Κεφάλαιο 3ο

Βασικά στοιχεία γλώσσας προγραμματισμού

Εισαγωγή

Η Python είναι μια σύγχρονη γλώσσα προγραμματισμού υψηλού επιπέδου

- εύκολη στην εκμάθηση και απλή στη χρήση
- διερμηνευόμενη
- δωρεάν αφού πρόκειται για Ελεύθερο Λογισμικό Ανοικτού Κώδικα
- φορητή
- πολύ διαδεδομένη και με μεγάλη υποστήριξη
- χρησιμοποιείται για πάρα πολλές εφαρμογές
- υποστηρίζει διάφορα προγραμματιστικά υποδείγματα -όπως **δομημένο, συναρτησιακό, αντικειμενοστρεφές-** και τεχνικές προγραμματισμού.
- Επεκτάσιμη αφού περιέχει **πλούσιες βιβλιοθήκες** από έτοιμο κώδικα προγραμματισμού και υποστηρίζει τον **αρθρωτό προγραμματισμό**.
  Τις βιβλιοθήκες μπορούμε να τις χρησιμοποιούμε στα προγράμματά μας ακολουθώντας ένα σύνολο απλών κανόνων και εντολών. Ο τρόπος αυτός μας επιτρέπει τη σχετικά εύκολη συγγραφή πιο σύνθετων προγραμμάτων

3.1 Γνωριμία με το ολοκληρωμένο περιβάλλον ανάπτυξης

Η Python μας έρχεται με ένα ολοκληρωμένο περιβάλλον προγραμματισμού (Integrated Development Environment), το IDLE. Το Python Shell το οποίο είναι μέρος του IDLE είναι ο διερμηνευτής της και ουσιαστικά επιτρέπει τη συγγραφή κάθε εντολής ξεχωριστά (η εντολή ακολουθεί μετά το σύμβολο της προτροπής που για την Python είναι τα τρία σύμβολα >>>) και την **άμεση εκτέλεσή** της με τη βοήθεια του διερμηνευτή της γλώσσας.

Tip: για να αντιγράψουμε μία εντολή που εκτελέσαμε κάνουμε κλικ πάνω της, και πατάμε το Enter

Για να γράψουμε κανονικά προγράμματα χρησιμοποιούμε τον Python Editor το οποίο και αυτό είναι μέρος του IDLE : Πηγαίνουμε στο μενού File → New File
Για να αποθηκεύσουμε το πρόγραμμα μας πηγαίνουμε στο μενού File → Save As
Τα προγράμματα της Python έχουν την κατάληψη .py
Για να τρέξουμε ένα πρόγραμμα πηγαίνουμε στο μενού Run → Run Module

Ας δοκιμάσουμε να δημιουργήσουμε το πρώτο μας πρόγραμμα με στόχο να εμφανιστεί στην οθόνη του υπολογιστή το μήνυμα χαιρετισμού: «Χαίρε, κόσμε!». Για το πρόγραμμά μας θα χρησιμοποιήσουμε την εντολή εμφάνισης **print,** μια εντολή εξόδου με τη δυνατότητα να εμφανιστεί στην έξοδο του υπολογιστή (οθόνη) ένα μήνυμα ή το αποτέλεσμα μιας πράξης.

>>> print ‘Χαίρε, κόσμε’ ή
>>> print (‘Χαίρε, κόσμε’)

Ο υπολογιστής θα εμφανίσει στην οθόνη το μήνυμα μας :

Χαίρε, κόσμε!

3.2 Μεταβλητές και τύποι δεδομένων

3.2.1 Τύποι δεδομένων

Οι *τύποι δεδομένων* προσδιορίζουν τον τρόπο παράστασης των δεδομένων εσωτερικά στον υπολογιστή, καθώς και το είδος της επεξεργασίας τους από αυτόν.

Οι χαρακτηριστικοί τύποι δεδομένων στην Python είναι
- ο αριθμητικός,
- ο λογικός (boolean)
- οι συμβολοσειρές ή αλφαριθμητικά (strings)

Αριθμοί

Οι **αριθμοί** στην Python είναι κυρίως τριών τύπων:

α) **ακέραιοι (Integer)**
π.χ. -3 , -2 , -1, 0 , 1 , 2

β) **αριθμοί κινητής υποδιαστολής (floating point)**

π.χ. 3.14, 28.2E-5, όπου το σύμβολο **E** δηλώνει το 10, οπότε το 28.2E-5 ισούται με $28.2 * 10^{-5}$

Παρατηρούμε ότι το δεκαδικό τμήμα διαχωρίζεται με το χαρακτήρα τελεία “.” και όχι το κόμμα “,”.

γ) **μιγαδικοί αριθμοί (complex numbers),** τύπος που δε θα μας απασχολήσουν (-2+3j)

Λογικός τύπος

Ο **λογικός τύπος** (bool) που δέχεται μόνο δύο τιμές, την τιμή **True** (Αληθής) και την τιμή **False** (Ψευδής) και έχει σκοπό την καταγραφή του αποτελέσματος ενός ελέγχου.

Συμβολοσειρές

Οι **συμβολοσειρές** είναι μια ακολουθία από χαρακτήρες.

- Μια συμβολοσειρά μπορεί να αποτελείται από περισσότερες από μία λέξεις. Οι λέξεις μπορούν να είναι σε κάθε γλώσσα που υποστηρίζεται από το πρότυπο Unicode, άρα σε ελληνική, αγγλική κ.ο.κ.
- Μπορούμε να ορίσουμε μια συμβολοσειρά με μονά εισαγωγικά ή με διπλά, αλλά όχι ανάμικτα. Με ότι ξεκινάμε, θα πρέπει πάλι να κλείνουμε.
  'Σήμερα είναι μία ηλιόλουστη μέρα!’
  ή "Σήμερα είναι μία ηλιόλουστη μέρα!"
  Μπορούμε να χρησιμοποιήσουμε ελεύθερα μονά εισαγωγικά μέσα σε διπλά εισαγωγικά και το ανάποδο.
  Χρησιμοποιώντας τριπλά εισαγωγικά μπορούμε να ορίσουμε συμβολοσειρές πολλαπλών γραμμών. Επίσης, μπορούμε να χρησιμοποιήσουμε ελεύθερα μονά και διπλά εισαγωγικά μέσα σε τριπλά εισαγωγικά.

3.3 Αριθμητικές και λογικές πράξεις και εκφράσεις

Χρησιμοποιώντας τιμές κάθε τύπου δεδομένων, μπορούμε να κάνουμε διάφορες πράξεις, χρησιμοποιώντας τους αντίστοιχους τελεστές.
Οι *τελεστές* (operators) είναι σύμβολα ή λέξεις για τη δημιουργία αριθμητικών και λογικών εκφράσεων.

Αριθμητικοί τελεστές

α) **Αριθμητικοί** τελεστές:
Είναι τα σύμβολα που χρησιμοποιούμε για να κάνουμε μαθηματικές πράξεις. Στη γλώσσα Python χρησιμοποιούμε τους παρακάτω βασικούς αριθμητικούς τελεστές:

Αριθμητικός τελεστής	Αριθμητική πράξη
+	Πρόσθεση
-	Αφαίρεση
*****	Πολλαπλασιασμός
/	Διαίρεση
//	Ακέραιο πηλίκο
%	Το υπόλοιπο της ακέραιας διαίρεσης
**	Ύψωση σε δύναμη

Παρατήρηση: Ο τελεστής της διαίρεσης /, με ακέραιους αριθμούς δίνει το ακέραιο πηλίκο.
Ενώ με δεκαδικούς δίνει ως αποτέλεσμα δεκαδικό αριθμό. Οπότε αν θέλουμε να βρούμε το ακέραιο πηλίκο της διαίρεσης δύο δεκαδικών αριθμών χρησιμοποιούμε το σύμβολο της διαίρεσης //.

Ιεραρχία αριθμητικών πράξεων

Σε κάθε έκφραση στην οποία υπάρχουν αριθμητικοί τελεστές ακολουθείται μια προσδιορισμένη ιεραρχία πράξεων, που είναι:
- Ύψωση σε δύναμη.
- Πολλαπλασιασμός, διαίρεση, υπόλοιπο ακέραιας διαίρεσης.
- Πρόσθεση, αφαίρεση.

Παραδείγματα

Αριθμητικές πράξεις	Αποτέλεσμα
2 + 8 * 2	18
2**2 + 4/2 - 3*4	-6
45 / 10	4
45 % 10	5
45.0 / 10	4.5

Αν θέλουμε να αλλάξουμε την ιεραρχία των πράξεων, μπορούμε να χρησιμοποιήσουμε παρενθέσεις.
Για παράδειγμα, στην έκφραση (2+3)*5 θα εκτελεστεί πρώτα η πρόσθεση μέσα στην παρένθεση και μετά, το αποτέλεσμα θα πολλαπλασιαστεί επί 5,
σε αντίθεση με την έκφραση 2+3*5 στην οποία, πρώτα θα γίνει ο πολλαπλασιασμός και μετά η πρόσθεση.

Εξάσκηση

Πράξεις	**Αναμενόμενο αποτέλεσμα**	**Αποτέλεσμα στην οθόνη**
2 + 3		
3560 – 160		
5 * 3 + 2		
5 * (3 + 2)		
5 * (3 + 2) / 10		
2 * 6.0 / 3		
5 ** 2		
2 ** 4		
3*2**3		

Λογικοί τελεστές σύγκρισης

Σχεσιακοί (ή συγκριτικοί) τελεστές: χρησιμοποιούνται για τη σύγκριση δύο τιμών ή μεταβλητών, με το αποτέλεσμα μιας σύγκρισης να είναι είτε **True** (Αληθής) είτε **False** (Ψευδής). Στη γλώσσα Python χρησιμοποιούνται οι παρακάτω βασικοί σχεσιακοί τελεστές:

Λογικός Τελεστής	**Σύγκριση**
==	Ισότητα (Ίσο με)
!=	Ανισότητα (διάφορο από)
<	Μικρότερο
<=	Μικρότερο ή ίσο
>	Μεγαλύτερο
>=	Μεγαλύτερο ή ίσο

Δεν υπάρχει ιεραρχία για τους συγκριτικούς τελεστές

Παραδείγματα

Λογικές εκφράσεις με συγκριτικούς τελεστές	Αποτέλεσμα
23 == 22+1	True
34 != 45	True
56 <= 12	False

Έλεγχοι συμμετοχής

in	Ανήκει σε ένα σύνολο
not in	Δεν βρίσκεται μέσα σε ένα σύνολο

Λογικοί τελεστές πράξεων

Τελεστές λογικών πράξεων: Στις λογικές πράξεις και εκφράσεις χρησιμοποιούνται οι λογικοί τελεστές **not** (ΟΧΙ), **and** (ΚΑΙ), **or** (Ή) με τις ακόλουθες λογικές λειτουργίες:

Λογικός Τελεστής	Λογική Πράξη
not	Άρνηση (ΟΧΙ)
and	Σύζευση ή τομή (ΝΑΙ)
or	Διάζευξη ή ένωση (Ή)

και με ιεραρχία, όπως αναφέρονται στον παραπάνω πίνακα : 1. **not**, 2. **and**, 3. **or**

Το αποτέλεσμα μιας λογικής πράξης είναι **True** (Αληθής) ή **False** (Ψευδής) σύμφωνα με τον παρακάτω πίνακα

P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Παραδείγματα

Λογικές εκφράσεις με τελεστές λογικών πράξεων	Αποτέλεσμα
12 < 11 and 23 > 10	False
12 < 11 or 23 > 10	True
not 56 <= 12	True

Ιεραρχία όλων των πράξεων

Όταν σε μία έκφραση υπάρχει συνδυασμός πράξεων, τότε αυτές εκτελούνται με την παρακάτω σειρά

1. Αριθμητικές πράξεις,
2. Έλεγχοι συμμετοχής,
3. Συγκρίσεις,
4. Λογικές πράξεις.

Προτεραιότητα τελεστών	
Τελεστής	**Περιγραφή**
****	Υψωση σε δύναμη
***, /, //, %**	Πολλαπλασιασμός, Διαίρεση, Ακέραια Διαίρεση, Υπόλοιπο
+, -	Πρόσθεση και αφαίρεση
<, <=, >, >=, !=, ==	Συγκρίσεις
in, not in	Έλεγχοι συμμετοχής
not x	Λογικό ΟΧΙ
and	Λογικό ΚΑΙ
or	Λογικό Ή

Παραδείγματα

Τελεστής	Όνομα	Εξήγηση	Παραδείγματα
+	Συν	Προσθέτει δύο αντικείμενα.	Το 3 + 5 δίνει 8. Το 'a' + 'b' δίνει 'ab'.
-	Μείον	Είτε δίνει έναν αρνητικό αριθμό, ή αφαιρεί έναν αριθμό από έναν άλλο.	Το -5.2 δίνει έναν αρνητικό αριθμό. Το 50 - 24 δίνει 26.
*	Επί	Δίνει το γινόμενο δύο αριθμών ή μιά συμβολοσειρά (string) επαναλαμβανόμενη τόσες φορές.	Το 2 * 3 δίνει 6. Το 'la' * 3 δίνει 'lalala'.
**	Δύναμη	Επιστρέφει το x υψωμένο στη δύναμη y.	Το 3 ** 4 δίνει 81 (δηλαδή 3 * 3 * 3 * 3).
/	Διά	Διαιρεί το x με το y.	Το 4 / 3 δίνει 1.3333333333333333.
//	Διαίρεση στρογγυλοποιημένη προς τα κάτω (Floor Division)	Επιστρέφει τον κοντινότερο (προς τα κάτω) ακέραιο στο πηλίκο.	Το 4 // 3 δίνει 1.
%	Υπόλοιπο	Επιστρέφει το υπόλοιπο της διαίρεσης.	Το 8 % 3 δίνει 2. Το -25.5 % 2.25 δίνει 1.5.
<	Μικρότερο από	Επιστρέφει το αν το x είναι μικρότερο από το y. Όλοι οι τελεστές σύγκρισης επιστρέφουν True (Αληθής) ή False (Ψευδής). Σημειώστε οτι τα ονόματα αυτά ξεκινούν με κεφαλαίο.	Το 5 < 3 δίνει False και το 3 < 5 δίνει True. Οι συγκρίσεις μπορούν να συνδυαστούν αλυσιδωτά κατά βούληση: Το 3 < 5 < 7 δίνει True.
>	Μεγαλύτερο από	Επιστρέφει το αν το x είναι μεγαλύτερο από το y.	Το 5 > 3 επιστρέφει True. Αν και οι δύο τελεστέοι είναι αριθμοί, πρώτα μετατρέπονται σε έναν κοινό τύπο. Αλλιώς, επιστρέφει πάντα False.
<=	Μικρότερο ή ίσο	Επιστρέφει το αν το x είναι μικρότερο από ή ίσο με το y.	Το x = 3; y = 6; x <= y επιστρέφει True.
>=	Μεγαλύτερο ή ίσο	Επιστρέφει το αν το x είναι μεγαλύτερο από ή ίσο με το y.	Το x = 4; y = 3; x >= 3 επιστρέφει True.
==	Ίσο	Συγκρίνει αν τα αντικείμενα είναι ίσα.	Το x = 2; y = 2; x == y επιστέφει True. Το x = 'str'; y = 'stR'; x == y επιστρέφει False. Το x = 'str'; y = 'str'; x == y επιστρέφει True.
!=	Διαφορετικό	Συγκρίνει αν τα αντικείμενα ΔΕΝ είναι ίσα.	Το x = 2; y = 3; x != y επιστρέφει True.
not	Λογικό ΟΧΙ	Αν το x είναι True, επιστρέφει False. Αν το x είναι False, επιστρέφει True.	x = True; not x επιστρέφει False.
and	Λογικό ΚΑΙ	Το x and y επιστρέφει False αν το x είναι False, αλλιώς επιστρέφει υπολογίζει και επιστρέφει την τιμή του y.	x = False; y = True; x and y επιστρέφει False αφού το x είναι False. Σε αυτή την περίπτωση, η Python δε θα ελέγξει την τιμή του y αφού γνωρίζει οτι η αριστερή πλευρά της έκφρασης 'and' είναι False που υποδηλώνει οτι ολόκληρη η έκφραση θα είναι False ανεξάρτητα από τις άλλες τιμές. Αυτή η τεχνική αποκαλείται short-circuit evaluation.
or	Λογικό Ή	Αν το x είναι True, επιστρέφει True, αλλιώς υπολογίζει και επιστρέφει την τιμή του y.	Το x = True; y = False; x or y επιστρέφει True. Η Short-circuit evaluation εφαρμόζεται και εδώ.

3.2.2. Μεταβλητές

Αυτά που είδαμε στις προηγούμενες δραστηριότητες αρκούν για να υλοποιήσουμε απλές εφαρμογές, ακόμα και πολύ απλά παιγνίδια. Ωστόσο αν θέλουμε να δημιουργήσουμε πιο περίπλοκες εφαρμογές θα πρέπει να χρησιμοποιήσουμε μεταβλητές. Αν θέλουμε για παράδειγμα να δημιουργήσουμε ένα παιγνίδι το οποίο θα μετράει το σκορ, πρέπει να το αποθηκεύσουμε κάπου. Θα το αποθηκεύσουμε λοιπόν σε μια μεταβλητή. Μπορούμε να φανταστούμε τις μεταβλητές σαν κουτάκια που έχουν ένα όνομα και μέσα τους βάζουμε μια τιμή. Κάθε μεταβλητή μπορεί να περιέχει μια μόνο τιμή. Αν αλλάξουμε αυτή την τιμή, η προηγούμενη θα χαθεί. Οι τιμές αυτές μπορούν να αλλάξουν, να διαβαστούν, ή να αντιγραφούν.
Οι μεταβλητές είναι συμβολικά ονόματα που αντιστοιχούν σε θέσεις μνήμης του υπολογιστή.
Οι μεταβλητές χρησιμοποιούνται για την αποθήκευση και διαχείριση δεδομένων σε ένα πρόγραμμα (π.χ. σε μια μεταβλητή μπορούμε να κρατάμε το σκορ ενός παιχνιδιού). Έχουν μία συγκεκριμένη τιμή κάθε στιγμή (στην οποία αναφερόμαστε με το όνομα που έχουμε δώσει στη μεταβλητή), η οποία όμως μπορεί να μεταβάλλεται κατά τη διάρκεια εκτέλεσης του προγράμματος μας.

Παρατηρήσεις:

- *Μια μεταβλητή (δηλαδή μια θέση μνήμης) μπορεί να έχει μόνο μια τιμή κάθε φορά.*
- *Η τιμή μιας μεταβλητής είναι η τελευταία που έχουμε εκχωρήσει σε αυτή. Η προηγούμενη τιμή, καθώς και όλες που υπήρχαν πριν από αυτή, χάνονται.*
- *Οι μεταβλητές τοποθετούνται ως ορίσματα σε όλες τις άλλες εντολές.*
- *Μία μεταβλητή μπορεί να δεχτεί ως όρισμα μία άλλη μεταβλητή.*

Αρκετές φορές σε ένα πρόγραμμα απαιτείται να αποθηκεύσουμε προσωρινά κάποια δεδομένα στη μνήμη του υπολογιστή. Για το σκοπό αυτό χρησιμοποιούμε τις **μεταβλητές**.
Οι μεταβλητές στον προγραμματισμό αντιστοιχούν σε μία θέση μνήμης του υπολογιστή. Κάθε φορά στη θέση αυτή μπορεί να αποθηκευτεί μόνο μία τρέχουσα τιμή.
Οι μεταβλητές χρησιμεύουν, ώστε εύκολα να μπορούμε να έχουμε πρόσβαση στο περιεχόμενό τους, που βρίσκεται προσωρινά αποθηκευμένο στη θέση μνήμης του υπολογιστή, που έχει δεσμευτεί για τη μεταβλητή αυτή.
Για τη χρησιμοποίηση μιας μεταβλητής δεν απαιτείται η δήλωσή της, ενώ μπορεί να εκχωρήσουμε διαφορετικούς τύπους τιμών σε μια μεταβλητή κατά τη διάρκεια ενός προγράμματος, όπως ακέραιες τιμές, κινητής υποδιαστολής και συμβολοσειρές.

x = 32

x = 2.14

x= “xaxaxaxa”

x = [1, 42, 8]

Για να χρησιμοποιήσουμε μια μεταβλητή απαιτείται να της δώσουμε ένα όνομα και να της εκχωρήσουμε κάποια τιμή.

Κανόνες για το όνομα μιας μεταβλητής.

Κανόνες για το όνομα μιας μεταβλητής.

- Αποτελείται μόνο από χαρακτήρες του αγγλικούς αλφαβήτου, αριθμητικά ψηφία και τη κάτω παύλα ‘_’
- Δεν επιτρέπεται να ξεκινά με αριθμό ή με κάποιο από τα δεσμευμένα σύμβολα (π.χ. -, +, *, #, &, =, >).
  Επιτρέπεται να ξεκινάει με κάτω παύλα ‘_’, αλλά δεν ενδείκνυται.
- Δεν επιτρέπεται να υπάρχουν κενά.
  Αντί θέλουμε να δώσουμε μία φράση για όνομα. Αντί για κενά συνήθως βάζουμε την κάτω παύλα ‘_’. (πχ emvado_trigonou). Προσοχή η κανονική παύλα - δεν επιτρέπεται όπως είπαμε πριν.
- Υπάρχει διάκριση μεταξύ των πεζών και των κεφαλαίων γραμμάτων.
  Μια άλλη τεχνική για να δίνουμε φράσεις ως ονόματα είναι αντί να τις χωρίζουμε με ‘_’ να ξεκινάμε το πρώτο γράμμα κάθε λέξης με κεφαλαίο γράμμα. (πχ EmvadoTrigonou).
- Δεν πρέπει να χρησιμοποιήσουμε κάποια δεσμευμένη λέξη (π.χ. print, and, not, or)
  Δεν πρέπει λοιπόν να είναι όμοιο με κάποιο όνομα ενσωματωμένης συνάρτησης ή εντολής, όπως τα παρακάτω :

```
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
```

Επίσης

- Συνηθίζεται να χρησιμοποιούμε λατινικούς χαρακτήρες.
- Δίνουμε περιγραφικά ονόματα.

Συνηθίζουμε να δίνουμε ένα όνομα σχετικό με το είδος της μεταβλητής, όπως για παράδειγμα: onoma1, onoma_2, timi, mesi_timi, emvado κ.ά

Παραδείγματα ορθής ονοματοδοσίας αναγνωριστικού: i, __my_name, name_23, a1b2_c3 και random_utf8_characters_ઊძ२ㄆㄘㄩ ڦڪ ༂ཀᥰεζᎶ ﻴﻠﻋﻝ□□ഖഭ.
Παραδείγματα εσφαλμένης ονοματοδοσίας αναγνωριστικού: 2things, this is spaced out, my-name, και "this_is_in_quotes".

Για να *εκχωρήσουμε* μια τιμή σε μια μεταβλητή, χρησιμοποιούμε τον τελεστή ίσον ‘ = ‘, όπως για παράδειγμα a = 125.

Στο αριστερό μέρος δίνουμε το όνομα της μεταβλητής, στη συνέχεια χρησιμοποιούμε τον τελεστή εκχώρησης ‘ = ’ και στο δεξιό μέρος βάζουμε την τιμή, μια άλλη μεταβλητή ή μια έκφραση που έχει ως αποτέλεσμα μια τιμή.

Όταν αναμειγνύουμε ακέραιους με δεκαδικούς αριθμούς σε μία έκφραση, η Python αυτόματα μετατρέπει τους ακέραιους σε δεκαδικούς.

Η **εντολή εκχώρησης** (**assignment statement**) δημιουργεί νέες μεταβλητές και τους δίνει τιμές. Οι μεταβλητές έχουν επίσης τύπους και μπορούμε να ρωτήσουμε την Python ποιοι είναι με την εντολή **type(μεταβλητή**). Ο τύπος μιας μεταβλητής είναι ο τύπος της τιμής στην οποία αναφέρεται. Για να χρησιμοποιήσουμε μια μεταβλητή, χρειάζεται απλά να γνωρίζουμε το όνομά της.

Μία **έκφραση** (**expression**) είναι ένας συνδυασμός από τιμές, μεταβλητές, τελεστές και κλήσεις σε συναρτήσεις. Οι **τελεστές** (**operators**) είναι λειτουργίες που κάνουν κάτι και μπορούν να αναπαρασταθούν με σύμβολα, όπως το ‘ + ’ ή με λέξεις κλειδιά, όπως το **and**. Η αποτίμηση μιας έκφρασης παράγει μία τιμή και αυτός είναι και ο λόγος που μία έκφραση μπορεί να βρίσκεται στο δεξιό μέρος μια εντολής εκχώρησης. Όταν μία μεταβλητή εμφανίζεται σε έκφραση, αντικαθίσταται από την τιμή της, προτού αποτιμηθεί η έκφραση.

Μία εντολή εκχώρησης αποτελείται από τρία μέρη: το αριστερό που υποχρεωτικά πρέπει να είναι μία μεταβλητή, το ίσον ‘ = ’ (τελεστής εκχώρησης) και το δεξιό μέρος που μπορεί να είναι μία τιμή, μία άλλη μεταβλητή ή μία έκφραση που αποτιμάται (υπολογίζεται) σε μία τιμή.

Παρατήρηση : το ίσον ‘ = ‘είναι τελεστής εκχώρησης και όχι ισότητας, και ότι δεν μπορούμε να εκχωρήσουμε μία τιμή σε μία άλλη τιμή.

Παραδείγματα

```
x = 10          # εκχωρεί την ακέραια τιμή 10 στο x
print x         # εμφανίζει το περιεχόμενο της x : 10
x = x + 15      # προσθέτει στο περιεχόμενο της x τον ακέραιο 15 και το
                  # αποτέλεσμα το εκχωρεί ξανά στη x
print x         # εμφανίζει το περιεχόμενο της x : 25
x = x * 0.1     # πολλαπλασιάζει στο νέο περιεχόμενο της x τον αριθμό
                  # κινητής υποδιαστολής 0.1 και το αποτέλεσμα που είναι πλέον
                  # αριθμός κινητής υποδιαστολής το εκχωρεί ξανά στη x.
print x         # εμφανίζει το περιεχόμενο της x : 2.5
print x * 100   # εμφανίζει το περιεχόμενο της x : 250.0
```

```
onoma = “Μυρτώ”   # εκχωρεί την τιμή της συμβολοσειράς Μυρτώ
                    # στη μεταβλητή onoma
print onoma         # εμφανίζει το περιεχόμενο της onoma : Μυρτώ
metavliti1 = metavliti2 = metavliti3 = 15
                    #πολλαπλή εκχώρηση της τιμής 15 σε τρεις μεταβλητές
x, y = 10, 18       # εκχωρεί τη τιμή 10 στη x, τη τιμή 18 στη y
print x,y           # εμφανίζει το περιεχόμενο της x, y : 10 18
x, y, z = 10, 20, “Μάγια”
                # εκχωρεί τη τιμή 10 στη x, τη τιμή 20 στη y και τη τιμή Μάγια στη z
print x,y,z     # εμφανίζει το περιεχόμενο της x, y, z : 10 20 Μάγια
print ‘Καλημέρα ’ + z  εμφανίζει το μήνυμα : Καλημέρα Μάγια
```

Εξάσκηση

Τα παρακάτω προγράμματα κάνουν την ίδια δουλειά. Ποιο προτιμάτε;

```
ores= 35.0
timi = 12.50
synolo = ores * timi
print synolo
```

```
hours = 35.0
rate = 12.50
pay = hours * rate
print pay
```

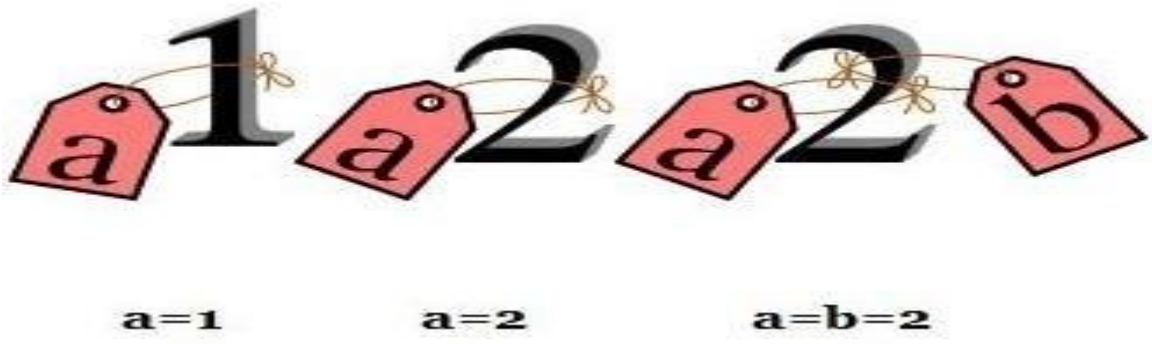
```
x13z9ahd = 35.0
x13z9afd = 12.50
x13p9afd = x13z9ahd * x13z9afd
print x13p9afd
```

```
a = 35.0
b = 12.50
c = a * b
print c
```

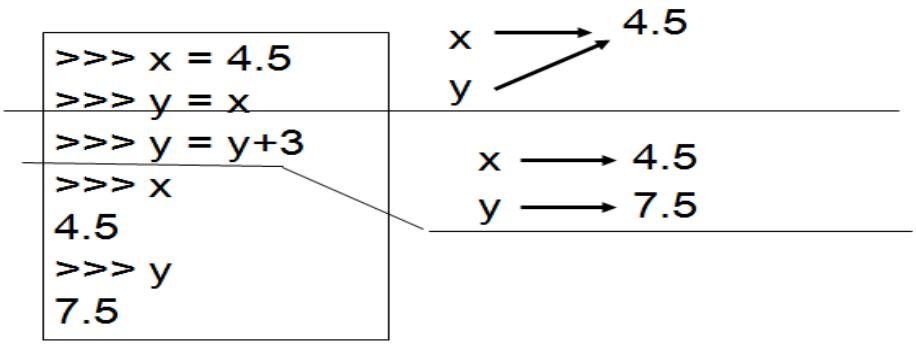
Παρατήρηση

Κατά τη συγγραφή ενός προγράμματος όταν γράφουμε μία εντολή δεν αφήνουμε κανένα κενό στην αρχή της γραμμής. Ξεκινάμε την εντολή από τη πρώτη στήλη.

Οι μεταβλητές στην Python λειτουργούν σαν ετικέτες με κάποιο όνομα, που χρησιμεύουν για να αναφερόμαστε (ή αλλιώς δείχνουν) σε κάποια αντικείμενα.



Παράδειγμα

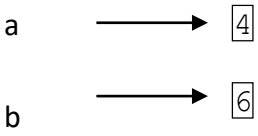


Η Python παρακολουθεί όλες τις τιμές και τις διαγράφει όταν πάψουν να υπάρχουν μεταβλητές που να αναφέρονται σε αυτές.

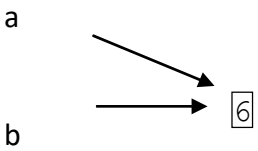
Η διαδικασία αυτή ονομάζεται συλλογή σκουπιδιών **(garbage collection).**

Παραδείγματα

Μετά την εκτέλεση των εντολών a = 4 και b = 6 η μεταβλητή a δείχνει στην τιμή 4 και η μεταβλητή b στην τιμή 6:



αλλά στη συνέχεια μετά την εκτέλεση της εντολής a = b και οι δύο μεταβλητές δείχνουν στην τιμή 6 και η τιμή 4 διαγράφεται αυτόματα από τη μνήμη:



3.3.2 Βασικές εντολές

Εντολή print ()

Η **print** χρησιμοποιείται για την εμφάνιση τιμών στην οθόνη του υπολογιστή, με ποικίλες μορφές, όπως:

print μεταβλητή, π.χ. print name
print αριθμός, π.χ. print 1045.34
print ‘συμβολοσειρά’, π.χ. print 'θαλασσινός αέρας'
print έκφραση, π.χ. print X+5

Παράδειγμα

```
name = ‘kostas’
print name              # εμφανίζει το κείμενο : Kostas
print 1045.34           # εμφανίζει την τιμή : 1045.34
message = "Τώρα "
print message*3         # εμφανίζει το κείμενο : Τώρα Τώρα Τώρα
```

Εντολή input ()

Ένας άλλος τρόπος εκχώρησης της τιμής σε μία μεταβλητή είναι η είσοδος της από το πληκτρολόγιο :
Σύνταξη: **μεταβλητή = input('μήνυμα').**
Εισαγωγή τιμής σε μια μεταβλητή από το πληκτρολόγιο. Το πρόγραμμα αναμένει από το χρήστη να πληκτρολογήσει μια τιμή. Η τιμή που εισάγεται αποδίδεται σε μια μεταβλητή.
Η εντολή αυτή εμφανίζει πρώτα το μήνυμα στην οθόνη και στη συνέχεια περιμένει μέχρι ο χρήστης να πληκτρολογήσει κάποια τιμή. Την τιμή που λαμβάνει από το πληκτρολόγιο την εκχωρεί στη συγκεκριμένη μεταβλητή στο αριστερό μέρος.

Παράδειγμα

```
number = input(' Δώσε έναν αριθμό:')
Δώσε έναν αριθμό: 34      # ο χρήστης δίνει τη τιμή : 34
print number              # εμφανίζει την ακέραια τιμή : 34
```

Η συνάρτηση **input ()** επιστρέφει στην μεταβλητή την τιμή που πληκτρολογήσαμε και αυτόματα λαμβάνει και τον αντίστοιχο τύπο.
Αν θέλουμε να διασφαλίσουμε ότι η είσοδος από το πληκτρολόγιο θα είναι ακέραιος αριθμός μαζί με τη συνάρτηση input () χρησιμοποιούμε και τη συνάρτηση int ()
a=int(input('Δώσε ένα αριθμό :'))

Παράδειγμα

```
a=int(input('Δώσε έναν ακέραιο αριθμό: '))
Δώσε έναν ακέραιο αριθμό: 2345.10  # ο χρήστης δίνει τη τιμή : 2345.10
print a              # εμφανίζει την ακέραια τιμή του αριθμού αποκόπτοντας τα
                         # δεκαδικά ψηφία : 2345
```

Η εντολή αυτή αν χρησιμοποιήσουμε τη σύνταξη μεταβλητή1 **= input** (μεταβλητή2) μας επιστρέφει την τιμή της μεταβλητής2.

Εντολή raw_input ()

Με τον ίδιο τρόπο με την input () αν θέλουμε να εισάγουμε ένα αλφαριθμητικό χρησιμοποιούμε την εντολή **raw_input** ()

Σύνταξη: **μεταβλητή = raw_input('μήνυμα'**).

Γενικά, ότι εισάγεται με τη raw_input θεωρείται αυτόματα αλφαριθμητικό,

Παράδειγμα

```
name = raw_input('Δώσε το όνομά σου : ')
Δώσε το όνομά σου : Kostas        # ο χρήστης πληκτρολογεί  : Kostas
print name                        # εμφανίζει τη συμβολοσειρά : Kostas
tree = raw_input('Δώσε το όνομα ενός δέντρου με άνθη που βλέπεις πηγαίνοντας
                  στο σχολείο: ')
Δώσε το όνομα ενός δέντρου με άνθη που βλέπεις πηγαίνοντας στο σχολείο:
Αμυγδαλιά                         # ο χρήστης πληκτρολογεί  : Αμυγδαλιά
print 'Το δέντρο ', tree, 'είναι όμορφο ανθισμένο'
   # εμφανίζει το κείμενο : Το δέντρο Αμυγδαλιά είναι όμορφο ανθισμένο
```

Tip: Όταν ξεκινάμε τη συγγραφή ενός ολοκληρωμένου προγράμματος στον Editor της Python τοποθετούμε τα δύο παρακάτω σχόλια τα αποκαλούμενα και μαγικά σχόλια :

 # -- coding: cp1253 -*-*

τα οποία αναλαμβάνουν να εξηγήσου στην python ότι υπάρχουν στο πρόγραμμα μας και κείμενα με ελληνικούς χαρακτήρες σύμφωνα με την unicode.

3.3 Βασικές (ενσωματωμένες) συναρτήσεις

Η Python παρέχει μια ποικιλία ενσωματωμένων συναρτήσεων για τη μετατροπή τιμών δεδομένων από έναν τύπο σε έναν άλλο, όπως οι: int(), float() και str().

- Η **float(x)** μετατρέπει ακεραίους και συμβολοσειρές σε δεκαδικούς αριθμούς.
  Για τη μετατροπή αυτή οι συμβολοσειρές θα πρέπει περιέχουν αριθμητικά ψηφία.
- Η **int(x)** δέχεται οποιαδήποτε αριθμητική τιμή και τη μετατρέπει σε ακέραιο κόβοντας τα δεκαδικά ψηφία, αν υπάρχουν.
- Η **str(n)** δέχεται οποιαδήποτε τιμή και την μετατρέπει σε συμβολοσειρά.
- Η **abs(x)** επιστρέφει την απόλυτη τιμή ενός αριθμού.
- Η **pow(x,y)** επιστρέφει τη δύναμη του x υψωμένη στο y.
- Η **divmod(x,y)** επιστρέφει το ακέραιο πηλίκο και το ακέραιο υπόλοιπο της διαίρεσης x/y.
- Η **round(x)** στρογγυλοποιεί δεκαδικούς αριθμούς.

Οι συναρτήσεις αυτές επιστρέφουν μία τιμή, αλλά δεν αλλάζουν τον τύπο μιας μεταβλητής και την τιμή στην οποία αυτή δείχνει. Αν θέλουμε, μπορούμε να εκχωρήσουμε την επιστρεφόμενη τιμή σε μια νέα μεταβλητή.

Παράδειγμα

```
float (10)       → 10.0
int (5.678)      →  5
str (5.678)      → '5.678'
abs (-45)        →  45
divmod (10,3) → 3, 1
pow (2,3)        → 8
```

Έλεγχος τύπου

Για να ελέγξουμε τον **τύπο δεδομένων** (κλάση στην Python) χρησιμοποιούμε την εντολή **type (x)**.

Παράδειγμα

```
>>> type (1)
<type 'int'>
>>> type (3.14)
<type 'float'>
>>> type (False)
<type 'bool'>
>>> type ('Αρετή')
<type 'str'>
>>>
```

Παράδειγμα

```
a = 10              # εκχωρεί την ακέραια τιμή 10 στη μεταβλητή a
type(a)             # εμφανίζει τον τύπο της a :  type 'int'
print a             # εμφανίζει το περιεχόμενο της a : 10
a = 2.14            # εκχωρεί την πραγματική τιμή 2.14 στη a
type(a)             # εμφανίζει τον τύπο της a :  type 'float'
print a             # εμφανίζει το περιεχόμενο της a : 2.14
a = True            # εκχωρεί την λογική τιμή true στη a
type(a)             # εμφανίζει τον τύπο της a :  type 'bool'
print a             # εμφανίζει το περιεχόμενο της a : True
a = 'Καλημέρα'      # εκχωρεί τη συμβολοσειρά τιμή 'Καλημέρα' στη a
type(a)             # εμφανίζει τον τύπο της a :  type 'str'
print a             # εμφανίζει το περιεχόμενο της a : Καλημέρα
```

Εδώ η μεταβλητή a παίρνει ως τιμή διαδοχικά

- ένα ακέραιο,
- ένα πραγματικό,
- μια λογική τιμή και
- μια συμβολοσειρά.

Κάθε φορά ελέγχεται ο τύπος δεδομένων με την εντολή type()

Εισαγωγή σχολίων

Τα σχόλια σε ένα πρόγραμμα διευκολύνουν την κατανόησή του. Στην Python, τα σχόλια εισάγονται θέτοντας μπροστά από αυτά το σύμβολο # .
Τα σχόλια μπορούν να αρχίζουν και μετά από εντολές στη μέση μιας γραμμής.
Ό,τι βρίσκεται δεξιά από το #, αγνοείται από το διερμηνευτή.

Παράδειγμα

```
Πρόγραμμα πολλαπλασιασμός δύο αριθμών
x=input('Δώσε τον πρώτο αριθμό: ')
Δώσε το πρώτο αριθμό: 34              # ο χρήστης πληκτρολογεί την τιμή  : 34
y=input('Δώσε το δεύτερο αριθμό: ')
Δώσε το δεύτερο αριθμό: 2             # ο χρήστης πληκτρολογεί την τιμή  : 2
ginomeno = x * y        # πολλαπλασιάζει τις τιμές των x και y και το αποτέλεσμα
                           # το εκχωρεί στη μεταβλητή ginomeno
print ginomeno          # εμφανίζει την τιμή του ginomenou : 68
```

Tip: Συνηθίζεται πριν αρχίσουμε να γράφουμε ένα πρόγραμμα στην python να ξεκινάμε με ένα σχόλιο με τον τίτλο του προγράμματος.
Π.χ. # Υπολογισμός μέσου όρου Ν αριθμών

Αρθρώματα (Modules)

Ένα **άρθρωμα** (**module**) είναι μία θεματική συλλογή συναρτήσεων. Ένα άρθρωμα αποθηκεύεται σε ένα αρχείο με κατάληξη .py. Η Python διαθέτει πάρα πολλά ενσωματωμένα αρθρώματα.

Math module

Η γλώσσα Python διαθέτει μια μαθηματική μονάδα λογισμικού (math module) η οποία περιέχει τις περισσότερο γνωστές μαθηματικές συναρτήσεις. Μια **μονάδα ή άρθρωμα λογισμικού** (module) είναι ένα αρχείο το οποίο περιέχει μια συλλογή από σχετικές συναρτήσεις.
Για να χρησιμοποιήσουμε ένα άρθρωμα σε ένα πρόγραμμα, θα πρέπει να το εισαγάγουμε με χρήση της εντολής **import άρθρωμα.** Ένα πάρα πολύ χρήσιμο παράδειγμα αρθρώματος είναι το math που περιέχει μαθηματικές συναρτήσεις. Αφού το εισαγάγουμε με την εντολή **import math** για να έχουμε πρόσβαση σε μια από τις ενσωματωμένες συναρτήσεις, θα πρέπει να προσδιορίσουμε το όνομα της μονάδας και το όνομα της συνάρτησης χωρισμένα με μία τελεία. Αυτή η μορφή ονομάζεται **συμβολισμός με τελεία** (dot notation).
Με αυτόν τον τρόπο αφού εισάγουμε το άρθρωμα math μπορούμε να καλέσουμε τις συναρτήσεις του με τη σύνταξη :

- **math.pi** : επιστρέφει το 3,14
- **math.sqrt(x) :** επιστρέφει την τετραγωνική ρίζα ενός αριθμού σε δεκαδική μορφή.
- **math.trunc (x)** : κάνει αποκοπή ενός δεκαδικού αριθμού.

Παράδειγμα

```
import math
riza = math.sqrt(2)
print riza              # εμφανίζει την τιμή : 1.41421356237
math.sqrt(3)            # εμφανίζει την τιμή : 1.7320508075688772
x = math.pi
print x                  # εμφανίζει την τιμή : 3.14159265359
```

Με την εντολή **from math import *** μπορούμε να χρησιμοποιούμε όλες τις μαθηματικές συναρτήσεις χωρίς να αναφέρουμε το άρθρωμα math.

3.5 Δομή προγράμματος και καλές πρακτικές

Μερικές καλές πρακτικές και συντακτικές συμβάσεις που πρέπει να τηρούμε είναι οι παρακάτω:

- Να δίνουμε ένα χαρακτηριστικό τίτλο στο πρόγραμμα με τη μορφή σχολίων, τα οποία ξεκινάνε με το σύμβολο #. Αυτό, παρότι δεν είναι αναγκαίο, είναι ιδιαίτερα χρήσιμο.
- Να προσέχουμε τα κενά διαστήματα πριν την κάθε εντολή, καθώς, όπως θα δούμε στο επόμενο κεφάλαιο, η Python βασίζεται σε αυτά, για να ορίσει ομάδες εντολών.
- Να επιλέγουμε τους κατάλληλους τελεστές.
- Να χρησιμοποιούμε τα ίδια εισαγωγικά (μονά εισαγωγικά με μονά, διπλά εισαγωγικά με διπλά) για μια συμβολοσειρά.
- Να προσθέτουμε, όπου κρίνουμε χρήσιμο, επεξηγηματικά σχόλια μέσα στον κώδικα. Όταν ένα πρόγραμμα μεταφράζεται, τα σχόλια αγνοούνται. Τα σχόλια αποτελούν παράδειγμα καλού προγραμματιστικού στυλ.
- Να δίνουμε ονόματα μεταβλητών που έχουν σχέση με τη χρήση τους.

Συντακτικά λάθη

Μερικά συνηθισμένα συντακτικά λάθη, που χρειάζεται να προσέχουμε:

- Κεφαλαία αντί μικρά γράμματα. Η Python πολλές φορές ξεχωρίζει τα κεφαλαία γράμματα από τα μικρά και δεν τα θεωρεί ως ίδια λέξη. Για παράδειγμα αν γράψουμε την εντολή
  Print με κεφαλαίο δεν θα καταλάβει ότι είναι η εντολή print.
- Δεν πρέπει να μπερδεύουμε τα διπλά εισαγωγικά με τα μονά. Όταν ανοίγουμε εισαγωγικά πρέπει να κλείνουμε με τα όμοια τους (μονά με μονά, διπλά με διπλά).
- Δεν πρέπει να μπερδεύουμε την κάτω _ με την μεσαία - παύλα.
- Να αποφεύγουμε να χρησιμοποιούμε ελληνικούς χαρακτήρες με λατινικούς στα ονόματα μεταβλητών.
- Όταν ανοίγουμε παρενθέσεις πρέπει να τις κλείνουμε με το αντίστοιχο σύμβολο (),{},[].
- Ιδιαίτερα προσοχή απαιτείται στα κενά διαστήματα στην αρχή μιας γραμμής, μια και στην Python τα κενά διαστήματα έχουν σημασία.
- Ελέγχουμε την ορθότητα της ορθογραφίας της κάθε εντολής, καθώς συχνά ξεχνάμε κάποιο γράμμα.
- Πρέπει να προσέχουμε στην περίπτωση που μεταφέρουμε έτοιμο κώδικα από διαφορετικές εκδόσεις της Python (όπως 2 και 3), διότι υπάρχουν διαφορές σε ορισμένες εντολές ως προς τη σύνταξη μεταξύ των εκδόσεων.

3.6 Διαδικασία συγγραφής, μετάφρασης και εκτέλεσης προγράμματος

3.6.1 Διερμηνέας και μεταγλωττιστής

Όταν γράψουμε κώδικα σε μορφή κειμένου, για να μπορέσει να εκτελεστεί στον υπολογιστή θα πρέπει να μετατραπεί σε γλώσσα μηχανής που είναι «κατανοητή» από την Κεντρική Μονάδα Επεξεργασίας (CPU) του υπολογιστή.

Τα προγράμματα που μετατρέπουν τις εντολές μας μπορούν να χωριστούν σε δύο κατηγορίες:

- στους **μεταγλωττιστές** (compilers) και
- στους **διερμηνείς** (interpreters).

Ένας **διερμηνευτής** διαβάζει και ελέγχει **μία εντολή την φορά**, την εκτελεί και μετά προχωράει στην επόμενη.
Ένας **μεταγλωττιστής** διαβάζει ολόκληρο το πρόγραμμα και το μεταφράζει, πριν ξεκινήσει η εκτέλεσή του.

Σε αυτό το πλαίσιο, το πρόγραμμα υψηλού επιπέδου ονομάζεται **πηγαίος κώδικας** (source code), και στη γενική περίπτωση, το μεταφρασμένο πρόγραμμα **εκτελέσιμο** (executable).
Όταν ένα πρόγραμμα μεταγλωττιστεί, μπορεί να εκτελεστεί επανειλημμένα χωρίς περαιτέρω μετάφραση.

Η Python αποτελεί γλώσσα με δυνατότητα τα προγράμματά της να εκτελούνται από διερμηνευτή.
Υπάρχουν δύο τρόποι χρήσης του διερμηνευτή:

- **διαδραστική λειτουργία** (interactive mode) και
- **σεναριακή λειτουργία** (script mode).

Στην **διαδραστική λειτουργία**, πληκτρολογούμε προγράμματα σε Python και ο διερμηνέας εμφανίζει το αποτέλεσμα:
>>> 15 + 1 16**.**

Το σύμβολο, >>>, είναι ο **προτροπέας** (prompt) που χρησιμοποιεί ο διερμηνευτής για να υποδείξει ότι είναι έτοιμος.
Όταν πληκτρολογήσουμε 15 + 1, ο διερμηνέας ελέγχει την έκφραση, την μεταφράζει ώστε να εκτελεστεί και στην οθόνη εμφανίζεται το αποτέλεσμα 16.
Εναλλακτικά, μπορούμε να αποθηκεύσουμε κώδικα σε ένα αρχείο και να χρησιμοποιήσουμε το διερμηνέα για να εκτελέσει το περιεχόμενο του αρχείου, το οποίο ονομάζεται ένα **σενάριο.**

Τύποι και δομές δεδομένων στις γλώσσες προγραμματισμού

Ένας *Τύπος Δεδομένων* (Data Type) είναι ένα σύνολο τιμών δεδομένων και λειτουργιών επί αυτών των τιμών.
Οι τύποι δεδομένων

- είτε είναι προκαθορισμένοι από τις γλώσσες προγραμματισμού και καλούνται ***Πρωτογενείς Τύποι Δεδομένων*** **(Primitive Data Types)**
- είτε δημιουργούνται από τον προγραμματιστή, οπότε καλούνται ***Μη Πρωτογενείς τύποι δεδομένων*** **(Non-Primitive Data Types**

Τύποι και Δομές Δεδομένων. Εννοιολογικοί προσδιορισμοί

Οι τύποι δεδομένων διακρίνονται επίσης ανάλογα με τη σύσταση των μερών που τους αποτελούν, σε:

- **Απλούς τύπους.**

Στους απλούς τύπους δεδομένων, οι τιμές των δεδομένων είναι στοιχεία μη περαιτέρω-χωριζόμενα (άτομα). Δηλαδή κάθε δεδομένο έχει μία και μοναδική τιμή.

- **Σύνθετους τύπους δεδομένων.**

Σύνθετος τύπος δεδομένων (Composite Data Type) είναι εκείνος, που αποτελείται από Πρωτογενείς ή / και άλλους σύνθετους τύπους, όπου μια μεταβλητή μπορεί να πάρει ως τιμή μια ενότητα τιμών. Οι σύνθετοι τύποι καλούνται και **Δομές Δεδομένων**. Παραδείγματα σύνθετων τύπων είναι η Εγγραφή, το Σύνολο, ο Πίνακας.

Συνήθεις τύποι σε γλώσσες προγραμματισμού

Απλοί

- **Ακέραιος** (Integer), με τιμές τους ακέραιους αριθμούς μέσα σε ένα κάτω και ένα άνω όριο και πράξεις: +, -,*, /, mod, div, := καθώς και οι συγκρίσεις >, =, <
- **Πραγματικός** (real/float).
- **Χαρακτήρας** (Character), με τιμές από το σύνολο των χαρακτήρων που διαθέτει ο υπολογιστής.
- **Λογικός** (Boolean), για την αναπαράσταση Λογικών δεδομένων, με τιμές True (σωστό), False (λάθος) και επιτρεπτές πράξεις για τις τιμές αυτές τις: and, or, not.
- **Αλφαριθμητικός** - string.

Σύνθετοι Τύποι Δεδομένων

- **Πίνακας (Array).** *Συνήθως* τα στοιχεία του καθορίζονται με τη βοήθεια δεικτών. Μπορεί να είναι πολλών διαστάσεων.
- **Εγγραφή** (Record/tuple/struct).
- **Λίστα** (List).
- **Σύνολο** (Set).
- **Σωρός** (Heap).
- **Στοίβα** (Stack).
- **Ουρά** (Queue).
- **Δένδρο** (Tree).
- **Γράφος** (Graph).