ονομάζεται **IDLE (Intergrated Development Environment) (Εικόνα 1.5)**



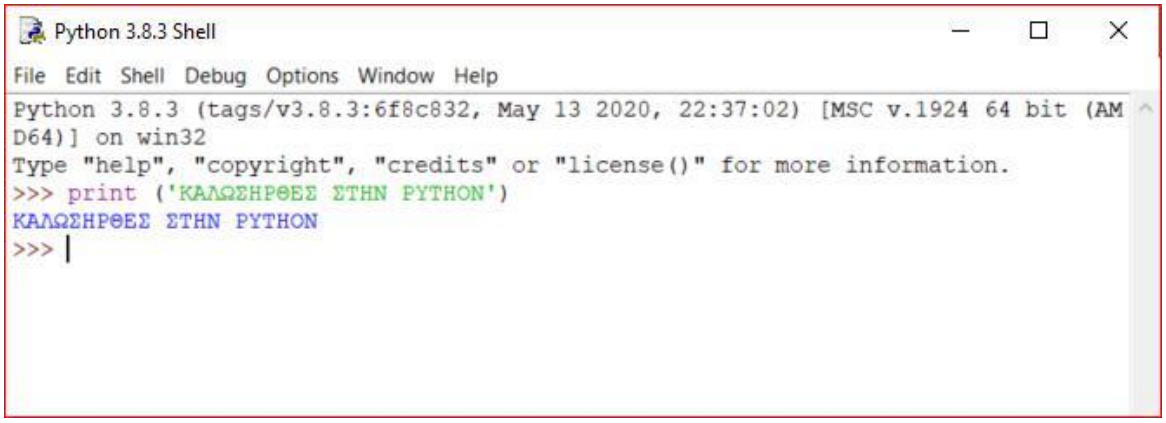
Εικόνα 1.6 *Το περιβάλλον IDLE*

Το IDLE (Integrated Development Environment) χωρίζεται στα παρακάτω μέρη:

Python Shell (κονσόλα διερμηνευτή της Python) : Το λεγόμενο κέλυφος στο οποίο γίνεται η εκτέλεση εντολών απο τον διερμηνευτή .Υπάρχει χρωματική επισήμανση στη σύνταξη στα δίαφορα τμήματα της εντολής .

Παράθυρο σύνταξης (Editor): Στο συγκεκριμένο παράθυρο υπάρχει η δυνατότητα στοίχισης των εντολών και αυτόματη συμπλήρωση τους .Εδώ γράφεται το πρόγραμμα (Program mode),αποθηκεύουμε σε αρχείο και το εκτελούμε.
Απασφαλματωτής (Debugger): σκοπός του απασφαλμωτή είναι η προσπέλαση του προγράμματος για να εντοπίσει ενδεχόμενα σφάλματα.

1.6 Το Πρώτο μας πρόγραμμα



Εικόνα 1.7 *Το πρώτο μας πρόγραμμα*

Στο Παραπάνω παράδειγμα (Εικόνα 1.7) βλέπουμε το πρώτο μας πρόγραμμα σε Python .Στο πάνω μέρος του παραθύρου βλέπουμε την έκδοση της Python που είναι εγκαταστημένη στον υπολογιστή μας. Στην προτροπή (>>>) δίνουμε την εντολή :

```
>>> print ('ΚΑΛΩΣΗΡΘΕΣ ΣΤΗΝ PYTHON')
```

Η εντολή αύτη τυπώνει στην οθόνη μας τη φράση

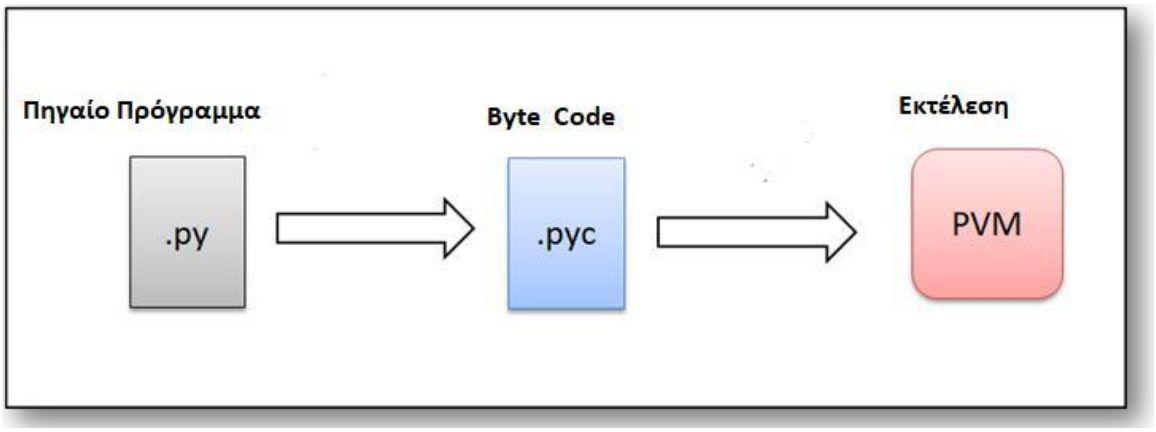
```
ΚΑΛΩΣΗΡΘΕΣ ΣΤΗΝ PYTHON
```

Τέλος, η προτροπή (>>>) εμφανίζεται ξανά περιμένοντας την επόμενη εντολή μας.

1.7 Το μοντέλο Εκτέλεσης Προγράμματος σε Python

Εκτελώντας το Πρόγραμμα ,η Python εσωτερικά μεταφράζει τον πηγαίο κώδικα σε μορφή που ονομάζεται **Byte Code** . Ο Byte Code είναι μια πλατφόρμα αναπαράστασης του κωδικά μας. Δεν είναι ορατή απο μάς και μπορεί να τρέξει σε διαφορετικά υπολογιστικά συστήματα .

Ο Byte Code εκτελείται μέσα απο την εικονική **Μηχανή της Python - PVM (Python Virtual Maschine).** Η Εικονική μηχανή της Python είναι λογισμικό το οποίο προσομειώνει έναν υπολογιστή που είναι για εκτέλεση προγραμμάτων της Python. (Σχημα 1.1)



ΣΧΗΜΑ 1.8 *Το Μοντέλο Εκτέλεση της Python*

Η Python αποτελείται απο τρία βασικά μέροι:

- Τον **ερμηνευτή (Interpreter)** για εκτέλεση εντολών
- Έναν **μεταφραστή (Compiler)** για μετατροπή .Py αρχείων σε .Pyc αρχεία
- Μία **εικονική μηχανή (PVM)** για εκτέλεση .Pyc αρχείων

Κεφάλαιο 2- Μεταβλητές και Βασικοί Τελεστές

Σε αυτό το κεφάλαιο περιλαμβάνονται βασικές έννοιες που υπάρχουν σχεδόν σε όλες τις γλώσσες προγραμματισμού , όπως οι έννοιες των μεταβλητών και οι βασικοί τελεστές τους ,καθώς είναι τα εργαλέια για να μπορέσουμε να δημιουργήσουμε πρόγραμμα στην Python.

2.1 Τιμές και Τύποι

Τιμή : Στα μαθηματικά , τη λογική και τον προγραμματισμό υπολογιστών , μία τιμή είναι συγκεκριμένο αντικείμενο.Για παράδειγμα ο αριθμός 1 , το γράμμα A και ο μοναδικό συνδυασμός γραμμάτων που σχηματίζει την λέξη P-y-t-h-o-n είναι όλες τιμές .

Σε υπολογιστές , μία τιμή είναι συνήθως ένας αριθμός , ένας χαρακτήρας ή μια σειρά χαρακτήρων αποτελούμενη απο Bit η οποία ανήκει σε κάποιο τύπο δεδομένων.

Τύποι Δεδομένων : Σε μία γλώσσα προγραμματισμού , ό τύπος δεδομένων καθορίζει τι είδους τιμές μπορέι να έχει ένα αντικέιμενο και ποιές λειτουργίες μπορούν να εκτελεστούν στο αντικέιμενο .

Στο παρακάτω παράδειγμα τυπώνουμε τιμές (με την χρήση της εντολής **Print**) , με διαφορετικούς τύπους δεδομένων :

```
>>> Print (7)
7
>>> Print (9.1)
9.1
>>> Print ('Καλωσήρθατε στην Python')


Καλωσήρθατε στην Python
```

ΣΧΗΜΑ 1.9 *Η εντολή print*

- Το **7** είναι **ακέραιος αριθμός (integer)**
- Το **9.1** είναι **αριθμός κινητής υποδιαστολής (floating)**
- Και το **'Welcome to Python'** είναι **συμβολοσειρά (string)**

Στην Python δεν είναι υποχρεωτικό να δηλώσουμε τους τύπους δεδομένων που χρησιμοποιούμε. Αυτό ονομάζεται **Δυναμική απόδοση τύπων (dynamic typing)**.

2.2 Μεταβλητές

Οι μεταβλητές χρησιμοποιούνται για την αποθήκευση των δεδομένων και έχουν σαν τιμή τα δεδομένα που έχουμε αποθηκεύσει στην συγκεκριμένη μεταβλητή.Οπώς οι τιμές έτσι και οι μεταβλητές έχουνε τύπους . Ο **τύπος μεταβλητής** είναι ο τύπος δεδομένων της τιμής στην οποία αναφέρεται .

Εντολή εκχώρησης (Assigment statement)

Η συγκεκριμένη εντολή παράγει νέες μεταβλητές και τους αναθέτει τιμές.

```
>>>  city = 'Patra'
>>> print (city)

>>> type (city)
<class 'str'>
>>> X=210
>>> print (X)
210
>>> type (X)
<class 'int'>
```

Σχήμα 2.0 Εντολή εκχώρησης

Στο παράδειγμα υπάρχουν 2 εντολές εκχώρησης:

- Η Λέξη **Patra** εκχωρέιται στη μεταβλητή **City**
- Και ο ακέραιος αριθμός **210** εκχωρέιται στην μεταβλητή **X**

Επίσης στο παραπάνω παράδειγμα υπάρχει και η εντολη **type(Μεταβλητή)**

Η οποία μας επιστρέφει τον τύπο της μεταβλητής και το τύπο δεδομένων που έχει η εκχωρηθέι στην μεταβλητή .

Garbage Collection

Η διαδικασία κατα την οποία η Python απελευθερώνει περιοδικά τμήματα μνήμης όταν δεν γίνεται αναφορά σε τιμές απο κάποια μεταβλητή.(όπως στο παρακάτω παράδειγμα)

```
>>> y = 4
>>> i = y
>>> print ( i )
4
>>> print ( y )
4
```

Σχήμα 2.1 Garbage Collection

Στο παραπάνω παράδειγμα βλέπουμε οτι μόλις εκτελεστούν οι εντολές οι μεταβλητές **(i,y)** μας δίνουν την τιμή 4 και η τιμή 2 διαγράφεται αυτόματα απο την μνήμη.

i=y=4

2.3 Εκφράσεις και τελεστές και σχόλια

Μία **έκφραση (expression)** είναι ένας συνδυασμός τιμών, μεταβλητών ,τελεστών και κλήσεις συναρτήσεων.

Τελεστές (operators) είναι σύμβολα που αντιπροσωπεύει μία ενέργεια η διαδικασία . Αυτά τα σύμβολα προσαρμόστηκαν απο τα μαθηματικά και την λογική και χρησιμοπούνται απο πολύ απλές λειτουργίες έως πολύπλοκους αλγόριθμους.

Κατηγορίες Τελεστών :

- Αριθμητικοί τελεστές (+, - , * , \ , %)
- Λογικοί τελεστές (And , or , not , xor)
- Σχεσιακοί τελεστές (< , > , =)

```
>>> x = 10
>>> y = 15
>>> x + y
25
>>> x > y is True
False
```

Σχήμα 2.1 παράδειγμα έκφρασης

Όταν γράφουμε κώδικα στην Python είναι σημαντικό ο κώδικας μας να μπορεί να γίνει εύκολα κατανοητός απο άλλους .Ένας τρόπος για να αυξήσουμε την αναγνωσιμότητα του κώδικα μας είναι χρησιμοποιώντας **σχόλια (comments).**

Τα σχόλια στην Python αρχίζουν με το σύμβολο # και οτιδήποτε ακολουθεί μετά αγνοείται απο την Python μέχρι το τέλος της γραμμής .

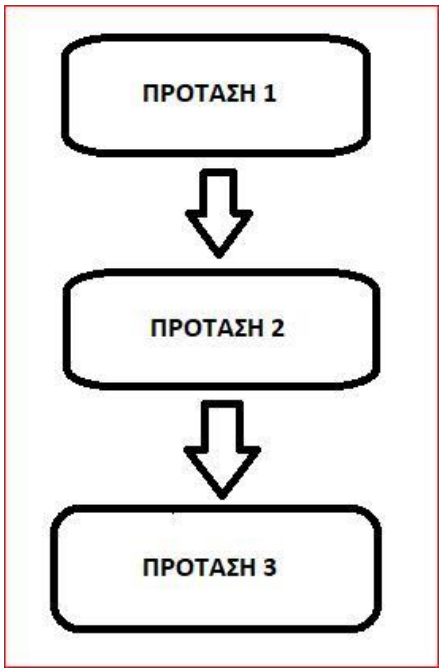
```
>>>     # Υπολογισμός του εμβαδού ενός ορθογωνίου
>>> width = 5     # πλάτος
>>> length = 9     # Μήκος
>>> area = width * length  # υπολογισμός Εμβαδού
>>> print ( 'Εμβαδόν' = area)
```

Σχήμα 2.2 παράδειγμα σχολίου

2.4 Έλεγχος ροής εκτέλεσης

Έλεγχος ροής : Είναι η σειρά με την οποία εκτελούνται οι κλήσεις,οι εντολές και οι δηλώσεις σε ένα πρόγραμμα**.**

Ο πιος διαδεδομένος τρόπος εκτέλεσης είναι η **ακολουθιακή εκτέλεση** στην οποία δύο ή περισσότερες προτάσεις βρίσκονται διατεταγμένες η μία μετά την άλλη και εκτελούνται διαδοχικά**.**(όπως φαίνεται στο παρακάτω παράδειγμα)



Εικόνα 2.3 *Ακολουθιακή εκτέλεση προτάσεων*

2.4.1 Τύπος δεδομένων αλήθειας (Boolean)

Στην πληροφορική οι τύποι δεδομένων αληθείας (Boolean) πέρνουν δύο τιμές , **Αληθής/True** ή **Ψευδής/False**) , με στόχο την αναπαράσταση της άλγεβρας bool, που πήρε το όνομα της απο τον George Boole, στα μέσα του 19^{ου} αιώνα.

Η σύνδεση των Boolean τιμών πραγματοποιείται με την χρήση των λογικών τελεστών **AND** , **NOT** και **OR**.

Παρακάτω ακολουθούν οι Πίνακες Αληθείας που περιέχουν τις τιμές (X και Y) και οι βασικοί λογικοί τελεστές.

X	Y	Not X	X And Y	X Or Y
Ψευδής	**Ψευδής**	**Αληθής**	**Ψευδής**	**Ψευδής**
Ψευδής	**Αληθής**	**Αληθής**	**Ψευδής**	**Αληθής**
Αληθής	**Ψευδής**	**Ψευδής**	**Ψευδής**	**Αληθής**
Αληθής	**Αληθής**	**Ψευδής**	**Αληθής**	**Αληθής**

Πίνακας 2.4 *Πίνακας αληθείας για τους βασικούς λογικούς τελεστές*

- Η έκφραση **(Not X)** είναι αληθής όταν η X είναι Ψευδής και Ψευδής όταν είναι Αληθής.
- Η έκφραση **(X Or Y)** είναι αληθής όταν η X ή Y είναι αληθής ή μία απο τις δύο είναι αληθής.
- Η έκφραση **(X And Y)** είναι αληθής μόνο όταν η X και η Y είναι αληθής.

Επίσης στις λογικές εκφράσεις έχουμε και τους **τελεστές σύγκρισης**:

i == y	i είναι ίσος με y
i != y	i δεν είναι ίσος με y
i > y	i είναι μεγαλύτερος απο y
i < y	i είναι μικρότερος απο y
i >= y	i είναι μεγαλύτερος ή ίσος με y
i <= y	i είναι μικρότερος ή ίσος με y

Πίνακας 2.5 *τελεστών σύγκρισης*

Προτεραιότητα τελεστών απο την μεγαλύτερη στην μικρότερη:

- Αριθμητικοί τελεστές
- Τελεστές σύγκρισης (< , > , = , < = , >= , != , ==)
- Λογικοί τελεστές (And , Not , Or)

```
>>> 10 != ( 3 + 2) and 5 == ( 3 + 2)
True
```

Σχήμα 2.6 παράδειγμα τελεστών σύγκρισης

2.4.2 Δομή ελέγχου if , if else

Η συγκεκριμένη δομή ελέγχου χρησιμοποιείτε για να ελέγχει μία συνθήκη και σύμφωνα με το αποτέλεσμα της (Αληθής ή ψευδής) εκτελέιται ή δεν εκτελούνται οι επόμενες εντολές του προγράμματος.Η σύνταξη τής εντολής if (στην πιο απλή της μορφή)είναι :

```
If (συνθήκη)
{
        ....// ομάδα εντολών }
 }
```

Η συνθήκη μπορεί να είναι μια λογική έκφραση που μπορεί να πάρει την έκφραση **True/ Αληθής** ή **false /ψευδής**.

- Αν είναι **True/ Αληθής** τότε εκτελούνται οι εντολές που βρίσκονται μέσα στα άγκιστρα ({,}).

```
>>> i = 6
>>> y = 9
>>> if i < y :
        Print ( ' Το i είναι μεγαλύτερο του y ')
Το i είναι μεγαλύτερο του y
```

Σχήμα 2.7 παράδειγμα πρώτης if

- Αν είναι **false /ψευδής** τότε οι εντολές μέσα στα άγκιστρα ({,}) δεν εκτελούνται.

```
>>> i = 1
>>> y = 5
>>> if i > y :
        Print ( '' Λάθος '' )

Λάθος
```

Σχήμα 2.8 παράδειγμα if

Για πολλές συνθήκες χρησιμοποιούμε τη λέξη **elif (συντομογραφία του else if) .**

```
>>> if num > 0 :
      Print ( '' θετικός αριθμός '')
elif   num == 0 :
      Print ( '' Μηδέν '' )
else :
          Print ( '' Αρνητικός αριθμός '' )
```

Σχήμα 2.9 παράδειγμα elif

2.4.3 Εντολή επανάληψης for

Στα προβλήματα που καλούμαστε να επιλύσουμε χρειαζόμαστε ένα πλήθος εντολών να εκτελεστεί πολλές φορές , οπότε είναι απαραίτητη η χρήση εντολών επανάληψης ,καθως μας βοηθάει στην εισαγωγή και επεξεργασία των δεδομένων .Η ομάδα των εντολών που επαναλαμβάνεται ονομάζεται **Βρόγχος (Loop).**Όταν γνωρίζουμε τον αριθμό των επαναλήψεων που θέλουμε να πραγματοποιηθούν χρησιμοποιούμε την εντολή επανάληψης **For.**

Στο παρακάτω παράδειγμα βλέπουμε ο κώδικας να τυπώνει τους ακέραιους αριθμούς απο το 1 μέχρι το 20.

```
>>>  for x in range (21) :
        Print ( x )


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Σχήμα 2.10 παράδειγμα for

Άν θέλουμε να ξεκινάει απο το 1 αντί για το 0 τροποιούμε τη συνάρτηση range.

```
>>>  for x in range ( 1,21 ) :
        Print ( x )



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Σχήμα 3.0 παράδειγμα for και range

Βλέπουμε στο παραπάνω παράδειγμα ότι η μεταβλητή X παίρνει τον αριθμό 1 ως αρχική τιμή, και σε κάθε επανάληψη αυξάνεται κατα 1.Αν δεν ορίσουμε το βήμα τότε θεωρέιτε οτι είναι 1.

2.4.4 Εντολή επανάληψης while

Εκτός απο την εντολή επανάληψης for , υπάρχει και η εντολή επανάληψης **while** η οποία εκτελείται όσο ισχύει η συνθήκη που ορίζουμε.

Στο παρακάτω παράδειγμα βλέπουμε την εκτύπωση των αριθμών απο το 1 έως το 20:

```
>>>  x = 0

>>> while x < 21 :

        Print ( x )

        x = x + 1


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Σχήμα 3.1 παράδειγμα while

Ο βρόγχος **while** αρχίζει με την λέξη **while** , την συνθήκη (λογική έκφραση) και τελειώνει με μία άνω κάτω τελεία και στην συνέχεια το μπλοκ εντολών που επαναλαμβάνονται ,όσο η συνθήκη είναι αληθής / True.Όταν η συνθήκη γίνει ψευδής (false) το μπλοκ εντολών δεν εκτελείται ,και η ροή εκτέλεσης μεταφέρεται στην επόμενη εντολή μετά τον βρόγχο.

Ατέρμων βρόγχος (infinite loop) : Ο βρόγχος που δεν σταματάει να εκτελείται , επειδή στο μπλοκ των εντολών δεν αλλάζουν οι τιμές ,ώστε να γίνει η συνθήκη ψευδής και να τερματιστεί ο βρόγχος.Για να αποφύγουμε το συγκεκριμένο πρόβλημα θα πρέπει στις μεταβλητές που αναφέρεται η συνθήκη να έχουν αρχικοποιηθεί.

2.4.5 Εντολή Break

Η εντολή **Break** χρησιμοποιείται για έξοδο απο οποιοδήποτε σημείο του βρόχου (όπως φένεται στο παρακάτω παράδειγμα).

```
>>> sum = 0
>>> while True :
        X = int ( input ( ' Δώσε έναν θετικό αριθμό ( - 1 για τέλος ) : '))
        If X == 1 :
                Break
       Sum = Sum + X
      Print ( ' Άθροισμα = ' , sum )
Δώστε έναν θετικό αριθμό ( -1 για τέλος ) : 3
Άθροισμα = 3
Δώστε έναν θετικό αριθμό ( -1 για τέλος ) : -3
Άθροισμα = 0
Δώστε έναν θετικό αριθμό ( -1 για τέλος ) : -1
>>>
```

Σχήμα 3.2 παράδειγμα break

Κεφάλαιο 3. Δομές Δεδομένων

3.1 Ορισμός Δομών Δεδομένων

Στην τεχνολογική εποχή που διανύουμε , Διαπιστώνεται μεγάλη αύξηση του αριθμού των Δεδομένων που διακινούνται μέσα απο το Διαδίκτυο και χρησιμοποιείται απο τους χρήστες στην καθημερινότητα τους .Είναι αναγκαία λοιπόν η καταγραφή και η αποτελεσματική Διαχείριση τους. Η έννοια της Δομής Δεδομένων είναι όλοι οι δυνατοί τρόποι ,αποθήκευσης και οργάνωσης των

δεδομένων ώστε τα δεδομένα να μπόρουν να καταγραφούν , να ταξινομηθούν και να χρησιμοποιηθούν αποδοτικά.

Οι Δομές δεδομένων χρησιμοποιούνται σε κάθε λογισμικό η Πρόγραμμα παίζει μεγάλο ρόλο ως προς την ομαλή λειτουργία

και την αποδοτικά του ,και έτσι το καθιστά απο τα βασικά κριτήρια σχεδίασης του Λογισμικού.

Ορισμός :Με τον όρο δομή δεδομένων (data structure) εννοούμε ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργειών στα δεδομένα αυτά. Μια μορφή δομής δεδομένων είναι η εγγραφή(record), που μπορεί οι μεταβλητές της να χαρακτηρίζουν ένα πρόσωπο, ένα είδος κλπ. Οι μεταβλητές της αυτές ονομάζονται πεδία (fields)

3.2 Λειτουργίες Δομών Δεδομένων

Κάθε Δομή Δεδομένων απαρτίζεται απο μία ομάδα κόμβων (nodes) .

Οι βασικές λειτουργίες των Δομών Δεδομένων είναι οι εξής :

- **Εισαγωγή** : Πρόσθεση ενός κόμβου στη Δομή
- **Αναζήτηση** : Η προσπέλαση τών κόμβων ,για να βρεθούν ενάς η περισσότεροι κόμβοι που έχουν μια δεδομένη ιδιότητα.
- **Διαγραφή** : Η αφαίρεση ενός κόμβου απο την Δομή.

Επίσης υπάρχουν και επιπρόσθετες λειτουργίες , όπως οι ακόλουθες :

- **Προσπέλαση** : Πρόσβαση σε ένα κόμβο με σκοπό να προβληθεί ή να τροποποιηθεί .
- **Αντιγραφή** : Αντιγραφή των κόμβων σε άλλη δομή.
- **Ταξινόμηση** : Οι κόμβοι εμφανίζονται κάτα αύξουσα η φθίνουσα σειρά.
- **Συγχώνευση** : Δύο ή περισσότεροι κόμβοι συνενώνονται σε μία ενιαία δομή.
- **Διαχωρισμός** : Η αντίστροφη πράξη της συγχώνευσης .

3.3 Πίνακες σε Python

Οι πίνακες είναι δομές δεδομένων, μια συλλογή από στοιχεία ή αριθμούς όπου το καθένα αναγνωρίζεται από ένα ευρετήριο πίνακα ή ένα κλειδί.

Α)**Το κλειδί ή αλλιώς ευρετήριο** αρχίζει από το 0

Β)Λόγω των ευρετηρίων : είναι δυνατή η τυχαία πρόσβαση

Οι πίνακες χωρίζονται σε **μονοδιάστατους** και σε **πολυδιάστατους**. Οι πολυδιάστατοι πίνακες μπορούν να αποδειχθούν πολύ σημαντικοί στους μαθηματικούς υπολογισμούς.

pinakas[] για μονοδιάστατο πίνακα

Pinakas[][] για δισδιαστατος πινακα

Στους δισδιάστατους πίνακες εχουμε τις γραμμές και τις στήλες για τα κλειδια π.χ ο pinakas[1][3] βρισκει τον αριθμο η το στοιχειο που βρισκεται στην γραμμη 1 και στην στηλη 3 δηλαδη πρωτα μπαινει το στοιχειο της γραμμης και μετα το στοιχειο της στηλης.

- Οι πίνακες είναι δομες δεδομενων για να αποθηκεύσεις **πραγματα του ιδιου τυπου** πχ αριθμους , ονοματα
- Στους πινακες χρησιμοποιουμε τους **δεικτες** ως ΄΄**κλειδια**΄΄
- Μπορούμε να εχουμε **οσες διαστασεις θελουμε** : μονοδιαστατος και διασδιαστατος να είναι οι πιο συνηθισμενοι
- Μπορούμε να εχουμε **δυναμικούς πινακες** δηλαδη πινακες που το μεγεθος αλλαζει δυναμικα

3.3.1 Τα πλεονεκτήματα των πινάκων

1. Μπορούμε να βρούμε πολύ γρήγορα κάποιο στοιχείο η κάποιον αριθμό λόγω των κλείδιων μέσω των δεικτών.

2. Είναι πολύ ευκολοι στην υλοποίηση τους και στην χρήση τους.

3. Είναι πολύ γρηγορη δομή δεδομένων

4. Θα πρέπε να χρησιμοποιούμε τους πίνακες όταν εμείς θελούμε να προσθέσουμε στοιχεία ολη την ώρα και πέρνουμε πολύ εύκολα πράγματα που εχουν δωθεί ευρετήρια ή κλειδιά ετσι θα είναι πολύ πιο γρήγορη η διαδικασία

3.3.2 Τα μειονεκτήματα των πινάκων

1. Πρέπει να ξέρουμε το μέγεθος του πίνακα την στιγμή της μεταγλήτισσης όποτε αυτό μας δείχνει ότι δεν είναι μια τοσο δυναμική δομή δεδομένων.
2. Εάν είναι γεμάτος ο πίνακας θα πρέπει να δημιουργήσουμε έναν μεγαλύτερο πίνακα και θα πρέπει να αντιγράψουμε τις τιμές ξεχωριστά.
3. Δεν είναι δυνατόν να αποθηκεύσουμε διαφορετικά γλώσσες προγραμματισμού στους πίνακες εκτός από την python.

3.3.3 Οι λειτουργίες των Πινάκων

- Η προσθήκη (add) , μπορούμε να προσθέτουμε στοιχεία η αριθμούς οσο δεν εχει γεμίσει ο πινακας
- Μπορούμε να εισάγουμε (insert) στοιχεία σε έναν πίνακα , παρόλαυτα όταν εισαγουμε μια τιμή σε έναν δείκτη η τιμή που βρίσκεται σε αυτόν τον δείκτη θα πρέπει να μετατοπιστεί στον επόμενο δείκτη.
- Η αφαίρεση (remove) τιμων από έναν πινακα οπου συνηθως είναι το ιδιο αποτελεσματικη με την insert διοτι όταν θελουμε να αφαιρέσουμε ένα στοιχειο από τον πινακα θα πρέπει να αντικατασταθεί οπότε καλύτερα θα ήταν να ξεκινάμε να αφαιρούμε από το τέλος.

Παράδειγμα Πίνακα σε Python

new 1 | pinakes

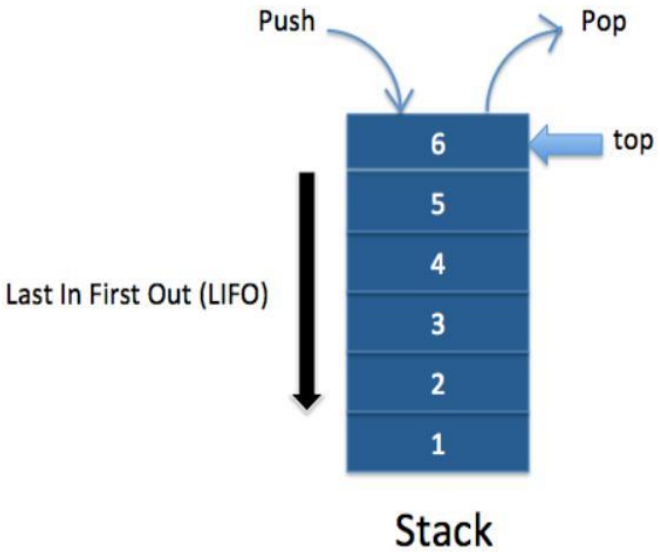
```
pinakas = [10,20,300,40.5,50] ;

#random indexing--> O(1) get items  if we know the index !!!
#print (pinakas[4]);

#pinakas[1] = 'Sadler' ;
#print (pinakas[1]);

#for num in pinakas :
#        print(num);

#for i in range(len(pinakas)) ;
#     print(pinakas[i]);

#print(pinakas[:-2]);

#O(N) search running time
Maximum = pinakas[0];
For num in pinakas :
        If num > Maximum :
            Maximum = num ;
print(Maximum);
```

Εικόνα 3.3 *Παράδειγμα πίνακα σε python*

Οπότε σε αυτό το παράδειγμα έχουμε ένα πίνακα με δεδομένες τιμές μέσα στους δείκτες του αυτό κάνει πιο γρήγορη την διαδικασία του προγράμματος. Στο παράδειγμα δίνω και διαφορετικού τύπου στοιχείο και όχι νούμερο και το πρόγραμμα δουλεύει κανονικά διότι όπως ειχαμε αναφέρει στην python μπορεί να συμβεί αυτό. Ενδιάμεσα χρησιμοποιώ το print(pinakas[:-2]); αυτό μας δείνει τις τιμές ολου του πινακα αλλα χωρις τις 2 τελευταιες δηλαδή [10 , Sadler , 30] και το Sadler μπαινει διοτι το ειχα αλλάξει ενδιάμεσα .Στο τέλος κάνω μια αναζήτηση σε ολο τον πίνακα να βρω τον μεγαλύτερο αριθμό και χρησιμοποιώ μια επανάληψη for οπου πιο πάνω εχω δώσει τον πρωτο αριθμό σε μια μεταβλητή για να μπορέσει να τον συγκρινει με όλες για να βγαλει το μεγαλύτερο

3.4 Στοίβες σε Python

Οι στοίβες είναι **αφηρημένος τύπος δεδομένων** και είναι μια ‘λίστα’ που είσαγεις και διαγράφεις στοιχεία μόνο στο ένα της άκρο. Στις περισσότερες γλώσσες προγραμματισμού μια στοίβα μπορεί να υλοποιηθεί εύκολα είτε με πίνακες είτε με λίστες. Μια σειρά από γλώσσες προγραμματισμού είναι προσανατολισμένες στη στοίβα, που σημαίνει ότι ορίζουν τις περισσότερες βασικές λειτουργίες (προσθήκη δύο αριθμών, εκτύπωση ενός χαρακτήρα) ως λήψη των επιχειρημάτων τους από τη στοίβα και τοποθέτηση οποιωνδήποτε τιμών επιστροφής στη στοίβα. Η μέθοδος επεξεργασίας των δεδομένων της στοίβας λέγεται **«Εξαγωγή κατά ανάστροφη σειρά εισαγωγής» (Last In – First Out, LIFO).**



Εικόνα 3.4 *Δομή της στοίβας*

3.4.1 Εφαρμογές που μπορούμε να χρησιμοποιήσουμε Στοίβες

- Σε **γλώσσες προγραμματισμού** προσανατολισμένες στο σωρό.
- Σε **αλγόριθμους γραφημάτων**: αναζήτηση βάθους-πρώτη μπορεί να εφαρμοστεί με στοίβα
- Βρίσκοντας **κύκλους Euler** σε ένα γράφημα
- Εύρεση ισχυρά συνδεδεμένων στοιχείων σε ένα **γράφημα**

3.4.2 Βασικές λειτουργίας της Στοίβας

Βασικές λειτουργίες της είναι : pop(), push() και peek()

Να δούμε ένα παράδειγμα αυτών των βασικών λειτουργιών :

Η **stack.push(23)** θα προσθέσει στην στοίβα το 23 οπότε στην ουσία με την push() προσθέτουμε απλά στοιχεία μεχρι να φουλάρει η στοίβα

Η **stack.pop()** θα αφαιρέσει το τελευταίο στοιχείο που μπήκε στην λίστα (γι'αυτό και το LIFO) αλλα δν μπορεί να αφαιρέσει το πρώτο στοιχείο χωρίς να αφαιρεθούν πρώτα τα άλλα

Η **stack.peek()** θα επιστρέψει το στοιχείο που είναι τελευταίο αλλά χώρις να το αφαιρέσει έτσι ώστε η στοίβα να παραμείνει ίδια

3.4.3 Η στοίβα στην διαχείρηση μνήμης , οι διαφορές με τις σωρούς και η εφαρμογή τους

Μια πολύ σημαντική εφαρμογή των στοιβών είναι το **stack memory** όπου είναι και μια ειδική περιοχή της μνήμης (in the RAM). Μια στοίβα κλήσεων είναι ένας αφηρημένος τύπος δεδομένων που αποθηκεύει πληροφορίες σχετικά με τις ενεργές **υπορουτίνες / μεθόδους / λειτουργίες(subroutines/methods/functions)** ενός προγράμματος υπολογιστή. Οι λεπτομέρειες είναι συνήθως κρυφές και αυτόματες σε γλώσσες προγραμματισμού υψηλού επιπέδου.

Είναι πολύ χρήσιμο γιατί παρακολουθεί το σημείο στο οποίο κάθε ενεργή υπορουτίνα πρέπει να επιστρέφει τον έλεγχο όταν τελειώσει η εκτέλεση και γιατι αποθηκεύει προσωρινές μεταβλητές που δημιουργούνται από κάθε συνάρτηση. Επίσης κάθε φορά που μια συνάρτηση δηλώνει μια νέα μεταβλητή ωθείται στη στοίβα

Διαφορές μεταξύ των στοιβών και των σωρών στην μνήμη :

- Στην περιπτωση **stack memory** εχουμε περιθωρια στην μνημη ενώ στις σωρους όχι.
- Στις στοιβες εχουμε **γρηγορη προσβαση** ενώ στις σωρους όχι.
- Στις στοιβες εχουμε **τοπικες μεταβλητες** ενώ στις σωρους **αντικείμενα.**
- Στο stack memory ο χωρος διαχειριζεται αποτελεσματικα από την **CPU** ενώ στις σωρους η **μνημη** μπορει να είναι **κατακερματισμενη.**
- Και τέλος στις στοιβες οι μεταβλητές δν μπορούν να αλλάξουν μάγεθος ενώ στις σωρούς μπορούν**.**

Η **Στοίβα σε εφαρμογή**

Όπου σε αυτή την εκτελεση της παρακατω εικονας όταν την τρεχουμε μας δειχνει την εφαρμογη των βασικών λειτουργιων **στην pop(), peek(),** και **push()** . Οπου φαινεται ότι

θα βγει στην οθονη το τι ακριβως καναμε pop το ποσα προσθεσαμε στην στοιβα και το τι διαγραψαμε και τι αφησαμε στην στοιβα.

```
Class  Stack :
          def __init__( self ) :
               self .stack = [ ]
          def  isEmpty ( self ) :
               return self.stack == [ ]
          def pop ( self ) :
               data = self.stack [-1]
               del self.stack [-1]
               return data
          def peek ( self ) :
               return self.stack [-1]
          def sizeStack ( self ) :
              return len ( self.stack)

stack = Stack ( )
stack.push ( 1 )
stack.push ( 2 )
stack.push ( 3 )
print ( stack.sizeStack ( ) )
print ( '' Popped : '', stack.pop ( ) )
print ( '' Popped : '', stack.pop ( ) )
print ( stack.size.Stack ( ) )
print ( '' Peek : “ , stack.peek ( ) )
print ( stack.sizeStack ( ) )
```

Εικόνα 3.5 *Εφαρμογές στη στοίβα*

Κατα την εκτέλεση της θα μας εμφανίσει :

```
3
Popped:     3
Popped:     3
1
Peek:       1
1
```

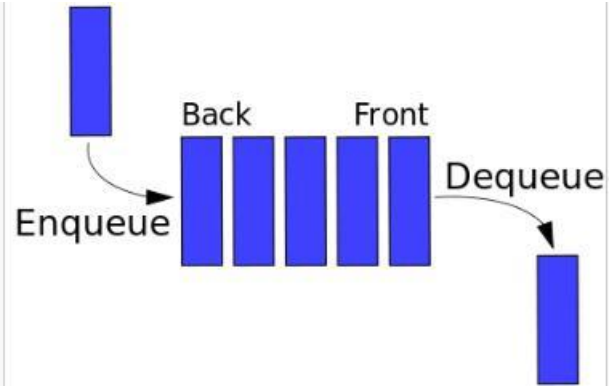
Εικόνα 3.6 *Εκτέλεση*

3.5 Ουρές

Είναι και αυτές **αφηρημένος τύπος δεδομένων** , είναι σημαντικές κατά την εφαρμογή αλγορίθμου **BFS** για γραφήματα , μπορούν να υλοποιηθούν εύκολα με **dynamic πινακες** και **λίστες** .

Βασικές λειτουργίες : enqueue() and dequeue(), peek()

Δομή FIFO: first in first out όπως βλέπουμε στην είκονα από κάτω



Εικόνα 3.7 *Δομή της Ουράς*

3.5.1 Εφαρμογές της Ουράς

Οι ουρές χρησιμοποιούνται κύριως σε **εφαρμογές** όπως είναι **CPU scheduling** δηλαδή όταν κυρίως θέλουμε να μοιράσουμε έναν πόρο σε πολλούς καταναλωτές. Επίσης σε εφαρμογές **IO buffers (input output buffers)** οπου τα δεδομένα μεταφέρονται ασύγχρονα μεταξύ δύο διεργασιών.

Ακόμα υπάρχουν **επιχειρησιακές εφαρμογές έρευνας** η **στοχαστικά μοντέλα** που βασίζονται σε μεγάλο βαθμό στις ουρές.

Μια εφαρμογή της ουράς σε python :

```
Class  Queue :
          def  __init__( self ) :
                self .queue = [ ]
def                     isEmpty (
                           self ) :
                return self.queue == [ ]
          def   enqueue ( self , data ) :
                self . queue.append ( data )
          def  dequeue ( self ) :
                data = self.queue [ 0 ]
                del self.queue [ 0 ]
                return data

          def  peek ( self ) :
                return self.queue [ 0 ]
          def sizeQueue ( self ):
                return len (self.queue)
queue = Queue ( )
queue.enqueue ( 10 )
queue.enqueue ( 20 )
queue.enqueue ( 30 )
print ( queue.sizeQueue ( ) )
print ( ''Dequeue : '', queue.dequeue ( ) )
print ( ''Dequeue : '', queue.dequeue ( ) )
print ( queue .sizeQueue ( ) )
```

Εικόνα 3.8 *Εφραμογή της Ουράς σε Python*

Όταν την εκτελέσουμε θα μας εμφανίσει :

```
3
Dequeue: 10
Dequeue: 20
1
```

Εικόνα 3.9 *Εκτέλεση της ουράς*

Όπου μας δείχνει ότι βάλαμε **3** στην ουρά , ότι αφαιρέσαμε **2** ψηφία από την ουρά και έμεινε μονο **1.**

Κεφάλαιο 3.6 Λίστες

Λίστα (List) είναι μία συλλογή τιμών οι οποίες συσχετίζονται με δείκτες .Οι τιμές της Λίστας ονομάζονται **στοιχεία (Elements)**.Τα στοιχεία τις λίστας μπορουν να εμφανίζονται παραπάνω απο μία φορά και δεν χρειάζεται να είναι του ίδιου τύπου δεδομένων .

Ακολουθίες (sequences) είναι οι λίστες οι οποίες συμπεριφέρονται σαν διατεταγμένες συλλογές τιμών.

Μία λίστα μέσα σε κάποια άλλη καλείται **εμφωλευμένη λίστα (nested list)** Τα στοιχεία μίας λίστας χωρίζονται με κόμμα μέσα σε τετράγωνες αγκύλες (**[στοιχείο]**). Μία λίστα που δεν περιέχει στοιχεία (Elements) ονομάζεται **άδεια λίστα** και συμβολίζεται με **[]**

Παράδειγμα :

```
>>> numbers = [ 1 , 2 , 3 , -4 ]
>>> Citys = [ 'ΑΘΗΝΑ' , ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ' , ' ΛΑΜΙΑ ' ]
>>> lst = [ 2 , 5.1 , 'ΣΠΑΡΤΗ' , [ 6, 7 , 8 ]]
>>> empty_list = [ ]
>>> print (numbers , '\n' , Citys , ' \n' , 1st , '\n' , empty_list)

[1,2,3,-4]
[ ' ΑΘΗΝΑ ', ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ', ' ΛΑΜΙΑ ' ]
[ 2 , 5.1 , ' ΣΠΑΡΤΗ ', [6, 7 ,8 ]]
[]
```

Εικόνα 3.10 *Παράδειγμα 1 στις Λίστες*

Λίστα με αλφαριθμητικά στοιχεία

Cities Δείκτες

0	1	2	3
'ΑΘΗΝΑ '	'ΠΑΤΡΑ '	'ΘΕΣΣΑΛΟΝΙΚΗ '	'ΛΑΜΙΑ '
-4	-3	-2	-1

```
>>> stringList = [ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ', ' ΛΑΜΙΑ ' ]

>>> print ( stringList)

[ ' ΑΘΗΝΑ ' , 'ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ' , ' ΛΑΜΙΑ ' ]
```

Εικόνα 4.0 *Παράδειγμα 2 στις Λίστες*

Εμφάνιση στοιχείου -1:

```
>>>  citiesList = [ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ', ' ΘΕΣΣΑΛΟΝΙΚΗ ', ' ΛΑΜΙΑ ' ]

>>> print (citiesList)

[ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ' , ' ΛΑΜΙΑ ' ]

>>> print ( citiesList [ -1 ] )

ΛΑΜΙΑ
```

Εικόνα 4.1 *Παράδειγμα 3 στις Λίστες*

Εμφάνιση σύνθετου αλφαρηθμιτικού:

```
>>> print (" Favorite cities : " + citiesList [ 1 ] + " , " + citiesList [3] )

Favorite cities : ΠΑΤΡΑ , ΛΑΜΙΑ
```

Εικόνα 4.2 *Παράδειγμα 4 στις Λίστες*

Λίστα με αριθμητικά στοιχεία

Numbers Δείκτες

0	1	2	3
1	2	3	-4
-4	-3	-2	-1

```
>>> numberList = [ 1 , 2 , 3 ,4 ]

>>> print ( numberList )

[1,2,3,4]
```

Εικόνα 4.3 *Παράδειγμα 5 στις Λίστες*

Πράξη στοιχείων Πίνακα:

```
>>> numberList = [ 2 , 4 , 6 , 8 ]

>>> print ( numberList [ 0 ] + numberList [ 3 ] )

10
```

Εικόνα 4.4 *Παράδειγμα 6 στις Λίστες*

3.6.1 Συναρτήσεις σε Λίστες

Στις Λίστες που τα στοιχεία τους είναι **συνεχόμενοι ακεραίοι αριθμοί** η Python μας παρέχει έναν εύκολο τρόπο για να τις δημιουργούμε.

Η συνάρτηση RANGE

Η συνάρτηση **range** επιστρέφει μια ακολουθία ακέραιων αριθμών ξεκινώντας απο το 0 απο προεπιλογή και αυξάνεται κατα ένα. Σταματάει πριν απο έναν προκαθορισμένο αριθμό. (Εικόνα 2.7)

Η συνάρτηση LIST

Η συνάρτηση **list** αρχικοποιεί τη νέα λίστα με τους ακεραίους αριθμούς της ακολουθίας. Χωρίς όρισμα η list επιστρέφει άδεια λίστα. (Εικόνα 2.7)

```
>>> numbers = list ( range ( 1 , 50 ) )

>>> print ( numbers )

[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23 ,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,

45,46,47,48,49]
```

Εικόνα 4.5 *Η συνάρτηση Range Και List*

Προσπέλαση στοιχείων μιας Λίστας

Για να προσπελάσουμε τα στοιχεία της λίστας , χρησιμοποιούμε παρόμοια σύνταξη με αυτή της προσπέλασης των χαρακτήρων μιας συμβολοσειράς.Κάθε ακέραια έκφραση μπορεί να χρησιμοποιηθεί ως δείκτης .Αν επιχειρήσουμε να προσπελάσουμε στοιχείο που δεν υπάρχει , τότε θα εμφανιστεί μήνυμα λάθους.

Παράδειγμα:

```
>>> numbers = [ 1 , 3 , 5 ,7 ]
>>> print ( numbers [ 0 ] )
1
>>> print ( numbers [ 3 ] )
7
>>> print ( numbers [ 2 – 1 ] )
3
```

Εικόνα 4.6 *Παράδειγμα 2 στις Λίστες*

3.6.2 Πρόσθεση στοιχείων στη Λίστα

Μέγεθος Λίστας LEN

Για να πάρουμε το μέγεθος μιας Λίστας χρησιμοποιούμε τη μέθοδο **len().** Στο παράδειγμα της εικόνας 1.9 χρησιμοποιούμε μία λίστα αριθμών 6 στοιχείων και χρησιμοποιούμε τη μέθοδο **len**() για να εμφανίσουμε το μέγεθος της:

```
>>> numberList = [1 , 2 , 3 , 4 , 5 , 6 ]

>>> print ( len (numberList ) )

6
```

Εικόνα 4.7 *Παράδειγμα μέθοδος **len** στις Λίστες*

Προσθήκη στοιχείου σε Λίστα

Για να προσθέσουμε ένα στοιχείο σε μια Λίστα χρησιμοποιούμε τη μέθοδο **append()** η οποία προσθέτει ένα στοιχείο στο τέλος της Λίστας**.** Για να εισάγουμε ένα στοιχείο σε μία συγκεκριμένη θέση χρησιμοποιούμε τη μέθοδο **insert(numberOfposition,item)**.

```
>>>  citiesList = [ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ' , ' ΛΑΜΙΑ ' ]
>>>  print ( citiesList )
[ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ' , ' ΛΑΜΙΑ ' ]
>>> citiesList.append ( ' ΒΟΛΟΣ ' )
>>> print ( citiesList )
[ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ' , ' ΛΑΜΙΑ ' , ' ΒΟΛΟΣ ' ]
>>> citiesList.insert ( 2 , ' ΠΕΙΡΑΙΑΣ ')
>>> print ( citiesList)
[ ' ΑΘΗΝΑ ' , ' ΠΑΤΡΑ ' , ' ΠΕΙΡΑΙΑΣ ' , ' ΘΕΣΣΑΛΟΝΙΚΗ ', ' ΛΑΜΙΑ ', ' ΒΟΛΟΣ ' ]
```

Εικόνα 4.8 *Παράδειγμα μέθοδος **append(), insert()** στις Λίστες*

3.6.3 Αφαίρεση στοιχείων απο τη Λίστα

Διαγραφή στοιχείου από Λίστα

Για να διαγράψουμε ένα στοιχείο σε μια Λίστα χρησιμοποιούμε τη μέθοδο **remove()** η οποία αφαιρεί ένα συγκεκριμένο στοιχείο της Λίστας**.** Επίσης μπορούμε να χρησιμοποιήσουμε τις μεθοδους **pop**() και **del** οι οποίες διαγράφουν το τελευταίο στοιχείο της λίστας ή ένα συγκεκριμένο στοιχείο. Η μέθοδος del μπορείνα διαγράψει τελείως τη Λίστα και η μέθοδος clear() αφαιρεί όλα τα στοιχεία της λίστας. Στο παράδειγμα χρησιμοποιούμε τις παραπάνω μεθόδους σε μία Λίστα με αλφαριθμητικά στοιχεία.

```
>>> namesList = [ ' ΝΙΚΟΛΑΣ ' , ' ΓΙΑΝΝΗΣ ' , ' ΣΟΥΛΗΣ ' , ' ΔΗΜΗΤΡΗΣ ' , ' ΦΟΙΒΟΣ ' ]
>>> print (namesList )
[ ' ΝΙΚΟΛΑΣ ' , ' ΓΙΑΝΝΗΣ ' , ' ΣΟΥΛΗΣ ' , ' ΔΗΜΗΤΡΗΣ ' , ' ΦΟΙΒΟΣ ']
>>> nameList.remove ( ' ΣΟΥΛΗΣ ')
>>> print (nameList)
[ ' ΝΙΚΟΛΑΣ ' , ' ΓΙΑΝΝΗΣ ', ' ΔΗΜΗΤΡΗΣ ' , ' ΦΟΙΒΟΣ ']
>>> namesList.pop (
) ' ΦΟΙΒΟΣ '
>>> print ( namesList)
[ ' ΝΙΚΟΛΑΣ ' , ' ΓΙΑΝΝΗΣ ', ' ΔΗΜΗΤΡΗΣ ']
>>> del namesList [ 0 ]
>>> print ( namesList )
[' ΓΙΑΝΝΗΣ ', ' ΔΗΜΗΤΡΗΣ ']
>>> namesList.clear ( )
>>> print (namesList)
[ ]
>>> del nameList
>>> print (namesList)
```

Εικόνα 4.9 *Παράδειγμα μέθοδος* ***remove(),pop(),del*** *στις Λίστες*

3.7 Δέντρα

Ορισμός:

Δένδρα είναι ένα σύνολο κόμβων που συνδέονται από ακμές και τόξα.

Στα δένδρα υπάρχει καλύτερη οργάνωση και όλες οι διαδικασίες είναι πιο αποδοτικές πχ από τις λίστες.

Από τι αποτελείται:

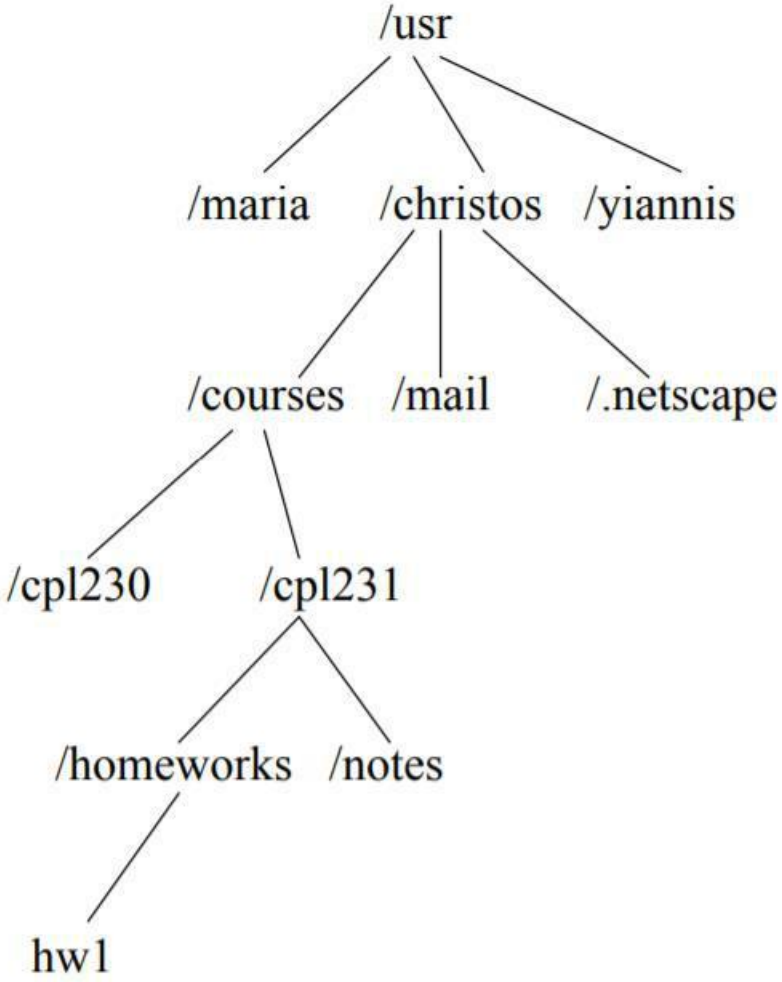
Ένα δένδρο έχει πάντα μια ρίζα ή είναι κενό.

Σε αυτή τη ρίζα ξεκινάνε κάποια υπόδενδρα το καθένα ξεχωριστό από τη ρίζα και από τα υπόλοιπα υπόδενδρα .

Επίσης στα δέντρα όλα είναι αναδρομικά είτε η αναζήτηση της είτε οτιδήποτε ακόμα και η δομή.

Όπως είπαμε τα δέντρα αποτελούνται από κόμβους όπου οι κόμβοι αποτελούνται από ακμές . Σε κάθε κόμβο εκτός από τη ρίζα καταλήγει μόνο μια ακμή , στη ρίζα πάλι καμία.

Παράδειγμα: Unix File System



Εικόνα 5.0 *Παράδειγμα Unix file system*

Πάμε σε κάποιους ορισμούς ή αλλιώς ονομασίες που έχουν δοθεί για τους κόμβους και τις ακμές.

Όταν υπάρχει μια ακμή από ένα κόσμο a σε ένα κόμβο b τότε ο a είναι ο πατέρας του b και ο b είναι το παιδί του a.

Αυτό γίνεται ώστε να μιλούμε για προγόνους και απογόνους κόμβων.

Κόμβοι που έχουν τον ίδιο πατέρα λέγονται αδέρφια.

Οι κόμβοι που δεν έχουν παιδιά λέγονται φύλλα . Οι υπόλοιποι λέγονται εσωτερικοί.

Βαθμός ενός κόμβου είναι ο αριθμός των παιδιών του.

Βαθμός ενός δέντρου είναι ο μέγιστος από τους βαθμούς των κόμβων του.

Συχνή χρήση δέντρων είναι τα δυαδικά(binary), τα τετραδικά(quadtrees) και τα οκταδικά (octtrees).

Βάθος ενός δέντρου ονομάζουμε το μέγιστο επίπεδο κόμβων του δέντρου.

Βάθος ενός κόμβου ονομάζουμε τον αριθμό των προγόνων του +1.

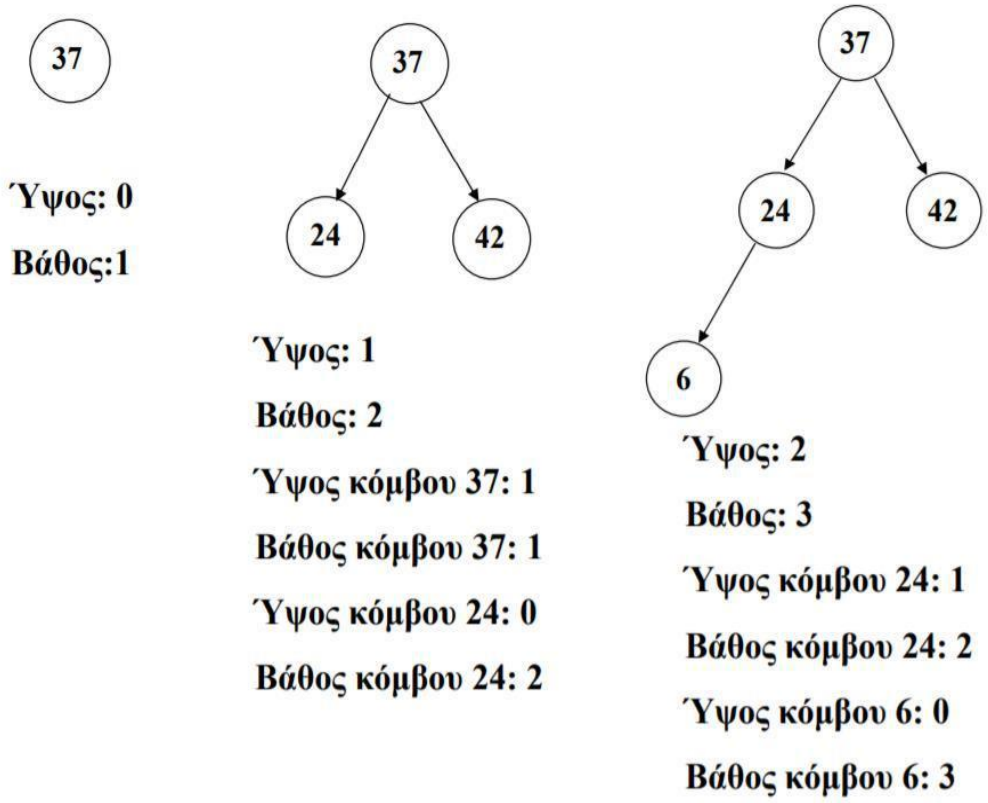
Ύψος ενός δέντρου ονομάζουμε το ύψος της ρίζας του δέντρου.

Ύψος ενός κόμβου ονομάζουμε το μήκος του μέγιστου μονοπατιού από τον κόμβο προς κάποιο φύλλο

Διαδρομή (path) ενός δένδρου είναι μια ακολουθία κόμβων v1, v2,, …, vk, όπου κάθε vi είναι πατέρας του vi+1. Το μήκος του μονοπατιού v1, v2,, …, vk είναι k-1.

Διατεταγμένο δένδρο ονομάζεται ένα δένδρο όταν για κάθε παιδί u κάποιου κόμβου v γνωρίζουμε αν ο u είναι το k-οστό παιδί του v.

Δίνουμε ένα παράδειγμα για τους περισσότερους ορισμούς στα δέντρα:



Εικόνα 5.1 *Παράδειγμα ύψους βάθους*

Όλοι αυτοί οι ορισμοί βοηθάνε πολύ όταν φτιάχνουμε ένα πρόγραμμα για την οργάνωση του και την εύρεση ενός στοιχείου για παράδειγμα :

Parent(a) { μας δίνει τον πατέρα του a} **Children(a)** {μας δίνει

το παιδί του a} **FirstChild(a)** {μας δίνει το πρώτο παιδί του a}

RightSibling(a) { μας δίνει τον κόμβο στα δεξιά του a}

LeftSibling(a) {μας δίνει τον κόμβο στα αριστερά του a}

IsLeaf(a) { εάν το a είναι φύλλο τότε TRUE εάν όχι τότε

FALSE}

IsRoot(a) { εάν το a είναι η ρίζα του δέντρου τότε TRUE εάν όχι τότε FALSE}
και **Depth(a)** {δώσε μας το βάθος του δέντρου}

3.7.1 Δυαδικά Δέντρα

Ένα δέντρο λέγεται δυαδικό όταν όλοι οι κόμβοι του έχουν βαθμό μικρότερη ή ίση του 2.

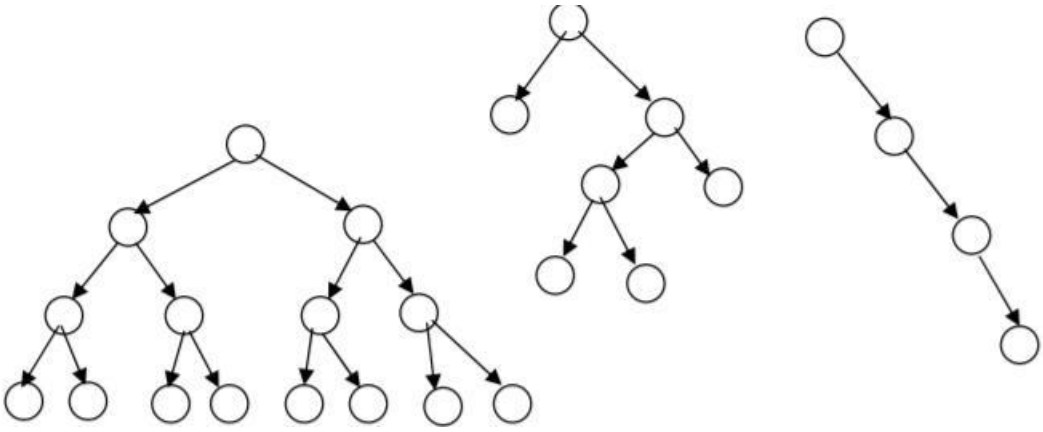
Ορισμός: Δυαδικό δένδρο λέγεται ένα δένδρο το οποίο είτε είναι κενό, είτε αποτελείται από μια ρίζα και δύο δυαδικά υπόδενδρα, το καθένα διακριτό από το άλλο και από τη ρίζα. Αναφερόμαστε στα δύο υπόδενδρα ως το αριστερό και το δεξιό υπόδενδρο.

Ένα δυαδικό δέντρο είναι γεμάτο , αν κάθε εσωτερικός του κόμβος έχει δύο παιδιά. Τέλειο λέγεται ένα δυαδικό δέντρο , αν είναι γεμάτο και όλα τα άλλα φύλλα έχουν το ίδιο βάθος.

Πλήρες είναι ένα δυαδικό δέντρο αν τηρεί αυτές τις προϋποθέσεις :

1) Έχει ύψος 0 και έναν κόμβο,
2) Έχει ύψος 1 και η ρίζα του έχει είτε 2 παιδιά είτε ένα αριστερό παιδί
3) Έχει ύψος n και η ρίζα του έχει ένα τέλειο αριστερό υπόδενδρο ύψους n-1 και ένα πλήρες δεξιό υπόδενδρο ύψους n-1 ή ένα πλήρες αριστερό υπόδενδρο n-1 και ένα τέλειο δεξιό υπόδενδρο ύψους n-2.

Παράδειγμα δυαδικού δέντρου:



Εικόνα 5.2 *Παράδειγμα δυαδικών δέντρων*

Όπως βλέπουμε και στο παράδειγμα έχουν όλοι οι κόμβοι το πολύ 2 παιδιά.

Τα δυαδικά δέντρα έχουν πολύ αποδοτική αναζήτηση σε ένα σύνολο στοιχείων τα οποία είναι τα δυαδικά δέντρα αναζήτησης(ΔΔΑ) .

Ένα δυαδικό δέντρο αναζήτησης(ΔΔΑ) είναι ένα δυαδικό δένδρο κάθε κόμβος u του οποίου ικανοποιεί τα εξής:

1) τα κλειδιά του αριστερού υποδένδρου του u είναι μικρότερα από το κλειδί του u.

2) τα κλειδιά του δεξιού υποδένδρου του u είναι μεγαλύτερα από το κλειδί του u.

Διαδικασία εύρεσης στοιχείου

Απλή αναδρομική στρατηγική: συγκρίνουμε το στοιχείο που μας ενδιαφέρει α με το στοιχείο της ρίζας του δένδρου β (αν υπάρχει) και

1) Αν α=β , σταματούμε
2) Αν α<β , προχωρούμε στο αριστερό υπόδεντρο,
3) Αν α>β , προχωρούμε στο δεξί υπόδεντρο

```
BSTnode* Find (BSTnode* r, int n) {
    If(r ==null) return null;
     Else
         If (n < (*r).Val)
                  return Find( (*r) . left, n) ;
         Else if ( n==(*r).Val ) )
                  Return r;
        Else return Find((*r). right, n);
```

Αυτό είναι ένα παράδειγμα για την απλή διαδικασία εύρεσης στοιχείου σε κώδικα .

Για να αφαιρέσουμε ένα κλειδί i από ένα ΔΔΑ:

1. Βρίσκουμε τον κόμβο u που περιέχει το i. Ας υποθέσουμε πως v είναι ο πατέρας του u.

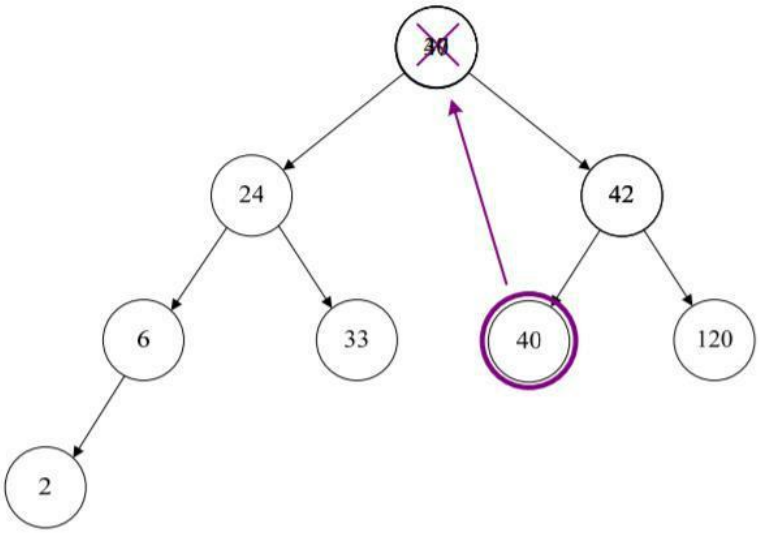
2. Αν u είναι φύλλο, τότε αλλάζουμε τον δείκτη του v που δείχνει το u, ώστε να γίνει null.

3. Αν ο u έχει ένα παιδί, τότε αλλάζουμε τον δείκτη του v που δείχνει τον u, ώστε να δείχνει στο παιδί του u.

4. Αν ο u έχει δύο παιδιά, – αλλάζουμε το κλειδί του u ώστε να γίνει το μικρότερο από τα κλειδιά όλων των απογόνων του που έχουν κλειδιά μεγαλύτερα του i. – καλούμε τη διαδικασία DeleteMin στο δεξιό παιδί του u.

Όλες αυτές τις διαδικασίες τις χρησιμοποιούμε για να αφαιρέσουμε ένα κόμβο ας δούμε ένα παράδειγμα εικονικά :

- Αφαίρεση του στοιχείου 37:



Εικόνα 5.3 *Παράδειγμα αφαίρεσης στοιχείου από δυαδικό δέντρο*

Διαγραφή κόμβου

• Το κόστος κάθε διαδικασίας είναι ανάλογο του βάθους/ύψους του δένδρου. – Θ(N) στη χειρότερη περίπτωση, – Θ(log N) στη μέση περίπτωση, υποθέτοντας πως όλα τα δένδρα είναι εξίσου πιθανά.

• Σημειώστε πως η διαδικασία διαγραφής κόμβου που έχουμε περιγράψει αφαιρεί πάντα κόμβο δεξιού υποδένδρου. Έτσι βοηθά την ύπαρξη μιας ασυμμετρίας στη δομή των ΔΔΑ (πιο βαθιά στα αριστερά).

Συμπέρασμα τα Δέντρα είναι μια δομή δεδομένων πολύ οργανωτική και δομημένη ώστε να μπορεί να πας από κόμβο σε κόμβο και να κάνεις εύκολες αναζητήσεις και διαγραφές και οτιδήποτε άλλο.

3.8 Λεξικά

Τα dictionaries είναι μια δομή δεδομένων όπου υποστηρίζεται από την python. Ο ορισμός της είναι ίδιος με την λίστα απλά ενώ στην λίστα υπάρχουν δείκτες για στα λεξικά (dictionaries) υπάρχουν λέξεις – κλειδιά.

Οι δείκτες στις λίστες είναι ακέραιοι αριθμοί ενώ οι λέξεις -κλειδιά μπορεί να είναι οτιδήποτε.

Στην συγκεκριμένη δομή δεδομένων μπορούμε να ανακτήσουμε δεδομένα μετά την καταχώρηση τους βεβαίως με το όνομα τους που είναι το κλειδί.

Ας δώσουμε ένα παράδειγμα:

```
A={‘S’:25, ‘Z’:40 , ‘T’:50}
```

Όπως βλέπουμε στο παράδειγμα από πάνω στην καταχώρηση μας έχουμε λέξεις-κλειδιά ζεύγη δηλαδή όπου το ‘S’ για παράδειγμα είναι το όνομα του και το 25 είναι η τιμή αυτού .

Τώρα αν λοιπόν θέλουμε να ανακτήσουμε κάποιες τιμές από αυτές θα γράψουμε:

```
>>> print A[‘S’] , A[‘Z’]
25 40
```

‘Ετσι γίνεται η ανάκτηση καλώντας από το A να μας δώσει το κάθε όνομα .

Αντίθετα με τις λίστες δεν χρειάζεται να γνωρίζουμε την θέση του στοιχείου για να το ανακτήσουμε μόνο το όνομα του.

Προσθέτουμε στοιχεία απλά με μια καταχώρηση

```
>>> A[‘Y’] = 36
>>> print A
{‘S’:25, ‘Z’:40 , ‘T’:50 , ‘Y’:36 }
```

Τώρα για να σβήσουμε απλά χρησιμοποιούμε την εντολή del

```
>>> del A[‘S’]
>>> print A
{‘Z’:40 , ‘T’:50 , ‘Y’:36}
```

Για να ορίσουμε ένα λεξικό στη python , ορίζουμε ένα όνομα για τη μεταβλητή και έπειτα μέσα σε άγκιστρα βάζουμε τα κλειδιά και τις τιμές. Τα ζεύγη τα χωρίζουμε με κόμμα . Για παράδειγμα:

```
>>>βαθμοί[={‘μαθηματικά’: 5, ‘φυσική’: 8, ‘γλώσσα’ :9} ]
>>>print( βαθμοί)
{‘ μαθηματικά’ : 5, ‘φυσική’: 8, ‘γλώσσα’ : 9}
>>>print( βαθμοί[‘φυσική’])
8
```

Πίνακας 5.4 *Παράδειγμα στα λεξικά*

Επίσης εδώ στο παράδειγμα εκτυπώνουμε και μόνο ένα στοιχείο από τους βαθμούς.

Ένας τρόπος να φτιάξουμε ένα λεξικό είναι να φτιάξουμε ένα άδειο λεξικό και μετά να προσθέσουμε όσα στοιχεία θέλουμε:

```
>>>Λαϊκή{}

>>>Λαϊκή[1]=’Πορτοκάλια’
>>>Λαϊκή[2]=’Μανταρίνια’

>>>Λαϊκή[3]=’καρπούζια’

>>>Λαϊκή[4]=’πεπόνια’

>>>Λαϊκή

(1:΄Πορτοκάλια’ , 2:’Μανταρίνια’, 3:’καρπούζια’,
4:’πεπόνια’)

>>>Λαϊκή[1]=’Μήλα’

>>>Λαϊκή

{1:’Μήλα’, 2:’Μανταρίνια’, 3:’καρπούζια’,
4:’πεπόνια’)
```

Πίνακας 5.5 *Παράδειγμα στα λεξικά 2*

Στο συγκεκριμένο παράδειγμα αλλάξαμε και στοιχεία ως εξής :

>>>Λαική[1] =’Μήλα’

>>>Λαική{1: ‘Μήλα’ , 2 : ‘Μανταρίνια’ , 3; ‘καρπούζια’, 4: ‘πεπόνια’}

Στην ουσία τα λεξικά (dictionaries) είναι πιο απλά στη χρήση τους από τις λίστες και μας βοηθάνε όταν θέλουμε να εστιάσουμε στα ονόματα και όχι να ψάχνουμε τους δείκτες από τις λίστες.

Οι βασικές συναρτήσεις για τα λεξικά είναι οι παρακάτω:

Συνάρτηση	Περιγραφή
dict(k)	Μετατρέπει την ακολουθία ζευγών k σε λεξικά
del(a[x])	Διαγράφει τα στοιχεία με κλειδί x από το λεξικό a
len(a)	Επιστρέφει το πλήθος των στοιχείων του λεξικού a
sorted(a)	Επιστρέφει σε λίστα τα κλειδιά του λεξικού a ταξινομημένα

```
>>>k= [ (1, 'a'), (2, 'b'), (3, 'c')]
>>>a= dict (k)
>>>a
(1: 'a', 2: 'b', 3: 'c')
      >
>>>del (a[3])
>>>a
(1: 'a', 2: 'b')
```

```
>>>a={3:'c',5:'e',1:'a',4:'d'}
>>>len(a)
4
>>>sorted(a)
(1, 3, 4, 5)
```

Πίνακες 5.6 *Παραδείγματα συναρτήσεων στα λεξικά*

Και βασικές μέθοδοι για λεξικά :

Μέθοδος	Περιγραφή
a.clear()	Διαγράφει όλα τα στοιχεία του λεξικού a
a.update(a1)	Συγχωνεύει τα στοιχεία του λεξικού a1 στο a
a.items()	Επιστρέφει όλα τα στοιχεία του λεξικού a σε ζευγάρια λίστας(τα ζευγάρια είναι πλειάδες)
a.keys()	Επιστρέφει όλα τα κλειδιά του λεξικού a σε λίστα
a.values	Επιστρέφει όλες τις τιμές του λεξικού a σε λίστα
a.copy()	Αντιγράφει όλα τα στοιχεία του λεξικού a

```
>>>a={1: 'a', 3: 'c', 4: 'd', 5: 'e'}
>>>a.clear()
>>>a
{}
```

```
>>>a= (1:'a' ,2:'b')
>>>a1= (3:'c', 4: 'd')
>>>a.update(a1)
>>>a
(1 : 'a' ,2: 'b' , 3: 'c' , 4: 'd')
>>>a.items()
[(1, 'a') , (2,'b'), (3,'c') , (4, 'd') ]
>>>a.keys()
[1, 2, 3, 4]
>>>a.values()
['a' , 'b' , 'c' , 'd']
>>>n = a.copy()
>>>n
(1 : 'a' , 2: 'b' , 3 : 'c' , 4: 'd')
```

Πίνακας 5.7 *Παράδειγμα μεθόδων στα λεξικά*

3.9 Πλειάδες

Ορισμός:

Οι πλειάδες είναι μια ομαδοποίηση ή μια ακολουθία στοιχείων με συγκεκριμένη σειρά όπου αντιστοιχίζονται με δείκτες. Αυτά τα στοιχεία μπορεί να είναι αριθμοί , λίστες , πλειάδες και συμβολοσειρές.

Οι πλειάδες μοιάζουν με τις λίστες που έχουμε αναφέρει πιο πάνω στην χρήση των δεικτών και στον τρόπο χρήσης τελεστών άνω κάτω τελείας σε αγκύλες.

Αξίζει να σημειωθεί ότι οι πλειάδες διαφέρουν πολύ από τις άλλες δομές διότι είναι αμετάβλητες. Αυτό δυσκολεύει λίγο την χρήση τους και τις κάνει να χρησιμοποιούνται για πιο αποκλειστικές χρήσεις όπου θα υπάρχει μια ακολουθία στοιχείων που δεν πρόκειται να αλλάξει.

Όπως και στις λίστες έτσι και στις πλειάδες στον κώδικα περικλείονται από παρενθέσεις και χωρίζονται με κόμματα τα στοιχεία.

Ας δούμε ένα απλό παράδειγμα:

Στο παρακάτω παράδειγμα δίνουμε τρείς πλειάδες ξεχωριστά στην πρώτη πλειάδα k βάζουμε 4 τιμές , την δεύτερη την δημιουργούμε κενή με μια παρένθεση γιατί όπως είπαμε πάντα χρησιμοποιούμε παρενθέσεις και στην $3^{η}$ πλειάδα βάζουμε μόνο έναν αριθμό και το διαχωρίζουμε με κόμμα γιατί αλλιώς δεν θα το δεχτεί το πρόγραμμα μας.

```
k = (1,2,3,4)
l = ()
p = (6,)
print k , l , p
(1,2,3,4) () (6,)
```

Πίνακας 5.8 *Παράδειγμα πλειάδας*

Ας δούμε ένα παράδειγμα με το πώς επιλέγουμε στοιχεία από τη πλειάδα χρησιμοποιώντας τους δείκτες:

```
p ('t', 1, 'o', 2, 'e', 3)
print p[2] , p[-3] , p[0]
0 2 t
print p[:3] , p[:2]
('t', 1, 'o') ('t' ,1)
```

Πίνακας 5.9 *Παράδειγμα πλειάδας 2*

Σε αυτό το παράδειγμα εκτός από το να επιλέγουμε στοιχεία επιλέγουμε και διάστημα τιμών με την άνω και κάτω τελεία.

Στις πλειάδες μπορούμε να κάνουμε δύο ειδών πράξεις πρόσθεση και πολλαπλασιασμό αυτές τις δύο υποστηρίζουν.

Ας δούμε ένα παράδειγμα πρόσθεσης (η πρόσθεση γίνεται μόνο ανάμεσα σε πλειάδες):

```
a= (1,2,3,4)
b= ('t', 'k', 'l')
z=a+b
z=(1,2,3,4, 't', 'k', 'l')
type(z)
<type 'tuple'>
```

Πίνακας 6.0 *Παράδειγμα πρόσθεση πλειάδας*

Στο παράδειγμα από πάνω γίνεται η πρόσθεση με την σειρά που την δίνουμε εμείς (το type'tuple' είναι για να κάνει το z πλειάδα θα το αναφέρουμε πιο κάτω στις βασικές συναρτήσεις των πλειάδων)

Αντίθετα ο πολλαπλασιασμός δεν γίνεται ανάμεσα σε πλειάδες αλλά ανάμεσα σε πλειάδες και ακέραιους μόνο αριθμούς. Δηλαδή:

Εικόνα 6.1 *Παράδειγμα πολλαπλασιασμού πλειάδας*

Γενικά ότι ισχύει στις πράξεις για τις λίστες ισχύει και για τις πλειάδες.

Ας αναφέρουμε μερικές βασικές συναρτήσεις και μεθόδους που χρησιμοποιούμε στις πλειάδες :

Συνάρτηση	Περιγραφή
tuple(k)	Μετατρέπει την ακολουθία k σε πλειάδα
len(a) min(a) max(a)	Λειτουργούν όπως και στις λίστες. Επιστρέφουν το πλήθος των στοιχείων, το ελάχιστο και το μέγιστο της πλειάδας a
any(a)	Επιστρέφει True, αν κάποιο στοιχείο της πλειάδας a είναι True
all(a)	Επιστρέφει True , αν όλα τα στοιχεία της πλειάδας a είναι True

Μέθοδος	Περιγραφή
a.index(x)	Επιστρέφει τον πρώτο δείκτη του στοιχείου x από την πλειάδα a
a.count(x)	Επιστρέφει το πλήθος των εμφανίσεων του x στη πλειάδα a

Εικόνα 6.2 *Συναρτήσεις και μέθοδοι πλειάδας*

```
>>>a= (1, 'x' , 3, 'x' , 'xx' )
>>>a.index('x')
1
>>>a.count('x')
2
```

```
>>>k= [ ‘a’, ‘b’, [1,2,3] ,4]
>>>a= tuple(k)
>>>a
(‘a’, ‘b’, [1,2,3], 4)
>>>len (a)
4
>>>min(a)
4
>>>max(a)
‘b’
>>>any(a)
True
>>>all(a)
True
```

Πίνακας 6.3 *Συναρτήσεις και μέθοδοι πλειάδας παραδείγματα*

3.10 Σύνολα

Τα σύνολα όπως τα χρησιμοποιούμε στα μαθηματικά έτσι και όπως είναι λογικό βρίσκονται στις δομές δεδομένων.

Γενικά τα σύνολα είναι μια ακολουθία τιμών που είναι μη διατεταγμένη δηλαδή που δεν έχει σημασία η σειρά. Οι τιμές που περιλαμβάνονται στα σύνολα ονομάζονται στοιχεία όπως και στις υπόλοιπες δομές δεδομένων. Στα σύνολα δεν υπάρχουν δείκτες όπως είναι προφανές αφού είναι μη διατεταγμένες δομές δεδομένων.

Τα σύνολα δέχονται μόνο αμετάβλητα στοιχεία όπως αριθμούς, συμβολοσειρές ή πλειάδες , αντίθετα τα ίδια τα σύνολα είναι μεταβαλλόμενα δηλαδή μπορεί να γίνει αλλαγή σε ένα ή πολλά στοιχεία του. Όπως ξέρουμε και από τα μαθηματικά τα σύνολα περικλείονται σε άγκιστρα και χωρίζονται με κόμμα.

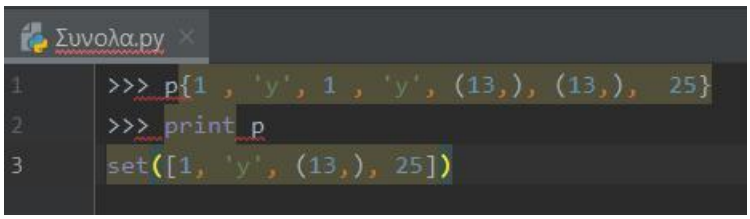
Παρακάτω θα δούμε ένα απλό παράδειγμα που θα περιέχει αριθμούς αλλά και συμβολοσειρές αλλά και πώς ορίζουμε ένα κενό σύνολο.

```
>>>b={25,14,'c',2,'t'}
>>>print b
set([2, 't', 14, 25 , 'c'])
>>>h={}
>>>h=set(h)
>>>type(b), type(h)
(<type 'set'>, <type 'set'>)
```

Πίνακας 6.4 *Παράδειγμα συνόλων*

Όπως βλέπουμε τα στοιχεία δεν εκτυπώνονται με την σωστή σειρά γιατί δεν έχει σημασία επειδή είναι σύνολο . Επίσης για να ορίσουμε κενό σύνολο μετατρέποντας με set() σε σύνολο διαφορετικά το δέχεται ως λεξικά το πρόγραμμα.

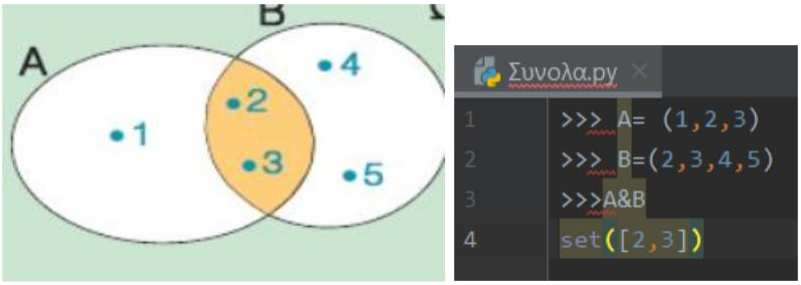
Επιπρόσθετα πρέπει να αναφέρουμε ότι στα σύνολα τα επαναλαμβανόμενα στοιχεία τα αφαιρεί αυτόματα το πρόγραμμα . Παράδειγμα:



Εικόνα 6.5 *Παράδειγμα συνόλων 2*

3.10.1 Τομή συνόλων

Η τομή συνόλων είναι μια από τις πράξεις των συνόλων όπου λέγοντας τομή εννοούμε τα κοινά στοιχεία μεταξύ συνόλων . Ο τελεστής της τομής είναι το σύμβολο &. Για παράδειγμα έχουμε δύο σύνολα Α και Β όπου Α = 1 ,2 ,3 και Β= 2,3,4,5



Εικόνα 6.6 *Παράδειγμα τομή συνόλων*

Η τομή τους όπως βλέπουμε είναι το 2 , 3

Για κάθε πράξη υπάρχουν δύο μέθοδοι , στις τομές έχουμε την intersection και την intersection_update.

Χρησιμοποιώντας τα σύνολα Α και Β η A.intersection(B) επιστρέφει την τομή των Α και Β χωρίς να αλλάξει τα σύνολα Α και Β ενώ η A.intersection_update(B) μετατρέπει το σύνολο Α στην τομή των συνόλων Α και Β.

Μέθοδοι για τη τομή συνόλων	
Y=A.intersection(B)	Επιστρέφει την τομή των Α και Β χωρίς να αλλάζει τα σύνολα Α,Β

X=A.intersection_update(B)	Μετατρέπει το σύνολο A στη τομή των συνόλων A και B

Πίνακας 6.7 *Παράδειγμα μεθόδων για τομή συνόλων*

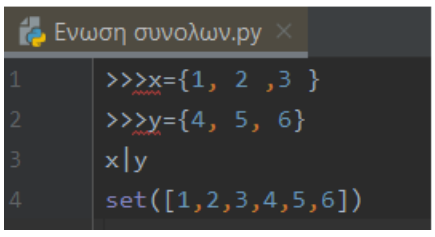
```
>>>A={5,6,1,2,3}
>>>B={7,8,1,2,3}
>>>Y=A.intersection(B)
>>>Y
set([1,2,3])
>>>X=A.intersection_update(B)
>>>A
set([1,2,3])
```

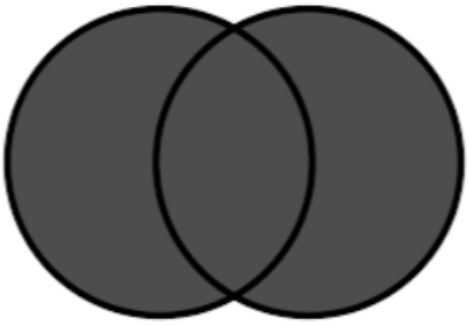
Εικόνα 6.8 *Παράδειγμα μεθόδων στη τομή συνόλων*

3.10.2 Ένωση συνόλων

Η ένωση συνόλων είναι άλλη μια πράξη των συνόλων όπου περιλαμβάνει όλα τα στοιχεία των συνόλων. Ο τελεστής είναι το σύμβολο της καθέτου | .

Έστω ότι έχουμε 2 σύνολα x και y όπου x = 1,2,3 και y = 4,5,6 :





Εικόνα 6.9 *Παράδειγμα ένωσης συνόλων*

Όπως αναφέραμε πριν έχουμε 2 μεθόδους στην κάθε πραξη εδώ έχουμε την union και την update.

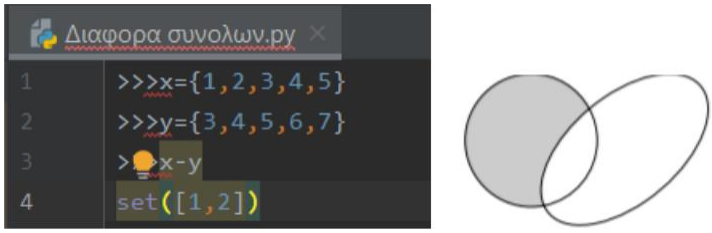
Μέθοδοι για την ένωση συνόλων	
L=x.union(y)	Εισχωρεί στο L την ένωση των δύο συνόλων x και y
x.update(y)	Μετατρέπει το σύνολο x στην ένωση των συνόλων x και y

```
>>>x = (1,2,3)
>>>y = (4,5,6)
>>>L=x.union(y)
>>>L
 set([1,2,3,4,5,6])
>>>x.update(y)
>>>x
 set([1,2,3,4,5,6])
```

Εικόνα 6.10 *Παράδειγμα μεθόδων ένωσης συνόλων*

3.10.3 Διαφορά συνόλων

Η διαφορά είναι τα μη κοινά στοιχεία του πρώτου συνόλου , του συνόλου πριν τον τελεστή. Ο τελεστής είναι το μείον - . Σε ένα παράδειγμα x και y όπου x = 1,2,3,4,5 και y= 3,4,5 ,6,7



Εικόνα 7.0 *Παράδειγμα διαφοράς συνόλων*

Και οι μέθοδοι που χρησιμοποιούμε εδώ είναι η difference και η difference_update.Η difference επιστρέφει τη διαφορά του συνόλου που είναι πρίν την τελεία και η difference_update μετατρέπει το σύνολο πριν την τελεία στη διαφορά του από το σύνολο που βρίσκεται στη παρένθεση της μεθόδου.

Μέθοδοι για την διαφορά συνόλων	
K=x.difference(y)	Εκχωρεί στο K τη διαφορά συνόλου x από το y
x.difference_update(y)	Μετατρέπει το σύνολο x στη διαφορά του από το y

```
>>>x={1,2,3,4,5}
>>>y={3,4,5,6,7}
>>>k= x.difference(y)
>>>k
set([1,2,3])
>>>x.difference_update(y)
>>>x
set([3,2,1])
```

Εικόνα 7.1 *Παράδειγμα μεθόδων διαφοράς συνόλων*

3.10.4 Συμμετρική διαφορά συνόλων

Η συμμετρική διαφορά συνόλων είναι η πράξη με τα μη κοινά στοιχεία και των δύο συνόλων μαζί. Ο τελεστής είναι το σύμβολο ^ .

Παμε πάλι με το παράδειγμα x και y όπου x =1,2,3,4,5 και y=3,4,5,6,7:

Εικόνα 7.2 *Παράδειγμα συμμετρικής διαφοράς συνόλων*

Στην συμμετρική διαφορά έχουμε τις μεθόδους symmetric_difference και symmetric_difference_update:

Μέθοδοι για συμμετρική διαφορά	
K=x.symmetric_difference(y)	Εκχωρεί στο K τη συμμετρική διαφορά των συνόλων x και y
x.symmetric_difference_update(y)	Μετατρέπει το σύνολο x στην συμμετρική διαφορά των συνόλων x και y

```
>>>x={1,2,3,4,5}
>>>y={3,4,5,6,7}

>>>k=x.symmetric_difference(y)

>>>x.symmetric_difference_update(y)
>>>k,x
set([1,2,6,7]) , set([1,2,6,7])
```

Εικόνα 7.3 *Παράδειγμα μεθόδων συμμετρικής διαφοράς*

3.10.5 Υποσύνολα και τελεστές σύγκρισης

Στα σύνολα υπάρχει και η έννοια του υποσυνόλου. Ένα σύνολο y λέμε ότι είναι υποσύνολο ενός άλλου συνόλου x όταν κάθε στοιχείο του y υπάρχει και στο x. Το αντίθετο του υποσυνόλου είναι το υπερσύνολο. Για να δούμε αν είναι υποσύνολο κάποιου άλλου χρησιμοποιούμε τους τελεστές σύγκρισης. Ας κάνουμε ένα παράδειγμα:

```
>>>x={1,2,3,4,5}
>>>y={1,2,3}
>>>x>=y
True
>>>x<y
False
>>>x==y
False
```

Εικόνα 7.4 *Παράδειγμα υποσυνόλων*

Στο παράδειγμα αυτό το x είναι υπερσύνολο του y και το y είναι υποσύνολο του x .Με τους τελεστές βλέπουμε ότι έχουμε απαντήσεις TRUE or FALSE αυτό μας βοηθάει να καταλάβουμε ποιο είναι υποσύνολο και ποιο υπερσύνολο.

Υπάρχουν και εδώ τρείς μέθοδοι για το υποσύνολα.

Μέθοδοι υποσυνόλων	
x.issubset(y)	Επιστρέφει TRUE,αν το x είναι υποσύνολο του y,αλλιώς FALSE
x.issuperset(n)	Επιστρέφει TRUE, αν το x είναι υπερσύνολο του n, αλλιώς FALSE
x.isdisjoint(m)	Επιστρέφει TRUE αν και τα δύο σύνολα x και m δεν έχουν ούτε ένα κοινό στοιχείο

Επίσης έχουμε και κάποιες συναρτήσεις για τα σύνολα και μεθόδους.

Συνάρτηση	Περιγραφή
set(k)	Μετατρέπει την ακολουθία k σε σύνολο
len(a) **min(a)** **max(a)**	Λειτουργούν όπως και στις λίστες. Επιστρέφουν το πλήθος των στοιχείων, το ελάχιστο και το μέγιστο του συνόλου a.

Μέθοδος	Περιγραφή
a.add(b)	Προσθέτει το στοιχείο b στο σύνολο a
a.clear()	Διαγράφει όλα τα στοιχεία του συνόλου a
a.remove(b)	Αφαιρεί απο το σύνολο α το στοιχείο b.Αν δεν υπάρχει εμφανίζει μήνυμα σφάλματος.
a.discard(b)	Αφαιρεί απο το σύνολο a το στοιχείο b. Αν δεν υπάρχει δεν εμφανίζει μήνυμα σφάλματος.

```
>>>k='abcde'

>>>a=set(k)

>>>a

set(['a', 'c', 'b', 'e' ,'d'])

>>>min(a), max(a), len(a)


('a', 'e', 5)
```

```
>>>a=(1, ‘s’, 18)
>>>a.add(100)

>>>a
set([1, ‘s’ ,100 ,18])

>>>a.clear()

>>>a

set([])
```

```
>>>a={1,2,3,4}

>>>a.remove(1)

>>>a.discard(3)

>>>a

set([2, 4])
```

Εικόνα 7.5 *Παράδειγμα μεθόδων και συναρτήσεων υποσυνόλω*

3.11 Αλφαριθμητικά

Τα αλφαριθμητικά είναι μια δομή δεδομένων που την βλέπουμε σχεδόν σε όλες τις γλώσσες προγραμματισμού.

Ορισμός: Τα αλφαριθμητικά είναι μια ακολουθία από χαρακτήρες και τα λέμε αλφαριθμητικά γιατί περιέχουν αριθμούς , χαρακτήρες ή άλλα σύμβολα.

Εκτός από ότι χρησιμοποιούνται σε πολλές γλώσσες χρησιμοποιούνται και σε πολλές δομές δεδομένων π.χ print (‘Kwstas’) .

Εκτός από γράμματα και ψηφία υπάρχουν και πολλοί ειδικοί χαρακτήρα

Σταθερές χαρακτήρων Η έκφραση ’z’ είναι μια σταθερά και αντιστοιχεί σε έναν int με τα 8 τελευταία του bits ίσα με τον ASCII κωδικό του χαρακτήρα

Τα Strings είναι πάντα πίνακες από χαρακτήρες .Ένα String είναι δείκτης στον 1ο χαρακτήρα του πίνακα Τιμή του string είναι η διεύθυνση του 1ου χαρακτήρα του πίνακα

Ας αναφέρουμε μερικά strings στην python

Η συνάρτη print	`print("Hello")` `print('Hello')`
String μέσα σε μεταβλητή	`a = "Hello"` `print(a)`
Strings σαν πίνακες	`a = "Hello, World!"` `print(a[1])`
Length των strings με το len	`a = "Hello, World!"` `print(len(a))`
Με το strip αφαιρούμε όποιο κένο υπάρχει από την αρχή	`a = " Hello, World! "` `print(a.strip()) # returns "Hello, World!"`
Με το lower βγάζουμε τα κεφαλαία γράμματα	`a = "Hello, World!"` `print(a.lower())`

Εικόνα 7.6 *Παράδειγμα Αλφαριθμητικών strings*

Κεφάλαιο 4 - Γιατί η η Γλώσσα προγραμματισμού Python είναι κατάλληλη για εκμάθηση και για διδασκαλία

Η Python είναι μία γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί σε μία μεγάλη ποικιλία εφαρμογών, είναι μια εξαιρετική γλώσσα προγραμματισμού για επαγγελματίες άλλα και αρχάριους που μπαίνουν στον συναρπαστικό κόσμο της πληροφορικής.

Είναι μία γλώσσα προγραμματισμού **απλή** και **εύκολη** στην εκμάθηση και δεν έχει περίπλοκη σύνταξη κώδικα και μοιάζει με την αγγλική γλώσσα , έτσι την καθιστά στις πιο δημοφιλείς γλώσσες προγραμματισμού.

Υπάρχουν συγκεκριμένα τομείς της πληροφορικής που η Python υπερέχει.Κάποιοι απο τους τομείς είναι :

- Ανάπτυξη API
- Επιστήμη δεδομένων (Date Science)
- Γλώσσα προγραμματισμού σεναρίων (Scripting)
- Ανάπτυξη εφαρμογών στο διαδίκτυο

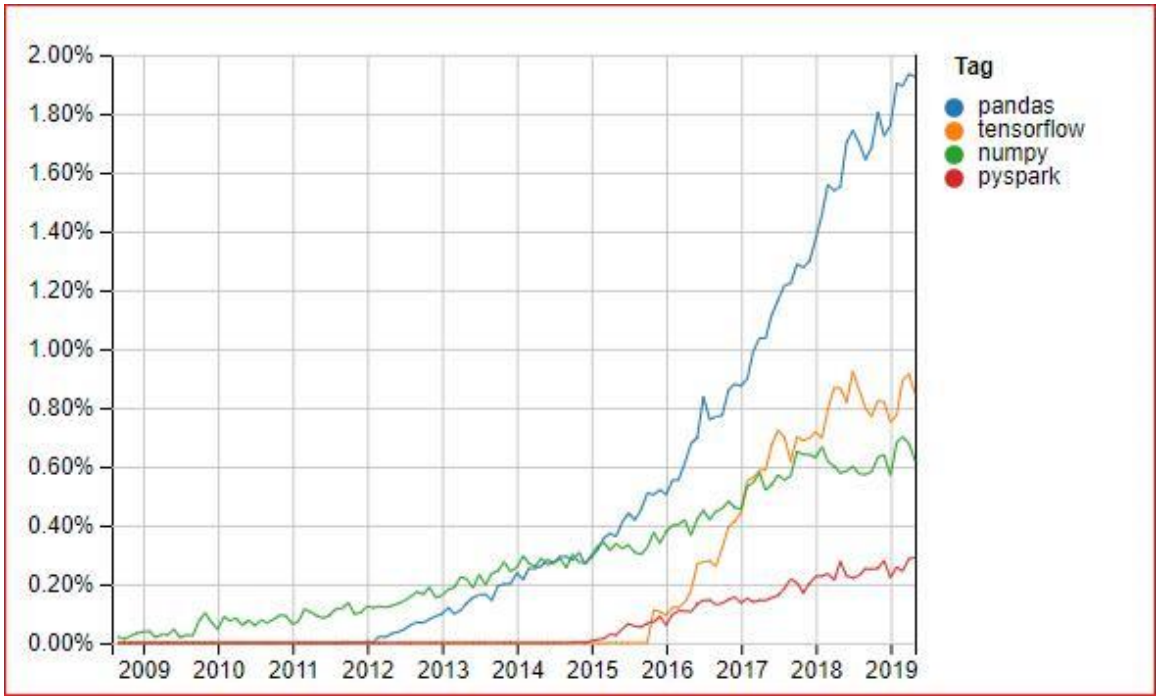
4.1 Η χρηση της Python στην επιστήμη των δεδομένων

Η Python χρησιμοποιείται συχνά για την **επιστήμη των δεδομένων (Date Science)** σε σχέση με άλλες γλώσσες προγραμματισμού , καθώς διαθέτει ένα μεγάλο αριθμό βιβλιοθηκών για **στατιστική ανάλυση**.Ο κώδικας θεωρείται ευκολότερος στην **συντήρηση** και **επεκτάσιμος**.Αυτοί είναι και οι λόγοι για τους οποίους έχει αυξηθεί η δημοτικότητα της γλώσσας στην επιστήμη των δεδομένων και ειδικά μεταξύ επαγγελματιών χωρίς προηγμένη εκπαίδευση στη στατιστική και τα μαθηματικά.Αυτές οι βιβλιοθήκες επιτρέπουν στον προγραμματιστή να αναλύει μεγάλο όγκο δεδομένων αποφεύγοντας την πολυπλοκότητα άλλων γλωσσων.

Κάποιες απο αυτές τις **βιβλιοθήκες στατιστικής ανάλυσης** είναι :

- **Pandas**
- **Tensoflow**
- **Numpy**
- **Pyspark**

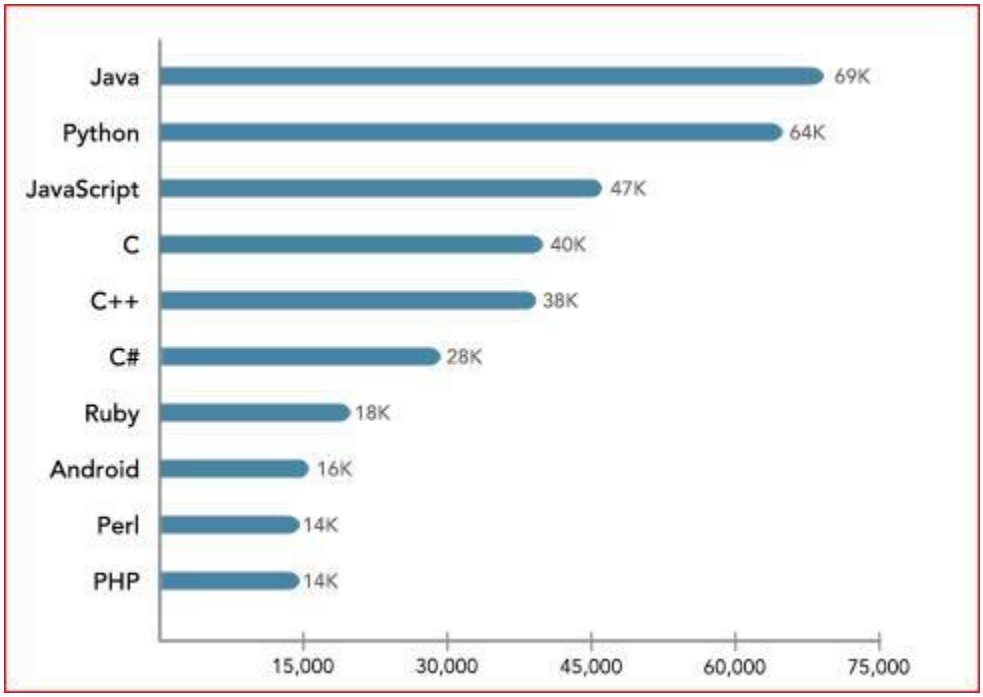
Στο παρακάτω παράδειγμα παρουσιάζεται η ανάπτυξη των βιβλιοθηκων στατιστικής ανάλυσης απο το 2009 μέχρι σήμερα.



Εικόνα 7.7 *Παράδειγμα ανάπτυξης των βιβλιοθηκών στατιστικής ανάλυσης*

4.2 Python - η γλώσσα προγραμματισμού με την μεγαλύτερη ζήτηση

Η Python βρίσκεται στην δεύτερη θέση της λίστας με τις γλώσσες προγραμματισμού που έχουνε την μεγαλύτερη ζήτηση στην αγορά εργασίας ,με βάση την ζήτηση θέσεων εργασίας σε μια απο τις μεγαλύτερες πλατφόρμες (www. Indeed.com) .Αυτά τα στοιχεία αντικατοπρίζουν την δυναμική που έχει γλώσσα και το μεγάλο πεδίο εφαρμογής που έχει δημιουργήσει .



Εικόνα 7.8 *Παράδειγμα ανάπτυξης των γλωσσών προγραμματισμού*

4.3 Μηχανική Εκμάθηση (Maschine Learning)

Ένας άλλος λόγος που οι προγραμματιστές μαθαίνουν την Python είναι η ανάπτυξη της μηχανικής μάθησης τα τελευταία χρόνια.Οι αλγόριθμοι γίνονται όλο και πιο πολύπλοκοι στις μέρες μας ,όπως οι αλγόριθμοι της Google για παράδειγμα.Υπάρχει ένας μεγάλος αριθμός απο βιβλιοθήκες μηχανικής μάθησης,οι οποίες την καθιστούν την καλύτερη γλώσσα προγραμματισμου στον συγκεκριμένο τομέα, και την κοινότητα των προγραμματιστών.

Εικόνα 7.9 *Παράδειγμα μηχανικής εκμάθησης*

4.4 Ανάπτυξη Διαδικτυακών εφαρμογών (Web development)

Σε σύγκριση με άλλες γλώσσες προγραμματισμού όπως Php,Java ή C++ μπορούμε να πούμε ότι η Python είναι πιο εύκολη στην εκμάθηση της.Πολλές απο τις κορυφαίες εφαρμογές στο διαδίκτυο χρησιμοποιούν την Python :

- **Instagram**: μία πλατφόρμα κοινωνικών μέσων που βασίζεται στην Python για να επιτρέπει στους χρήστες να φωτογραφίζουν,να επεξεργάζονται και να μοιράζονται τις φωτογραφίες τους σε ένα ψηφιακό album.
- **Spotify**: Από τις μεγαλύτερες πλατφόρμες διαμοιρασμού μουσικής .
- **Facebook** : Η μεγαλύτερη πλατφόρμα κοινωνικής δικτύωσης, στην οποία οι χρήστες ανεβάζουν φωτογραφίες αλληλεπιδρούν μεταξύ τους .

Επίσης ένα ακόμα θετικό χαρακτηριστικό που έχει η Python στην ανάπτυξη διαδικτυακών εφαρμογών(web developments) είναι ότι διαθέτει πολλά πλαίσια (frameworks) που απλοποιούν τη διαδικασία ανάπτυξη της εφαρμογής .Τα πιο διαδεδομένα frameworks είναι :

Εικόνα 8.0 *Παραδείγματα frameworks*

5. ΒΙΒΛΙΟΓΡΑΦΙΑ- ΑΝΑΦΟΡΕΣ

- ΜΑΘΑΙΝΩ PYTHON & TKINTER – Κωνσταντίνος Παπαστεργίου
- Εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό με Python – Μαγκούτης Κωνσταντίνος , Νικολάου Χρήστος
- https://www.learnpython.org/
- https://www.cs.ucy.ac.cy/courses/EPL231/2010-winter/notes/notes9-10.pdf Πανεπιστήμιο Κύπρου Μάθημα Δέντρων
- https://repository.kallipos.gr/bitstream/11419/2751/1/chap05.pdf ΚΑΛΛΙΠΟΣ
- https://www.cs.ucy.ac.cy/~dzeina/courses/epl032/slides/19.pdf
- https://stackoverflow.blog/2017/09/06/incredible-growth-python/ ΕΡΕΥΝΕΣ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΤΗΣ PYTHON
- https://www.python.org/doc/
- https://el.wikipedia.org/wiki/Python
- https://www.w3schools.com/python/default