ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΣΧΟΛΗ: ΔΙΟΙΚΗΣΗ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ: ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ (ΠΑΤΡΑ)
(ΠΡΩΗΝ ΤΜΗΜΑ ΕΣΠΣ)

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΝΟΣ ΜΑΖΙΚΟΥ ΔΙΑΔΙΚΤΥΑΚΟΥ ΜΑΘΗΜΑΤΟΣ (MOOC) ΓΙΑ ΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

Καρρίτσα Βαλεντίν ΑΜ: 11825
Επιβλέπων: **Κουτσονίκος Ιωάννης**

ΠΑΤΡΑ 2018

ΠΡΟΛΟΓΟΣ

Η ελκυστικότητα της γλώσσας προγραμματισμού Python σε συνδυασμό με την πρόκληση της διδασκαλίας της στο πλαίσιο ενός μαζικού ανοικτού διαδικτυακού μαθήματος (MOOC) ήταν οι βασικοί λόγοι επιλογής αυτού του θέματος πτυχιακής εργασίας για την ολοκλήρωση των σπουδών μου στο ΤΕΙ Δυτικής Ελλάδας.

Η Python είναι μια δυναμική γλώσσα προγραμματισμού υψηλού επιπέδου, η οποία διανέμεται δωρεάν και ανήκει στο ελεύθερο λογισμικό. Η δομή της πτυχιακής εργασίας χωρίζεται σε δυο μέρη. Στο πρώτο μέρος παρουσιάζονται τα μαζικά ανοικτά διαδικτυακά μαθήματα (Massive Open Online Courses, MOOCs) και στο δεύτερο παρουσιάζεται το εκπαιδευτικό υλικό που έχει αναπτυχθεί σε μορφή βίντεο.

Το εκπαιδευτικό υλικό χωρίζεται σε 6 ενότητες που αντιστοιχούν σε 6 εβδομάδες μελέτης και κάθε ενότητα διαθέτει από 2 έως 3 μαθήματα, ενώ κάθε μάθημα απαρτίζεται από 1 έως 3 βίντεο. Στην πρώτη ενότητα γίνεται μια εισαγωγή στην Python, την ιστορία και τη φιλοσοφία της, κάποια βασικά στοιχεία που πρέπει να γνωρίζει κανείς για τη συγγραφή κώδικα και εξηγείται πώς γίνεται η εγκατάσταση. Στη δεύτερη ενότητα, ο εκπαιδευόμενος έρχεται σε επαφή με βασικές αρχές προγραμματισμού γενικότερα και με την ακολουθιακή δομή. Στην τρίτη ενότητα αναλύεται η δομή ελέγχου και στην τετάρτη ενότητα παρουσιάζεται η δομή επανάληψης. Στην πέμπτη ενότητα αναλύονται οι δομές δεδομένων της Python. Τέλος, στην έκτη ενότητα εξηγούνται οι συναρτήσεις και οι συμβολοσειρές. Τα παραδείγματα που παρατίθενται είναι μικρά ώστε να επιτρέπουν στον εκπαιδευόμενο να επικεντρώνεται στο ζήτημα που εξηγείται κάθε φορά.

Θέλω να ευχαριστήσω τον κ. Κουτσονίκο Ιωάννη για τη συνεργασία μας και τις συμβουλές του.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία αναπτύχθηκε εκπαιδευτικό υλικό σε μορφή βίντεο με σκοπό την υλοποίηση του μαζικού ανοικτού διαδικτυακού μαθήματος «Βασικές αρχές προγραμματισμού χρησιμοποιώντας τη γλώσσα Python».

Το μάθημα αυτό απευθύνεται σε όποιον δεν είναι απαραίτητο να διαθέτει γνώσεις προγραμματισμού και επιθυμεί να μάθει τη γλώσσα προγραμματισμού Python, χωρίς την παρουσία εκπαιδευτή. Μετά από μια σύντομη εισαγωγική παρουσίαση των MOOCs (τι είναι, ποια είναι τα πλεονεκτήματα και μειονεκτήματά τους), παρουσιάζονται οι έξι ενότητες του εκπαιδευτικού υλικού. Σε αυτές τις ενότητες αναφέρεται η ιστορία και φιλοσοφία της Python, εξηγείται ο τρόπος σύνταξης των εντολών, καθώς και έννοιες απαραίτητες για τη δημιουργία προγραμμάτων. Επίσης, παρουσιάζονται οι δομές δεδομένων, οι αλγοριθμικές δομές και πώς μπορούμε να δημιουργήσουμε μια συνάρτηση.

Η μελέτη κάθε ενότητας εκτιμάται να έχει διάρκεια μίας εβδομάδας. Κάθε ενότητα αποτελείται από 2 έως 3 μαθήματα (υποενότητες) και κάθε μάθημα περιέχει από 1 έως 2 βίντεο. Τα βίντεο είναι διάρκειας από 4 έως 12 λεπτά.

ABSTRACT

In this dissertation, educational material in video format was developed in order to implement the Massive Open Online Course "Basic programming principles using the Python language".

This course is aimed at anyone who does not necessarily have previous programming experience and wishes to learn the Python programming language without the presence of a trainer. After a brief introductory presentation of MOOCs (what they are, what are their strengths and weaknesses), the six sections of the educational material are presented. Python's history and philosophy is explained in these sections, explaining the way the commands are written, and the concepts needed to create programs. Also, data structures and algorithmic structures are presented, as well as how we can create a function.

The study of each module is estimated to last for a week. Each module consists of 2 to 3 lessons (sub-sections) and each lesson contains from 1 to 2 videos. Videos last from 4 to 12 minutes.

1. ΕΙΣΑΓΩΓΗ

1.1. ΤΙ ΕΙΝΑΙ ΤΑ MOOCs

Ο όρος (MOOCs) «Massive Open Online Courses» η αλλιώς «Μαζικά Ανοικτά Διαδικτυακά Μαθήματα» όπως αποδίδεται στην Ελληνική γλώσσα, είναι διαδικτυακά μαθήματα υπό μορφή κλασικού μαθήματος. Έκανε την εμφάνιση του για πρώτη φορά στο χώρο της εκπαίδευσης το 2008 από τους καθηγητές George Siemens και David Cormier του πανεπιστημίου της Manitoba του Καναδά. Αποτελούν μια καινοτομία όσον αφορά την πανεπιστημιακή εκπαίδευση και έχει προσελκύσει το ενδιαφέρον πολλών διεθνών αναγνωρισμένων εκπαιδευτικών ιδρυμάτων. Οι βασικοί λόγοι που τα MOOCs έχουν γίνει τόσο δημοφιλή, είναι επειδή έχουν σχεδιαστεί ώστε να δέχονται απεριόριστο αριθμό εκπαιδευομένων χωρίς προϋποθέσεις. Επίσης, είναι ελεύθερα και ο καθένας μπορεί να εγγραφεί, το μόνο που απαιτείται είναι η σύνδεση στο διαδίκτυο. Το μεγαλύτερο μέρος (αν όχι όλο) των συζητήσεων και των αλληλεπιδράσεων των εκπαιδευομένων με τους εκπαιδευτές ή και των εκπαιδευομένων μεταξύ τους γίνεται διαδικτυακά. Τέλος επειδή έχουν τη μορφή ενός μαθήματος, έχουν ημερομηνίες έναρξης / λήξης και γίνεται αξιολόγηση των εκπαιδευομένων.

Όσον αφορά τη διαδικασία εκπαίδευσης μέσω των MOOCs, αυτή γίνεται με διάφορους τύπους εκπαιδευτικού υλικού. Παρακάτω παρουσιάζονται οι πιο βασικοί τύποι εκπαιδευτικού υλικού που μπορούν να υπάρχουν σε μια πλατφόρμα ενός MOOC.

- **Βίντεο**: Τα εκπαιδευτικά βίντεο (Educational Videos) περιέχουν οπτικοακουστικό υλικό από τον διδάσκοντα που περιγράφει διάφορες ενότητες εκπαιδευτικού περιεχομένου. Διατίθενται σε μορφή που μπορεί να αναπαραχθούν μέσω φυλλομετρητών στο διαδίκτυο.
- **Υπερκείμενο**: Το υπερκείμενο (Hypertext) είναι ένα ηλεκτρονικό κείμενο που συνδέει τον εκπαιδευόμενο σε άλλα κείμενα ή ιστοσελίδες.
- **Παρουσίαση**: Η παρουσίαση (Presentation) είναι οποιασδήποτε μορφής διαφάνειες από τον εκπαιδευτή, οι οποίες περιέχουν σχετικό εκπαιδευτικό υλικό.
- **Δοκίμιο**: Το δοκίμιο (Document) είναι αρχεία σε μορφή PDF τα οποία περιέχουν εκπαιδευτικό υλικό.
- **Κουίζ**: Τα κουίζ είναι σε μορφή ερωτήσεων από τον εκπαιδευτή που συνήθως υπάρχουν μετά τα βίντεο, τις παρουσιάσεις ή το δοκίμιο ενός μαθήματος.
- **Εργασίες**: Αποτελούν οποιαδήποτε εργασία (Project) που δίνεται στον εκπαιδευόμενο.

- **ü** Wiki: Ένα Wiki είναι συνήθως μια ιστοσελίδα που επιτρέπει στους χρήστες της να προσθέσουν, αφαιρέσουν ή να επεξεργαστούν το περιεχόμενό της, γρήγορα και εύκολα.

Το πιστοποιητικό παρακολούθησης αποκτούν οι εκπαιδευόμενοι στο τέλος, όταν θα έχουν διδαχθεί όλο το εκπαιδευτικό υλικό, και δεν ισοδυναμεί με ένα κανονικό πτυχίο. Επίσης, για την απόκτησή του ο εκπαιδευόμενος μπορεί να χρεωθεί ένα χρηματικό ποσό, το οποίο όμως δεν συμπεριλαμβάνει το εκπαιδευτικό υλικό ή τη χρήση της εκπαιδευτικής πλατφόρμας.

1.2. **ΤΥΠΟΙ** MOOCs

Οι διάφορες απόψεις και ιδεολογίες έχουν οδηγήσει τα MOOCs προς τρεις παιδαγωγικές κατευθύνσεις: (Cooper&Sahami, 2013)

i. cMOOC: Είναι βασισμένο στο διαδίκτυο (Σεψάκος, 2013). Τα μαθήματα αυτής της κατηγορίας έχουν ως στόχο τη δημιουργία ενός συμμετοχικού περιβάλλοντος μάθησης. Τα MOOCs αυτά χρησιμοποιούν ως εκπαιδευτικό σύστημα τις αρχές του κονεκτιβισμού.

ii. vMOOC: Είναι βασισμένο στην εργασία. Τα vMOOCs (vocational MOOCs) αναπτύσσονται μέσα από την ολοκλήρωση μιας σειράς εργασιών. Ο εκπαιδευόμενος έχει πλήθος επιλογών ολοκλήρωσης των εργασιών, ενώ καθορίζονται εξ αρχής συγκεκριμένος αριθμός και είδη εργασιών που πρέπει να ολοκληρωθούν.

iii. xMOOC: Είναι βασισμένο στο περιεχόμενο. Την κατηγορία αυτή των MOOCs χρησιμοποιούν οι εκπαιδευτικές πλατφόρμες, Coursera, Edx και Udacity. Σε αυτή την κατηγορία δίνεται μεγαλύτερη σημασία στο εκπαιδευτικό περιεχόμενο και στην κατανόησή του από τους εκπαιδευόμενους και λιγότερο στην ολοκλήρωση των εργασιών. Τα MOOCs αυτού του τύπου βασίζονται στις αρχές του αντικειμενισμού (instructivism).

1.3. **ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΩΝ** MOOCs

Εξ αρχής τα MOOCs καθιερώθηκαν για να προσφέρουν ανοιχτή μάθηση και ελεύθερη πρόσβαση πανεπιστημιακού επιπέδου σε όσο το δυνατόν ευρύτερο πληθυσμό. Τα κύρια γνωρίσματα και, παράλληλα, πλεονεκτήματα των MOOCs είναι τρία: (Τομαράς, 2014)

I. **Η Μαζικότητα** (Massiveness)
   Τα MOOCs σχεδιάστηκαν ώστε να παρέχουν μαζικές υπηρεσίες ανεξαρτήτως του πλήθους των ανθρώπων, ανεξάρτητα με την ώρα και τον τόπο

πρόσβασης του καθενός μεμονωμένου χρήστη. Παράλληλα, η χωρίς χρέωση παροχή μαθημάτων και η online μορφή προσφέρει πρόσβαση και ευελιξία. Αυτά τα στοιχεία προσδίδουν στις ηλεκτρονικές πλατφόρμες μάθησης το προνόμιο να συμβάλλουν στην ισότητα παροχής της τριτοβάθμιας εκπαίδευσης, καθώς όλοι οι μεμονωμένοι χρήστες αντιμετωπίζονται με ίδια κριτήρια.

II. **Η Ανοικτότητα** (Openness)
Τα MOOCs αποτελούν ένα open source σύστημα, καθώς όλες τους οι πληροφορίες είναι ανοικτές προς τον καθένα. Από το υλικό των μαθημάτων, έως και τις σωστές απαντήσεις στις εξετάσεις είναι ελεύθερα και προσβάσιμα από κάθε χρήστη. Τα MOOCs έχουν ως στόχο να διδάξουν και όχι να εξετάσουν τι διδάχθηκαν οι συμμετέχοντες. Το ζήτημα αυτό το αφήνουν στην κρίση των συμμετεχόντων.

III. **Η Φιλοσοφία του Κονεκτιβισμού** (Connectivism philosophy)
Τα MOOCs έχουν ως θεμέλιους λίθους τις αρχές της αυτονομίας, της ποικιλομορφίας, της ανοικτότητας (όπως αναφέραμε προηγουμένως) και της αλληλεπίδρασης. Ο ρόλος του διδάσκοντα είναι να διευκολύνει την προσπάθεια των εκπαιδευομένων να συμμετέχουν και να αποκτήσουν γνώσεις.

Ωστόσο, η εξέλιξη των MOOCs προκάλεσε και αρνητικές κριτικές με την αμφισβήτηση σχετικά με την προσφορά ουσιαστικής εκπαίδευσης (Τομαράς, 2014) που αναλύονται στη συνέχεια.

1. **Πνευματικά Δικαιώματα** (Copyright)
Αν και το βασικό πλεονέκτημα των MOOCs είναι το υποστηρικτικό υλικό που παρέχεται δωρεάν, πρέπει να αναρωτηθούμε πώς διασφαλίζονται τα δικαιώματα πνευματικής ιδιοκτησίας. Ως εκ τούτου πρέπει να βρεθούν μηχανισμοί ώστε το υλικό που διατίθεται να είναι νόμιμο και να εξασφαλίζει τα πνευματικά δικαιώματα των συγγραφέων.
2. **Πιστοποίηση** (Certification)
Το απλό πιστοποιητικό που παρέχεται είναι αρκετό για τους συμμετέχοντες; Αναγνωρίζεται; Είναι επίσης πολύ σημαντική η σωστή επιλογή του μαθήματος και του φορέα που το παρέχει, μια που ολοένα και περισσότεροι οργανισμοί (αλλά και επιχειρήσεις) συμμετέχουν στη διαδικασία, η εγκυρότητα των οποίων δεν είναι πάντα διασφαλισμένη.
3. **Καθοδήγηση** (Guidance)
Μία από τις αρχές στις οποίες βασίζονται τα MOOCs είναι η αλληλεπίδραση μεταξύ των εκπαιδευομένων, η οποία καλείται να υποκαταστήσει την έλλειψη καθοδήγησης από εκπαιδευτές. Μέσα από αυτή τη δράση τονίζεται η έννοια της κοινωνικότητας των ατόμων και σε αυτή στηρίζεται το σύστημα

καθοδήγησης των εκπαιδευομένων στις ηλεκτρονικές πλατφόρμες μάθησης από απόσταση.

4. **Εκπαιδευτική διαδικασία** (Educational process)
   Η εκπαιδευτική διαδικασία στις ηλεκτρονικές πλατφόρμες μαθημάτων από απόσταση εξελίσσεται ανεξέλεγκτα από τον διδάσκοντα, γεγονός που υποβαθμίζει τον ρόλο του εκπαιδευτικού. Ο εκπαιδευτικός πλέον ορίζεται ως ο συντονιστής του συστήματος και όχι ως ο πανταχού παρών σε όλη την εκπαιδευτική διαδικασία.

1.4. **ΔΗΜΟΦΙΛΕΣΤΕΡΕΣ ΠΛΑΤΦΟΡΜΕΣ** MOOCs

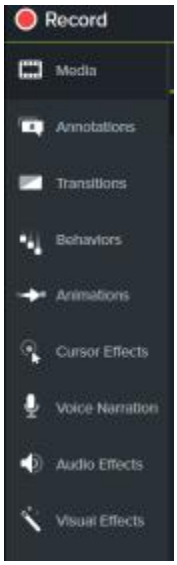
Οι πιο δημοφιλές υπηρεσίες παροχής MOOCs είναι οι εξής:

- Η edx (http://www.edx.org/) είναι μια μη κερδοσκοπική πλατφόρμα παροχής MOOCs που δημιουργήθηκε από τα πανεπιστήμια MIT και Harvard, τα οποία τη χρηματοδότησαν με 60 εκατομμύρια δολάρια. Το 2013 προσφέρθηκαν μέσω της πλατφόρμας 92 μαθήματα. Στη συγκεκριμένη πλατφόρμα παρέχονται 3 είδη διαφορετικών πιστοποιητικών (Honor Code, ID Verified, XSeries Certificates).
- Η Αμερικανική εταιρία Coursera (http://www.coursera.org/) είναι μια αφιλοκερδούς χαρακτήρα εκπαιδευτική εταιρία, η οποία ξεκίνησε με χρηματοδότηση 22 εκατομμυρίων δολαρίων από διάφορους επενδυτές επιχειρηματικών κεφαλαίων. Στη συγκεκριμένη πλατφόρμα υπάρχουν 4 πανεπιστημιακοί εταίροι, τα πανεπιστήμια Stanford, Princeton, Michigan, Pennsylvania.

  Το έτος 2013 η Coursera παρείχε 540 μαθήματα σε 18 διαφορετικά γνωστικά αντικείμενα από 107 συνεργάτες, ενώ είχε 5,5 εκατομμύρια εγγεγραμμένους χρήστες. Ορισμένα από τα συνεργαζόμενα πανεπιστήμια προσφέρουν σε όσους εκπαιδευόμενους είναι διατεθειμένοι να καταβάλουν μια συνδρομή, επιπλέον εκπαιδευτικό υλικό με εργασίες και δραστηριότητες, καθώς και τη δυνατότητα απόκτησης επίσημου πιστοποιητικού.
- Η Udacity (http://www.udacity.com/) είναι μια κερδοσκοπική οργάνωση που ιδρύθηκε από τους Sebastian Thrun, David Stavens, Mike Sokolsky και έχει χρηματοδοτηθεί από ιδιωτικές εταιρίες επενδύσεων. Το έτος 2013 η Udacity προσέφερε δωρεάν 28 μαθήματα από 5 διαφορετικούς γνωστικούς τομείς. Στην περίπτωση που κάποιος εκπαιδευόμενος επιθυμεί την απόκτηση πιστοποίησης μετά το πέρας των μαθημάτων και εφόσον καταβάλει ένα χρηματικό ποσό, θα λάβει πιστοποιητικό ολοκλήρωσης το οποίο θα παρουσιάζει τις επιδόσεις του, υπογεγραμμένο από τους εκπαιδευτές.

1.5. ΠΕΡΙΓΡΑΦΗ ΔΗΜΙΟΥΡΓΙΑΣ ΒΙΝΤΕΟ

Τα βίντεο περιέχουν εκπαιδευτικό υλικό για τη γλώσσα προγραμματισμού Python. Δημιουργήθηκαν με σκοπό την υλοποίηση του διαδικτυακού μαθήματος με τίτλο «Βασικές αρχές προγραμματισμού χρησιμοποιώντας τη γλώσσα Python». Τα βίντεο αποτελούνται από εικόνα και ήχο. Για την εικόνα έχουν δημιουργηθεί διαφάνειες σε PowerPoint ενώ για τον ήχο χρησιμοποιήθηκε κείμενο που εκφωνείται. Έχουν δημιουργηθεί εικοσιένα βίντεο και ο χρόνος που διαρκεί το καθένα κυμαίνεται από 4 έως 12 λεπτά. Επίσης, κάθε βίντεο αντιστοιχεί σε κάποιο μάθημα και κάθε μάθημα αντιστοιχεί σε κάποια εβδομάδα. Μία εβδομάδα περιέχει ένα, δύο ή τρία μαθήματα, ενώ ένα μάθημα περιέχει ένα ή δύο βίντεο. Ακόμα, κάθε βίντεο έχει από έξι έως δέκα διαφάνειες που με τη σειρά τους περιέχουν θεωρία και παραδείγματα. Τα παραδείγματα είναι σε μορφή εικόνας ή βίντεο. Το κείμενο που εκφωνείται περιγράφει και αναλύει τις διαφάνειες.



Για τη δημιουργία των βίντεο χρησιμοποιήθηκε το πρόγραμμα Camtasia Studio 9. Το Camtasia Studio είναι ένα software το οποίο ηχογραφεί και καταγράφει κάθε δραστηριότητα η οποία λαμβάνει χώρα στην οθόνη του υπολογιστή μας, σε τύπο αρχείου βίντεο. Το Camtasia Studio μας προσφέρει μεγάλη ποικιλία εφέ, τα οποία εφαρμόζονται στα βίντεο που δημιουργηθήκαν με κατάλληλη επεξεργασία. Τα εφέ αφορούν την ποιότητα ήχου κατά τη διάρκεια ηχογράφησης και την κατάσταση προβολής του δείκτη του ποντικιού. Υπάρχουν οπτικά εφέ, εφέ συμπεριφοράς κ.ά. Επίσης, μας προσφέρει τη δυνατότητα αποθήκευσης των βίντεο σε πολλές επεκτάσεις όπως MP4, WMV, AVI, GIF, M4A. Μπορεί να χρησιμοποιηθεί για να ενισχύσει PowerPoint και πολυμεσικές εφαρμογές, μέσω βίντεο στο οποίο προστίθεται η αφήγηση για επιπλέον επεξηγήσεις.

2. **ΕΙΣΑΓΩΓΗ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ** PYTHON

2.1. **ΜΑΘΗΜΑ** 1. **ΕΙΣΑΓΩΓΗ ΣΤΗΝ** PYTHON

2.1.1. **ΙΣΤΟΡΙΑ ΚΑΙ ΦΙΛΟΣΟΦΙΑ ΤΗΣ** PYTHON

Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού, η οποία δημιουργήθηκε από τον Ολλανδό προγραμματιστή Γκβίντο βαν Ρόσσουμ (Guido van Rossum) το 1990. Το όνομα της γλώσσας προέρχεται από το Monty Pythons' Flying Circus, μια Αγγλική σατυρική σειρά του '70.


Guido Van Rossum 1956-

Τα πλεονεκτήματά της είναι η αναγνωσιμότητα του κώδικα και η ευκολία χρήσης της. Για το λόγο αυτό είναι ιδανική για αρχάριους, αλλά χρησιμοποιείται και από έμπειρους προγραμματιστές για ανάπτυξη ολοκληρωμένων εφαρμογών σε μεγάλους οργανισμούς, όπως η NASA, Google, Disney και CIA. Ο τρόπος σύνταξής της επιτρέπει στους προγραμματιστές που τη χρησιμοποιούν να εκφράσουν δύσκολες έννοιες σε μικρότερο κώδικα απ' ότι θα γινόταν σε άλλες γλώσσες προγραμματισμού όπως η C++ ή η Java. Ένα πρόγραμμα που γράφεται σε μεταγλωττιζόμενη γλώσσα όπως η C ή η C++ μετατρέπεται σε μια γλώσσα που μιλάει ο υπολογιστής μας (δυαδικός κώδικας) χρησιμοποιώντας έναν μεταγλωττιστή. Η Python, από την άλλη μεριά, δεν χρειάζεται μεταγλώττιση σε δυαδικό αρχείο. Απλά τρέχουμε το πρόγραμμα απευθείας από τον πηγαίο κώδικα. Εσωτερικά, η Python μετατρέπει τον πηγαίο κώδικα σε μια ενδιάμεση μορφή που ονομάζεται bytecode και μετά το μεταφράζει στη γλώσσα του υπολογιστή και το εκτελεί. Όλη αυτή η διαδικασία στην πραγματικότητα κάνει τη χρήση της Python πιο εύκολη, αφού δεν χρειάζεται να ανησυχούμε για τη μεταγλώττιση του προγράμματος, τη σύνδεση με τις κατάλληλες βιβλιοθήκες κλπ. Αυτό επίσης κάνει τα προγράμματα της Python εύκολα φορητά, αφού μπορούμε να αντιγράψουμε σε έναν άλλο υπολογιστή το πρόγραμμα της Python που δημιουργήσαμε και να δουλέψει έτσι απλά.

Η φιλοσοφία της Python επικεντρώνεται κυρίως στην απλότητα και αναγνωσιμότητα του κώδικα. Αναπτύσσεται ως λογισμικό ανοικτού κώδικα με βάση το μοντέλο της κοινότητας προγραμματιστών που εργάζονται για την ανάπτυξή της και ο συντονισμός και η διαχείριση γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation (PSF). Ο κώδικας διανέμεται με την άδεια Python Software Foundation License.

Η Python είναι μια εύκολη στην εκμάθηση γλώσσα προγραμματισμού. Η γλώσσα είναι ιδιαίτερα δημοφιλής για πολλούς λόγους:

- Μπορεί να χρησιμοποιηθεί όπως και μια αριθμομηχανή. Είναι ένας εύκολος τρόπος να εξοικειωθεί κάποιος με τον προγραμματισμό.
- Αν και γλώσσα υψηλού επιπέδου, η σύνταξή της είναι πολύ απλή.
- Συνδυάζεται εύκολα με άλλες γλώσσες προγραμματισμού όπως Java, C, C++.
- Διατίθεται δωρεάν και υποστηρίζεται από όλα τα Λειτουργικά Συστήματα.

Η Python 1.0 κυκλοφόρησε το 1994 και στις 16 Οκτωβρίου 2000 κυκλοφόρησε η Python 2.0. Στις 3 Δεκεμβρίου 2008 κυκλοφόρησε η έκδοση 3.0, γνωστή ως Python 3000. Πολλά από τα νέα χαρακτηριστικά αυτής της έκδοσης έχουν μεταφερθεί στις εκδόσεις 2.6 και 2.7, οι οποίες είναι προς τα πίσω συμβατές. Η συμβατότητα προς τα πίσω είναι μια ιδιότητα που επιτρέπει συστήματα και τεχνολογίες να λειτουργούν και να συνδέονται με παλαιότερες εκδόσεις τους. Η Python 3 είναι η πρώτη γλώσσα προγραμματισμού που σπάει τη συμβατότητα προς τα πίσω με όλες τις προηγούμενες εκδόσεις της ώστε να επιτρέπεται η διόρθωση σε λάθη που υπήρχαν σε προηγούμενες εκδόσεις και να γίνει ακόμα πιο εύκολη η χρήση της. Στη συνέχεια των μαθημάτων θα αναφέρουμε αρκετές διαφορές ανάμεσα στις εκδόσεις 2 και 3 όπως και παραδείγματα.

2.1.2. ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ

Από οποιονδήποτε browser πληκτρολογούμε https://www.python.org/. Επιλέγουμε το σύνδεσμο Downloads και κατεβάζουμε μια έκδοση της Python. Στη συγκεκριμένη περίπτωση έχουν επιλεγεί και οι δύο εκδόσεις.

Τα περισσότερα παραδείγματα που θα αναφέρουμε κατά τη διάρκεια των μαθημάτων θα είναι στην έκδοση 2.7.14, θα πούμε επίσης και τις διαφορές ανάμεσα στις εκδόσεις 2 και 3. Όταν κατέβει το αρχείο το ανοίγουμε και το εκτελούμε ακολουθώντας τις οδηγίες ώστε να γίνει η εγκατάσταση.

Η Python προσφέρει ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών. Το περιβάλλον αυτό ονομάζεται IDLE (Integrated Development and Learning Environment). Συνήθως ένα IDLE περιλαμβάνει κάποιον επεξεργαστή πηγαίου κώδικα, έναν μεταγλωττιστή κ.ά. Το IDLE είναι ένα απλό περιβάλλον, ειδικά για αρχάριους προγραμματιστές, στο οποίο μπορούμε να γράψουμε και να εκτελέσουμε τα προγράμματά μας.

Για να ανοίξουμε την κονσόλα του IDLE που ονομάζετε Shell, πληκτρολογούμε στη γραμμή αναζήτησης των Windows «IDLE» και μπορούμε να την κλείσουμε με την εντολή exit(). Επίσης μπορούμε να την ανοίξουμε από το μενού, όπου θα γράφει "Python 2.7", επιλέγοντας το IDLE και να την κλείσουμε πατώντας το παραδοσιακό x πάνω δεξιά.

Η κονσόλα του IDLE χρησιμοποιείται για άμεση εκτέλεση εντολών, δηλαδή κάθε εντολή που γράφεται εκτελείται αμέσως πατώντας enter. Για παράδειγμα, γράφοντας print('Hello World') εκτυπώνεται Hello World, αν πληκτρολογήσουμε και άλλες εντολές θα παρατηρήσουμε ότι χρωματίζονται κατάλληλα (χρωματική συντακτική επισήμανση). Η επισήμανση αυτή βοηθά στη σωστή σύνταξη των εντολών, αφού κάθε μία έχει διαφορετικό χρώμα και είναι εύκολο να ανιχνευθούν σφάλματα, όπως για παράδειγμα λανθασμένη σύνταξη των εντολών ή λάθη λόγω βιασύνης στην πληκτρολόγηση των εντολών. Τα λάθη χρωματίζονται με κόκκινο χρώμα.

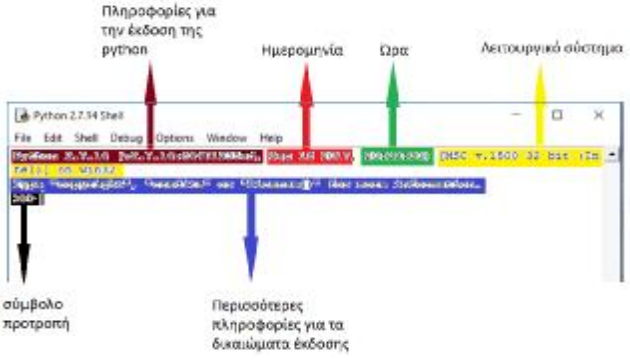
Ακόμη επισημαίνεται ότι το παράθυρο αυτό υποστηρίζει και επανάκληση των προηγούμενων εντολών που έχουμε δώσει. Αυτό γίνεται, αν κάνουμε κλικ με το ποντίκι σε προηγούμενη εντολή που έχουμε γράψει και πατήσουμε Enter.

Για λόγους παρουσίασης θα μάθουμε πώς να αλλάξουμε το φόντο της κονσόλας του IDLE. Στο Python Shell **à**, επιλέγουμε Options **à** Configure IDLE **à** Highlighting **à**, αλλάζουμε το IDLE Classic σε IDLE Dark **à** Apply **à** Ok.

2.2. **ΜΑΘΗΜΑ** 2. **ΣΥΝΤΑΞΗ ΚΑΙ ΧΡΗΣΗ ΤΗΣ** PYTHON

2.2.1. **ΣΥΝΤΑΞΗ ΣΤΟ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ**

Όπως είδαμε και στο προηγούμενο μάθημα, καθώς εκτελούμε το πρόγραμμα της Python ανοίγει η κονσόλα (shell) του IDLE. Εκεί θα παρατηρήσετε ότι στις πρώτες γραμμές υπάρχουν μερικές πληροφορίες



για την έκδοση της Python, την ώρα, την ημερομηνία και σε ποιο Λειτουργικό Σύστημα τρέχει. Ακριβώς από κάτω δείχνει τι πρέπει να πατήσουμε για περισσότερες πληροφορίες, ενώ στο τέλος έχει το σύμβολο προτροπής που βοηθά στη σύνταξη ώστε να είναι πιο εύκολα αναγνώσιμος ο κώδικας.

Εκτός της κονσόλας του IDLE, η Python παρέχει και έναν επεξεργαστή κώδικα (editor). Μπορούμε να τον ανοίξουμε από την κονσόλα (Shell) με File **à** New File ή με Ctrl+N. Θα δούμε ότι ανοίγει ένα αρχείο με όνομα "Untitled", οπότε μπορούμε να συντάξουμε το πρόγραμμα. Για να τρέξει το πρόγραμμα πρέπει να γραφεί ολόκληρος ο κώδικας και να αποθηκευτεί. Όταν το τρέξουμε θα παρατηρήσουμε ότι το αποτέλεσμα εμφανίζεται στην κονσόλα του IDLE.

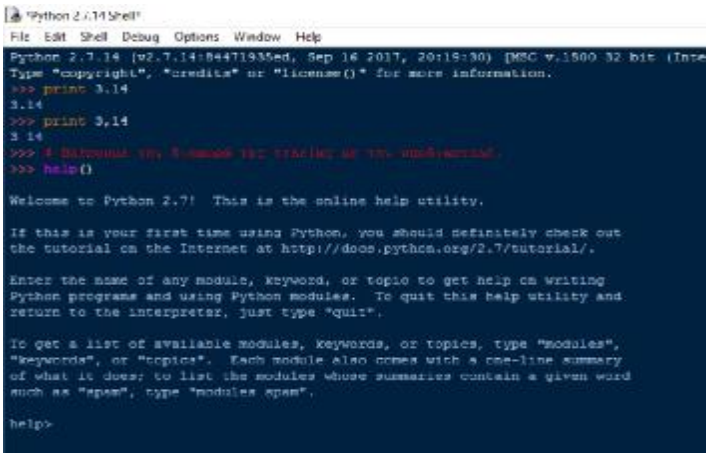
Για παράδειγμα, ανοίγουμε τον επεξεργαστή κώδικα, όπως αναφέραμε, και γράφουμε το πρώτο μας πρόγραμμα, που περιλαμβάνει την εντολή print 'Hello World!' και από κάτω την εντολή print 'το πρώτο μας πρόγραμμα'. Αποθηκεύουμε πατώντας File **à** Save **à**, δίνουμε ένα όνομα και επιλέγουμε το φάκελο που θέλουμε. Πατάμε OK και συνεχίζουμε στην εκτέλεση του προγράμματος, επιλέγουμε run module και βλέπουμε ότι τα δύο μηνύματα εμφανίζονται στην κονσόλα του IDLE. Αυτή είναι και η διαδικασία αποθήκευσης στην Python.

Στη συνέχεια, θα δούμε τις διαφορές μεταξύ **κονσόλας** (Shell) και **Επεξεργαστή κώδικα** (Editor) του IDLE. Στην κονσόλα, όπως είδαμε, υπάρχουν μερικές πληροφορίες στις πρώτες γραμμές και το σύμβολο προτροπής, ενώ στον επεξεργαστή κώδικα όχι. Επίσης, στην κονσόλα εκτελούνται μία - μία οι εντολές, ενώ στον επεξεργαστή κώδικα πρέπει να γραφεί ολόκληρος ο κώδικας και τα αποτελέσματά του θα εμφανιστούν στην κονσόλα. Μια άλλη διαφορά είναι ότι για να κάνουμε αποθήκευση στην Python πρέπει να χρησιμοποιήσουμε τον επεξεργαστή κώδικα. Τέλος, τα περιεχόμενα της γραμμής εργαλείων δεν είναι ίδια.

Ας μιλήσουμε λίγο για τη σύνταξη. Μπορούμε να έχουμε πολλά παράθυρα ανοιχτά στην κονσόλα ή στον επεξεργαστή κώδικα. Όπως είδαμε, όταν γράφουμε τις εντολές είτε στην κονσόλα είτε στον επεξεργαστή κώδικα, αυτές χρωματίζονται κατάλληλα.

- Οι εντολές γράφονται πάντα με μικρούς λατινικούς χαρακτήρες.
- Η υποδιαστολή συμβολίζεται με τελεία.
- Η εισαγωγή σχολίων γίνεται με τη χρήση του συμβόλου της δίεσης (#). Το σχόλιο μπορεί να μπει οπουδήποτε στον κώδικα, αλλά μέσα στην εντολή print θα εμφανιστεί ακριβώς όπως γράφηκε.
- Αν στην κονσόλα πληκτρολογήσετε την εντολή help() θα εμφανιστεί ένα κείμενο που θα παρέχει βοήθεια.

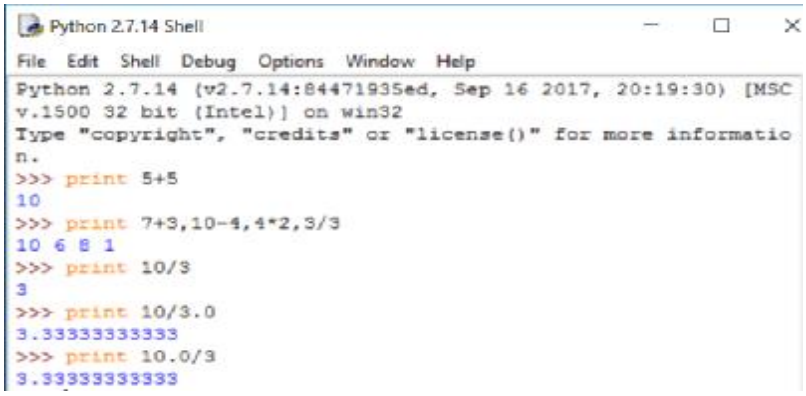
Στην εικόνα βλέπουμε τη διαφορά της τελείας με την υποδιαστολή. Επίσης, βλέπουμε ότι τα σχόλια χρωματίζονται κόκκινα. Τέλος, φαίνονται τα αποτελέσματα της εντολής help().



2.2.2. **ΧΡΗΣΗ ΤΗΣ** PYTHON **ΩΣ ΑΡΙΘΜΟΜΗΧΑΝΗ**

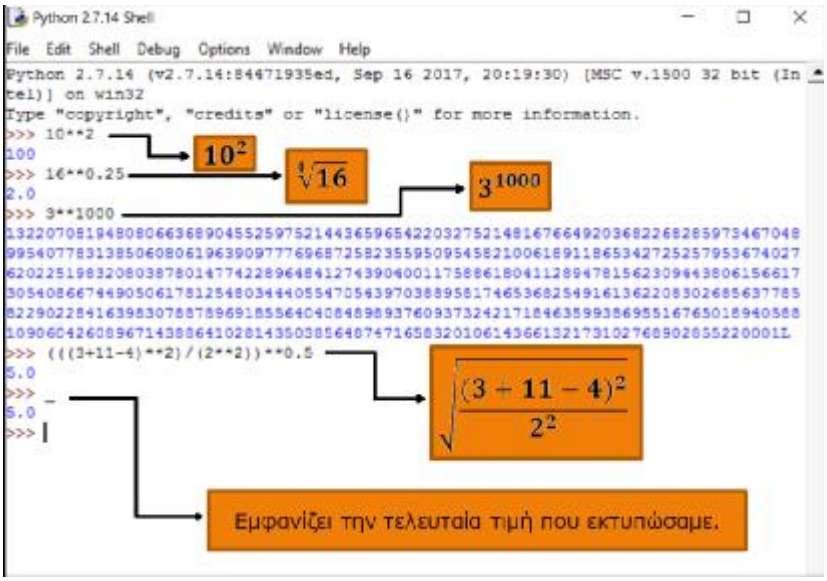
Η Python μπορεί να χρησιμοποιηθεί ως υπολογιστική μηχανή, όπου οι βασικές πράξεις γίνονται με τα σύμβολα +, -, *, /. Έτσι, μπορούμε να εκτελέσουμε απλές πράξεις αλλά και πιο σύνθετες με τη χρήση παρενθέσεων. Κάθε φορά που θέλουμε το αποτέλεσμα μιας πράξης αρκεί να γράψουμε την εντολή print και την πράξη που θέλουμε. Αυτό ισχύει και για την κονσόλα αλλά και για τον επεξεργαστή. Μπορούμε επίσης να τυπώσουμε πολλά αποτελέσματα μαζί, αρκεί αυτά να χωρίζονται με κόμμα. Αν και δεν έχουμε ορίσει ακόμα τους τύπους των δεδομένων, αξίζει να σημειώσουμε ότι όταν κάνουμε μια πράξη μεταξύ ακέραιων αριθμών που το αποτέλεσμά της μπορεί να είναι πραγματικός αριθμός, τότε αυτό που θα εμφανιστεί θα είναι το ακέραιο μέρος του. Έτσι, στη διαίρεση, αν ο διαιρέτης και ο διαιρετέος είναι ακέραιοι αριθμοί τότε το πηλίκο θα εμφανιστεί ως ακέραιος, ακόμα και αν είναι πραγματικός. Αυτό ισχύει για την έκδοση 2.7, η έκδοση 3.6 μας εμφανίζει το πηλίκο ως πραγματικό αριθμό.

Για παράδειγμα, αν γράψουμε την εντολή print (5+5) θα εμφανιστεί το αποτέλεσμα 10. Επίσης, βλέπουμε πώς γίνεται και η εμφάνιση πολλών αποτελεσμάτων μαζί. Με την εντολή print(7+3, 10-4, 4*2, 3/3) θα εμφανιστεί το αποτέλεσμα όλων των πράξεων: 10, 6, 8, 1. Βλέπουμε ότι στη διαίρεση μεταξύ ακεραίων αριθμών, το αποτέλεσμα θα είναι επίσης ακέραιος, ενώ αν ένας τουλάχιστον από τους διαιρέτη και διαιρετέο είναι πραγματικός αριθμός, το αποτέλεσμα θα είναι πραγματικός.



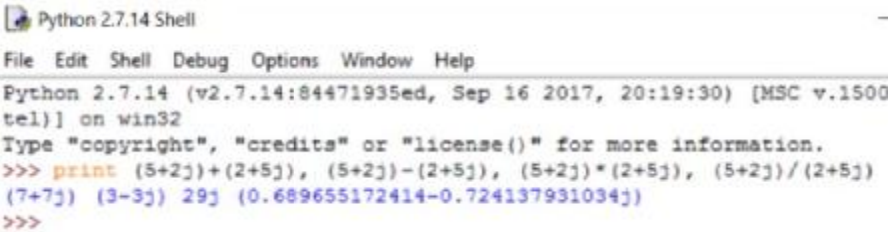
Στην Python μπορούμε να κάνουμε τις πράξεις κατευθείαν στην κονσόλα (Shell) και να μας εμφανίσει τα αποτελέσματα χωρίς να χρειαστεί η εντολή print. Για παράδειγμα, γράφουμε 3*2+6-4/2 και πατώντας enter θα εμφανίσει το αποτέλεσμα 10. Αν θέλουμε να δώσουμε προτεραιότητα σε κάποια πράξη, απλά τη βάζουμε σε παρένθεση, π.χ. γράφουμε 3*(2+6-4)/2 οπότε θα εμφανίσει το αποτέλεσμα 6.

Αν θέλουμε να υψώσουμε έναν αριθμό σε δύναμη χρησιμοποιούμε τα δύο αστεράκια. Ο υπολογισμός των ριζών γίνεται με τη μαθηματική ιδιότητα $\sqrt[n]{x} = x^{\frac{1}{n}}$. Να αναφέρουμε επίσης ότι η Python έχει απεριόριστο μέγεθος για ακέραιους αριθμούς και ότι μας παρέχει τη δυνατότητα αποθήκευσης της τελευταίας τιμής που εκτυπώσαμε με το χαρακτήρα της κάτω παύλας.



Βλέπουμε στην εικόνα πώς γίνεται ο υπολογισμός ύψωσης σε δύναμη και εύρεσης της ρίζας ενός αριθμού. Δηλαδή, 10 στο τετράγωνο μας κάνει 100 ενώ η τέταρτη ρίζα του 16 είναι 2. Ακόμα, βλέπουμε πόσο μεγάλους αριθμούς χρησιμοποιούμε στην Python, εμφανίζοντας το 3 υψωμένο στη χιλιοστή. Στο άλλο παράδειγμα, βλέπουμε πώς μπορούμε να συντάξουμε μια πολύπλοκη αριθμητική έκφραση και, τέλος, ποια είναι η χρησιμότητα της κάτω παύλας.

Αυτό που αξίζει να αναφέρουμε είναι ότι η Python χειρίζεται εξίσου καλά και μιγαδικούς αριθμούς εκτός από ακέραιους και πραγματικούς. Οι μιγαδικοί γράφονται με τη μορφή a+bj. Για παράδειγμα, δίνοντας την εντολή print (5+2j) + (2+5j), (5+2j) - (2+5j), (5+2j) * (2+5j), (5+2j) / (2+5j) εμφανίζονται τα αντίστοιχα αποτέλεσμα των πράξεων.



3. ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ ΕΞΟΔΟΥ

3.1. ΜΑΘΗΜΑ 3. ΔΕΔΟΜΕΝΑ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ

3.1.1. ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Στο μάθημα αυτό θα μιλήσουμε για τους ΤΥΠΟΥΣ ΔΕΔΟΜΕΝΩΝ της Python. Είναι πολύ σημαντικό για την εκμάθηση οποιασδήποτε γλώσσας προγραμματισμού να καταλάβουμε τους βασικούς τύπους δεδομένων, ώστε να μπορούμε να γράψουμε σωστά τον κώδικα. Οι τύποι δεδομένων στην Python χωρίζονται σε:

Ακέραιους αριθμούς που συμβολίζονται με «int»

Πραγματικούς αριθμούς που συμβολίζονται με «float»

Ακολουθίες χαρακτήρων που συμβολίζονται με «str»

Λογικούς που συμβολίζονται με «bool» και έχουν μόνο δύο εκδοχές, αληθή (True) και ψευδή (False).

Μιγαδικούς αριθμούς που συμβολίζονται με «complex».

Παράδειγμα:

2 **à** ακέραιος αριθμός «int»

3.14 **à** πραγματικός αριθμός «float»

'Hello World!' **à** συμβολοσειρά «str»

True **à** λογικού τύπου και είναι αληθείς «bool»

5+2j **à** μιγαδικός αριθμός «complex»

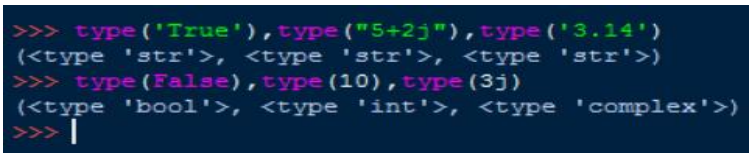
Οι τιμές αυτές ανήκουν σε διαφορετικό τύπο δεδομένων. Ο αριθμός 2 είναι ακέραιος, το 3.14 είναι πραγματικός και το 'Hello World!' μια συμβολοσειρά (ακολουθία από χαρακτήρες), ενώ το True συμβολίζει ότι η μεταβλητή είναι λογικού τύπου και είναι αληθής.

Οι συμβολοσειρές μπαίνουν πάντα σε εισαγωγικά, διπλά ή απλά. Αν οποιοσδήποτε αριθμός είναι μέσα σε εισαγωγικά είτε αυτός είναι ακέραιος είτε είναι μιγαδικός τότε εκλαμβάνεται ως συμβολοσειρά. Στη συνέχεια θα αφιερώσουμε ειδικό μάθημα για να αναλύσουμε τις συμβολοσειρές. Στις τιμές λογικού τύπου, που είναι True και False, το πρώτο τους γράμμα γράφεται πάντα με κεφαλαίο για να τις αναγνωρίσει η Python. Ενώ στους πραγματικούς αριθμούς, η υποδιαστολή πρέπει να είναι με τελεία (.) και όχι με κόμμα (,). Στην Python δεν είναι αναγκαίο να δηλώνουμε με ακρίβεια τους τύπους δεδομένων που χρησιμοποιούμε. Ο έλεγχος

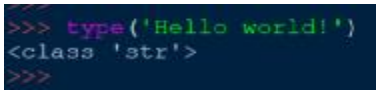
γίνεται καθώς εκτελείται το πρόγραμμα. Η ιδιότητα αυτή ονομάζεται δυναμική απόδοση τύπων (dynamic typing).

Αν δεν είμαστε σίγουροι σε ποια κατηγορία ανήκει η τιμή που θέλουμε να εισάγουμε, τότε μπορούμε να ζητήσουμε βοήθεια. Πληκτρολογούμε τη συνάρτηση "type" και μέσα σε παρένθεση γράφουμε την τιμή για τον τύπο της οποίας έχουμε αμφιβολία. Ως αποτέλεσμα από κάτω ακριβώς θα εμφανιστεί το μήνυμα <type και ο τύπος που ανήκει η τιμή>. Βλέπετε και στο παράδειγμα ότι αν γράψουμε 2 θα εμφανιστεί int αν γράψουμε 3.14 θα εμφανιστεί float κλπ.

Επίσης μπορούμε να χρησιμοποιήσουμε πολλές φορές τη συνάρτηση type() στην ίδια γραμμή αρκεί να τις χωρίζουμε με κόμμα, όπως στο παράδειγμα, οπότε τα αποτελέσματα θα εμφανιστούν σε μια γραμμή.



Μια μικρή λεπτομέρεια, στην έκδοση 3.6.4 η συνάρτηση type()μας εμφανίζει class και τον τύπο της τιμής, όπως βλέπετε και στο παράδειγμα.

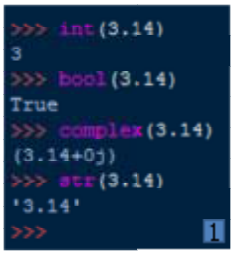


Έκδοση 3.6.4

Αναφερθήκαμε προηγουμένως στον όρο συνάρτηση. Τι είναι όμως μια συνάρτηση; Συνάρτηση είναι μια αυτόνομη μονάδα κώδικα που σχεδιάστηκε για να εκτελεί μια συγκεκριμένη εργασία. Δηλαδή είναι ένα υποπρόγραμμα που μπορούμε να το επαναχρησιμοποιήσουμε και οι εντολές του έχουν μια συγκεκριμένη λειτουργία. Τα πλεονεκτήματα των συναρτήσεων είναι ότι με τη βοήθειά τους ο κώδικας καταλαμβάνει λιγότερο χώρο και ότι μπορούμε να τις επαναχρησιμοποιήσουμε όσες φορές θέλουμε χωρίς να γράφουμε εξ αρχής ολόκληρο τον κώδικα. Στην Python υπάρχουν δύο είδη συναρτήσεων, αυτές που έχει η Python, τις ενσωματωμένες, και αυτές που μπορούμε να δημιουργήσουμε εμείς ως προγραμματιστές. Μέχρι τώρα έχουμε δει δύο ενσωματωμένες συναρτήσεις την exit() και την type(). Οι συναρτήσεις στην Python έχουν μωβ χρώμα.

Οι παρακάτω συναρτήσεις κάνουν μετατροπή στον αντίστοιχο τύπο δεδομένων. Η int() σε ακέραιο, η float() σε πραγματικό, η str() σε συμβολοσειρά κλπ. Στην παρένθεση μπορούμε να βάλουμε οποιοδήποτε τύπο ή μεταβλητή ή ακόμα και άλλη συνάρτηση.

Όπως βλέπετε στις εικόνες, η εικόνα 1 δείχνει πώς μετατρέπεται ο αριθμός 3.14 στους άλλους τύπους, η εικόνα 2 δείχνει πώς μετατρέπεται το true και το false σε ακέραιο ή πραγματικό, ενώ οι μιγαδικοί δεν μπορούν να μετατραπούν σε ακέραιο αριθμό.



Επίσης, η εικόνα 3 δείχνει ότι η συμβολοσειρά δεν μπορεί να μετατραπεί σε ακέραιο. Η εικόνα 4 μας δείχνει ότι μπορούμε να μετατρέψουμε τη συμβολοσειρά '2.5' σε πραγματικό αριθμό, όχι όμως σε ακέραιο. Το μετατρέπουμε σε ακέραιο βάζοντας την float() ως παράμετρο της int(). Μπορείτε να κάνετε μερικές δοκιμές για να δείτε σε ποιες τιμές γίνεται η μετατροπή και σε ποιες όχι.

```
>>> int(True)
1
>>> float(False)
0.0
>>> int(3+5j)

Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    int(3+5j)
TypeError: can't convert complex to int
>>>
```
2

```
>>> int('a')

Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    int('a')
ValueError: invalid literal for int() with base 10: 'a'
>>>
```
3

```
>>> float('2.5')
2.5
>>> int('2.5')

Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    int('2.5')
ValueError: invalid literal for int() with base 10: '2.5'
>>> int(float('2.5'))
2
>>>
```
4

3.1.2. **ΜΕΤΑΒΛΗΤΕΣ**

Μεταβλητή λέγεται ένα όνομα που αναφέρεται σε μία τιμή. Οι τιμές των μεταβλητών είναι δυνατό να αλλάξουν κατά τη διάρκεια εκτέλεσης ενός προγράμματος, εξ ου και η ονομασία “μεταβλητή”. Ενώ η τιμή της μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος, το όνομα της μεταβλητής παραμένει ίδιο. Οι μεταβλητές μπορούν να ανήκουν σε οποιονδήποτε από τους τύπους δεδομένων. Οι μεταβλητές είναι τμήματα της μνήμης του υπολογιστή μας, όπου μπορούμε να αποθηκεύσουμε πληροφορία. Για να δημιουργήσουμε μια μεταβλητή χρησιμοποιούμε την εντολή εκχώρησης η οποία της δίνει μια τιμή. Ο τύπος που ορίζει μια μεταβλητή είναι ίδιος με τον τύπο της τιμής που της εκχωρούμε στο πρόγραμμά μας.

Μεταβλητή = τιμή

Μια εντολή εκχώρησης αποτελείται από το αριστερό μέρος, όπου είναι το όνομα της μεταβλητής, το ίσον (=) και το δεξί μέρος, όπου είναι μια τιμή ή μια άλλη μεταβλητή ή μια έκφραση. Σε περίπτωση που στο δεξί μέρος υπάρχει μια άλλη μεταβλητή, εκχωρείται στην πρώτη μεταβλητή η τιμή της δεύτερης μεταβλητής. Θα ήταν συνετό να επιλέγουμε κατάλληλα ονόματα μεταβλητών που καταλαβαίνουμε τι τιμή έχει εκχωρηθεί.

Για παράδειγμα, έστω ότι θέλω να εισαγάγω στο πρόγραμμα και να εκτυπώσω τις τιμές Όνομα, Αριθμός Μητρώου και Βαθμολογία μιας φοιτήτριας.

```
>>> name='Maria'
>>> am=1030
>>> grade=5.6
>>> print name,am,grade
Maria 1030 5.6
>>>
```

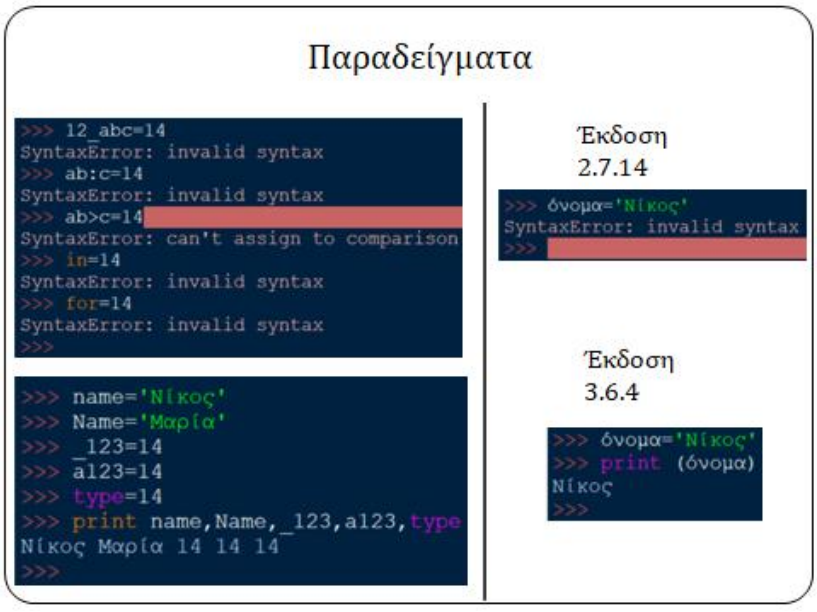
ΠΡΟΣΟΧΗ: Στην εντολή print, θα πρέπει να γράψουμε τις μεταβλητές όπως ακριβώς τις δηλώσαμε. Αν δηλαδή προσπαθήσω να τρέξω την εντολή print(Name) με κεφαλαίο Ν, τότε θα εμφανιστεί μήνυμα ότι δεν υπάρχει αυτή η μεταβλητή.
Επίσης, να σημειωθεί ότι οι τιμές των μεταβλητών θα εμφανιστούν με τη σειρά που θα τις γράψουμε στην εντολή print. Στο παράδειγμα που αναφέραμε, θα εμφανιστούν με τη σειρά το όνομα, ο αριθμός μητρώου και η βαθμολογία.

Υπάρχουν κάποιοι κανόνες για τη δημιουργία του ονόματος μιας μεταβλητής, που αν δεν τους ακολουθήσουμε θα εμφανιστεί μήνυμα ότι υπάρχει συντακτικό λάθος.

1. Ένα όνομα μπορεί να έχει όσους χαρακτήρες θέλουμε, μπορεί να περιέχει αλφαριθμητικούς χαρακτήρες, αλλά πρέπει πάντα να ξεκινάει με λατινικό γράμμα ή τον χαρακτήρα _ (underscore / κάτω παύλα).
2. Τα πεζά διακρίνονται από τα κεφαλαία γράμματα (case sensitive), για παράδειγμα όπως αναφέραμε παραπάνω το name (με πεζό το πρώτο γράμμα) και το Name (με κεφαλαίο το πρώτο γράμμα) είναι δύο διαφορετικές μεταβλητές.
3. Δεν επιτρέπονται σύμβολα (εισαγωγικά, τελείες, κόμματα και άλλα) και κενά.
4. Στην έκδοση 2 δεν μπορούμε να χρησιμοποιήσουμε Ελληνικούς χαρακτήρες, ενώ στην έκδοση 3 μπορούμε.
5. Δεν πρέπει να χρησιμοποιούνται κάποιες από τις λέξεις - κλειδιά (keywords) της Python. Οι λέξεις - κλειδιά συνδέονται με τους κανόνες και τη δομή της γλώσσας και δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών. Η Python έχει τις εξής λέξεις - κλειδιά: and, del, from, not, while, as, elif, global, or, with, assert, else, if, pass, yield, break, except, import, print, class, exec, in, raise, continue, finally, is, return, def, for, lambda, try.

Παράδειγμα

Για παράδειγμα, οι μεταβλητές 12_abc, ab:c, ab>c δεν είναι αποδεκτές, γιατί η πρώτη ξεκινά από αριθμό και οι άλλες δύο περιέχουν σύμβολο. Οι μεταβλητές in και for επίσης δεν είναι αποδεκτές, γιατί είναι λέξεις - κλειδιά. Οι



μεταβλητές name (με το πρώτο γράμμα πεζό) και Name (με το πρώτο γράμμα κεφαλαίο) είναι διαφορετικές. Ενώ οι μεταβλητές _123 a123 και type είναι σωστές. Το τελευταίο είναι όνομα συνάρτησης, αλλά μπορούμε να το χρησιμοποιήσουμε. Δίπλα βλέπουμε ότι στην έκδοση 2.7.14 δεν μπορούμε βάλουμε Ελληνικούς χαρακτήρες, ενώ στην έκδοση 3.6.4 μπορούμε.

Εκτός από την τυπική εντολή εκχώρησης, μπορούμε να δηλώσουμε πολλές τιμές σε πολλές μεταβλητές στην ίδια γραμμή, αρκεί να χωρίζονται με κόμμα και ο αριθμός των μεταβλητών να είναι ίδιος με τον αριθμό των τιμών. Βλέπετε και στο παράδειγμα, οι μεταβλητές x, y και t πήραν τις τιμές 2, True και 'ένα', αντίστοιχα.

```
>>> x,y,t = 2,True,'ένα'
>>> print x,y,t
2 True ένα
>>>
```

Μπορούμε, επίσης, να αναθέσουμε μια τιμή σε πολλές μεταβλητές στην ίδια γραμμή, όπως για παράδειγμα οι x, y και t πήραν την τιμή 'one'.

```
>>> x=y=t='one'
>>> print x,y,t
one one one
>>>
```

Οι παραστάσεις x=x+1 και x+=1 είναι ισοδύναμες. Βλέπετε και στο παράδειγμα, αν αναθέσουμε την τιμή 2 στο x και γράψουμε τις παραστάσεις θα μας εμφανίσει το ίδιο αποτέλεσμα, δηλαδή το 3.

```
>>> x=2
>>> x+=1
>>> x
3
>>>
```

```
>>> x=2
>>> x=x+1
>>> x
3
>>>
```

Ας δούμε μερικές συναρτήσεις που χρησιμεύουν στις μεταβλητές. Η συνάρτηση type() εμφανίζει τον τύπο της μεταβλητής, ενώ οι συναρτήσεις int(), float(), str(), bool() και complex() μετατρέπουν τον τύπο της μεταβλητής, όπως είδαμε και στο μάθημα τύποι δεδομένων.

Δίπλα βλέπουμε τη συνάρτηση εμφάνισης ταυτότητας μιας μεταβλητής την id(). Η id() εμφανίζει την ταυτότητα σε ένα αντικείμενο. Η ταυτότητα ενός αντικειμένου είναι η θέση που δεσμεύει στη μνήμη του υπολογιστή. Βλέπετε και στο παράδειγμα, οι μεταβλητές x και y έχουν πάρει τις τιμές 12 και 'ένα' ως συμβολοσειρά και η id() του x και y μας εμφανίζει την ταυτότητα των δύο μεταβλητών.

```
>>> x,y=12,'ένα'
>>> id(x),id(y)
(77559900, 88586384)
>>>
```

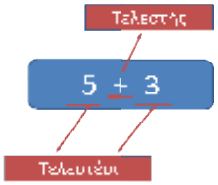
Τέλος, η εντολή del() διαγράφει μια μεταβλητή. Βλέπετε και στο παράδειγμα, δηλώνουμε x ίσον με 3, εκτυπώνουμε το x και εμφανίζει 3, μετά πατάμε del x και πάμε ξανά να εμφανίσουμε το x, οπότε εμφανίζεται μήνυμα ότι το όνομα x δεν εντοπίστηκε.

```
>>> x=3
>>> x
3
>>> del x
>>> x

Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    x
NameError: name 'x' is not defined
>>>
```

3.2. ΜΑΘΗΜΑ 4. ΤΕΛΕΣΤΕΣ ΚΑΙ ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ

3.2.1. ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ ΠΡΑΞΕΩΝ



Ένας τελεστής είναι ένα σύμβολο που αναπαριστά μια λειτουργία. Για παράδειγμα, στην έκφραση 5+3 το + είναι ένας τελεστής. Οι αριθμοί 5 και 3 ονομάζονται τελεστέοι. Οι τελεστές (operators, στα Αγγλικά) είναι λειτουργίες που εκτελούν κάποια ενέργεια και μπορούν να αναπαρασταθούν με σύμβολα όπως το + ή με λέξεις κλειδιά όπως το and.

Οι αριθμητικοί τελεστές είναι τα σύμβολα που χρησιμοποιούνται για να εκτελεστούν αριθμητικές πράξεις. Παραδείγματα αριθμητικών τελεστών φαίνονται στον παρακάτω πίνακα.

Αριθμητικοί Τελεστές

Σύμβολο	Ονομασία	Επεξήγηση	Παραδείγματα	Σειρά Προτεραιότητας	
**	Ύψωση σε δύναμη	Ύψωση αριθμού σε μια δύναμη	2**3=8	1	Υψηλή
*	Πολλαπλασιασμός	Πολλαπλασιασμός αριθμών ή επανάληψη συμβολοσειρών τόσες φορές	4*3=12 α*3=ααα	2	
/	Διαίρεση	Διαίρεση αριθμών	6/5=1 6/5.0=1.2		
%	Υπόλοιπο ακέραιας διαίρεσης	Το υπόλοιπο της ακέραιας διαίρεσης	10%3=1		
+	Πρόσθεση	Πρόσθεση αριθμών ή συμβολοσειρών	5+3=8 α+α=αα	3	
-	Αφαίρεση	Αφαίρεση αριθμών	5-2=3		Χαμηλή

Εκτός από τους τελεστές, κάποιες πράξεις μπορούμε να τις κάνουμε και μέσω συναρτήσεων. Εξηγήσαμε την προηγούμενη εβδομάδα τι είναι συνάρτηση. Οι συναρτήσεις που θα χρησιμοποιήσουμε είναι ενσωματωμένες στην Python. Θα ξεκινήσουμε με τη συνάρτηση divmod(διαιρετέος, διαιρέτης) όπου γίνεται η ακέραια διαίρεση, αυτό που θα μας εμφανίσει είναι το πηλίκο και το υπόλοιπο. Στον διαιρέτη και στον διαιρετέο μπορούμε να βάλουμε όποιον αριθμό θέλουμε ακόμα και αρνητικούς ή πραγματικούς, εκτός βέβαια από τον περιορισμό που έχουμε, ο διαιρέτης να είναι πάντα διάφορος του μηδέν.

```
>>> divmod(10,3)
(3, 1)
>>> divmod(10,3.5)
(2.0, 3.0)
>>> divmod(10.1,3.3)
(3.0, 0.20000000000000018)
>>>
```

```
>>> divmod(-10,3)
(-4, 2)
>>> divmod(-10,-3)
(3, -1)
>>> divmod(10+0j,3+0j)
((3+0j), (1+0j))
>>>
```

-10=(-3)*3 1

Η συνάρτηση ύψωσης σε δύναμη είναι η pow και στην παρένθεση βάζουμε πρώτα τη βάση και μετά τον εκθέτη. Βλέπετε και στο παράδειγμα $10^2 = 100$ και $10^{-2} = 0,01$ ενώ $-10^3 = -1000$. Επίσης, μπορούμε να βάλουμε και μιγαδικούς.

```
>>> pow(10,2)
100
>>> pow(10,-2)
0.01
>>> pow(-10,3)
-1000
>>> pow(10+0j,2+0j)
(100+0j)
>>>
```

Η συνάρτηση απόλυτης τιμής είναι η abs και στην παρένθεση βάζουμε τον αριθμό που θέλουμε. Όπως βλέπετε στο παράδειγμα, και εδώ μπορούμε να βάλουμε μιγαδικούς.

```
>>> abs(3)
3
>>> abs(-3)
3
>>> abs(-3.5)
3.5
>>> abs(3+0j)
3.0
>>>
```

Στα παρακάτω παραδείγματα βλέπουμε πώς μπορούμε να στρογγυλοποιήσουμε έναν αριθμό, με τη συνάρτηση round που μέσα στην παρένθεση βάζουμε πρώτα τον αριθμό και μετά το ψηφίο στρογγυλοποίησης. Αν το ψηφίο στρογγυλοποίησης είναι θετικός αριθμός, η στρογγυλοποίηση γίνεται στα δεξιά της υποδιαστολής στο δεκαδικό μέρος, ενώ αν είναι αρνητικός γίνεται στα αριστερά της υποδιαστολής, σε δεκάδες, εκατοντάδες κλπ. Αν δεν υπάρχει ψηφίο στρογγυλοποίησης, η στρογγυλοποίηση γίνεται στον πλησιέστερο ακέραιο. Στα παραδείγματα της διαφάνειας φαίνονται τα αποτελέσματα των στρογγυλοποιήσεων για όλες τις περιπτώσεις που αναφέρθηκαν.

Παραδείγματα:

```
>>> round(5.123456789,3)
5.123
>>> round(5.123456789,6)
5.123457
>>>
```

```
>>> round(987654321.43,-3)
987654000.0
>>> round(987654321.43,-6)
988000000.0
>>>
```

```
>>> round(2.0)
2.0
>>> round(2.4)
2.0
>>> round(2.5)
3.0
>>> round(2.9)
3.0
>>>
```

Στα επόμενα παραδείγματα, βλέπουμε τις συναρτήσεις αθροίσματος, ελαχίστου και μεγίστου, που είναι sum, min και max, αντίστοιχα. Μέσα στην παρένθεση βάζουμε τους αριθμούς που πρέπει να είναι σε αγκύλες, άγκιστρα ή παρενθέσεις και να χωρίζονται με κόμμα. Οι συναρτήσεις min και max εκτός από αριθμούς επιστρέφουν αποτέλεσμα και σε συμβολοσειρές, μόνο που η μέτρηση γίνεται αλφαβητικά. Στα παραδείγματα φαίνονται τα

Παραδείγματα:

```
>>> sum((1.3,2.2,4.52,3,9))
20.02
>>> max((1.3,2.2,4.52,3,9))
9
>>> min((1.3,2.2,4.52,3,9))
1.3
>>>
```

```
>>> min(('abcdefg'))
'a'
>>> max(('abcdefg'))
'g'
>>> max(('one','tow','three'))
'tow'
>>> min(('one','tow','three'))
'one'
>>>
```

αποτελέσματα των συναρτήσεων sum, min και max για όλες τις περιπτώσεις που αναφέρθηκαν.

Βλέπουμε επίσης πώς γίνεται η μετατροπή αριθμών μεταξύ διαφορετικών αριθμητικών συστημάτων. Με τη συνάρτηση eval μετατρέπουμε αριθμούς από οποιοδήποτε αριθμητικό σύστημα στο δεκαδικό, με τη συνάρτηση bin στο δυαδικό σύστημα και με τις συναρτήσεις oct και hex στο οκταδικό και στο δεκαεξαδικό σύστημα, αντίστοιχα. Στο παράδειγμα που ακολουθεί, βλέπουμε την μετατροπή του αριθμού 25 από τα άλλα αριθμητικά συστήματα σε δεκαδικό, στο δυαδικό, στο οκταδικό και στο δεκαεξαδικό σύστημα. Παρατηρούμε ότι μόνο στην συνάρτηση eval ο αριθμός πρέπει να είναι σε εισαγωγικά. Επίσης, βλέπουμε ότι η αναγνώριση του αριθμητικού συστήματος στο οποίο ανήκει ο αριθμός γίνεται με το πρόθεμα '0b' για τους δυαδικούς αριθμούς, με το πρόθεμα '0' για τους οκταδικούς αριθμούς και με το πρόθεμα '0x' για τους δεκαεξαδικούς αριθμούς. Αν δεν υπάρχει πρόθεμα, ο αριθμός λογίζεται ως δεκαδικός.

Μετατροπή σε δεκαδικό

```
>>> eval('0b11001')
25
>>> eval('031')
25
>>> eval('0x19')
25
>>>
```

Μετατροπή σε δυαδικό

```
>>> bin(25)
'0b11001'
>>> bin(031)
'0b11001'
>>> bin(0x19)
'0b11001'
>>>
```

Μετατροπή σε οκταδικό

```
>>> oct(0b11001)
'031'
>>> oct(0x19)
'031'
>>> oct(25)
'031'
>>>
```

Μετατροπή σε δεκαεξαδικό

```
>>> hex(0b11001)
'0x19'
>>> hex(031)
'0x19'
>>> hex(25)
'0x19'
>>>
```

3.2.2. ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ ΕΞΟΔΟΥ

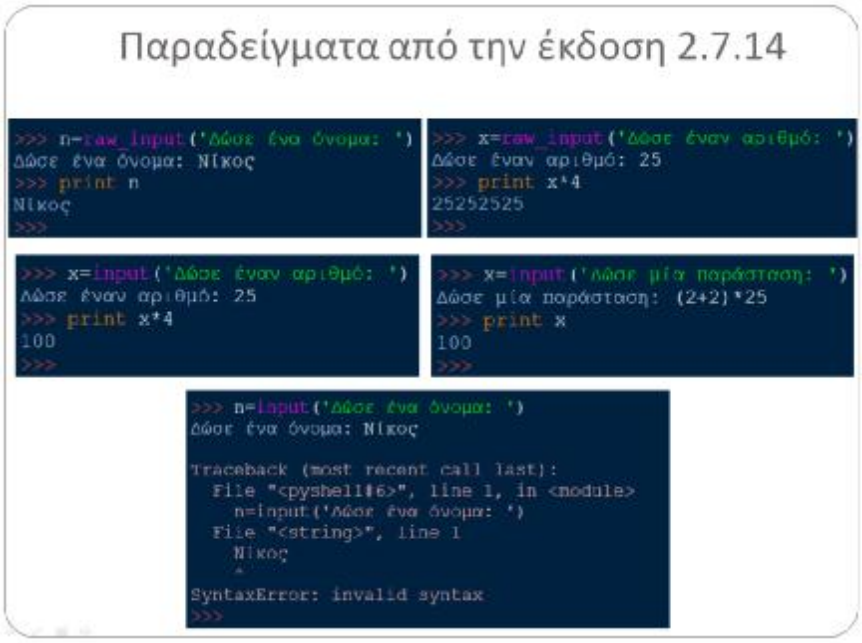
Ένα πρόγραμμα μπορεί να πάρει τιμές για τις μεταβλητές του από τον χρήστη. Τα δεδομένα εισάγονται κατά τη διάρκεια της εκτέλεσης του προγράμματος από μια μονάδα εισόδου (π.χ. το πληκτρολόγιο). Υπάρχουν αρκετές διαφορές στις εκδόσεις 2 και 3 όσον αφορά στη διαδικασία εισόδου, που θα δούμε αναλυτικά. Στην έκδοση 2, για να εισάγει ο χρήστης δεδομένα σε μορφή συμβολοσειράς χρησιμοποιούμε τη συνάρτηση raw_input(). Ας δούμε πώς συντάσσεται η raw_input():

a=raw_input('μήνυμα').

Η μεταβλητή a παίρνει την τιμή που εισάγει ο χρήστης σε μορφή συμβολοσειράς. Στη raw_input() βάζουμε μια παράμετρο (μήνυμα) που μπορεί να είναι συμβολοσειρά, σταθερά η μεταβλητή. Η παράμετρος δεν είναι υποχρεωτική, τη

χρησιμοποιούμε μόνο για να εμφανίσει στο χρήστη ένα μήνυμα που θα τον προτρέπει στην εισαγωγή των δεδομένων.

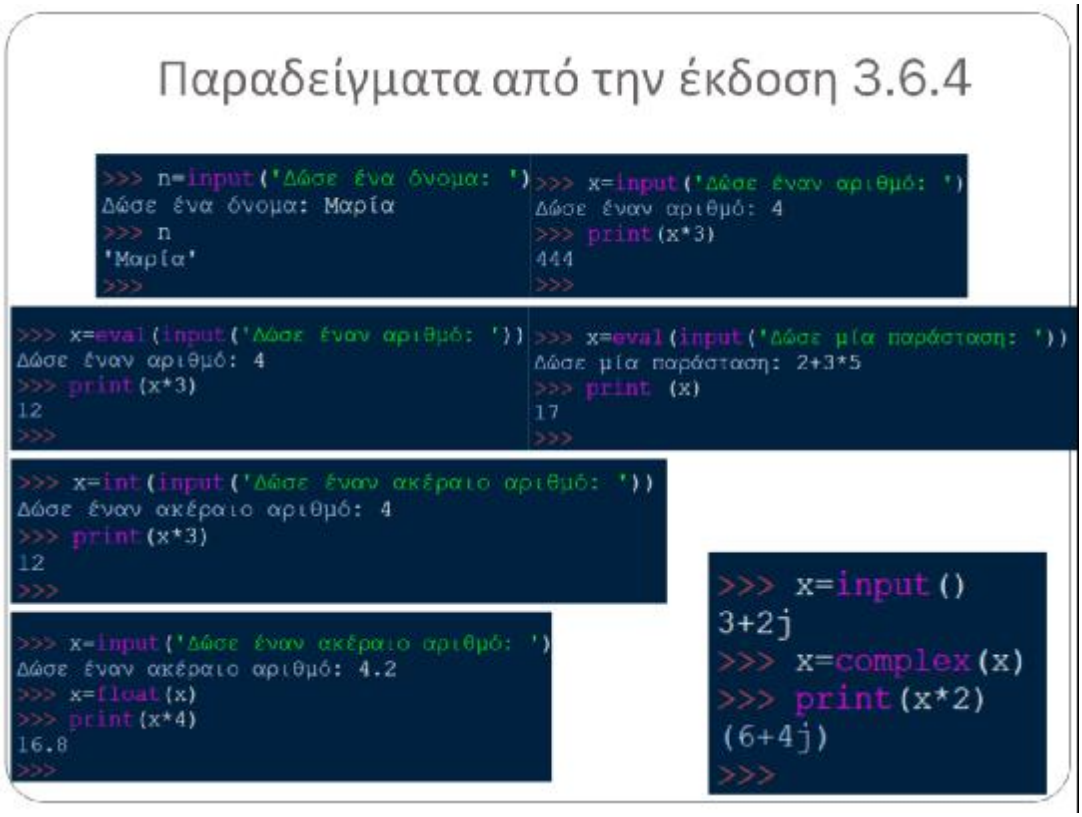
Για να εισάγει ο χρήστης αριθμητικά δεδομένα, χρησιμοποιούμε τη συνάρτηση input(). Η σύνταξή της είναι ίδια με της raw_input(). Η input() όμως δεν δέχεται για εισαγωγή δεδομένων αλφαβητικούς χαρακτήρες. Ας δούμε μερικά παραδείγματα.



Στο πρώτο παράδειγμα, ζητάμε από τον χρήστη να δώσει ένα όνομα που θα εκχωρηθεί στη μεταβλητή n. Βλέπουμε ότι αν εκτυπώσουμε τη μεταβλητή n, θα εμφανιστεί το όνομα που εισήχθηκε. Δίπλα, στο άλλο παράδειγμα ζητάμε από τον χρήστη να δώσει έναν αριθμό, εισάγει το 25 και η τιμή αυτή εκχωρείται στη μεταβλητή x. Εκτυπώνουμε τη μεταβλητή x επί 4 και παρατηρούμε ότι το 25 είναι σε μορφή συμβολοσειράς. Δηλαδή, η συνάρτηση raw_input() εισάγει δεδομένα μόνο σε μορφή συμβολοσειράς. Κάτω έχουμε το ίδιο παράδειγμα, με τη συνάρτηση input(), όπου βλέπουμε ότι στην πράξη x επί 4 μας εμφανίζει το αποτέλεσμα 100. Εκτός από αριθμούς στη συνάρτηση input() μπορεί ο χρήστης να εισάγει και μια παράσταση, αυτό όμως που καταχωρείται στη μεταβλητή δεν είναι η παράσταση, αλλά το αποτέλεσμά της. Βλέπουμε στο τέταρτο παράδειγμα ότι ο χρήστης εισάγει την παράσταση (2+2)*25 και η μεταβλητή x έχει πάρει την τιμή 100. Στο τελευταίο παράδειγμα, βλέπουμε ότι η συνάρτηση input() εμφανίζει συντακτικό λάθος αν εισάγουμε αλφαβητικούς χαρακτήρες.

Στην έκδοση 3 της Python, χρησιμοποιούμε τη συνάρτηση input() για εισαγωγή δεδομένων σε μορφή συμβολοσειράς. Έχει ίδια σύνταξη με τις συναρτήσεις που είδαμε. Για εισαγωγή δεδομένων σε μορφή αριθμού χρησιμοποιούμε την input() σε συνδυασμό με τις συναρτήσεις eval(), int(), float() και complex() που έχουμε ήδη συναντήσει σε προηγούμενα μαθήματα.

Ας δούμε μερικά παραδείγματα εισόδου για την έκδοση 3 της Python. Στο πρώτο παράδειγμα εκχωρείται στη μεταβλητή n το όνομα 'Μαρία'. Στο διπλανό παράδειγμα ζητάμε από τον χρήστη να δώσει έναν αριθμό και δίνει το 4. Το 4 έχει καταχωρηθεί στη μεταβλητή x σε μορφή συμβολοσειράς. Στο επόμενο παράδειγμα βλέπουμε πώς μπορούμε να χρησιμοποιήσουμε τη συνάρτηση eval() σε συνδυασμό με την input() για εισαγωγή αριθμών και παραστάσεων. Με τον ίδιο τρόπο χρησιμοποιούμε και την int() δηλαδή χρησιμοποιούμε την input() ως παράμετρο της int(), η οποία δέχεται μόνο ακέραιους αριθμούς. Στα τελευταία δύο παραδείγματα, βλέπουμε ότι στη μεταβλητή x έχει εκχωρηθεί μια συμβολοσειρά που μετατρέπουμε σε πραγματικό και μιγαδικό με τις συναρτήσεις float() και complex(), αντίστοιχα.



print()

```
>>> print('Year',2018)
('Year', 2018)
>>> print 'Year',2018
Year 2018
>>>
```

Τα δεδομένα που εισήχθησαν στο πρόγραμμα επεξεργάζονται και εξάγουν τα αποτελέσματα από μία μονάδα εξόδου (π.χ. την οθόνη). Μια εντολή εξόδου είναι η print(), για την οποία υπάρχουν αρκετές διαφορές ανάμεσα στις εκδόσεις 2 και 3 της Python. Στην έκδοση 2, η print() είναι εντολή γι' αυτό χρωματίζεται πορτοκαλί και οι παρενθέσεις δεν απαιτούνται. Όπως βλέπετε, αν βάλουμε παρενθέσεις εμφανίζει το αποτέλεσμα ακριβώς έτσι όπως το έχουμε γράψει, με εισαγωγικά, κόμμα και

παρενθέσεις, ενώ αν δεν βάλουμε παρενθέσεις εμφανίζει απλά τη συμβολοσειρά year και τον ακέραιο αριθμό 2018.

Η print() στην έκδοση 3 μετατράπηκε σε συνάρτηση, όπου χρωματίζεται με μωβ και οι παρενθέσεις είναι υποχρεωτικές. Στο σχετικό παράδειγμα, η print() με παρενθέσεις εμφανίζει τη συμβολοσειρά και τον αριθμό έτσι όπως πρέπει, ενώ αν δεν βάλουμε τις παρενθέσεις εμφανίζει συντακτικό λάθος.

print()

```
>>> print('Year',2018)
Year 2018
>>> print 'Year',2018
SyntaxError: Missing pa
rentheses in call to 'p
rint'. Did you mean pri
nt('Year',2018)?
>>>
```

Ας δούμε μερικές χρήσιμες πληροφορίες ακόμα για την εντολή - συνάρτηση print() που ισχύουν και για τις δύο εκδόσεις. Με το κόμμα εκτυπώνουμε πολλές τιμές διαφορετικού τύπου ταυτόχρονα, ενώ η ανάποδη κάθετος n (\n) προκαλεί αλλαγή γραμμής.

```
>>> print'Year',2018,'\n',True
Year 2018
True
>>>
```

Ο ρόλος των διπλών και απλών εισαγωγικών όταν συνδυάζονται σε μία print() είναι για να εκτυπώσουμε το ένα είδος από αυτά. Στο ένα παράδειγμα εκτυπώνουμε τα απλά εισαγωγικά που είναι εμφωλευμένα στα διπλά και στο άλλο παράδειγμα εκτυπώνουμε τα διπλά εισαγωγικά που είναι εμφωλευμένα στα απλά.

```
>>> print "Μαθαίνοντας την 'Python'"
Μαθαίνοντας την 'Python'
>>> print 'Μαθαίνοντας την "Python"'
Μαθαίνοντας την "Python"
>>>
```

Στην έκδοση 2 της Python, αν χρησιμοποιήσουμε παρενθέσεις η ανάποδη κάθετος n (\n) δεν λειτουργεί και η εκτύπωση των ελληνικών χαρακτήρων εμφανίζεται κωδικοποιημένη.

```
>>> print('Έτος','\n',2018)
('\xb8\xf4\xef\xf2', '\n', 2018)
>>> print'Έτος','\n',2018
Έτος
2018
>>>
```

Στην έκδοση 3 μπορούμε να χρησιμοποιήσουμε τις παραμέτρους sep και end. Η sep είναι για να εισάγουμε μια συμβολοσειρά ανάμεσα στις τιμές μέσα στην print() και η end για να εισάγουμε μια συμβολοσειρά στο τέλος των τιμών. Στο σχετικό παράδειγμα, η πρώτη print() εμφανίζει τις τιμές 10, 20, 30 και 40 με το σύμβολο του ευρώ ανάμεσα ενώ η δεύτερη print() εμφανίζει το σύμβολο του ευρώ στο τέλος, μετά και την τελευταία τιμή.

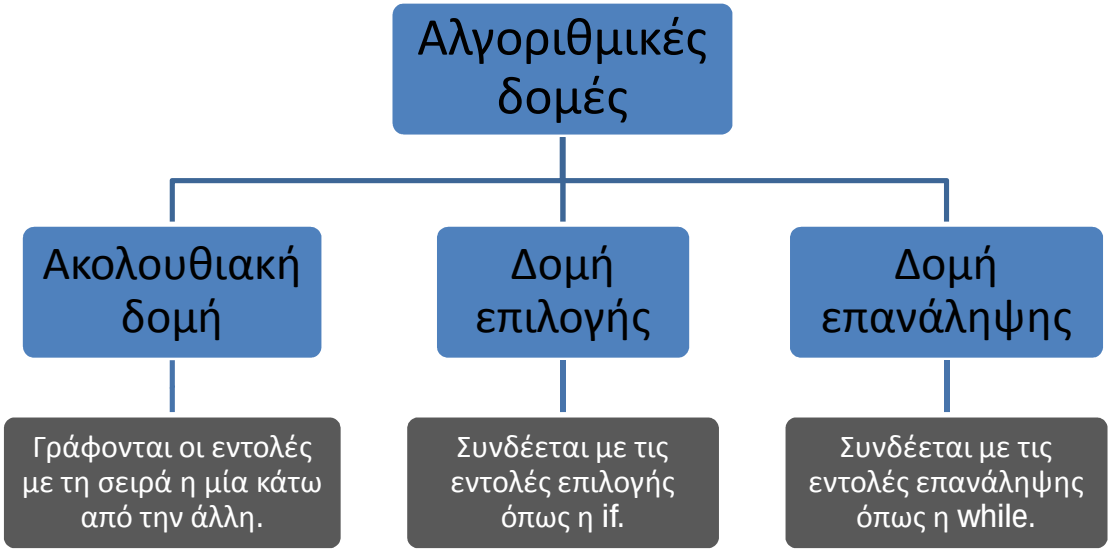
```
>>> print(10,20,30,40,sep='€ ')
10€ 20€ 30€ 40
>>> print(10,20,30,40,end='€')
10 20 30 40€
>>>
```

4. ΔΟΜΕΣ ΕΠΙΛΟΓΗΣ

4.1. ΜΑΘΗΜΑ 5. ΤΕΛΕΣΤΕΣ

4.1.1. ΣΥΓΚΡΙΤΙΚΟΙ ΚΑΙ ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ

Η Python ακολουθεί τις αρχές του δομημένου προγραμματισμού. Ο δομημένος προγραμματισμός περιέχει τρεις αλγοριθμικές δομές.



- Την ακολουθιακή δομή, όπου οι εντολές γράφονται με τη σειρά η μία κάτω από την άλλη. Η δομή αυτή είναι η πιο απλή και την έχουμε χρησιμοποιήσει έως τώρα.
- Τη δομή επιλογής, που συνδέεται με τις εντολές επιλογής, όπως η if που θα δούμε στο επόμενο μάθημα.
- Τέλος, τη δομή επανάληψης, που συνδέεται με τις εντολές επανάληψης, όπως η while.

Με λίγα λόγια, ένα πρόγραμμα στην Python συντάσσεται με συνδυασμό όλων των αλγοριθμικών δομών.

Πριν αναλύσουμε τις εντολές επιλογής, ας δούμε πρώτα τους συγκριτικούς και λογικούς τελεστές. Θα τους χρειαστούμε για να δημιουργήσουμε συνθήκες στις εντολές επιλογής και επανάληψης. Ξεκινάμε με τους συγκριτικούς τελεστές που βλέπετε στον πίνακα που ακολουθεί. Οι συγκριτικοί τελεστές είναι τα σύμβολα που χρησιμοποιούνται για να συγκριθούν μεταξύ τους δύο αριθμητικές εκφράσεις ή μεταβλητές.

ΣΥΓΚΡΙΤΙΚΟΙ ΤΕΛΕΣΤΕΣ			
Σύμβολο	**Ονομασία**	**Επεξήγηση**	**Παραδείγματα**
>=	Μεγαλύτερο ή ίσο	Το πρώτο μέλος είναι μεγαλύτερο ή ίσο του δεύτερου	5>=5
>	Μεγαλύτερο	Το πρώτο μέλος είναι μεγαλύτερο του δεύτερου	5>2
<=	Μικρότερο ή ίσο	Το πρώτο μέλος είναι μικρότερο ή ίσο του δεύτερου	1<=1
<	Μικρότερο	Το πρώτο μέλος είναι μικρότερο του δεύτερου	1<3
!=	Διάφορο	Το πρώτο μέλος είναι διαφορετικό του δεύτερου	a=3, b=4 άρα a!=b
==	Ίσο	Το πρώτο μέλος είναι ίσο με το δεύτερο	a=3, b=3 άρα a==b

Συνεχίζουμε με τους λογικούς τελεστές που βλέπετε στον παρακάτω πίνακα.

ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ			
Σύμβολο	Ονομασία	Επεξήγηση	Παραδείγματα
NOT	NOT X	Αν το X είναι αληθές, τότε επιστρέφει ως αποτέλεσμα ψευδές.	X = αληθές, άρα not X = ψευδές
AND	X AND Y	Για να έχει αποτέλεσμα αληθές πρέπει και το X και το Y να είναι αληθή, αλλιώς το αποτέλεσμα είναι ψευδές.	X = αληθές, Y = ψευδές, τότε X and Y είναι ψευδές X = αληθές, Y = αληθές, τότε X and Y είναι αληθές
OR	X OR Y	Αρκεί ένα από τα δύο να είναι αληθές για να είναι το αποτέλεσμα αληθές.	X = αληθές, Y = ψευδές, επομένως X or Y είναι αληθές

Εκτός των τελεστών σύγκρισης που είδαμε προηγουμένως, υπάρχουν δύο ακόμα συγκριτικοί τελεστές, ο is (ίσο) και ο is not (διάφορο). Οι τελεστές is και is not δεν ταυτίζονται με τους συγκριτικούς τελεστές «ίσο» και «διάφορο». Η διαφορά ανάμεσα στους τελεστές αυτούς φαίνεται στο παράδειγμα, όπου έχουμε δύο μεταβλητές x και y με την ίδια τιμή. Ο τελεστής με διπλό ίσον εμφανίζει true, ενώ ο τελεστής is εμφανίζει false. Αυτό γίνεται γιατί οι μεταβλητές μπορεί να έχουν την ίδια τιμή, αλλά έχουν διαφορετικό αντικείμενο. Το βλέπουμε με τη συνάρτηση id, όπου η ταυτότητα των δύο μεταβλητών δεν είναι ίδια. Συνεχίζοντας, οι μεταβλητές a και b έχουν ίδια ταυτότητα, γι' αυτό ο τελεστής is εμφανίζει true. Το ίδιο ισχύει και για τον τελεστή is not.

```
>>> x,y='ένα','ένα'
>>> x==y, x is y
(True, False)
>>> id (x), id (y)
(89512080, 89512104)
>>> a=b='δύο'
>>> a==b, a is b
(True, True)
>>> id (a), id (b)
(88022744, 88022744)
>>>
```

Η Python περιέχει και δύο επιπλέον τελεστές, τον in και τον not in. Οι τελεστές αυτοί ελέγχουν αν ένα αντικείμενο ανήκει σε μια ακολουθία ή όχι. Στο παράδειγμα, υπάρχει μια μεταβλητή a που έχει πάρει την τιμή 'e'. Η παράσταση a in 'abcde' εμφανίζει true γιατί ο χαρακτήρας 'e' που είναι η τιμή της μεταβλητής a ανήκει στη συμβολοσειρά 'abcde'. Στο επόμενο παράδειγμα, όπου η τιμή της μεταβλητής a δεν ανήκει στην ακολουθία 'abcd', το αποτέλεσμα είναι false. Τον τελεστή in θα τον χρησιμοποιήσουμε στην εντολή επανάληψης for.

```
>>> a='e'
>>> a in 'abcde'
True
>>> a in 'abcd'
False
>>>
```

Σε μια συνθήκη μπορεί να χρησιμοποιήσουμε πολλούς τελεστές και γι' αυτό το λόγο πρέπει να γνωρίζουμε τη σειρά προτεραιότητας μεταξύ των διαφόρων τελεστών. Στον πίνακα βλέπουμε τη σειρά προτεραιότητας όλων των τελεστών.

Προτεραιότητα Τελεστών

Σύμβολο	Ονομασία	Σειρά Προτεραιότητας	
**	Ύψωση σε δύναμη	1	Υψηλή
*, /, %	Πολλαπλασιασμός, Διαίρεση, Υπόλοιπο ακέραιας διαίρεσης	2	
+, -	Πρόσθεση, Αφαίρεση	3	
<=, <, >=, >	Τελεστές σύγκρισης	4	
==, !=, is, is not	Τελεστές σύγκρισης - ισότητας, ανισότητας	5	
not	Λογικοί τελεστές	6	
and		7	Χαμηλή
or		8	

4.2. **ΜΑΘΗΜΑ** 6. **ΔΟΜΗ ΕΠΙΛΟΓΗΣ** if

4.2.1. **ΔΟΜΗ ΕΠΙΛΟΓΗΣ** if-else

Χρησιμοποιούμε την εντολή if για να ελέγξουμε τη ροή της εκτέλεσης ενός προγράμματος. Ελέγχεται μία συνθήκη και ανάλογα με το αποτέλεσμα (Αληθής ή Ψευδής) εκτελείται ή δεν εκτελείται μία ομάδα (μπλοκ) εντολών. Συντάσσουμε την εντολή if ως εξής:

Σύνταξη της if:

```
if (συνθήκη):
        (εντολή / εντολές)
```

Στην πρώτη γραμμή θέτουμε μία επικεφαλίδα με τη δεσμευμένη λέξη if, μία συνθήκη και, στο τέλος, μία άνω κάτω τελεία (:). Η συνθήκη μπορεί να είναι μία οποιαδήποτε λογική έκφραση που παίρνει τιμή True ή False. Αν η συνθήκη είναι αληθής (True), τότε εκτελούνται οι εντολές που ακολουθούν σε εσοχή. Αν η συνθήκη είναι ψευδής (False), τότε δεν εκτελείται καμία από τις εντολές. Δίπλα στο σχήμα βλέπουμε το διάγραμμα ροής της if.

Μετά την άνω κάτω τελεία μπαίνουμε μέσα στο μπλοκ εντολών της if, όπου μπορούμε να βάλουμε όσες εντολές θέλουμε, αρκεί να υπάρχει τουλάχιστον μία. Οι εντολές πρέπει να είναι μετατοπισμένες προς τα δεξιά. Η τυπική μετατόπιση ή εσοχή (indentation) των εντολών είναι τέσσερα κενά. Η Python μας βοηθά σε αυτό, ρυθμίζοντας αυτόματα τις εσοχές για μας, απλά πατώντας Enter μετά την πληκτρολόγηση της άνω κάτω τελείας. Αν λείπει ένα κενό ή υπάρχουν παραπάνω από 4 τότε μπορεί να προκύψει λάθος στη συνέχεια του προγράμματος. Δηλαδή την αρχή και το τέλος των εντολών την ορίζει η εσοχή και η τήρησή της είναι κανόνας σύνταξης στην Python.

Μπορούμε να αλλάξουμε τα κενά της εσοχής πηγαίνοντας είτε στην κονσόλα είτε στον επεξεργαστή **à** options **à** configure idle. Εκεί θα ανοίξει το παράθυρο που βλέπετε δίπλα. Στο πεδίο Indentation Width μπορούμε να αλλάξουμε τα κενά της εσοχής. Η προεπιλογή στην Python είναι τέσσερα κενά. Εκτός αυτού, στο πεδίο Base Editor Font μπορούμε να αλλάξουμε γραμματοσειρά, να ρυθμίσουμε το μέγεθός της, αλλά και τις ιδιότητές της, αν θέλουμε για παράδειγμα να είναι έντονη ή κανονική. Οι ρυθμίσεις των κενών της εσοχής γίνονται μόνο για τον επεξεργαστή, ενώ οι άλλες ρυθμίσεις γίνονται και για την κονσόλα Shell.

Ας επιστρέψουμε στην εντολή if, για να δούμε ένα απλό παράδειγμα. Στο αριστερό μέρος είναι ο κώδικας στον επεξεργαστή, όπου έχει μια συνάρτηση input() που ζητά από το χρήστη να δώσει βαθμολογία για να την καταχωρίσει στη μεταβλητή x, ενώ από κάτω έχει δύο εντολές if με δύο συνθήκες σύγκρισης του x με το 5. Αν το x είναι μικρότερο του 5, εκτυπώνει «κόπηκες», ενώ αν είναι μεγαλύτερο ή ίσο του 5 εκτυπώνει «πέρασες». Δίπλα είναι τα αποτελέσματα στην κονσόλα Shell. Έχουμε τρέξει δύο φορές το πρόγραμμα, την πρώτη φορά δώσαμε τη βαθμολογία 7 και εμφανίζει «πέρασες» και τη δεύτερη φορά δώσαμε τη βαθμολογία 4 και εμφανίζει «κόπηκες».

Το πρόγραμμα στον επεξεργαστή.

```
x=input('Δώσε βαθμολογία: ')
if x<5:
    print('Κόπηκες')
if x>=5:
    print('Πέρασες')
```

Αποτελέσματα στην κονσόλα (Shell).

```
Δώσε βαθμολογία: 7
Πέρασες
>>>
```

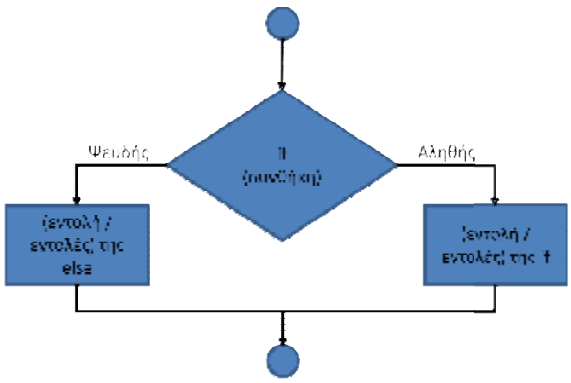
```
Δώσε βαθμολογία: 4
Κόπηκες
>>>
```

Είδαμε την απλή εντολή ελέγχου if, υπάρχει όμως και η if - else που η μόνη διαφορά με την if είναι ότι αν δεν ισχύει η συνθήκη τρέχει μια άλλη ομάδα εντολών της else. Βλέπετε πώς συντάσσεται η if - else και δίπλα το διάγραμμα ροής της.

Σύνταξη της if-else:

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

Διάγραμμα ροής της if-else:



Στο ίδιο παράδειγμα με την if παρατηρούμε ότι εδώ χρησιμοποιήσαμε μία συνθήκη και εμφανίζει για τη βαθμολογία 2 «κόπηκες» ενώ για τη βαθμολογία 8 «πέρασες». Θεωρούμε ότι η βαθμολογία κυμαίνεται στην κλίμακα από 0 έως 10.

Το πρόγραμμα στον επεξεργαστή.

```
x=input('Δώσε βαθμολογία: ')
if x<5:
    print('Κόπηκες')
else:
    print('Πέρασες')
```

Αποτελέσματα στην κονσόλα (Shell).

```
Δώσε βαθμολογία: 2
Κόπηκες
>>>
```

```
Δώσε βαθμολογία: 8
Πέρασες
>>>
```

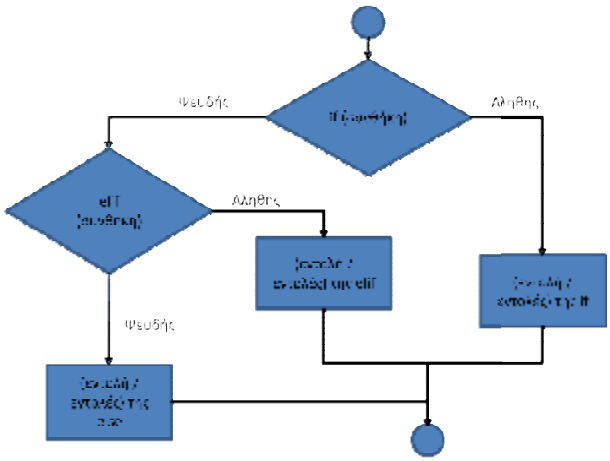
4.2.2. **ΔΟΜΗ ΕΠΙΛΟΓΗΣ** elif

Στην Python μπορούμε να έχουμε αλυσιδωτές συνθήκες (με πολλές συνθήκες και εντολές). Εκτελούνται μόνο οι εντολές που υπακούουν σε μία συνθήκη. Κάθε συνθήκη ελέγχεται με τη σειρά, αν η **πρώτη είναι ψευδής**, πάμε στη δεύτερη κ.ο.κ. Αν κάποια συνθήκη είναι αληθής, τότε εκτελείται η αντίστοιχη ομάδα εντολών και η εντολή if ολοκληρώνεται, ακόμη και αν ακολουθούν και άλλες συνθήκες που αληθεύουν. Για αλυσιδωτές συνθήκες χρησιμοποιούμε τη δεσμευμένη λέξη elif (συντόμευση του else if). Βλέπετε τη σύνταξη της elif και δίπλα το διάγραμμα ροής της.

Σύνταξη της elif:

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

Διάγραμμα ροής της elif.



Για παράδειγμα, ζητάμε τη βαθμολογία ενός φοιτητή και, ανάλογα με το βαθμό, εμφανίζονται τέσσερις απαντήσεις: αν είναι μικρότερος του 5 εμφανίζεται «Κόπηκες», από 5 έως μικρότερο του 6,5 εμφανίζεται «Καλώς», από 6,5 έως μικρότερο του 8,5 εμφανίζεται «Λίαν καλώς» και από 8,5 και πάνω εμφανίζεται «Άριστα». Με αυτή τη λογική χρησιμοποιήσαμε την elif στον επεξεργαστή κώδικα. Τα αντίστοιχα αποτελέσματα εκτέλεσης του προγράμματος με διαφορετικές τιμές της βαθμολογίας εμφανίζονται στην κονσόλα Shell.

Το πρόγραμμα στον επεξεργαστή κώδικα.

```
x=input('Δώσε βαθμολογία: ')
if x<5:
    print('Κόπηκες')
elif (x>=5) and (x<6.5):
    print('Καλώς')
elif(x>=6.5)and (x<8.5):
    print('Λίαν καλώς')
else:
    print('Άριστα')
```

Αποτελέσματα στην κονσόλα (Shell).

```
========== RESTART
Δώσε βαθμολογία: 2
Κόπηκες
>>>
========== RESTART
Δώσε βαθμολογία: 5
Καλώς
>>>
========== RESTART
Δώσε βαθμολογία: 7
Λίαν καλώς
>>>
========== RESTART
Δώσε βαθμολογία: 8.6
Άριστα
>>>
```

Η Python μας επιτρέπει να έχουμε εμφωλευμένες εντολές ελέγχου, δηλαδή μια εντολή if μέσα σε μια άλλη. Οι εμφωλευμένες εντολές ελέγχου δεν χρησιμοποιούνται τόσο συχνά γιατί καθιστούν δυσανάγνωστα τα προγράμματα. Οι εμφωλευμένες συνθήκες συντάσσονται όπως φαίνεται στο ακόλουθο σχήμα.

Σύνταξη

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

Το πρόγραμμα που ακολουθεί διαβάζει την ηλικία του χρήστη και πόσα χρόνια διαθέτει δίπλωμα αυτοκινήτου (αν έχει) και εμφανίζει αν μπορεί ο χρήστης να ενοικιάσει αυτοκίνητο. Το ελάχιστο όριο ηλικίας είναι 23 έτη και το μέγιστο 70 έτη. Η άδεια οδήγησης θα πρέπει να έχει εκδοθεί 1 έτος πριν την ημερομηνία ενοικίασης.

Το πρόγραμμα στον επεξεργαστή.

```
age=input('Δώσε την ηλικία σου\n:')
if age<18:
    print('Δεν έχεις δίπλωμα')
else:
    years=input('Πόσα χρόνια έχεις δίπλωμα οδήγησης;\n:')
    if (years>1) and (age>=23) and (age<=70):
        print 'Μπορείς να ενοικιάσεις αμάξι'
    else:
        print 'Δεν μπορείς να ενοικιάσεις αμάξι'
```

Αποτελέσματα στην κονσόλα (Shell).

```
Δώσε την ηλικία σου
:25
Πόσα χρόνια έχεις δίπλωμα οδήγησης;
:5
Μπορείς να ενοικιάσεις αμάξι
>>>
========== RESTART: C:/Users/B
Δώσε την ηλικία σου
:71
Πόσα χρόνια έχεις δίπλωμα οδήγησης;
:31
Δεν μπορείς να ενοικιάσεις αμάξι
>>>
========== RESTART: C:/Users/B
Δώσε την ηλικία σου
:17
Δεν έχεις δίπλωμα
>>>
```

5. ΔΟΜΕΣ ΕΠΑΝΑΛΗΨΗΣ

5.1. ΜΑΘΗΜΑ 7. Η ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ for

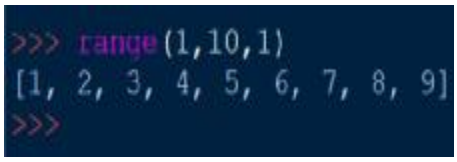
5.1.1. ΣΥΝΑΡΤΗΣΗ range()

Πριν αναφερθούμε στην εντολή επανάληψης for, θα δούμε τη συνάρτηση range(), που σημαίνει εύρος, επειδή χρησιμοποιείται πολύ συχνά στην εντολή for. Με αυτή τη συνάρτηση μπορούμε να δημιουργήσουμε μια λίστα, δηλαδή μια συλλογή στοιχείων που περικλείονται από αγκύλες. Θα δούμε αναλυτικά τι είναι λίστα στα επόμενα μαθήματα. Μια λίστα μπορεί να περιέχει όλους τους τύπους δεδομένων όπως float, int, str κλπ. Όμως η συνάρτηση range() δημιουργεί λίστα που περιέχει μόνο ακεραίους αριθμούς. Η σύνταξη της range είναι απλή, γράφουμε range και μέσα στις παρενθέσεις βάζουμε τρεις παραμέτρους «αρχή», «τέλος» και «βήμα».

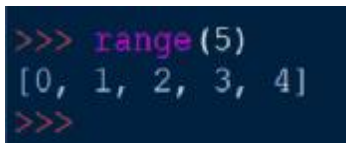
Οι παράμετροι στην παρένθεση πρέπει να είναι ακέραιοι αριθμοί, διαφορετικά η Python θα μας εμφανίσει συντακτικό λάθος. Η παράμετρος που συμβολίσαμε ως «αρχή» δηλώνει τον πρώτο αριθμό της λίστας. Ο τελευταίος αριθμός της λίστας θα είναι μικρότερος από την παράμετρο «τέλος». Το «βήμα» είναι ο αριθμός που προσθέτουμε σε κάθε τιμή για να προκύψει η επόμενη, ξεκινώντας από την παράμετρο «αρχή».

Ας δούμε μερικά παραδείγματα:

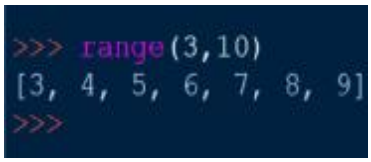
- Το πρώτο είναι απλό, δηλώνουμε για αρχή το 1, για τέλος το 10 και το βήμα 1 και μας εμφανίζει τη λίστα ακέραιων αριθμών από το ένα έως το εννέα. Παρατηρούμε ότι το τελευταίο στοιχείο της λίστας, δηλαδή το 9, είναι μικρότερο της παραμέτρου «τέλος» που είναι το 10. Επίσης, οι αριθμοί μέσα στη λίστα αυξάνονται κατά ένα, όπως το έχουμε δηλώσει με το «βήμα».



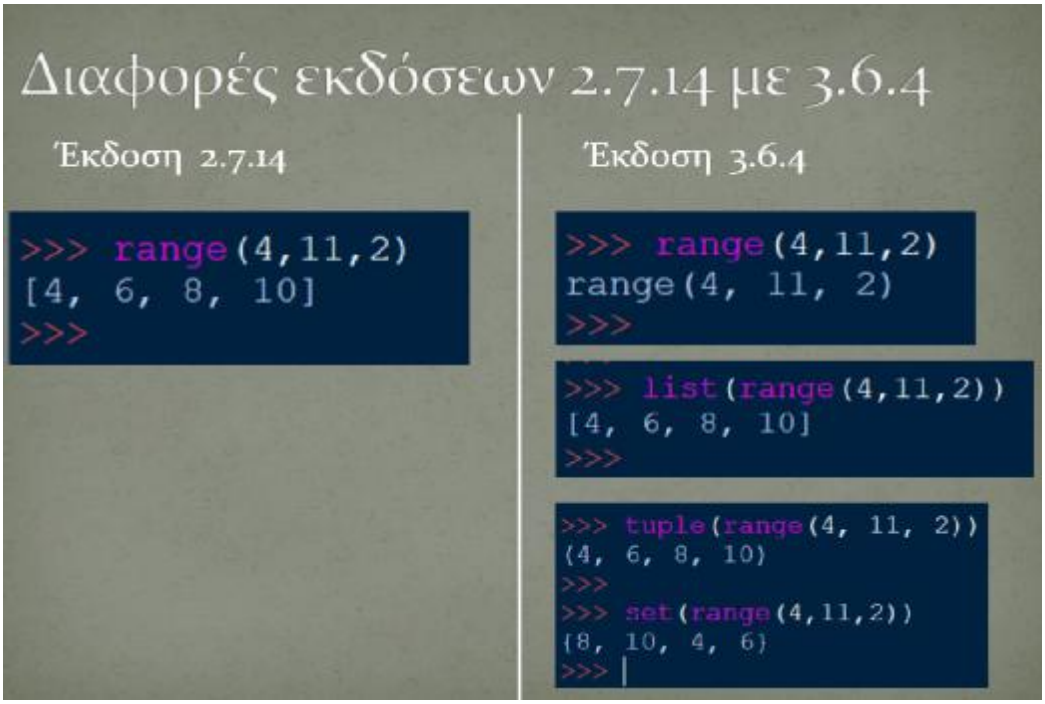
- Στο δεύτερο παράδειγμα, έχουμε δηλώσει μόνο μία παράμετρο, το πέντε. Σε αυτή την περίπτωση η Python αναθέτει αυτόματα το πέντε ως τέλος, ενώ ορίζει ως αρχή το μηδέν και ως βήμα το ένα. Βλέπουμε ότι μας εμφανίζει τους αριθμούς από το μηδέν έως το τέσσερα.



- Στο τελευταίο παράδειγμα έχουμε βάλει δύο παραμέτρους το 3 και το 10. Εδώ, η Python αναθέτει το 3 ως αρχή, το 10 ως τέλος και ορίζει ως βήμα το 1 και εμφανίζει τους αριθμούς από το 3 έως το 9.



Ας δούμε τις διαφορές της έκδοσης 2 με την έκδοση 3, όσον αφορά στη συνάρτηση range(). Στο παρακάτω παράδειγμα, η έκδοση 2.7.14 μας εμφανίζει τη λίστα τιμών ενώ η έκδοση 3.6.4 μας εμφανίζει τη συνάρτηση range με τις παραμέτρους τις. Αν θέλουμε στην έκδοση 3 να μας εμφανίσει τη λίστα τιμών, πρέπει να χρησιμοποιήσουμε τη συνάρτηση list(), όπως βλέπουμε στο παράδειγμα. Αυτό γίνεται γιατί στην έκδοση 3 με τη range() μπορούμε να εμφανίσουμε εκτός από λίστες και πλειάδα και σύνολο με τις συναρτήσεις tuple και set αντίστοιχα. Τις συναρτήσεις αυτές θα τις δούμε αναλυτικά στα μαθήματα που αφορούν στις δομές δεδομένων.



5.1.2. **ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ** for

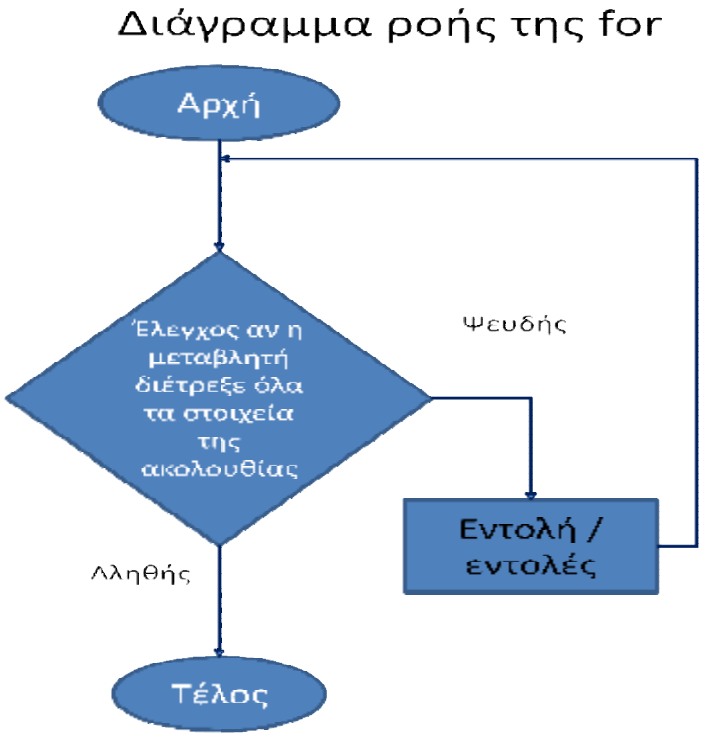
Ένας κώδικας μπορεί να γίνει πιο απλός και πιο εύκολος στη σύνταξή του αν χρησιμοποιήσουμε εντολές επανάληψης. Αυτό σημαίνει ότι κατά την εκτέλεση του προγράμματος μπορούμε να επαναλάβουμε πολλές φορές την εκτέλεση της ίδιας εντολής ή ομάδας εντολών. Μια ομάδα εντολών που επαναλαμβάνονται ονομάζεται βρόχος (loop). Η εντολή for επαναλαμβάνει ένα μπλοκ εντολών για συγκεκριμένο πλήθος φορών. Περιέχει μια μεταβλητή ελέγχου, τον τελεστή in, μια ακολουθία στοιχείων και την άνω κάτω τελεία.

Η σύνταξη της for είναι ως εξής: γράφουμε την εντολή for, τη μεταβλητή, τον τελεστή in, την ακολουθία τιμών και, τέλος, την άνω κάτω τελεία. Η μεταβλητή μπορεί να είναι οποιουδήποτε τύπου, όπως και τα στοιχεία

Σύνταξη της for:

```
for «μεταβλητή» in «ακολουθία»:
    Εντολή /εντολές
```

της ακολουθίας. Αυτό θα το καταλάβουμε καλύτερα με τα παραδείγματα που θα αναφέρουμε στη συνέχεια. Ο τελεστής in ελέγχει αν η μεταβλητή πέρασε από όλα τα στοιχεία της ακολουθίας. Παρακάτω βλέπουμε το διάγραμμα ροής της for. Η διαδικασία ξεκινά με τη μεταβλητή να παίρνει την τιμή του πρώτου στοιχείου της ακολουθίας και εκτελούνται οι εντολές μέσα στη for. Στη συνέχεια, η μεταβλητή παίρνει την τιμή του δεύτερου στοιχείου της ακολουθίας κ.ο.κ., μέχρι να πάρει όλες τις τιμές της ακολουθίας, οπότε και τελειώνει η επανάληψη.



Παραδείγματα

Στο πρώτο παράδειγμα, η ακολουθία είναι μία λίστα από ακέραιους αριθμούς. Παρατηρούμε ότι η εντολή print εκτελείται όσες φορές η μεταβλητή i παίρνει μία τιμή της ακολουθίας και την εμφανίζει.

```
>>> for i in [1,2,3,4,5]:
        print i


1
2
3
4
5
>>>
```

Στο δεύτερο παράδειγμα, η ακολουθία είναι μία συμβολοσειρά. Όπως είπαμε, η ακολουθία μπορεί να είναι οποιουδήποτε τύπου, λίστα, πλειάδα, συμβολοσειρά κλπ. Το πρόγραμμα εμφανίζει τα στοιχεία της ακολουθίας.

```
>>> for i in 'Ενα':
        print i


Ε
ν
α
>>>
```

Στο τελευταίο παράδειγμα, έχουμε ως ακολουθία μία λίστα που τα στοιχεία της είναι διαφορετικού τύπου και αυτό είναι ένα από τα πλεονεκτήματα της Python. Δηλαδή, η μεταβλητή στην εντολή for μπορεί να πάρει τιμή διαφορετικού τύπου σε κάθε επανάληψη. Όπως και στα άλλα παραδείγματα, το πρόγραμμα εμφανίζει τα στοιχεία της ακολουθίας.

```
>>> for i in ['Ενα','Δύο',3,True]:
        print i


Ενα
Δύο
3
True
>>>
```

Είδαμε ως τώρα πώς μπορούμε να χρησιμοποιήσουμε την εντολή for για μικρό πλήθος επαναλήψεων. Τι κάνουμε, όμως, αν ο αριθμός των επαναλήψεων πρέπει να είναι πολύ μεγάλος; Σε αυτή την περίπτωση μας βοηθά η συνάρτηση range() που εξηγήσαμε. Για παράδειγμα, θέλουμε να υπολογίσουμε το άθροισμα των αριθμών από το 1 έως το 100. Δηλώνουμε μια μεταβλητή - αθροιστή a=0 και στην εντολή for χρησιμοποιούμε τη range(101). Μέσα στο βρόχο αυξάνουμε το a με την εντολή a=a+i. Έτσι όταν εκτυπώσουμε το a εκτός βρόχου θα μας εμφανίσει το άθροισμα 5050. Με την ίδια φιλοσοφία βρίσκουμε το γινόμενο των αριθμών 10, 20, 30, 40, 50, 60, 70, 80, 90, 100. Μόνο που εδώ πρέπει να αρχικοποιήσουμε το a σε ένα και όχι σε μηδέν.

```
>>> a=0
>>> for i in range(101):
        a=a+i

>>> print a
5050
>>>
```

```
>>> a=1
>>> for i in range(10,101,10):
        a=a*i


>>> print a
3628800000000000
>>>
```

Χρησιμοποιούμε την εντολή break όταν θέλουμε να "σπάσει" ο κώδικας και να έχουμε άμεση έξοδο από το βρόχο. Η σύνταξή της έχει ως εξής: μέσα στο βρόχο της for χρησιμοποιούμε μια εντολή if με τη συνθήκη της. Αν ισχύει η συνθήκη τότε εκτελούνται οι εντολές της if και εκεί γράφουμε και την break που τερματίζει την επανάληψη. Αν δεν ισχύει η συνθήκη τότε συνεχίζονται κανονικά οι επαναλήψεις της for.

Σύνταξη

```
for μεταβλητή in ακολουθία
    εντολή /εντολές
    if συνθήκη
        εντολή/ εντολές
        break
```

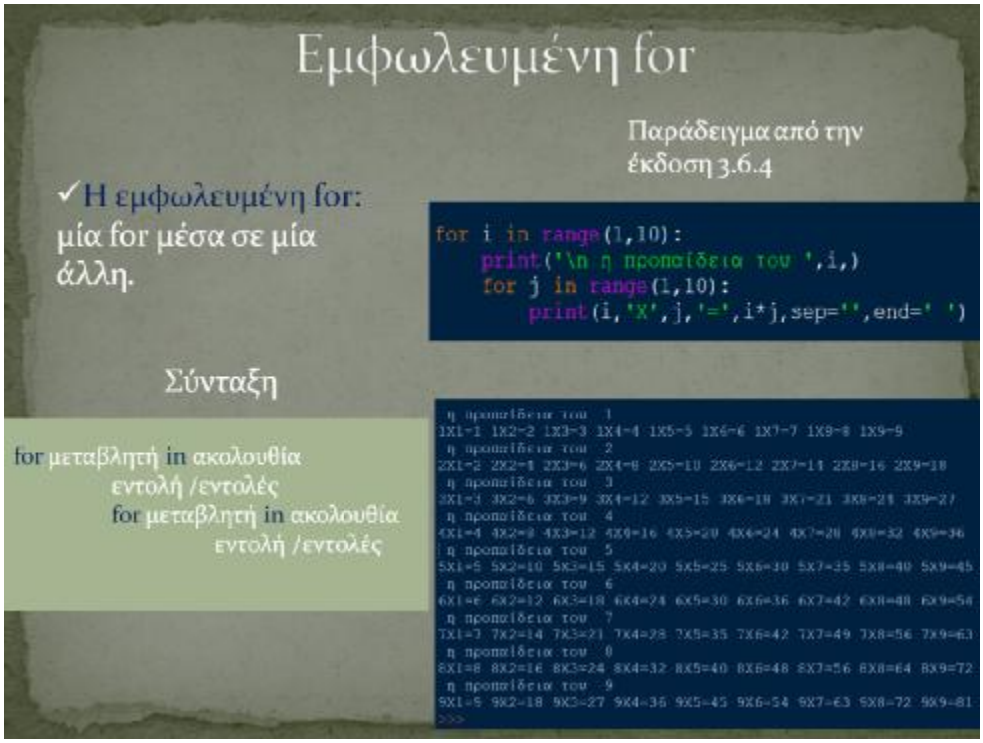
Εκτός της break, έχουμε και την continue που τη χρησιμοποιούμε όταν θέλουμε να διακόψουμε μία η μερικές επαναλήψεις και όχι όλες. Η σύνταξη της continue είναι λίγο πιο πολύπλοκη. Μέσα στο βρόχο της for η

```
for μεταβλητή in ακολουθία
    if συνθήκη
        εντολή/ εντολές
        continue
    εντολή /εντολές
```

πρώτη εντολή που θα χρησιμοποιήσουμε είναι η if με μια συνθήκη και μετά οι υπόλοιπες εντολές. Αν ισχύει η συνθήκη τότε εκτελούνται οι εντολές της if, στις οποίες συμπεριλαμβάνεται και η continue. Μόλις ο κώδικας συναντήσει την continue σταματάει η συγκεκριμένη επανάληψη και η μεταβλητή της for παίρνει την επόμενη τιμή.

Ας δούμε ορισμένα παραδείγματα. Στο πρώτο παράδειγμα θέλουμε να βρούμε το άθροισμα των αριθμών από το 1 έως το 100, αλλά να διακοπεί η επανάληψη μόλις το άθροισμα γίνει μεγαλύτερο από 1000. Δηλώνουμε τη μεταβλητή a=0 που αυξάνεται μέσα στη for με την εντολή a = a + i. Χρησιμοποιώντας την εντολή if, ελέγχουμε αν το a είναι μεγαλύτερο η ίσο του 1000, οπότε εκτυπώνονται το a και το i και τερματίζεται η επανάληψη. Έτσι, μας εμφανίζει το άθροισμα 1035 για τους αριθμούς από το 1 έως το 45. Στο επόμενο παράδειγμα, φαίνεται η χρησιμότητα της εντολής continue. Για τους αριθμούς από το 1321 έως το 1330 θέλουμε να δούμε ποιος είναι περιττός και ποιος άρτιος. Μέσα στη for η συνθήκη της if ελέγχει αν το υπόλοιπο της διαίρεσης i/2 είναι μηδέν οπότε εκτυπώνεται ότι ο αριθμός i είναι άρτιος και ακολουθεί η continue ώστε να μην εκτυπωθεί ότι ο αριθμός i είναι περιττός.

Επίσης, στην εντολή for μπορούμε να έχουμε και έναν βρόχο μέσα σε έναν άλλο βρόχο (εμφωλευμένος βρόχος). Η σύνταξη μιας εμφωλευμένης for είναι απλός. Το μόνο που κάνουμε είναι να γράψουμε την εμφωλευμένη for στην εσοχή της for που χρησιμοποιούμε. Για καλύτερη κατανόηση χρησιμοποιήσαμε ένα παράδειγμα από την έκδοση 3.6.4 της Python που εμφανίζει την προπαίδεια των αριθμών 1 έως 9.



5.2. **ΜΑΘΗΜΑ** 8. **Η ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ** while

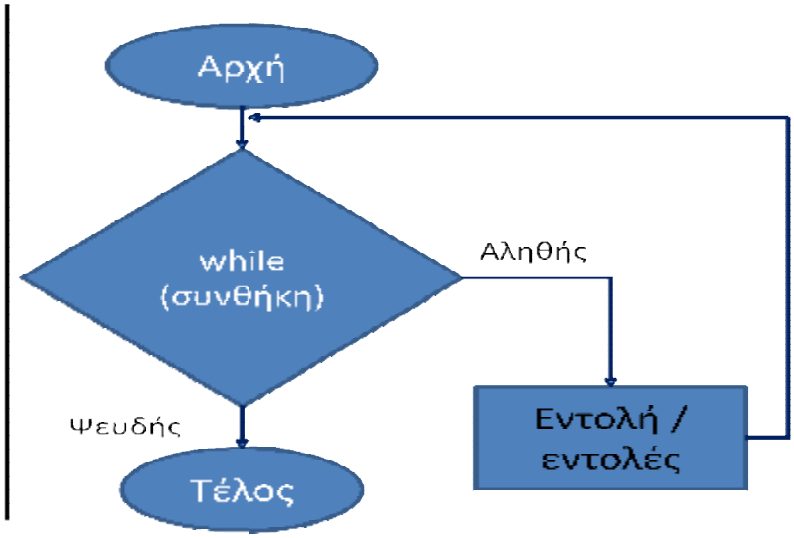
Σύνταξη της while:

```
while «συνθήκη»:
    Εντολή /εντολές
```

Η εντολή επανάληψης while εκτελεί επαναλαμβανόμενα μια ομάδα εντολών, όσο μια συνθήκη που έχουμε ορίσει παραμένει αληθής. Η σύνταξη της εντολής ξεκινά με τη δεσμευμένη λέξη while, μία συνθήκη (που μπορεί να είναι οποιαδήποτε λογική έκφραση) και τελειώνει με άνω κάτω τελεία. Στη συνέχεια ακολουθεί σε εσοχή τεσσάρων κενών μια ομάδα εντολών που εκτελούνται επαναλαμβανόμενα όσο η συνθήκη είναι αληθής. Όταν η συνθήκη γίνει ψευδής, το «σώμα» δεν εκτελείται και η ροή εκτέλεσης των εντολών μεταφέρεται στην πρώτη εντολή μετά το βρόχο. Αν η συνθήκη είναι ψευδής από την πρώτη φορά που ελέγχεται, το «σώμα» της επανάληψης δεν εκτελείται ποτέ.

Στην εντολή while, πρέπει να ορίσουμε μια μεταβλητή πριν ξεκινήσει η εντολή επανάληψης για να τη χρησιμοποιήσουμε στον έλεγχο της συνθήκης. Το «σώμα» του βρόχου θα πρέπει να αλλάζει την τιμή της μεταβλητής κάθε φορά που γίνεται η επανάληψη, ώστε η συνθήκη να γίνεται κάποια στιγμή ψευδής και ο βρόχος να τερματίζεται. Διαφορετικά, ο βρόχος θα επαναλαμβάνεται επ' άπειρο, στην περίπτωση αυτή ονομάζεται ατέρμων βρόχος (infinite loop).

Διάγραμμα ροής της while



Ας δούμε ορισμένα παραδείγματα:

- Στο πρώτο παράδειγμα θέλουμε να εκτυπωθούν οι αριθμοί από το 5 έως το 9. Αρχικοποιούμε τη μεταβλητή i σε 5 πριν την εντολή while και στη συνθήκη της εντολής while γράφουμε i<10 που σημαίνει όσο το i είναι μικρότερο του 10 θα επαναλαμβάνονται οι εντολές print i και i=i+1. Έτσι θα έχουμε την εκτύπωση τον αριθμών 5 έως 9.

```
>>> i=5
>>> while i<10:
        print i
        i=i+1


5
6
7
8
9
>>>
```

- Στο επόμενο παράδειγμα θέλουμε να εκτυπωθούν οι αριθμοί 1, 3, 5, 7, 9. Οι διαφορές με το προηγούμενο παράδειγμα είναι ότι η i παίρνει την τιμή 1 και στο βρόχο η i αυξάνεται κατά 2 δηλαδή το βήμα είναι 2.

```
>>> i=1
>>> while i<10:
        print i
        i=i+2


1
3
5
7
9
>>>
```

- Στο τρίτο παράδειγμα βλέπουμε ότι το βήμα μπορεί να είναι αρνητικό. Εδώ η αρχική τιμή είναι το 100, το βήμα -10 και η τελική τιμή 1. Οι τιμές που θα εμφανιστούν είναι 100, 90 κ.ο.κ. μέχρι και το 10.

```
>>> i=100
>>> while i>1:
        print i
        i=i-10


100
90
80
70
60
50
40
30
20
10
>>>
```

Στην Python, όταν ξέρουμε τον αριθμό τον επαναλήψεων είναι καλύτερο να χρησιμοποιήσουμε τη for. Όμως, αν δεν ξέρουμε τον αριθμό των επαναλήψεων χρησιμοποιούμε πάντα τη while. Στο παρακάτω παράδειγμα θέλουμε να υπολογίσουμε το άθροισμα κάποιων αριθμών που θα μας δώσει ο χρήστης. Δηλώνουμε τη μεταβλητή ath και την αρχικοποιούμε σε μηδέν. Όσο το ath είναι μικρότερο του 50 ζητάμε από το χρήστη να εισάγει έναν ακέραιο αριθμό. Ο βρόχος διακόπτεται όταν το ath πάρει την τιμή 54, την οποία και εκτυπώνουμε.

```
>>> ath=0
>>> while ath<50:
        ath=ath+input('Δώσε έναν ακέραιο αριθμό απο 1 έως 10:')


Δώσε έναν ακέραιο αριθμό απο 1 έως 10:10
Δώσε έναν ακέραιο αριθμό απο 1 έως 10:9
Δώσε έναν ακέραιο αριθμό απο 1 έως 10:8
Δώσε έναν ακέραιο αριθμό απο 1 έως 10:10
Δώσε έναν ακέραιο αριθμό απο 1 έως 10:5
Δώσε έναν ακέραιο αριθμό απο 1 έως 10:6
Δώσε έναν ακέραιο αριθμό απο 1 έως 10:6
>>> print ath
54
>>>
```

Αμυντικός Προγραμματισμός

Ο Αμυντικός Προγραμματισμός είναι μια προγραμματιστική τεχνική με την οποία ελέγχεται η εγκυρότητα των δεδομένων εισόδου. Δηλαδή, όταν ο χρήστης πληκτρολογεί την είσοδο, ελέγχεται αν η τιμή αυτή είναι έγκυρη. Αν δεν είναι έγκυρη, τότε ζητείται από τον χρήστη να την πληκτρολογήσει ξανά. Αν είναι ορθή, τότε προχωράμε στην επεξεργασία των δεδομένων εισόδου.

Παράδειγμα

```
#ζητάμε από τον χρήστη να εισάγει τον μήνα που επιθυμεί
m=input('Εισάγετε μήνα')
#ελέγχουμε αν αυτό που εισήγαγε δεν είναι μεταξύ του 1 και του 12
while (m<1 or m>12):
    print 'Λάθος είσοδος'
    print ' Εισάγετε σωστό μήνα'
    m=input('Εισάγετε μήνα')
```

```
>>> m=input('Εισάγετε μήνα: ')
Εισάγετε μήνα: 25
>>> while m<1 or m>12:
        print 'Λάθος είσοδος'
        print 'Εισάγετε σωστό μήνα'
        m=input('Εισάγετε μήνα: ')


Λάθος είσοδος
Εισάγετε σωστό μήνα
Εισάγετε μήνα: 13
Λάθος είσοδος
Εισάγετε σωστό μήνα
Εισάγετε μήνα: 7
>>>
```

Σε κάποιες γλώσσες προγραμματισμού, υπάρχει άλλη μία εντολή επανάληψης που η Python δεν την διαθέτει. Αυτή είναι η repeat - until που στα Ελληνικά μεταφράζεται επανάλαβε - μέχρις ότου. Παρόλο που η Python δεν έχει

ενσωματωμένη την repeat - until, μπορεί με τη βοήθεια των εντολών while και break να δημιουργήσει αντίστοιχο πρόγραμμα. Να αναφέρουμε πως όπως χρησιμοποιήσαμε τις εντολές break και continue στη for, μπορούμε να τις χρησιμοποιήσουμε και στη while. Για παράδειγμα, ζητάμε από το χρήστη να βρει το αποτέλεσμα μιας παράστασης. Δηλώνουμε ως συνθήκη της while το True που σημαίνει ότι επιτρέπει την είσοδο στο βρόχο χωρίς όρους και ο βρόχος τερματίζεται μόνο όταν ισχύει η συνθήκη της if που περιέχει την break. Δηλαδή, αν ο χρήστης δεν δώσει τη σωστή απάντηση, που είναι το 12, οι εντολές θα επαναλαμβάνονται.

```
>>> while True:
        a=input('Ποιο είναι το αποτέλεσμα της παράστασης((3+5)*3)/2= ')
        if a==12:
                print 'Σωστή απάντηση'
                break
        else:
                print 'Λάθος απάντηση'


Ποιο είναι το αποτέλεσμα της παράστασης((3+5)*3)/2= 13
Λάθος απάντηση
Ποιο είναι το αποτέλεσμα της παράστασης((3+5)*3)/2= 6
Λάθος απάντηση
Ποιο είναι το αποτέλεσμα της παράστασης((3+5)*3)/2= 12
Σωστή απάντηση
>>>
```

Σύγκριση for και while

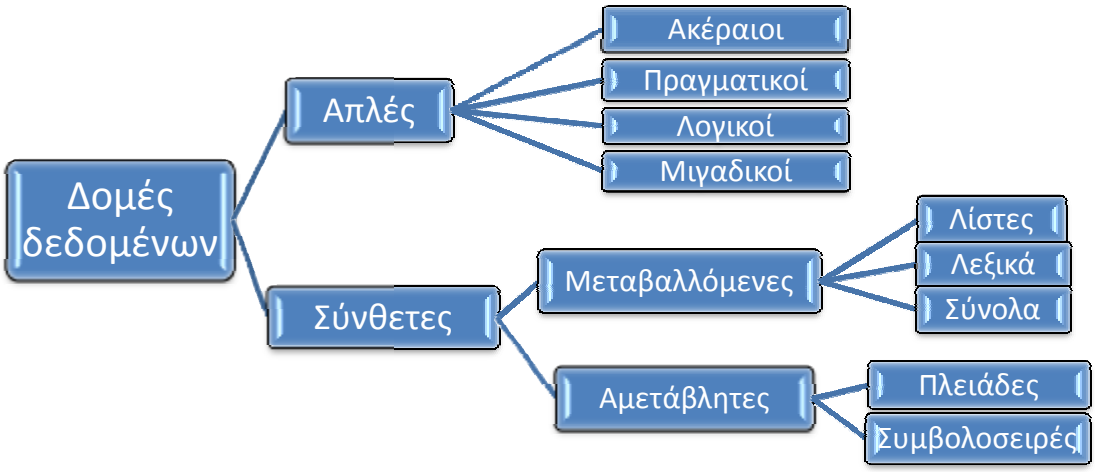
- Στην εντολή for, ο αριθμός των επαναλήψεων είναι προκαθορισμένος, ενώ στην εντολή while δεν είναι πάντα.
- Στην εντολή while υπάρχουν περιπτώσεις που δεν μπορούμε να αποδείξουμε διαβάζοντας απλά τον κώδικα αν θα έχουμε ή όχι ατέρμονα βρόχο κατά την εκτέλεση.
- Συνήθως η while μας δίνει πιο πολύπλοκο κώδικα.

6. ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

6.1. ΜΑΘΗΜΑ 9. ΛΙΣΤΕΣ

6.1.1. ΕΙΣΑΓΩΓΗ ΣΤΙΣ ΛΙΣΤΕΣ

Μία δομή δεδομένων (data structure) είναι ένα σύνολο δεδομένων (τιμών) μαζί με ένα σύνολο επιτρεπτών λειτουργιών (πράξεων) επί αυτών. Οι πιο βασικές λειτουργίες είναι οι ακόλουθες: προσπέλαση, εισαγωγή, διαγραφή, αναζήτηση, ταξινόμηση, συγχώνευση και διαχωρισμός. Τα δεδομένα που χειρίζονται τα διάφορα προγράμματα που αναπτύσσουμε θα πρέπει να είναι οργανωμένα σε δομές δεδομένων. Ουσιαστικά, ένα πρόγραμμα είναι ένα σύνολο αλγορίθμων και δομών δεδομένων. Η Python διαθέτει αρκετές δομές δεδομένων και παρέχει επαρκή υποστήριξη αυτών. Οι δομές δεδομένων χωρίζονται σε απλές και σύνθετες, τις οποίες θα αναπτύξουμε στη συνέχεια. Οι απλές δομές είναι οι τύποι δεδομένων που έχουμε ήδη αναφέρει: ακέραιοι, πραγματικοί, λογικοί, μιγαδικοί. Οι σύνθετες δομές χωρίζονται σε δύο ομάδες, μεταβαλλόμενες και αμετάβλητες. Οι μεταβαλλόμενες δομές, αυτές δηλαδή που αλλάζουν, είναι οι λίστες, τα λεξικά και τα σύνολα. Οι αμετάβλητες δομές είναι οι πλειάδες και οι συμβολοσειρές.



Η πρώτη σύνθετη δομή δεδομένων που θα δούμε είναι οι λίστες. Μία λίστα είναι μία συλλογή τιμών σε σειρά και συμβολίζεται ως list. Οι τιμές που περιλαμβάνονται στη λίστα ονομάζονται στοιχεία (elements), τα οποία αντιστοιχούν σε δείκτες. Τα στοιχεία μιας λίστας δεν χρειάζεται να είναι ίδιου τύπου. Επίσης, ένα στοιχείο μπορεί να υπάρχει περισσότερες από μία φορές σε μία λίστα. Μία λίστα μέσα σε μία άλλη λίστα ονομάζεται εμφωλευμένη λίστα (nested list). Ακόμα, τόσο οι λίστες όσο και οι συμβολοσειρές, που συμπεριφέρονται ως συλλογές τιμών, ονομάζονται ακολουθίες (sequences). Τα στοιχεία μιας λίστας διαχωρίζονται με κόμμα και περικλείονται σε αγκύλες [ ]. Μία λίστα που δεν περιέχει στοιχεία ονομάζεται κενή λίστα.

Ας γράψουμε, για παράδειγμα:

```
>>> mathimata=['Μαθηματικά','Φυσική','Έκθεση']
>>> bathmoi=[5,8.5,7]
>>> emfwleumeni=[5.2,'Χημεία',[5,6,7]]
>>> adeialista=[]
>>> print (mathimata,bathmoi,emfwleumeni,adeialista)
['Μαθηματικά', 'Φυσική', 'Έκθεση'] [5, 8.5, 7] [5.2, 'Χημεία', [5, 6, 7]] []
>>>
```

Αν εκτελέσουμε την εντολή print, θα εμφανιστεί η λίστα mathimata με όλα τα στοιχεία της και με τη σειρά που ορίστηκαν. Έτσι γίνεται και με τις υπόλοιπες λίστες.

```
>>> list(range(11))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list('abcde')
['a', 'b', 'c', 'd', 'e']
>>>
```

Η συνάρτηση list() λαμβάνει μία παράμετρο που πρέπει να είναι ακολουθία και τη μετατρέπει σε λίστα. Για παράδειγμα, η list(range(11)) μετατρέπει την ακολουθία αριθμών από το 0 έως το 10 σε λίστα. Ενώ η list('abcde') μετατρέπει τη συμβολοσειρά 'abcde' σε λίστα των 5 γραμμάτων που απαρτίζουν την ακολουθία.

Κάθε ακέραια έκφραση μπορεί να χρησιμοποιηθεί ως δείκτης. Έστω η λίστα n=['a', 'b', 'c', 'd']. Θέτουμε το δείκτη του πρώτου αριθμού στη λίστα σε 0 (Προσοχή! ΟΧΙ σε 1) και συνεχίζουμε μέχρι να αριθμήσουμε όλους τους δείκτες. Αν θέλουμε να μετρήσουμε την απόσταση από το τέλος θέτουμε το τελευταίο στοιχείο της λίστας σε -1, το προτελευταίο σε -2 κ.ο.κ. Στην επάνω γραμμή του πίνακα, βλέπουμε τους θετικούς δείκτες της λίστας, στη μεσαία γραμμή τα στοιχεία της λίστας και κάτω τους αρνητικούς δείκτες. Έτσι εκπροσωπείται το κάθε στοιχείο χωρίς να ξέρουμε απαραίτητα την τιμή του. Σημειώνεται ότι δεν χρειάζεται να εκτελεστεί κάποια εντολή για να θέσουμε δείκτες στο πρόγραμμα γιατί αυτό γίνεται αυτόματα.

Θετικοί δείκτες	→			
	0	1	2	3
Στοιχεία	'a'	'b'	'c'	'd'
Αρνητικοί δείκτες	-4	-3	-2	-1
	←			

Έστω, λοιπόν, ότι θέλουμε να εκτυπώσουμε το στοιχείο ‘d’ της λίστας. Ο δείκτης του στοιχείου είναι το 3 ή το -1. Οι εντολές print n[3] και print n[-1] θα εκτυπώσουν το στοιχείο ‘d’.

Παράδειγμα

Εμφάνιση στοιχείου d →

Εμφάνιση στοιχείου a και c →

```
>>> n=['a','b','c','d']
>>> print n[3],n[-1]
d d
>>> print n[0],n[-2]
a c
>>>
```

Εκτός από την προσπέλαση μεμονωμένων στοιχείων, μπορούμε να εμφανίσουμε και τμήμα μιας λίστας.

Δείκτες «+»	0		1		2		3		4		5		6
Στοιχεία		'a'		'b'		'c'		'd'		'e'		'f'	
Δείκτες «-»	-6		-5		-4		-3		-2		-1		-

Μεταβλητή[αρχή : τέλος : βήμα]

Έστω η λίστα n=[‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]. Για να εμφανίσουμε ένα τμήμα λίστας, γράφουμε τη μεταβλητή της λίστας, βάζουμε σε αγκύλες τον δείκτη αρχής, τον δείκτη τέλους και το βήμα και τα διαχωρίζουμε με άνω κάτω τελεία. Ας δούμε ένα παράδειγμα. Το n[0:6:2] εμφανίζει τα στοιχεία από την αρχή έως το τέλος ανά δύο δηλαδή τα ‘a’, ‘c’, ‘e’. Το n[0:6] εμφανίζει όλα τα στοιχεία της λίστας, καθώς η Python ορίζει αυτόματα το βήμα ως 1 όταν αυτό παραλείπεται. Με το n[0:5] εμφανίζονται τα στοιχεία ανάμεσα στους δείκτες 0 έως 4, παρατηρούμε δηλαδή ότι το ‘f’ δεν εμφανίζεται. Αν θέλουμε να εμφανίσουμε μόνο ένα στοιχείο χρησιμοποιούμε απλά τον δείκτη του, που για το ‘f’ είναι το 5. Στη συνέχεια, στο n[:], όπου δεν έχουμε ορίσει αρχή και τέλος, η Python ορίζει αυτόματα ως πρώτο δείκτη την αρχή και ως τελευταίο δείκτη το τέλος. Έτσι, πάλι εμφανίζονται όλα τα στοιχεία του πίνακα. Στην τελευταία περίπτωση εμφανίζονται τα στοιχεία από την αρχή μέχρι το δείκτη 4 (η τιμή στη θέση 4 δεν εκτυπώνεται).

```
>>> n=['a','b','c','d','e','f']
>>> n[0:6:2]
['a', 'c', 'e']
>>> n[0:6]
['a', 'b', 'c', 'd', 'e', 'f']
>>> n[0:5]
['a', 'b', 'c', 'd', 'e']
>>> n[5]
'f'
>>> n[:]
['a', 'b', 'c', 'd', 'e', 'f']
>>> n[:4]
['a', 'b', 'c', 'd']
>>>
```

Αφαίρεση και προσθήκη στοιχείων σε μια λίστα

Μπορούμε να αφαιρέσουμε στοιχεία από μία λίστα εκχωρώντας τους μια άδεια λίστα. Σε αυτή την περίπτωση, θα εκτυπώσει τα στοιχεία ‘a’, ’d’, ’e’ και ’f’, αφού οι χαρακτήρες με δείκτες 1 έως 3 (μη συμπεριλαμβανομένου του 3) αντικαταστάθηκαν με κενό.

```
>>> mylist=['a','b','c','d','e','f']
>>> mylist[1:3]=[]
>>> print mylist
['a', 'd', 'e', 'f']
>>>
```

Παρομοίως, μπορούμε και να προσθέσουμε στοιχεία σε μια λίστα με φέτες από λίστες όπως είδαμε προηγουμένως. Για παράδειγμα:

mylist = [‘a’, ‘b’, ’c’, ’d’]

mylist[4:4]= [1, 2]

print mylist

```
>>> mylist=['a','b','c','d']
>>> mylist[4:4]=[1,2]
>>> print mylist
['a', 'b', 'c', 'd', 1, 2]
>>>
```

Στην εκτύπωση θα εμφανιστούν με την σειρά τα: ‘a’, ‘b’, ‘c’, ‘d’, 1 και 2, αφού προσθέσαμε στο δείκτη 4 δύο στοιχεία και οι επόμενοι δείκτες άλλαξαν.

Επειδή οι λίστες είναι μια μεταβαλλόμενη δομή δεδομένων, μπορούμε να αλλάξουμε κάποιο στοιχείο της λίστας, χωρίς να χρειαστεί να δημιουργήσουμε καινούρια. Αυτό γίνεται βάζοντας στους δείκτες της λίστας που θέλουμε τα στοιχεία που θέλουμε. Στο παράδειγμά μας, μετατρέψαμε τα στοιχεία ‘a’ και ‘b’ της mylist σε 1 και 2.

```
>>> mylist=['a','b','c','d']
>>> mylist[0:2]=[1,2]
>>> print mylist
[1, 2, 'c', 'd']
>>>
```

Προσοχή! Οι ενέργειες n[1:1] = [4, 5] και n[1] = [4, 5] είναι διαφορετικές. Όπως φαίνεται στο παράδειγμα, η πρώτη εισάγει στο δείκτη 1 της λίστας n τα στοιχεία 4 και 5, ενώ η δεύτερη μετατρέπει το στοιχείο με δείκτη 1 σε λίστα [4, 5].

```
>>> n=['a','b','c','d']
>>> n[1:1]=[4,5]
>>> n
['a', 4, 5, 'b', 'c', 'd']
>>> n=['a','b','c','d']
>>> n[1]=[4,5]
>>> n
['a', [4, 5], 'c', 'd']
>>>
```

6.1.2. ΠΕΡΙΣΣΟΤΕΡΑ ΓΙΑ ΤΙΣ ΛΙΣΤΕΣ

Αντικείμενα και ψευδώνυμα

Όλες οι τιμές στην Python αποθηκεύονται στη μνήμη του υπολογιστή με τη μορφή αντικειμένων. Είναι πολύ πιθανό δύο μεταβλητές να έχουν τις ίδιες τιμές, αλλά να αναφέρονται σε διαφορετικά αντικείμενα. Είναι δηλαδή ισότιμες αλλά όχι ταυτόσημες.

Ας δούμε ένα παράδειγμα:

```
>>> list1=['a','b','c']
>>> list2=['a','b','c']
>>> list1==list2
True
>>> list1 is list2
False
>>>
```

Αυτές οι δύο λίστες, list1 και list2, αν και έχουν τις ίδιες τιμές, είναι διαφορετικά αντικείμενα. Αυτό φαίνεται με τους τελεστές is, που εμφανίζει False, και == (σύγκρισης), που εμφανίζει True.

```
>>> list1=['a','b','c']
>>> list2=list1
>>> list1 is list2
True
>>> list1[0]=4
>>> list1
[4, 'b', 'c']
>>> list2
[4, 'b', 'c']
>>>
```

Αυτό θα άλλαζε αν εκχωρούσαμε τη list2 ως list1. Ας δούμε το παράδειγμα: ξέρουμε ότι όταν εκχωρούμε μια μεταβλητή σε μια άλλη, αυτομάτως παίρνουν τις ίδιες τιμές. Άρα, σε αυτή την περίπτωση είναι ίδια αντικείμενα. Εάν μία λίστα έχει δύο ονόματα (όπως στο παράδειγμα), λέμε ότι έχει ψευδώνυμα. Αλλαγές που γίνονται στο ένα ψευδώνυμο επηρεάζουν το άλλο. Δηλαδή, αν αλλάξει μια τιμή στη list1 τότε αυτομάτως θα αλλάξει και στη list2, αφού είναι το ίδιο αντικείμενο.

Λίστες - κλώνοι

Για να τροποποιήσουμε μια λίστα και να κρατήσουμε ένα αντίγραφο αναλλοίωτο την κλωνοποιούμε. Η κλωνοποίηση μιας λίστας γίνεται θέτοντας σε μια άλλη μεταβλητή το όνομά της με την άνω κάτω τελεία μέσα σε αγκύλες. Για παράδειγμα:

```
list1 = ['a', 'b', 'c']
list2 = list1[:]
list1[0]=5
print list1 list2
```

Στην εκτύπωση εμφανίζονται: 5, 'b', 'c' και 'a', 'b', 'c'.

```
>>> list1=['a','b','c']
>>> list2=list1[:]
>>> list1[0]=5
>>> print list1,list2
[5, 'b', 'c'] ['a', 'b', 'c']
>>>
```

Εμφωλευμένες λίστες, πίνακες

Όπως έχουμε δει, μια εμφωλευμένη λίστα είναι μια λίστα που εμφανίζεται ως στοιχείο άλλης λίστας. Μέσω μια εμφωλευμένης λίστας, μπορούμε να αναπαραστήσουμε πολυδιάστατους πίνακες στην Python. Για παράδειγμα, ο δίπλα πίνακας μπορεί να αναπαρασταθεί θέτοντας στη μεταβλητή table=[[1,2,3], [4,5,6], [7,8,9]].

1	2	3
4	5	6
7	8	9

Με την εντολή: print table[0], '\n', table[1], '\n', table[2] θα έχουμε τα ακόλουθα αποτελέσματα.

```
>>> table=[[1,2,3],[4,5,6],[7,8,9]]
>>> print table[0],'\n',table[1],'\n',table[2]
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
>>>
```

Αν θέλουμε να εμφανίσουμε ένα συγκεκριμένο στοιχείο του πίνακα, για παράδειγμα στη γραμμή 2 και στήλη 1, τότε η εντολή εκτύπωσης θα είναι η print table[2][1]. Το [2] δηλαδή ο δείκτης 2 συμβολίζει τη λίστα 7, 8, 9 και ο δείκτης 1 το 8. Δείχνω δηλαδή πρώτα σε ποια εμφωλευμένη λίστα ανήκει ο αριθμός και μετά μέσα από αυτή τη λίστα τον δείκτη του στοιχείου που θέλουμε.

```
>>> table=[[1,2,3],[4,5,6],[7,8,9]]
>>> print table[2][1]
8
>>>
```

Τώρα θα δούμε μερικές συναρτήσεις που μπορούμε να χρησιμοποιήσουμε στις λίστες.

Συνάρτηση	Περιγραφή
list(k)	Μετατρέπει την ακολουθία k σε λίστα.
del(a[x])	Διαγράφει το στοιχείο με δείκτη x από τη λίστα a.
len(a)	Επιστρέφει το πλήθος των στοιχείων της λίστας a.
min(a)	Επιστρέφει το ελάχιστο στοιχείο της λίστας a.
max(a)	Επιστρέφει το μέγιστο στοιχείο της λίστας a.
sum(a)	Επιστρέφει το άθροισμα των στοιχείων της αριθμητικής λίστας a.

Παράδειγμα

```
>>> k='abcdef'
>>> a=list(k)
>>> a
['a', 'b', 'c', 'd', 'e', 'f']
>>> del(a[5])
>>> a
['a', 'b', 'c', 'd', 'e']
>>> len(a)
5
>>> min(a)
'a'
>>> max(a)
'e'
>>> a=[1,2,3,4,5]
>>> sum(a)
15
>>>
```

Εκτός από συναρτήσεις, οι λίστες έχουν και μεθόδους. Μία μέθοδος μοιάζει με συνάρτηση, αλλά διαφέρει ως προς τη σύνταξή της. Η σύνταξη της μεθόδου έχει ως εξής: πρώτα βάζουμε τη μεταβλητή της λίστας, ύστερα μια τελεία που παραπέμπει σε μέθοδο, ακολουθεί το

Σύνταξη μεθόδου:

Μεταβλητή.μέθοδος(παράμετροι)

όνομα της μεθόδου και, τέλος, σε παρένθεση οι παράμετροι. Ας δούμε κάποιες μεθόδους που διαγράφουν στοιχεία από μία λίστα.

Μέθοδοι που διαγράφουν στοιχεία από μία λίστα	
Μέθοδος	Περιγραφή
a.clear()	Διαγράφει όλα τα στοιχεία τις λίστας a.
a.pop(i)	Επιστρέφει το στοιχείο με δείκτη i από τη λίστα a και μετά το διαγράφει.
a.remove(x)	Διαγράφει το στοιχείο με τιμή x από τη λίστα a.

Παράδειγμα.

```
>>> a=[1,2,'x','c']
>>> a.clear()
>>> a
[]
>>> a=[1,2,'x','c']
>>> a.pop(0)
1
>>> a.remove('x')
>>> a
[2, 'c']
>>>
```

Ας δούμε μερικές ακόμα μεθόδους.

Μέθοδος	Περιγραφή
Μέθοδοι που εισάγουν στοιχεία σε λίστες	
a.insert(i,x)	Εισάγει στη λίστα a το αντικείμενο x στη θέση με δείκτη i.
a.append(x)	Εισάγει το αντικείμενο x στο τέλος της λίστας a.
a.extend(a1)	Εισάγει όλα τα στοιχεία της λίστας a1 στο τέλος της λίστας a.
Μέθοδοι που αναδιατάσσουν δεδομένα	
a.reverse()	Αντιστρέφει τη σειρά των στοιχείων της λίστας a.
a.sort()	Ταξινομεί τα στοιχεία της λίστας a.
Μέθοδοι που αντλούν πληροφορίες από λίστες	
a.count(x)	Επιστρέφει το πλήθος των εμφανίσεων του x στη λίστα a.
a.index(x)	Επιστρέφει τον πρώτο δείκτη όπου εμφανίζεται το x στην λίστα a.

Παράδειγμα

```
>>> a=[1,2]
>>> a.insert(0,'x')
>>> a
['x', 1, 2]
>>> a.append('x')
>>> a
['x', 1, 2, 'x']
>>> a1=[3,4]
>>> a.extend(a1)
>>> a
['x', 1, 2, 'x', 3, 4]
>>>
>>> a.reverse()
>>> a
[4, 3, 'x', 2, 1, 'x']
>>> a.sort()
>>> a
[1, 2, 3, 4, 'x', 'x']
>>>
>>> a.count('x')
2
>>> a.index('x')
4
>>>
```

6.2. **ΜΑΘΗΜΑ** 10. **ΠΛΕΙΑΔΕΣ**

6.2.1. **ΠΛΕΙΑΔΕΣ**

Μία πλειάδα (tuple) είναι μια ακολουθία τιμών με συγκεκριμένη σειρά, οι οποίες αντιστοιχίζονται σε δείκτες. Οι τιμές που είναι μέλη μιας πλειάδας ονομάζονται στοιχεία (elements) και μπορεί να είναι οποιουδήποτε τύπου (αριθμοί, συμβολοσειρές, λίστες, πλειάδες). Οι πλειάδες μοιάζουν με τις λίστες στη χρήση δεικτών, στον τρόπο με τον οποίο διατρέχονται και στη χρήση του τελεστή της άνω κάτω τελείας σε αγκύλη. Όμως, οι πλειάδες, όπως και οι συμβολοσειρές, είναι αμετάβλητες. Οι πλειάδες αξιοποιούνται συνήθως στις περιπτώσεις όπου πρόκειται να χρησιμοποιηθεί μια ακολουθία τιμών (πλειάδα) που δεν πρόκειται να αλλάξει. Συντακτικά, μια πλειάδα είναι μια λίστα τιμών που χωρίζονται με κόμμα. Παρ' όλο που δεν είναι αναγκαίο, είναι σύνηθες να περικλείουμε τις πλειάδες με παρενθέσεις, για παράδειγμα: Example1 = (1, 2, 3, 4, 5).

Για να δημιουργήσουμε μια άδεια πλειάδα, χρησιμοποιούμε άδειες παρενθέσεις, όπως στο example2. Για να δημιουργήσουμε πλειάδα με ένα μόνο στοιχείο, πρέπει να προσθέσουμε ένα κόμμα δίπλα στο στοιχείο, όπως στο example3.

```
>>> example1=(1,2,3,4,5)
>>> example2=()
>>> example3=(1,)
>>> print example1, example2, example3
(1, 2, 3, 4, 5) () (1,)
>>>
```

Στο δεύτερο παράδειγμα βλέπουμε ότι μπορούμε να επιλέξουμε ένα στοιχείο από μια πλειάδα χρησιμοποιώντας τους δείκτες. Ο τελεστής της άνω κάτω τελείας επιλέγει διάστημα τιμών, όπως ακριβώς και στις λίστες.

```
>>>
>>> n=('a',1,'b',2,'c',3)
>>> print n[2],n[-3],n[0]
b 2 a
>>> print n[:3],n[-7:-1:3]
('a', 1, 'b') ('a', 2)
>>>
```

Στο τρίτο παράδειγμα, βλέπουμε ότι ο τελεστής in ελέγχει εάν μια τιμή ανήκει σε μια πλειάδα. Οι πλειάδες είναι αμετάβλητες. Αν αλλάξουμε ένα από τα στοιχεία μιας πλειάδας, θα εμφανιστεί μήνυμα λάθους. Για να αλλάξουμε ένα στοιχείο, μπορούμε να αντικαταστήσουμε μια πλειάδα με μια άλλη, όπου θα έχουμε αλλάξει το στοιχείο, αλλά τα υπόλοιπα στοιχεία παραμένουν ίδια, όπως στο τρίτο παράδειγμα.

```
>>> n=('a',1,'b',2,'c',3)
>>> 'a' in n
True
>>> n[0]='A'

Traceback (most recent call la
st):
  File "<pyshell#32>", line 1,
in <module>
    n[0]='A'
TypeError: 'tuple' object does
not support item assignment
>>> n=('A',)+n[1:6]
>>> n
('A', 1, 'b', 2, 'c', 3)
```

Οι πλειάδες υποστηρίζουν δύο πράξεις, την πρόσθεση και τον πολλαπλασιασμό. Η πρόσθεση γίνεται ανάμεσα σε πλειάδες και το αποτέλεσμα είναι η ένωσή τους. Στο παράδειγμα, όπου δημιουργήσαμε την πλειάδα k που είναι η πρόσθεση των πλειάδων n και m, με τη σειρά που τις προσθέσαμε.

```
>>> n=(1,2,3,4,5)
>>> m=('a','b','c')
>>> k=n+m
>>> k
(1, 2, 3, 4, 5, 'a', 'b', 'c')
>>> type(k)
<type 'tuple'>
>>>
```

Ο πολλαπλασιασμός γίνεται ανάμεσα σε μια πλειάδα και σε έναν ακέραιο αριθμό. Το αποτέλεσμα είναι τα στοιχεία της πλειάδας να επαναλαμβάνονται τόσες φορές όσες δηλώνει ο ακέραιος αριθμός. Στο παράδειγμα βλέπουμε ότι τα στοιχεία της πλειάδας m επαναλαμβάνονται δυο φορές στην πλειάδα k μετά την πράξη. Όσον αφορά τις πράξεις ισχύει για τις πλειάδες ότι ισχύει και για τις λίστες.

```
>>> m=('a','b','c')
>>> k=2*m
>>> k
('a', 'b', 'c', 'a', 'b', 'c')
>>>
```

Ας δούμε μερικές βασικές συναρτήσεις και μεθόδους που μπορούμε να χρησιμοποιήσουμε σε πλειάδες. Η tuple() μετατρέπει μια ακολουθία σε πλειάδα.

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, αντίστοιχα
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

Παράδειγμα

```
>>> 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
>>>
```

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 στην πλειάδα a

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

6.2.2. **ΛΕΞΙΚΑ**

Το λεξικό (dictionary) είναι μια δομή δεδομένων στην οποία εκχωρούμε δεδομένα που μπορούμε να ανακτήσουμε με το όνομά τους. Κάθε στοιχείο αποτελείται από ένα ζεύγος "κλειδί : τιμή" (key : value). Το κλειδί είναι το όνομα και χωρίζεται από την τιμή με το σύμβολο της άνω κάτω τελείας. Κάθε κλειδί αντιστοιχεί σε μια τιμή και είναι μοναδικό σε ένα λεξικό. Μπορούμε να χρησιμοποιούμε μόνο αμετάβλητα αντικείμενα (όπως ακέραιους αριθμούς, συμβολοσειρές) για κλειδιά ενός λεξικού, αλλά μπορούμε να έχουμε είτε αμετάβλητα ή μετατρέψιμα αντικείμενα για τις τιμές του. Τα λεξικά είναι μεταβαλλόμενα και μπορούμε εύκολα να προσθέσουμε και να διαγράψουμε στοιχεία. Επιπλέον, ένα λεξικό αποτελεί μια συλλογή από στοιχεία (ζεύγη κλειδιών - τιμών), τα οποία δεν ταξινομούνται με κανέναν τρόπο (απροσδιόριστη σειρά). Δεν υπάρχει η έννοια της θέσης δείκτη και έτσι σε ένα λεξικό δεν μπορούμε να κάνουμε πέρασμα ή να χρησιμοποιήσουμε φέτες.

Για να ορίσουμε ένα λεξικό στην 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:

bathmoi = {‘μαθηματικά’ : 5, ’φυσική’ : 8, ’έκθεση’ : 7}

Αν ζητήσουμε να εκτυπωθεί το λεξικό (με την εντολή print bathmoi), εμφανίζεται ότι υπάρχει μέσα στα άγκιστρα. Από την άλλη, αν θέλουμε να εκτυπωθεί μόνο ένα στοιχείο στο λεξικό, πρέπει να γράψουμε το όνομα της μεταβλητής (π.χ. bathmoi) και σε αγκύλες το κλειδί που θέλουμε από το λεξικό. Για παράδειγμα: print bathmoi['φυσική']

```
>>> Βαθμοί={'μαθηματικά':5,'φυσική':8,'έκθεση':7}
>>> print (Βαθμοί)
{'μαθηματικά': 5, 'φυσική': 8, 'έκθεση': 7}
>>> print(Βαθμοί['φυσική'])
8
>>>
```

Ένας άλλος τρόπος για να δημιουργήσουμε ένα λεξικό είναι να φτιάξουμε ένα άδειο λεξικό και μετά να προσθέσουμε όσα στοιχεία θέλουμε.

```
>>> day = {}
>>> day[1] = 'Δευτέρα'
>>> day[2] = 'Τρίτη'
>>> day[7] = 'Κυριακή'
>>> day
{1: 'Δευτέρα', 2: 'Τρίτη', 7: 'Κυριακή'}
```

```
>>> day={}
>>> day[1]='Δευτέρα'
>>> day[2]='Τρίτη'
>>> day[7]='Κυριακή'
>>> day
{1: 'Δευτέρα', 2: 'Τρίτη', 7: 'Κυριακή'}
>>>
>>> day[1]='Πέμπτη'
>>> day
{1: 'Πέμπτη', 2: 'Τρίτη', 7: 'Κυριακή'}
>>>
```

Επίσης, μπορούμε να αλλάξουμε στοιχεία στα λεξικά ως εξής:

```
>>> day[1] = 'Πέμπτη'
>>> day
{1: 'Πέμπτη', 2: 'Τρίτη', 7: 'Κυριακή'}
```

Να αναφέρουμε ότι τα παραδείγματα αυτά αναφέρονται στην έκδοση 3.6.4.

Οι βασικές συναρτήσεις για τα λεξικά είναι οι παρακάτω:

Συνάρτηση	Περιγραφή
dict(k)	Μετατρέπει την ακολουθία ζευγών k σε λεξικό.
del(a[x])	Διαγράφει το 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.

Παραδείγματα

```
>>> 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]
>>>
```

Βασικές μέθοδοι για λεξικά:

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

Παραδείγματα

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

```
>>> a={1:'a',2:'b'}
>>> a1={3:'c',4:'d'}
2
>>> a.update(a1)
>>> a
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
3
>>> a.items()
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
4
>>> a.keys()
[1, 2, 3, 4]
5
>>> a.values()
['a', 'b', 'c', 'd']
6
>>> n=a.copy()
>>> n
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
>>>
```

6.3. ΜΑΘΗΜΑ 11. ΣΥΝΟΛΑ

Η επόμενη δομή δεδομένων που θα δούμε είναι τα σύνολα (τύπου set). Ένα σύνολο είναι μια ακολουθία τιμών που δεν έχει σημασία η σειρά τους, δηλαδή είναι μη διατεταγμένη. Γι' αυτό το λόγο, στα σύνολα δεν χρησιμοποιούμε τους δείκτες. Οι τιμές που είναι μέλη ενός συνόλου ονομάζονται στοιχεία, όπως και στις άλλες δομές δεδομένων που έχουμε δει. Τα σύνολα είναι μεταβαλλόμενα, δηλαδή μπορούμε να αλλάξουμε ένα ή μερικά από τα στοιχεία τους. Όμως ένα σύνολο δέχεται ως στοιχεία μόνο αμετάβλητα δεδομένα, όπως συμβολοσειρές, αριθμούς ή πλειάδες. Επίσης, τα στοιχεία ενός συνόλου χωρίζονται με κόμμα και περικλείονται σε άγκιστρα. Τέλος κάθε στοιχείο σε ένα σύνολο είναι μοναδικό.

Ας δούμε μερικά παραδείγματα. Στο πρώτο παράδειγμα, δημιουργήσαμε το σύνολο a που περιέχει αριθμούς και συμβολοσειρές. Αν εκτυπώσουμε το a, η σειρά των στοιχείων δεν είναι η ίδια με την οποία τα έχουμε γράψει. Αυτό γίνεται γιατί σημασία δεν έχει η σειρά στα σύνολα, αλλά αν ένα στοιχείο είναι μέρος του συνόλου. Επίσης, μπορούμε να δημιουργήσουμε ένα κενό σύνολο, θέτοντας σε μια μεταβλητή v άγκιστρα και μετά μετατρέποντας τη μεταβλητή αυτή με τη συνάρτηση set() σε σύνολο, διαφορετικά η Python τη δέχεται ως λεξικό.

```
>>> a={45,19,'f',3,'c'}
>>> print a
set([3, 'c', 19, 45, 'f'])
>>>
>>> v={}
>>> v=set(v)
>>> type(a),type(v)
(<type 'set'>, <type 'set'>)
>>>
```

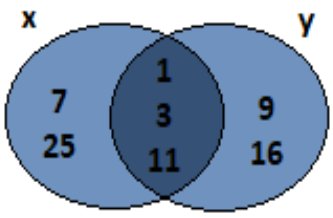
Στο επόμενο παράδειγμα, βλέπουμε πως στα σύνολα τα επαναλαμβανόμενα στοιχεία αφαιρούνται αυτόματα.

```
>>> k={1,'xx',34,'xx',1,(12,),(12,)}
>>> print k
set([1, 'xx', 34, (12,)])
>>>
```

Πράξεις σε σύνολα - Τομή

Τα σύνολα υποστηρίζουν τέσσερις πράξεις. Θα ξεκινήσουμε με την τομή. Η τομή δύο συνόλων είναι τα κοινά τους στοιχεία. Ο τελεστής της τομής είναι το σύμβολο &. Για παράδειγμα, έχουμε τα σύνολα x = 7, 25, 1, 3, 11 και y = 9, 16, 1, 3, 11. Η τομή των x και y είναι τα στοιχεία 11, 1 και 3.

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> x&y
set([11, 1, 3])
>>>
```



Επίσης, θα δούμε ότι για κάθε πράξη υπάρχουν δύο μέθοδοι. Στην προκειμένη περίπτωση έχουμε την intersection και την intersection_update. Χρησιμοποιώντας τα σύνολα x και y, η x.intersection(y) επιστρέφει την τομή των x και y χωρίς να αλλάζει τα σύνολα x και y, ενώ η x.intersection_update(y) μετατρέπει το σύνολο x στην τομή των συνόλων x και y.

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

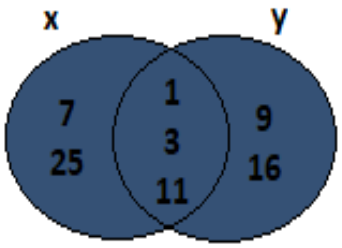
Παράδειγμα

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> k=x.difference(y)
>>> k
set([25, 7])
>>> x.difference_update(y)
>>> x
set([7, 25])
>>>
```

Ένωση

Η επόμενη πράξη είναι η ένωση που περιλαμβάνει όλα τα στοιχεία και των δύο συνόλων. Ο τελεστής της είναι το σύμβολο της καθέτου. Στο παράδειγμα με τα σύνολα x και y, η ένωσή τους είναι τα στοιχεία 1, 3, 7, 9, 11, 16 και 25.

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> x|y
set([1, 3, 7, 9, 11, 16, 25])
>>>
```



Σε αυτή την περίπτωση, οι μέθοδοι που χρησιμοποιούνται είναι η union και η update. Βλέπετε στο παράδειγμα ότι η k = x.union(y) εκχωρεί στη μεταβλητή k την ένωση των στοιχείων x και y, ενώ η x.update(y) μετατρέπει το σύνολο x στην ένωση των συνόλων x και y.

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

Παράδειγμα

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> k=x.union(y)
>>> k
set([1, 3, 7, 9, 11, 16, 25])
>>> x.update(y)
>>> x
set([1, 3, 7, 9, 11, 16, 25])
>>>
```

Διαφορά

Η πράξη της διαφοράς είναι τα μη κοινά στοιχεία του πρώτου συνόλου, δηλαδή του συνόλου που είναι πριν τον τελεστή. Στην περίπτωσή μας, η διαφορά x - y περιλαμβάνει τα στοιχεία του x που δεν υπάρχουν στο y, δηλαδή τα στοιχεία 7 και 25. Ο τελεστής είναι το μείον.

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> x-y
set([25, 7])
>>>
```

Οι μέθοδοι που χρησιμοποιούνται στη διαφορά είναι η difference και η difference_update. Η difference επιστρέφει τη διαφορά του συνόλου που είναι πριν την τελεία και η difference_update μετατρέπει το σύνολο πριν την τελεία στη διαφορά του από το σύνολο που βρίσκεται στην παρένθεση της μεθόδου.

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

Παράδειγμα

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> k=x.difference(y)
>>> k
set([25, 7])
>>> x.difference_update(y)
>>> x
set([7, 25])
>>>
```

Συμμετρική διαφορά

Τέλος, έχουμε την πράξη της συμμετρικής διαφοράς που είναι τα μη κοινά στοιχεία και των δύο συνόλων μαζί. Ο τελεστής είναι το σύμβολο του σημείου εισαγωγής ^. Στο παράδειγμα, τα στοιχεία της συμμετρικής διαφοράς των συνόλων x και y είναι 7, 9, 16 και 25.

```
>>> x={7,25,1,3,11}
>>> y={9,16,1,3,11}
>>> x^y
set([7, 9, 16, 25])
>>>
```

Στη συμμετρική διαφορά έχουμε τις μεθόδους symmetric_difference και symmetric_difference_update. Στο σχετικό παράδειγμα, η πρώτη εκχωρεί στο k τη συμμετρική διαφορά των δύο συνόλων, ενώ η δεύτερη μετατρέπει το σύνολο x στη συμμετρική διαφορά των συνόλων x και y.

Μέθοδοι για τη συμμετρική διαφορά συνόλων	
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 στη συμμετρική διαφορά των συνόλων x και y.

Παράδειγμα

```
>>> x,y={7,25,1,3,11},{9,16,1,3,11}
>>> k=x.symmetric_difference(y)
>>> x.symmetric_difference_update(y)
>>> k,x
(set([7, 9, 16, 25]), set([7, 9, 16, 25]))
>>>
```

Υποσύνολα και τελεστές σύγκρισης

```
>>> x={1,2,3,4,5,6}
>>> y={1,2,3,4}
>>> x>=y
True
>>> x<y
False
>>> x==y
False
>>>
```

Ας δούμε μερικά ακόμα στοιχεία για τα σύνολα. Στα σύνολα υπάρχει η έννοια του υποσυνόλου, για παράδειγμα ένα σύνολο y λέμε ότι είναι υποσύνολο ενός άλλου συνόλου x όταν κάθε στοιχείο του y υπάρχει και στο x. Το αντίθετο είναι το υπερσύνολο. Για να δούμε αν ένα σύνολο είναι υποσύνολο κάποιου άλλου, χρησιμοποιούμε τους τελεστές ανισότητας. Στο παράδειγμα, έχουμε τα σύνολα x και y, όπου το x είναι υπερσύνολο του y. Θα μπορούσαμε να πούμε επίσης ότι το y είναι υποσύνολο του x. Η παράσταση x>=y εμφανίζει True, οι παραστάσεις x<y και x==y εμφανίζουν False.

Χρησιμοποιούνται και εδώ μέθοδοι. Η πρώτη είναι η x.issubset(y) που επιστρέφει True αν το x είναι υποσύνολο του y, διαφορετικά επιστρέφει False. Η δεύτερη είναι η μέθοδος x.issuperset(n) που επιστρέφει True αν το x είναι υπερσύνολο του n, διαφορετικά επιστρέφει False και η τελευταία είναι x.isdisjoint(m) που επιστρέφει True αν τα σύνολα x και m δεν έχουν κανένα κοινό στοιχείο, διαφορετικά επιστρέφει False. Στην εικόνα που ακολουθεί βλέπουμε τις μεθόδους και ένα παράδειγμα για κάθε μέθοδο. Το σύνολο x είναι υποσύνολο του y, επίσης είναι υπερσύνολο του n και δεν έχει κανένα κοινό στοιχείο με το σύνολο m.

Παράδειγμα

Μέθοδοι για υποσύνολα	
x.issubset(y)	Επιστρέφει True, αν το x είναι υποσύνολο του y, διαφορετικά False.
x.issuperset(n)	Επιστρέφει True, αν το x είναι υπερσύνολο του n, διαφορετικά False.
x.isdisjoint(m)	Επιστρέφει True, αν τα σύνολα x και m δεν έχουν κανένα κοινό στοιχείο, διαφορετικά False.

```
>>> x={1,2,3,4}
>>> y={1,2,3,4,5}
>>> n={1,2,3}
>>> m={6,7,'a'}
>>> x.issubset(y)
True
>>> x.issuperset(n)
True
>>> x.isdisjoint(m)
True
>>>
```

Τέλος, όπως σε κάθε ακολουθία δεδομένων, θα δούμε τις συναρτήσεις και τις μεθόδους για τα σύνολα.

Παραδείγματα

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

```
>>> k='abcde'
>>> a=set(k)
>>> a
set(['a', 'c', 'b', 'e', 'd'])
>>> min(a), max(a), len(a)
('a', 'e', 5)
>>>
```

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 από το σύνολο a το στοιχείο b. Αν δεν υπάρχει εμφανίζει μήνυμα σφάλματος.
a.discard(b)	Αφαιρει από το σύνολο a το στοιχείο b. Αν δεν υπάρχει δεν εμφανίζει μήνυμα σφάλματος.

```
>>> 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])
>>>
```

7. ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

7.1. ΜΑΘΗΜΑ 12. ΣΥΜΒΟΛΟΣΕΙΡΕΣ

Η τελευταία σύνθετη δομή δεδομένων που θα δούμε είναι οι συμβολοσειρές. Μια συμβολοσειρά είναι μια ακολουθία χαρακτήρων σε σειρά και συμβολίζεται με το σύμβολο str. Οι χαρακτήρες μιας συμβολοσειράς, όπως και στις άλλες δομές δεδομένων, ονομάζονται στοιχεία και μπορεί να είναι αγγλικά γράμματα, αριθμοί, σύμβολα, ακόμα και γράμματα άλλης γλώσσας. Επίσης, τα στοιχεία μιας συμβολοσειράς περικλείονται από μονά ή διπλά ή τριπλά εισαγωγικά. Όπως και στις λίστες τα στοιχεία μιας συμβολοσειράς έχουν δείκτες. Η συμβολοσειρά είναι δομή δεδομένων αμετάβλητου τύπου, δεν μπορούμε δηλαδή να τη μετατρέψουμε ή να αλλάξουμε τη σειρά των στοιχείων τους.

Ο ρόλος των μονών η διπλών εισαγωγικών είναι για να εκτυπώνονται είτε τα μονά είτε τα διπλά εισαγωγικά κατά περίπτωση. Στο παράδειγμα, έχει εκτυπωθεί το hello world και η λέξη world είναι σε διπλά εισαγωγικά. Στο επόμενο παράδειγμα, η λέξη world είναι σε μονά εισαγωγικά. Τα τριπλά εισαγωγικά, δηλαδή τρεις φορές μονά ή τρεις φορές διπλά, χρησιμεύουν στο να μπορούμε να αλλάζουμε γραμμή με enter όταν γράφουμε τη συμβολοσειρά. Η εκτύπωση θα γίνει σε όσες γραμμές έχουμε γράψει τη συμβολοσειρά. Στο παράδειγμα, υπάρχει μια μεταβλητή a που έχει δηλωθεί ως συμβολοσειρά με τριπλά εισαγωγικά. Μόλις ξεκινά, έχουμε αλλάξει γραμμή και στη μέση της συμβολοσειράς αλλάζουμε και πάλι γραμμή. Η εκτύπωση της a γίνεται με μια κενή γραμμή στην αρχή και η συμβολοσειρά χωρίζεται σε άλλες δύο γραμμές.

```
>>> print 'Hello "world"!'
Hello "world"!
>>> print "Hello 'world'!"
Hello 'world'!
>>> a='''
Μπορούμε να γράψουμε όλα τα σύμβολα
σε μια συμβολοσειρά. !@#$%^&*()'''
>>> print a

Μπορούμε να γράψουμε όλα τα σύμβολα
σε μια συμβολοσειρά. !@#$%^&*()
>>>
```

Όταν η Python διαβάσει σε μια συμβολοσειρά το σύμβολο της ανάποδης καθέτου, ελέγχει πάντα τον επόμενο χαρακτήρα και εκτελεί την ανάλογη λειτουργία. Μερικές από τις λειτουργίες είναι:

- \n που προκαλεί αλλαγή γραμμής στην εκτύπωση.
- \t που προκαλεί την εκτύπωση του κενού διαστήματος tab.
- Με σκέτο \ μπορούμε να αλλάξουμε γραμμή αν πατήσουμε enter αμέσως μετά, χωρίς να υπάρχει συντακτικό

```
>>> print 'Hello \nworld'
Hello
world
>>> print 'Hello \tworld'
Hello   world
>>> print'Hello \
world'
Hello world
>>> print 'Hello \\ world'
Hello \ world
>>>
```

λάθος, αλλά στην εκτύπωση θα εμφανιστεί ενιαία η συμβολοσειρά (χωρίς αλλαγή γραμμής). Για να γίνει πιο κατανοητό, δείτε στο παράδειγμα το hello world. Παρ' όλο που το γράψαμε σε δύο γραμμές εκτυπώνεται σε μία.

- Τέλος, η διπλή ανάποδη κάθετος εμφανίζει την ίδια την ανάποδη κάθετο.

Δείκτες «+»	0		1		2		3		4		5		6
Στοιχεία		a		b		c		d		e		f	
Δείκτες «-»	-6		-5		-4		-3		-2		-1		-

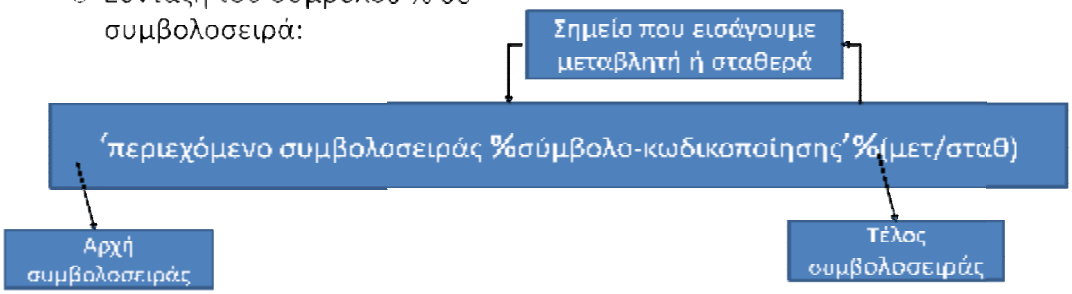
Συνεχίζουμε με τους δείκτες σε μια συμβολοσειρά. Οι δείκτες λειτουργούν ακριβώς το ίδιο όπως και στις άλλες δομές δεδομένων που έχουμε δει. Υπάρχουν θετικοί και αρνητικοί δείκτες, με τη βοήθειά τους και σε συνδυασμό με την άνω κάτω τελεία σε αγκύλες, μπορούμε να εμφανίζουμε ένα μέρος της συμβολοσειράς. Στο διπλανό παράδειγμα, η μεταβλητή x έχει δηλωθεί ως συμβολοσειρά με τιμή 'abcdef'. Το x[:] εμφανίζει όλα τα στοιχεία της συμβολοσειράς, το x[:3] εμφανίζει τα τρία πρώτα στοιχεία. Το x[5] εμφανίζει το στοιχείο με το δείκτη 5, δηλαδή το γράμμα 'f'. Το x[-2:-1] εμφανίζει το στοιχείο που βρίσκεται ανάμεσα στις θέσεις -2 και -1, δηλαδή το γράμμα 'e'. Το x[-4:] εμφανίζει τα στοιχεία από το -4 έως το τέλος. Στις συμβολοσειρές δεν επιτρέπεται να αλλάξουμε κάποιο στοιχείο, όπως κάναμε στις λίστες, γιατί όπως είπαμε είναι αμετάβλητου τύπου.

```
>>> x='abcdef'
>>> x[:]
'abcdef'
>>> x[:3]
'abc'
>>> x[5]
'f'
>>> x[-2:-1]
'e'
>>> x[-4:]
'cdef'
>>> x[1]=j

Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    x[1]=j
NameError: name 'j' is not defined
```

Όταν θέλουμε να εκτυπώσουμε μια συμβολοσειρά, έχουμε τη δυνατότητα να εισάγουμε στο εσωτερικό της μια μεταβλητή ή σταθερά. Σε αυτό μας βοηθά το σύμβολο %. Όταν στο εσωτερικό μιας συμβολοσειράς υπάρχει % ελέγχεται το επόμενο σύμβολο που ακολουθεί και εκτελείται μία λειτουργία. Κάθε λειτουργία κωδικοποιείται από ένα σύμβολο. Ας δούμε αναλυτά τη σύνταξη του συμβόλου %. Ξεκινάμε να γράφουμε τη συμβολοσειρά ανοίγοντας τα εισαγωγικά (μονά ή διπλά). Στο σημείο που θέλουμε να εισάγουμε την τιμή της μεταβλητής ή σταθεράς, βάζουμε το σύμβολο % και αμέσως μετά το σύμβολο κωδικοποίησης που μας δείχνει τον τύπο της μεταβλητής ή σταθεράς. Η συμβολοσειρά τελειώνει κλείνοντας τα εισαγωγικά, αμέσως μετά γράφουμε πάλι το σύμβολο % και μέσα σε παρένθεση βάζουμε τη μεταβλητή ή σταθερά που θέλουμε να εκτυπωθεί. Θα γίνει πιο κατανοητό με τα παραδείγματα που θα δούμε.

- Σύνταξη του σύμβολου % σε συμβολοσειρά:



Ας δούμε τα πιο βασικά σύμβολα κωδικοποίησης. Ξεκινάμε με αυτά που εισάγουν μεταβλητές ή σταθερές που έχουν μόνο αριθμητικές τιμές. Το σύμβολο f ή F εισάγει πραγματικούς αριθμούς. Όπως βλέπετε στο παράδειγμα που ακολουθεί, έχουμε δηλώσει τη μεταβλητή p = 3.14 και την εκτυπώνουμε στην επόμενη γραμμή μέσω συμβολοσειράς. Τα σύμβολα i, d και u εισάγουν μόνο ακέραιους αριθμούς. Στο παράδειγμα εκτυπώσαμε πάλι μέσω συμβολοσειράς τη μεταβλητή p αλλά με την κωδικοποίηση i, γι' αυτό εμφανίζεται μόνο το ακέραιο μέρος, δηλαδή το 3. Στην επόμενη γραμμή εκτυπώσαμε τους αριθμούς 12,5 και 100 ταυτόχρονα, χωρίζοντας τους με κόμμα και χρησιμοποιώντας δύο φορές το σύμβολο % με την κωδικοποίηση d και u, αντίστοιχα. Το σύμβολο o μετατρέπει τον αριθμό σε οκταδικό ενώ τα σύμβολα X κεφαλαίο και x πεζό μετατρέπουν τον αριθμό σε δεκαεξαδικό. Στο δεύτερο παράδειγμα βλέπουμε πώς λειτουργούν τα σύμβολα o, X κεφαλαίο και x πεζό. Παρατηρούμε ότι με το X κεφαλαίο τα γράμματα του δεκαεξαδικού συστήματος εμφανίζονται με κεφαλαία ενώ με το x πεζό τα γράμματα του δεκαεξαδικού συστήματος εμφανίζονται πεζά.

Παραδείγματα

Εισαγωγή μεταβλητής η σταθεράς μόνο για αριθμητικές τιμές	
Σύμβολο κωδικοποίησης	**Περιγραφή λειτουργίας**
f, F	Εισαγωγή πραγματικού αριθμού
i, d, u	Εισαγωγή ακέραιου αριθμού
o	Μετατρέπει τον αριθμό στο οκταδικό σύστημα
X, x	Μετατρέπει τον αριθμό στο δεκαεξαδικό σύστημα

```
>>> p=3.14
>>> print 'Πραγματικός %f'%(p)
Πραγματικός 3.140000
>>> print 'Ακέραιος: %i'%(p)
Ακέραιος: 3
>>> print'Ακέραιοι:%d,%u'%(12.5,100)
Ακέραιοι:12,100
>>>
```

```
>>> a='''ο αριθμός 9 σε οκταδικό
σύστημα είναι: %o'''%(9)
>>> print a
ο αριθμός 9 σε οκταδικό
σύστημα είναι: 11
>>> b='''ο αριθμός 10 σε
δεκαεξαδικό σύστημα
είναι:%X ή %x'''%(10,10)
>>> print b
ο αριθμός 10 σε
δεκαεξαδικό σύστημα
είναι:A ή a
>>> print
```

Συνεχίζουμε με το σύμβολο s που δέχεται στην παρένθεση οποιαδήποτε τιμή και τη μετατρέπει σε συμβολοσειρά μέσω την συνάρτησης string(). Στο παράδειγμα που ακολουθεί, έχουμε τη μεταβλητή z, τύπου λίστας, και την έχουμε καταχωρίσει στη μεταβλητή a που είναι μια συμβολοσειρά μαζί με τον αριθμό 25. Μπορούμε να εκτυπώσουμε ολόκληρη τη συμβολοσειρά a ή ένα μέρος της. Παρατηρούμε ότι ακόμα και οι αγκύλες της λίστας ή τα εισαγωγικά είναι στοιχεία της συμβολοσειράς. Το τελευταίο σύμβολο που θα δούμε είναι το c. Δέχεται στην παρένθεση μόνο ακέραιους αριθμούς που αντιστοιχούν σε ένα σύμβολο, δηλαδή κωδικούς ASCII συμβόλων, και εμφανίζει τα σύμβολα. Στο παράδειγμα εμφανίσαμε τη λέξη 'Python' χρησιμοποιώντας τους κωδικούς ASCII κάθε χαρακτήρα.

Σύμβολο κωδικοποίησης	Περιγραφή λειτουργίας
s	Μετατρέπει κάθε τιμή σε συμβολοσειρά μέσω της συνάρτησης string()
c	Δέχεται στην παρένθεση μόνο ακέραιους αριθμούς που αντιστοιχούν σε ένα σύμβολο (κωδικοί ASCII) και εμφανίζει το σύμβολο

Παραδείγματα

```
>>> z=[1,'ab']
>>> a='Λίστα: %s Αριθμός: %s'%(z,25)
>>> print a[:]
Λίστα: [1, 'ab'] Αριθμός: 25
>>> print a[:17]
Λίστα: [1, 'ab']
>>> print a[17:]
Αριθμός: 25
>>>
```

```
>>> p='%c%c%c%c%c%c'%(80,121,116,104,111,110)
>>> print p
Python
>>>
```

Στη συνέχεια, θα αναφέρουμε τις βασικές μεθόδους για συμβολοσειρές. Ξεκινάμε με τις μεθόδους που επεξεργάζονται πληροφορίες και επιστρέφουν νέα συμβολοσειρά. Δίπλα υπάρχει και από ένα παράδειγμα.

Μέθοδοι που επεξεργάζονται πληροφορίες και επιστρέφουν νέα συμβολοσειρά	
Μέθοδος	**Περιγραφή**
lower()	Μετατρέπει όλα τα κεφαλαία γράμματα στα αντίστοιχα πεζά.
upper()	Μετατρέπει όλα τα πεζά γράμματα στα αντίστοιχα κεφαλαία.
swapcase()	Μετατρέπει τα πεζά γράμματα σε κεφαλαία και τα κεφαλαία σε πεζά.
title()	Μετατρέπει σε όλες τις λέξεις της συμβολοσειράς το πρώτο γράμμα σε κεφαλαίο και τα άλλα σε πεζά.
capitalize()	Μετατρέπει το πρώτο γράμμα της συμβολοσειράς σε κεφαλαίο και όλα τα άλλα σε πεζά.
lstrip()	Διαγράφει τα κενά που βρίσκονται στο αριστερό μέρος του κειμένου.
rstrip()	Διαγράφει τα κενά που βρίσκονται στο δεξί μέρος του κειμένου.
strip()	Διαγράφει τα κενά που βρίσκονται και δεξιά και αριστερά του κειμένου.

Παραδείγματα

```
>>> a=' Μαθαίνοντας την PYTHON '
>>> a.lower()
' μαθαίνοντας την python '
>>> a.upper()
' ΜΑΘΑΙΝΟΝΤΑΣ ΤΗΝ PYTHON '
>>> a.swapcase()
' μΑΘΑΙΝΟΝΤΑΣ ΤΗΝ python '
>>> a.title()
' Μαθαίνοντας Την Python '
>>> a.capitalize()
' μαθαίνοντας την python '
>>> a.lstrip()
'Μαθαίνοντας την PYTHON '
>>> a.rstrip()
' Μαθαίνοντας την PYTHON'
>>> a.strip()
'Μαθαίνοντας την PYTHON'
>>>
```

Μέθοδοι που επεξεργάζονται πληροφορίες και επιστρέφουν νέα συμβολοσειρά	
Μέθοδος	**Περιγραφή**
zfill(n)	Επιστρέφει το κείμενο έχοντας προσθέσει n-x μηδενικά στην αρχή του. Το x είναι το πλήθος των στοιχείων του κειμένου.
replace (old,new,n)	Αλλάζει το old με new και το n καθορίζει πόσες αλλαγές θα γίνουν.
split(t,n)	Επιστρέφει σε λίστα τα τμήματα ενός κειμένου. Το t καθορίζει σε ποιο σημείο θα διαχωριστεί σε τμήματα το κείμενο. Το n καθορίζει πόσες φορές θα γίνει ο διαχωρισμός.

```
>>> a='Python'
>>> a.zfill(9)
'000Python'
>>> a.zfill(4)
'Python'
>>>
```

```
>>> a='1+1=2,2+2=4,4+4=8'
>>> a.replace('+','και')
'1και1=2,2και2=4,4και4=8'
>>> a.replace('=','ίσον',1)
'1+1ίσον2,2+2=4,4+4=8'
>>>
```

```
>>> a='1η λέξη -2η λέξη -3η λέξη'
>>> a.split('-')
['1η λέξη ', '2η λέξη ', '3η λέξη']
>>> a.split('-',1)
['1η λέξη ', '2η λέξη -3η λέξη']
>>>
```

Εδώ βλέπουμε τις μεθόδους που επιστρέφουν true η False.

Μέθοδοι που επιστρέφουν True ή False	
Μέθοδος	**Περιγραφή**
islower()	Επιστρέφει True αν όλα τα γράμματα της συμβολοσειράς είναι πεζά.
isupper()	Επιστρέφει True αν όλα τα γράμματα της συμβολοσειράς είναι κεφαλαία.
istitle()	Επιστρέφει True αν η συμβολοσειρά είναι τίτλος (δηλαδή το πρώτο γράμμα κάθε λέξης είναι κεφαλαίο και όλα τα άλλα πεζά).
isalpha()	Επιστρέφει True αν η συμβολοσειρά έχει μόνο γράμματα.
startswith(x)	Επιστέφει True αν η συμβολοσειρά αρχίζει με το κείμενο x.
endswith(x)	Επιστρέφει True αν η συμβολοσειρά τελειώνει με το κείμενο x.

Παραδείγματα

```
>>> a='python'
>>> a.islower()
True
>>> a='PYTHON'
>>> a.isupper()
True
>>> a='Μαθαίνοντας Την Python'
>>> a.istitle()
True
>>> a='Python'
>>> a.isalpha()
True
>>> a.startswith('Py')
True
>>> a.endswith('on')
True
>>>
```

Τέλος, βλέπουμε τις μεθόδους που επιστρέφουν πληροφορίες και τις βασικές συναρτήσεις για συμβολοσειρές.

Μέθοδοι που επιστρέφουν πληροφορίες	
Μέθοδος	**Περιγραφή**
count(x)	Επιστρέφει τον αριθμό των εμφανίσεων του κειμένου x.
find(x)	Επιστρέφει τον πρώτο δείκτη όπου εμφανίζεται το κείμενο x.

Συναρτήσεις για συμβολοσειρές	
Συνάρτηση	**Περιγραφή**
str()	Μετατρέπει μια μεταβλητή ή μια σταθερά σε συμβολοσειρά.
len()	Επιστρέφει το πλήθος των στοιχείων μιας συμβολοσειράς.
ord()	Επιστρέφει τον κωδικό ASCII ενός χαρακτήρα μιας συμβολοσειράς (δέχεται ως παράμετρο μόνο ένα στοιχείο).

Παραδείγματα

```
>>> a='01xxxx'
>>> a.count('x')
4
>>> a.find('x')
2
>>>
```

```
>>> a=str(12345)
>>> a
'12345'
>>> len(a)
5
>>>
```

```
>>> a='Python'
>>> for i in a:
        print( ord(i),end=',')


80,121,116,104,111,110,
>>>
```

7.2. **ΜΑΘΗΜΑ** 13. **ΣΥΝΑΡΤΗΣΕΙΣ**

Στο μάθημα «τύποι δεδομένων» είδαμε τον ορισμό της συνάρτησης και τα πλεονεκτήματά της. Επίσης είδαμε ότι υπάρχουν δύο ειδών συναρτήσεις, οι ενσωματωμένες και αυτές που δημιουργούμε εμείς ως προγραμματιστές. Σε αυτό το μάθημα θα μάθουμε πώς να δημιουργούμε τις δικές μας συναρτήσεις. Συνάρτηση είναι μια ομάδα εντολών που της έχουμε δώσει ένα όνομα και εκτελούμε αυτό το σύνολο εντολών χρησιμοποιώντας το όνομα της συνάρτησης, οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε. Η χρήση του ονόματος της συνάρτησης για να εκτελεστούν οι εντολές της λέγεται κλήση (calling) της συνάρτησης. Οι συναρτήσεις ορίζονται χρησιμοποιώντας τη λέξη κλειδί def, μετά την οποία ακολουθεί ένα όνομα που ταυτοποιεί τη συνάρτηση, κατόπιν ακολουθεί ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν μερικά ονόματα παραμέτρων και η γραμμή τελειώνει με άνω κάτω τελεία (:) που σηματοδοτεί την έναρξη των εντολών. Οι εντολές της συνάρτησης γράφονται με εσοχή, όπως είδαμε και στην εντολή if.

Σύνταξη της def:

```
def όνομα_συνάρτησης( παράμετροι / ορίσματα):
        εντολή /εντολές
        return τιμή
```

Οι παράμετροι ή ορίσματα της συνάρτησης μπαίνουν σε παρένθεση και μέσω αυτών γίνεται η μεταβίβαση πληροφοριών στη συνάρτηση. Δεν είναι αναγκαίο να βάλουμε πάντα παραμέτρους σε μια συνάρτηση, αλλά οι παρενθέσεις είναι υποχρεωτικές. Για παράδειγμα, δημιουργούμε τη συνάρτηση sayhello() χωρίς παραμέτρους που έχει μέσα την εντολή print 'hello world'. Όταν καλέσουμε τη συνάρτηση γράφοντας το όνομά της, θα μας εμφανίσει τη φράση 'hello world'. Ας δούμε τώρα ένα παράδειγμα με παραμέτρους. Δημιουργούμε τη συνάρτηση mo() που υπολογίζει το μέσο όρο τριών αριθμών και τον εμφανίζει. Προσοχή! όταν καλούμε τη συνάρτηση γράφοντας το όνομά της δεν ξεχνάμε να βάλουμε τις παραμέτρους. Στο παράδειγμά μας γράψαμε 5, 9 και 3. Παρατηρούμε ότι οι παράμετροι χωρίζονται με κόμμα, ενώ έχουμε τη δυνατότητα να χρησιμοποιήσουμε το ίδιο όνομα και στη μεταβλητή και στη συνάρτηση.

Παράδειγμα χωρίς παραμέτρους

```
>>> def sayhello():
        print 'Hello World!'


>>> sayhello()
Hello World!
>>>
```

Παράδειγμα με παραμέτρους

```
>>> def mo(a, b, c):
        mo = (a+b+c)/3
        print mo


>>> mo(5,9,3)
5
>>>
```

Υπάρχουν τρεις τρόποι για να καλέσουμε μια συνάρτηση. Ο πρώτος είναι να γράψουμε το όνομά της, όπως είδαμε και στα προηγούμενα παραδείγματα. Ο δεύτερος είναι να αναθέσουμε τη συνάρτηση σε μια μεταβλητή και να εκτυπώσουμε αυτή τη μεταβλητή. Ένας τρίτος τρόπος είναι να εκτυπώσουμε τη συνάρτηση με την εντολή print. Μέσα στη συνάρτηση, οι εντολές εκτελούνται η μία μετά την άλλη. Η εντολή return τερματίζει την εκτέλεση των εντολών και επιστρέφει την τιμή που έχει η παράσταση return. Ας δούμε δύο παραδείγματα, για να εξηγήσουμε καλύτερα την εντολή return.

Δημιουργούμε τη συνάρτηση pol() που πολλαπλασιάζει τις παραμέτρους a και b. Η διαφορά των δύο παραδειγμάτων είναι ότι στο πρώτο εμφανίζει το αποτέλεσμα με την εντολή print, ενώ στο δεύτερο με την εντολή return. Καλούμε τη συνάρτηση και με τους τρεις τρόπους που είδαμε παραπάνω. Όταν καλούμε τη συνάρτηση με το όνομά της, π.χ. pol(7, 8), εμφανίζει το γινόμενο 7 επί 8, δηλαδή 56, και στα δύο παραδείγματα. Επίσης, αν καλέσουμε τη συνάρτηση καταχωρώντας την σε μια μεταβλητή x, βλέπουμε ότι επιστρέφει το γινόμενο των παραμέτρων 3 και 6. Στο παράδειγμα που έχουμε χρησιμοποιήσει την εντολή print, στη συνάρτηση η μεταβλητή x δεν έχει τύπο. Όμως εκεί που χρησιμοποιήσαμε την εντολή return, η μεταβλητή x έχει τύπο και είναι ακέραια (int). Αυτή είναι η διαφορά της return με την print. Η return επιστρέφει μια τιμή ενώ η print απλά την εμφανίζει. Αν καλέσουμε τη συνάρτηση με την εντολή print στο πρώτο παράδειγμα εμφανίζει το

γινόμενο των παραμέτρων μαζί με None στο τέλος, δηλώνοντας ότι η συνάρτηση δεν έχει τιμή. Να αναφέρουμε ότι μια συνάρτηση επιστέφει μόνο μία τιμή, αλλά αυτή η τιμή μπορεί να είναι οποιουδήποτε τύπου.

✓ Χρησιμοποιώντας μια συνάρτηση χωρίς return σε εντολή print.

```
>>> def pol(a,b):
        p=a*b
        print p


>>> pol(7,8)
56
>>> x=pol(3,6)
18
>>> type(x)
<type 'NoneType'>
>>> print pol(5,9)
45
None
>>>
```

✓ Χρησιμοποιώντας την εντολή return σε μια συνάρτηση.

```
>>> def pol(a,b):
        p=a*b
        return p

>>> pol(7,8)
56
>>> x=pol(3,6)
>>> x
18
>>> type(x)
<type 'int'>
>>> print pol(5,9)
45
>>>
```

Η μεταβίβαση πληροφοριών σε μία συνάρτηση μπορεί να γίνει γράφοντας στις παραμέτρους μεταβλητές αντί για σταθερές, όπως έχουμε δει στα προηγούμενα παραδείγματα. Στην Python υπάρχουν δύο τρόποι για τη μεταβίβαση πληροφοριών, η μεταβίβαση τιμών με τιμή, που σημαίνει ότι η τιμή μέσα στη συνάρτηση δεν αλλάζει, και η μεταβίβαση τιμών με αναφορά που σημαίνει ότι η τιμή μέσα στη συνάρτηση αλλάζει. Για να έχουμε μεταβίβαση τιμών με τιμή, χρησιμοποιούμε αμετάβλητες τιμές στις μεταβλητές π.χ. τύπου συμβολοσειράς, πλειάδας ή αριθμών. Αν θέλουμε να έχουμε μεταβίβαση τιμών με αναφορά χρησιμοποιούμε μεταβαλλόμενες τιμές π.χ. τύπου λίστας, συνόλου κλπ.

1. Μεταβίβαση τιμών με τιμή.

Όπως βλέπουμε στο πρώτο παράδειγμα, δημιουργήσαμε τη συνάρτηση al() που αλλάζει τις παραμέτρους a σε a*3 και b σε b+1 και τις επιστρέφει με την εντολή return. Από κάτω υπάρχουν οι μεταβλητές x και y που έχουν πάρει τις τιμές 'k' και 2, αντίστοιχα. Καλούμε τη συνάρτηση al() με τις παραμέτρους x και y και επιστρέφει τη συμβολοσειρά 'kkk' και το 3, αν όμως εκτυπώσουμε τις μεταβλητές εκτός συνάρτησης θα εμφανιστούν οι αρχικές τιμές τους.

```
>>> def al(a,b):
        a*=3
        b=b+1
        return a,b

>>> x,y='k',2
>>> al(x,y)
('kkk', 3)
>>> x,y
('k', 2)
>>>
```

2. Μεταβίβαση τιμών με αναφορά.

```
>>> def al(a,b,c):
        a*=3
        b.append(2)
        c=c+c
        return a,b,c

>>> x,y,z=[1],[2],[3]
>>> al(x,y,z)
([1, 1, 1], [2, 2], [3, 3])
>>> x,y,z
([1, 1, 1], [2, 2], [3])
>>>
```

Στο δεύτερο παράδειγμα, έχουμε επίσης τη συνάρτηση al() που μετατρέπει τις παραμέτρους ως εξής: το a σε a*3, στη λίστα b επισυνάπτει το 2, το c σε c+c, και τις επιστρέφει. Από κάτω έχουμε δηλώσει τις μεταβλητές x, y και z στις λίστες [1], [2] και [3], αντίστοιχα. Αν καλέσουμε τη συνάρτηση al() με τις παραμέτρους x, y, z, θα εμφανιστούν οι λίστες [1,1,1], [2,2], [3,3]. Αν εκτυπώσουμε τις μεταβλητές θα δούμε ότι η x και η y έχουν αλλάξει, ενώ η z κράτησε την αρχική τιμή. Αυτό γίνεται γιατί στη συνάρτηση δεν αλλάξαμε την παράμετρο c, αλλά δημιουργήσαμε μια νέα μεταβλητή. Αυτό θα γίνει πιο κατανοητό στη συνέχεια.

Ας δούμε την εμβέλεια των μεταβλητών εντός και εκτός των συναρτήσεων. Οι μεταβλητές που βρίσκονται μέσα στη συνάρτηση είναι διαφορετικές από αυτές που βρίσκονται στον κύριο κώδικα, ακόμα και αν έχουν το ίδιο όνομα. Η συνάρτηση διαβάζει μεταβλητές δύο ειδών. Τις καθολικές μεταβλητές, αυτές δηλαδή που βρίσκονται στον κύριο κώδικα, και τις τοπικές μεταβλητές, αυτές δηλαδή που βρίσκονται στο σώμα της συνάρτησης. Αν στο σώμα μιας συνάρτησης έχει δημιουργηθεί μια μεταβλητή που υπάρχει και στον κύριο κώδικα, τότε η συνάρτηση τη θεωρεί ως τοπική και την αποθηκεύει σε διαφορετική θέση μνήμης, δηλαδή οι μεταβλητές δεν έχουν καμία σχέση μεταξύ τους. Βλέπετε και στο παράδειγμα, όπου έχουμε τη μεταβλητή x=100. Η συνάρτηση a() εκτυπώνει την τιμή της μεταβλητής x, δηλαδή το 100, γιατί την εκλαμβάνει ως καθολική. Η συνάρτηση b() εκλαμβάνει τη μεταβλητή x ως τοπική και τυπώνει την τιμή της, δηλαδή 200, όμως αυτό δεν επηρεάζει τη μεταβλητή x του κύριου προγράμματος. Το ίδιο ισχύει και για τις παραμέτρους όταν χρησιμοποιούμε μεταβλητές αντί για σταθερές.

Η συνάρτηση a() διαβάζει τη μεταβλητή x ως καθολική. Εκτυπώνει 100.

Η συνάρτηση b() διαβάζει την μεταβλητή x ως τοπική. Εκτυπώνει 200.

```
>>> x=100
>>> def a():
        print x


>>> a()
100
>>> def b():
        x=200
        print x


>>> b()
200
>>> print x
100
>>>
```

Τι γίνεται όμως αν θέλουμε μια μεταβλητή που είναι τοπική να τη μετατρέψουμε σε καθολική; Εδώ χρησιμοποιούμε την εντολή global. Η εντολή global χρησιμοποιείται μέσα στη συνάρτηση και συνήθως στην αρχή. Αν όχι στην αρχή της συνάρτησης, πρέπει οπωσδήποτε να γραφεί πριν τη χρήση της μεταβλητής που θέλουμε να γίνει καθολική. Ας δούμε το παράδειγμα: έχουμε πάλι τη μεταβλητή x=100, η συνάρτηση b() εκτυπώνει τη x και την εκτυπώνει ξανά με την καινούρια τιμή που έχει πάρει και την εκλαμβάνει ως καθολική. Αν δεν είχαμε χρησιμοποιήσει την εντολή global στη συνάρτηση b() τότε θα υπήρχε συντακτικό λάθος αν την καλούσαμε. Επίσης, με τη χρήση της global, αν αλλάξει η μεταβλητή μέσα στη συνάρτηση αλλάζει και εκτός.

```
>>> x=100
>>> def b():
        global x
        print x
        x=200
        print x


>>> b()
100
200
>>> print x
200
>>>
```

Με τη χρήση της global, αν αλλάξει η μεταβλητή μέσα στη συνάρτηση αλλάζει και εκτός.

8. **ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ**

- Wikipedia,(2015). ‘‘Massive Open Online Course’’, Ανακτήθηκε από http://en.wikipedia.org/wiki/Massive open online course
- http://europestarstsmooc.weebly.com/
- ΜΑΝΟΣ ΚΑΦΕΣ: ‘‘ΕΞΕΡΕΥΝΗΣΗ ΤΗΣ Python’’
- https://www.learnpython.org/
- Ένεμαρκ-Κρεμμύδα Ολυμπία, Λάζαρη Αλεξάνδρα: “ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ Ανάπτυξη εκπαιδευτικού υλικού για την υλοποίηση ενός Μαζικού Ανοικτού Διαδικτυακού Μαθήματος (Massive Open Online Course - MOOC) για την εκμάθηση της Γλώσσας Προγραμματισμού Python”
- Ν. Αβούρης, Κ. Σγάρμπας, Β. Παλιουράς, Μ. Κουκιάς: “Εισαγωγή στους υπολογιστές με την γλώσσα Python”
- https://wiki.python.org/moin/BeginnersGuide
- Νικόλαος Α. Αγγελιδάκης: “Εισαγωγή στον προγραμματισμό με την Python”, Ανακτήθηκε από http://aggelid.mysch.gr/pythonbook/
- Κατάταξη των MOOCs: http://www.wiredacademic.com/2013/02/mooc-giants-coursera-edx-add-more-university-partners-expand-abroad/