Απλοί τύποι δεδομένων (Simple data type) στην Python:

1. Αριθμητικός
    a. Ακέραιοι (integer) π.χ.:24
    b. Αριθμοί κινητής υποδιαστολής (floating point) π.χ.: 24.43, 5.0
2. Λογικός (boolean) τιμές: True ή False
3. Χαρακτήρας (character)
4. Συμβολοσειρές ή αλφαριθμητικά (strings) π.χ.: ‘Γρηγόρης’, ”Christos”

Σύνθετοι τύποι δεδομένων (Composite data type) στην Python:

1. Εγγραφή (record)
2. Πίνακας (array)
3. Λίστα (list)
4. Σύνολο (set)
5. Σωρός (heap)
6. Στοίβα (stack)
7. Ουρά (queue)
8. Δέντρο (tree)
9. Γράφος (graph)

Μεταβλητές (variable) - σκέψου ετικέτες με όνομα

Για να τις χρησιμοποιήσουμε πρέπει:

- Να τους δώσουμε ένα όνομα (δεν ξεκινά ποτέ με αριθμό, δεν είναι ποτέ όνομα εντολής)
- Να τους εκχωρήσουμε μία τιμή

Προσοχή! Γίνεται διάκριση μικρών και κεφαλαίων χαρακτήρων, δηλ. οι μεταβλητές sxoleio και Sxoleio είναι διαφορετικές. Οι έμπειροι προγραμματιστές αποφεύγουν τα κεφαλαία.

Κανόνες δημιουργίας ονομάτων μεταβλητών:

- Τα ονόματα μπορούν να είναι όσο μεγάλα θέλουμε, μπορούν να περιέχουν γράμματα και αριθμούς, αλλά πρέπει να ξεκινούν με ένα γράμμα ή τον χαρακτήρα _ (κάτω παύλα / underscore).
- Αν και μπορούμε να χρησιμοποιήσουμε ελληνικά γράμματα, καλό είναι να τα αποφεύγουμε.
- Τα πεζά διακρίνονται από τα κεφαλαία γράμματα (case sensitive), για παράδειγμα οι day, Day, DAy, DAY, dAy, daY, DaY, dAY είναι διαφορετικές μεταβλητές.

- Δεν επιτρέπονται κενά, εισαγωγικά, τελείες, κόμματα και άλλοι παρόμοιοι χαρακτήρες. Επιτρέπεται μόνο ο χαρακτήρας _ (underscore/κάτω παύλα), ο οποίος είναι χρήσιμος κυρίως σε ονόματα που αποτελούνται από πολλές λέξεις.

Τελεστές (Operators)

Οι τελεστές είναι σύμβολα ή λέξεις για τη δημιουργία αριθμητικών ή λογικών εκφράσεων.

1. Αριθμητικοί (+ - * / ** % //)
2. Σχεσιακοί (< <= > >= == !=)
3. Λογικών πράξεων (not, and, or)

Τελεστής	Όνομα	Εξήγηση	Παραδείγματα
+	Συν	Πρόσθεση αριθμών ή αλληλουχία συμβολοσειρών	Το 5 + 3 δίνει 8 Το 'a' + 'b' δίνει 'ab'
-	Μείον	Αφαίρεση ενός αριθμού από έναν άλλο	Το 50 - 26 δίνει 24
*	Επί	Γινόμενο δύο αριθμών ή επανάληψη μιας συμβολοσειράς τόσες φορές.	Το 2 * 3 δίνει 6 Το 'la' * 3 δίνει 'lalala'
**	Δύναμη	Ύψωση αριθμού σε δύναμη.	Το 3 ** 4 δίνει 81 Το 10 ** 2 δίνει 100 Το 2.0 ** 3 δίνει 8.0
/	Δια	Διαίρεση δύο αριθμών.	Το αποτέλεσμα είναι πάντα δεκαδικός αριθμός. Το 4 / 3 δίνει 1.3333333333333333 Το 10 / 2 δίνει 5.0 Το 1 / 2 δίνει 0.5

//	Ακέραια διαίρεση	Διαίρεση δύο αριθμών στρογγυλοποιημένη προς τα κάτω (floor division).	Το 4 // 3 δίνει 1 Το 1 // 2 δίνει 0 Το 17 // 3 δίνει 5 Το 17.1 // 3 δίνει 5.0
%	Υπόλοιπο	Υπόλοιπο διαίρεσης δύο αριθμών.	Το 11 % 3 δίνει 2

Βασικές εντολές/συναρτήσεις

print : Εμφάνιση μηνύματος ή περιεχομένου μεταβλητής στην οθόνη

input : Εισαγωγή (διάβασμα) τιμών από το πληκτρολόγιο. Κάθε φορά που καλείται η συνάρτηση αυτή από ένα πρόγραμμα, αυτό σταματάει και περιμένει από τον χρήστη να πληκτρολογήσει κάτι.

= Εντολή εκχώρησης τιμής, χρησιμοποιείται για την απόδοση τιμής σε μεταβλητές

Βασικές (ενσωματωμένες) συναρτήσεις

Η Python έχει ενσωματωμένες συναρτήσεις για μετατροπή από έναν τύπο σε έναν άλλο:

float(x): μετατρέπει ακέραιους αριθμούς και συμβολοσειρές σε δεκαδικό αριθμό.

int(x): μετατρέπει δεκαδικούς («κόβει» τα δεκαδικά ψηφία) και συμβολοσειρές(strings) σε ακέραιους αριθμούς.

str(n): μετατρέπει αριθμούς σε συμβολοσειρές (string).

round(x): Στρογγυλοποιεί δεκαδικούς αριθμούς.

Εξωτερικές βιβλιοθήκες

Εκτός από τις ενσωματωμένες συναρτήσεις οι οποίες περιλαμβάνονται στη γλώσσα Python, μπορεί κανείς να βρει πληθώρα εξωτερικών βιβλιοθηκών (modules) στους Διαδικτυακούς τόπους υποστήριξης της γλώσσας.

Μια βιβλιοθήκη είναι ένα αρχείο το οποίο περιέχει μια **συλλογή από σχετικές συναρτήσεις**. Ενδεικτικά, μπορούν να αναφερθούν: η Μαθηματική βιβλιοθήκη, η βιβλιοθήκη τυχαίων αριθμών και οι βιβλιοθήκες γραφικών. Οι βιβλιοθήκες αυτές για να χρησιμοποιηθούν, θα πρέπει πρώτα να εισαχθούν στο πρόγραμμά μας. Η εισαγωγή αυτή γίνεται με την **εντολή import**.

Για παράδειγμα, προκειμένου να χρησιμοποιήσουμε μαθηματικές συναρτήσεις σε ένα πρόγραμμα, θα πρέπει μία από τις αρχικές εντολές του προγράμματός μας να είναι: import math.
Για να έχουμε πρόσβαση σε μια από τις συναρτήσεις, θα πρέπει να δηλώσουμε το όνομα της μονάδας και το όνομα της συνάρτησης, χωρισμένα με μια τελεία, μορφή που ονομάζεται συμβολισμός με τελεία (dot notation).
Ας δούμε ένα παράδειγμα συνάρτησης για την τετραγωνική ρίζα, sqrt().

import math
riza = **math.sqrt(2)**
print riza # 1.41421356237
math.sqrt(3) # 1.7320508075688772
x = **math.pi**
print x # 3.14159265359

Δεύτερο παράδειγμα : Παραγωγή τυχαίων αριθμών

Πολλές φορές θέλουμε να παράγουμε αριθμούς με τυχαίο τρόπο. Η **βιβλιοθήκη random** περιέχει μια ποικιλία συναρτήσεων γι΄ αυτόν τον σκοπό. Δύο από αυτές που θα χρησιμοποιούμε πολύ συχνά, είναι η **randint** και η **randrange**, που επιστρέφουν τυχαίους ακέραιους αριθμούς εντός κάποιων ορίων.

import random
number = random.**randint**(1, 10) # επιστρέφει έναν τυχαίο αριθμό στο [1, 10]
number = random.**randrange**(1,10) # επιστρέφει έναν τυχαίο αριθμό στο [1, 9]
number = random.**randrange**(10) # επιστρέφει έναν τυχαίο αριθμό στο [0, 9]

Εκτέλεση υπό συνθήκη, η εντολή if

```
if συνθήκη:
    μπλοκ_εντολών_1
else:
    μπλοκ_εντολών_2
```

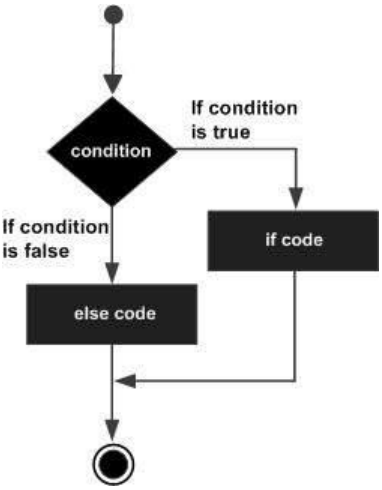
Οι **εντολές** ενός μπλοκ **πρέπει να είναι μετατοπισμένες προς τα δεξιά**. Η τυπική μετατόπιση ή εσοχή (indentation) των εντολών είναι τέσσερα κενά. Η Python μάς

βοηθάει σε αυτό ρυθμίζοντας αυτόματα τις εσοχές για μας, απλά πατώντας Enter μετά την πληκτρολόγηση της άνω κάτω τελείας. Η έλλειψη ενός κενού ή η ύπαρξη επιπλέον κενών μπορεί να οδηγήσει σε λάθος ή απρόσμενη συμπεριφορά σε ένα πρόγραμμα.

Παράδειγμα :

```
x = int(input('Δώστε έναν ακέραιο αριθμό:'))
if x % 2 == 0:
    print(x, 'είναι άρτιος')
else:
    print(x, 'είναι περιττός')
```

Σύνθετη επιλογή (*if...else* statement)



Παράδειγμα :

```
x = 20
y = 10

if x == y:
    print(x, '=', y)
else:
    if x < y:
        print(x, '<', y)
    else:
        print(x, '>', y)
```

Πολλαπλή επιλογή (*elif* statement)

```
x = 20
y = 10

if x < y:
    print(x, '<', y)
elif x > y:
    print(x, '>', y)
else:
    print(x, '=', y)
```

Πίνακας αληθείας (Truth table logical operators)

a	b	a or b	a and b	not a
true	true	true	true	false
true	false	true	false	false
false	true	true	false	true
false	false	false	false	true

Δομή επανάληψης (for και while)

Συχνά σε ένα πρόγραμμα, μια ομάδα εντολών είναι αναγκαίο να εκτελείται περισσότερες από μία φορές.

Υπάρχουν δύο τύποι επαναλήψεων:
● Οι προκαθορισμένοι, όπου το πλήθος των επαναλήψεων είναι δεδομένο, πριν αρχίσουν οι επαναλήψεις. Για παράδειγμα: ο υπολογισμός του μέσου όρου βαθμολογίας των μαθητών ενός τμήματος 22 μαθητών.
● Οι μη προκαθορισμένοι, όπου το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών του σώματος της επανάληψης. Για παράδειγμα: να διαβάζονται αριθμοί μέχρι να δοθεί ο αριθμός μηδέν, να υπολογίζεται ο μέσος όρος τους.

Στη γλώσσα προγραμματισμού Python, χρησιμοποιούμε την εντολή for για να εκτελεστεί ένα τμήμα του κώδικα για έναν καθορισμένο αριθμό επαναλήψεων, ενώ την εντολή while για να εκτελείται υπό συνθήκη και μάλιστα, όσο αυτή είναι αληθής. Στην εντολή for χρησιμοποιείται η συνάρτηση range() για τον καθορισμό των επαναλήψεων.

```
for variable in ange (start, end, step):
        Εντολή_1
        Εντολή_2 …..
        Εντολή_v
```

Η **range()** είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία, ανάμεσα σε άλλα, χρησιμοποιείται για την υπόδειξη του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο.

Η δομή της είναι της μορφής range (start, end, step), όπου start, end, step ακέραιοι αριθμοί.
Οι start και end δεν είναι υποχρεωτικές και, αν δεν αναφέρονται, θα αρχίσει από 0 και θα συνεχίσει με βήμα 1. Αντίθετα η ένδειξη end πρέπει πάντα να αναφέρεται.

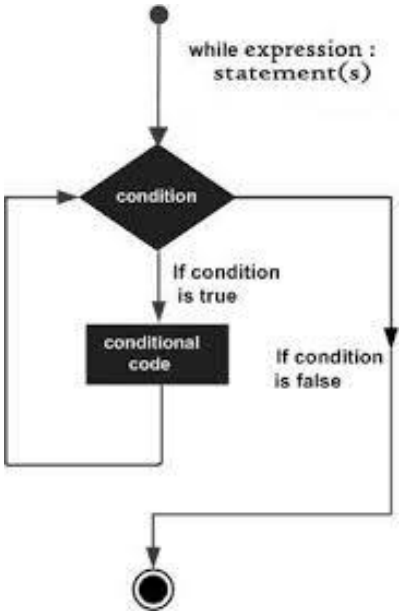
Παραδείγματα
- range(10), παράγει τη λίστα: [0,1,2,3,4,5,6,7,8,9].
- range(1, 8), παράγει τη λίστα: [1,2,3,4,5,6,7]
- range(8, -1, -1), παράγει τη λίστα [8, 7, 6, 5, 4, 3, 2, 1, 0]

Δομή Επανάληψης με while

Η δομή while (Όσο <συνθήκη> επανάλαβε) χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων. Σε κάθε επανάληψη (και στην αρχική) πραγματοποιείται ο έλεγχος της συνθήκης, πριν από την εκτέλεση των εντολών του βρόχου, κάτι που σημαίνει ότι υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου.

Αρχική τιμή μεταβλητής
while όνομα_μεταβλητής <συνθήκη>:
    Εντολή_1
    Εντολή_2 ….
    Εντολή_κ

Σημείωση 1: θα πρέπει μέσα στο μπλοκ εντολών να υπάρχει κατάλληλη εντολή, ώστε να εξασφαλίζεται ότι κάποια στιγμή η συνθήκη θα γίνει ψευδής και θα διακοπεί ο βρόχος. Διαφορετικά ο βρόχος δε θα τερματίζει.
Σημείωση 2: πριν το βρόχο while θα πρέπει αρχικά να δώσουμε μία τιμή στη μεταβλητή που ελέγχει τη συνθήκη του βρόχου, ώστε ανάλογα να εκτελεστεί ή όχι ο βρόχος.



Λίστες

Δημιουργία λίστας

- Οι τιμές που θα περιέχονται στη λίστα εσωκλείονται σε αγκύλες [ ] και χωρίζονται με κόμμα.
- Οι τιμές καταχωρούνται η μια μετά την άλλη.

Παραδείγματα:
nums = [3,5,8,13,21,34,55]
users = ['Αργυρώ', 'Γιάννης', 'Κυριάκος']
empty = [ ]

Πρόσβαση σε στοιχείο της λίστας

- Αποκτούμε πρόσβαση σε ένα στοιχείο της λίστας γράφοντας το όνομα της λίστας και τη θέση του στοιχείου μέσα σε [ ].
- Η αρίθμηση των θέσεων ξεκινάει πάντα από το 0 και φτάνει μέχρι το πλήθος των στοιχείων της λίστας μειωμένο κατά 1.
- Εναλλακτικά, η αρίθμηση των θέσεων γίνεται κι αντίστροφα, με την τελευταία θέση να αντιστοιχεί στον αριθμό -1.
- Μέσα στις αγκύλες μπορούμε να γράψουμε οποιαδήποτε ακέραια έκφραση.
- Σε περίπτωση που αναφερθούμε σε μια θέση που δεν υπάρχει στη λίστα, δηλαδή σε έναν αριθμό μεγαλύτερο ή ίσο με το πλήθος των στοιχείων της τότε προκύπτει σφάλμα.

Παραδείγματα:
nums[3] = 1, δίνει στο τέταρτο στοιχείο την τιμή 1
nums[-1] = 89, δίνει στο τελευταίο στοιχείο την τιμή 89
print(nums[0]), εμφανίζει το πρώτο στοιχείο της λίστας

Πλήθος στοιχείων - Συνάρτηση len
Δίνει το πλήθος των στοιχείων μιας λίστας.

Παράδειγμα:
```
if len(nums) == 0:
        print("Η λίστα είναι κενή")
```

Απαρίθμηση στοιχείων λίστας - Εντολή for

- Η εντολή for είναι μια εντολή επανάληψης που διατρέχει τα στοιχεία μιας ακολουθίας τιμών, όπως μια λίστα, με τη σειρά που εμφανίζονται.
- Σε κάθε επανάληψη η τιμή του επόμενου στοιχείου της ακολουθίας ανατίθεται σε μια μεταβλητή απαρίθμησης που χρησιμοποιούμε στην for

Παράδειγμα:
```
        for ar in nums: print(ar)
```

Συνένωση και πολλαπλασιασμός Τελεστές +, *

- Ο τελεστής + (συνένωση) χρησιμοποιείται ανάμεσα σε δύο λίστες και δημιουργεί μια νέα λίστα που περιέχει όλα τα στοιχεία των αρχικών.
- Ο τελεστής * έχει ως αποτέλεσμα τη δημιουργία μιας νέας λίστας που περιέχει πολλές φορές τα στοιχεία της αρχικής.

Παραδείγματα:
n = [1,2,3] + [4,5,6], το n θα περιέχει τα [1,2,3,4,5,6]
n = [1,2,3] * 3, το n περιέχει τα [1,2,3,1,2,3,1,2,3]

Έλεγχος ύπαρξης τιμής σε λίστα Τελεστής in
Ελέγχει αν η τιμή βρίσκεται στη λίστα και επιστρέφει αντίστοιχα την τιμή True ή False.
Παράδειγμα:
```
if "Athens" in List_Of_Cities:
        print('ok')
```

Τεμαχισμός λίστας
Δημιουργεί μια νέα λίστα που αντιστοιχεί σε ένα «τεμαχισμένο» τμήμα της αρχικής ο Για να τεμαχίσουμε μια λίστα γράφουμε μέσα σε αγκύλες [ ] τρεις αριθμούς: την αρχική θέση του τεμαχισμού, τη θέση τερματισμού του τεμαχισμού (που δεν περιλαμβάνεται στο τελικό τμήμα) και ανά πόσα στοιχεία θα περιλαμβάνονται στο τεμαχισμένο τμήμα, ξεκινώντας από την αρχική θέση.

- Αν παραλείψουμε την αρχική θέση, ο τεμαχισμός ξεκινάει από το πρώτο στοιχείο της λίστας.
- Αν παραλείψουμε την τελική θέση τότε ο τεμαχισμός φτάνει μέχρι το τέλος της λίστας.
- Αν παραλείψουμε το βήμα τότε παίρνει την τιμή +1.

Παραδείγματα:

nums[1:4], δημιουργεί μια νέα λίστα που περιέχει τα στοιχεία στις θέσεις 1 έως και 3 της αρχικής λίστας

nums[::2], ξεκινώντας από την αρχή της λίστας, δημιουργεί μια νέα λίστα που περιέχει τα στοιχεία της αρχικής που βρίσκονται σε ζυγές θέσεις

nums[::-1], δημιουργεί νέα λίστα, αντίστροφη της αρχικής

Προσθήκη νέου στοιχείου Μέθοδος append()

Προσθέτει ένα νέο στοιχείο στο τέλος της λίστας.

Παραδείγματα:

nums.append(89)

users.append('Μυρσίνη')

Μέθοδος insert()

Εισάγει ένα νέο στοιχείο σε οποιαδήποτε θέση της λίστας.

Η θέση και το στοιχείο εισαγωγής δίνονται ως παράμετροι.

Παραδείγματα:

users.insert(1, 'Μελίνα'), εισάγει την τιμή 'Μελίνα' στη 2η θέση της λίστας

Αφαίρεση στοιχείου Μέθοδος pop()

Αφαιρεί το τελευταίο στοιχείο της λίστας και το επιστρέφει.

Παράδειγμα:

lastnum = nums.pop()

Μέθοδος remove()

Αφαιρεί ένα στοιχείο της λίστας.

- Δέχεται ως παράμετρο το στοιχείο που θα αφαιρεθεί.
- Αν το στοιχείο δεν υπάρχει στη λίστα τότε προκύπτει σφάλμα

Παράδειγμα:

nums.remove(13) Δημιουργία αντιγράφου

Μέθοδος copy()

Επιστρέφει ένα αντίγραφο μιας λίστας.

Παράδειγμα:

otherNums = nums.copy()

Εύρεση θέσης στοιχείου Μέθοδος index()

Αναζητά τη θέση ενός στοιχείου σε μια λίστα.

Παρατηρήσεις:

- Δέχεται ως παράμετρο το στοιχείο της λίστας, για το οποίο αναζητούμε τη θέση του.

- Αν το στοιχείο δεν υπάρχει στη λίστα τότε προκύπτει σφάλμα

Παράδειγμα:
pos = nums.index('Στέλλα')

Ταξινόμηση στοιχείων Μέθοδος sort()
Η λίστα στην οποία εφαρμόζεται μεταβάλλεται.
Παράδειγμα:
nums.sort()

Συνάρτηση sorted()
Επιστρέφει μια νέα, ταξινομημένη λίστα χωρίς να μεταβάλλει τη λίστα που δέχεται σαν παράμετρο.
Παράδειγμα:
ordered = sorted(nums)

Αντιστροφή στοιχείων Μέθοδος reverse()
Παράδειγμα:
nums.reverse() Αντιστροφή μπορεί να γίνει και με τεμαχισμό: π.χ. Nums[::-1]

Ανακάτεμα στοιχείων Συνάρτηση shuffle()
Ανακατεύει τα στοιχεία μιας λίστας.
Δέχεται ως παράμετρο τη λίστα που θα ανακατέψει.
Παράδειγμα:
random.shuffle(nums)

Επιλογή ενός τυχαίου στοιχείου Συνάρτηση choice()
Η παράμετρος της είναι η λίστα από την οποία θα επιλέξει το τυχαίο στοιχείο.
Παράδειγμα:
element = random.choice(nums)

Επιλογή πλήθους τυχαίων στοιχείων Συνάρτηση sample()
Δέχεται ως παράμετρο μια λίστα και το πλήθος των στοιχείων που θα επιλεχθούν τυχαία.
Επιστρέφει τα τυχαία επιλεγμένα στοιχεία σε μια νέα λίστα.
Παράδειγμα:
mixed = random.sample(nums, 3)