Εισαγωγή στην Python

Βασικά σημεία θεωρίας - Ασκήσεις

Απλοί τύποι δεδομένων στην Python (Simple data types)

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 στην Python (Composite data types)

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

Μεταβλητές (variables) - 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.

Μεταβλητές (variables) – Ονόματα μεταβλητών

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), είναι χρήσιμος κυρίως σε ονόματα που αποτελούνται από πολλές λέξεις.

Άσκηση 1

Είναι σωστές οι παρακάτω ονομασίες μεταβλητών;

Γράψτε στο τετράδιό σας τον αριθμό και δίπλα τη λέξη **ΣΩΣΤΟ** ή **ΛΑΘΟΣ**.

Για όσες δεν είστε σίγουροι δοκιμάστε τις στο περιβάλλον της Python:

βάλτε μια τιμή (π.χ kostos$ = 100) και μετά τρέξτε το πρόγραμμα για να δείτε αν εμφανίζεται λάθος. Σε περίπτωση λάθους προσπαθήστε να απαντήσετε γιατί είναι λάθος.

Άσκηση 1

Είναι σωστές οι παρακάτω ονομασίες μεταβλητών;

1. kostos€
2. 67ktl
3. MesosOros
4. k+m
5. $dς
6. G(x)
7. print_input
8. Μ.Ο.
9. Δημοτικό
10. 3odimotiko
11. Ektoras
12. Έκτορας
13. Print

Τελεστές

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

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

Αριθμητικοί Τελεστές

+	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’

Αριθμητικοί Τελεστές

**	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

Αριθμητικοί Τελεστές

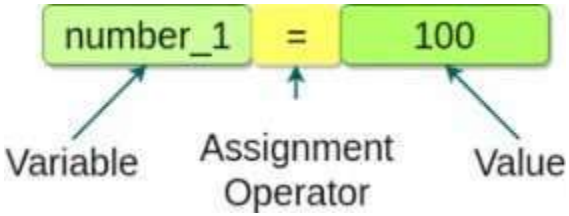
//	Ακέραια διαίρεση	Διαίρεση δύο αριθμών στρογγυλοποιημένη προς τα κάτω (floor division).	Το 4 // 3 δίνει 1 Το 1 // 2 δίνει 0 Το 17 // 3 δίνει 5 Το 17.1 // 3 δίνει 5.0
%	Υπόλοιπο	Υπόλοιπο διαίρεσης δύο αριθμών.	Το 11 % 3 δίνει 2

Βασικές εντολές

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



Άσκηση 2

Γράψτε στο τετράδιο σας ποιές από τις παρακάτω εντολές εκχώρησης είναι σωστές και ποιές είναι λάθος.

Δοκιμάστε τις στην Python και ελέγξτε τις απαντήσεις σας.

1

```
a = 10
a = a + 1 print (a)
```

2

```
num = '2o'
sxoleio = num + 'gymnasio’
print (sxoleio)
```

3

```
A = B = 5
print (A, B)
```

Άσκηση 2

4

```
A = 10
A + 3 = B
print (A, B)
```

5

```
B = -3.5
print (B *2)
```

6

```
b = 10
a, c = b
print (a, b, c)
```

7

```
Ektoras = ‘Programmer’
print (Ektoras)
```

Άσκηση 2

8

```
Ektoras = "Ektoras«
Nikos = Ektoras
print (Nikos, Ektoras)
```

9

```
x20 = 20
x21 = x20 + 2 + 1
print (x20, x21)
```

10

```
print = 5
```

Άσκηση 3

Να μετατραπούν σε εντολές εκχώρησης τιμής οι παρακάτω εκφράσεις (γράψτε μια εντολή = σε κάθε περίπτωση):

1. Η μεταβλητή A έχει διπλάσια τιμή από τη μεταβλητή B
2. Η μεταβλητή Average είναι ο μέσος όρος των A, B, C
3. Η μεταβλητή B αυξάνεται κατά 2
4. Η μεταβλητή K μειώνεται κατά A και B
5. Η μεταβλητή K είναι το μισό του αθροίσματος των A και B
6. Η μεταβλητή P είναι το 15% της μεταβλητής Price *(το 15% μπορεί να γραφεί 15/100)*

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(n): στρογγυλοποιεί δεκαδικούς αριθμούς.

Εξωτερικές βιβλιοθήκες (modules)

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 βιβλιοθήκη **math**, η βιβλιοθήκη **random** και η βιβλιοθήκη **turtle**. 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)

Εξωτερικές βιβλιοθήκες – Η βιβλιοθήκη math

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
```

Εξωτερικές βιβλιοθήκες – Η βιβλιοθήκη random

Παράδειγμα συνάρτησης για την παραγωγή τυχαίων αριθμών

Πολλές φορές θέλουμε να παράγουμε αριθμούς με τυχαίο τρόπο. Η **βιβλιοθήκη random** περιέχει μια ποικιλία συναρτήσεων για αυτόν τον σκοπό. Δύο από αυτές που 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]
```

Άσκηση 4

Να γράψετε πρόγραμμα που:

α) να εμφανίζει το μήνυμα "Ποιό είναι το όνομά σου;" και διαβάζει το όνομά σας

β) να εμφανίζει το μήνυμα "Ποιό είναι το επώνυμό σου;" και διαβάζει το επώνυμό σας

γ) να εμφανίζει τις λέξεις "Καλώς ήρθες", αμέσως μετά το όνομα και το επώνυμό σας και στο τέλος "στον προγραμματισμό."

Παράδειγμα εκτέλεσης:

```
Ποιό είναι το όνομά σου; Mike
Ποιό είναι το επώνυμό σου; Jordan
Καλώς ήρθες Mike Jordan στον προγραμματισμό.
```

Άσκηση 5

Στον πλανήτη Αφροδίτη το βάρος ενός αντικειμένου είναι 0,9 φορές το βάρος του στη Γη. Στον Ήλιο το βάρος ενός αντικειμένου είναι 27,07 φορές το βάρος του στη Γη, αλλά όταν βρίσκεται κανείς εκεί η αύξηση του βάρους δεν είναι το βασικότερο πρόβλημα.

Να γράψετε πρόγραμμα που θα ζητάει (διαβάζει) από το χρήστη το βάρος του στη Γη και θα εμφανίζει το βάρος του στην Αφροδίτη και τον Ήλιο.

Άσκηση 6

Ο κήπος μιας πολυκατοικίας έχει σχήμα ρόμβου με περίμετρο Χ μέτρα.

Να διαβαστεί η περίμετρος Χ (σε μέτρα) και να υπολογίσετε το μήκος κάθε πλευράς του σε μέτρα ή δέκατα ή εκατοστά ή χιλιοστά.

Παράδειγμα εκτέλεσης:

```
Δώστε περίμετρο πολυκατοικίας (σε μέτρα):
Μήκος πλευράς σε μέτρα:

Μήκος πλευράς σε δέκατα:

Μήκος πλευράς σε εκατοστά:

Μήκος πλευράς σε χιλιοστά:
```

Εκτέλεση υπό συνθήκη (*if* statement)

```
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.

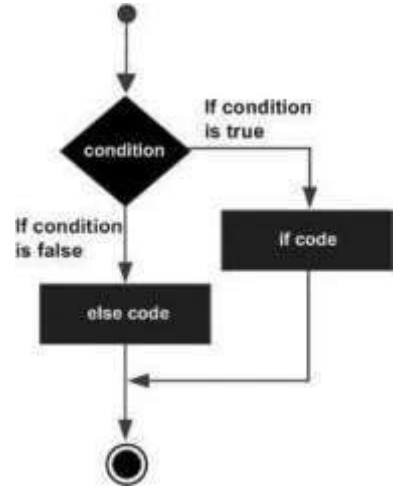
Η Python μας βοηθάει σε αυτό ρυθμίζοντας αυτόματα τις εσοχές για μας, απλά πατώντας Enter μετά την πληκτρολόγηση της άνω κάτω τελείας (:).

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)
```

Λογικοί Τελεστές - Πίνακας Αληθείας

x	y	x or y	x and y	not x
True	True	True	True	False
True	False	True	False	False
False	True	True	False	True
False	False	False	False	True

Άσκηση 7

Να γράψετε πρόγραμμα που να διαβάζει (input) το μήκος των 2 πλευρών ενός παραλληλογράμμου και να υπολογίζει και εμφανίζει (print) την περίμετρο και το εμβαδόν του.

Σε περίπτωση που οι πλευρές είναι ίσες να εμφανίζει «Το σχήμα είναι τετράγωνο» ενώ σε περίπτωση που είναι άνισες να εμφανίζει «Το σχήμα είναι παραλληλόγραμμο».

Παράδειγμα εκτέλεσης:

```
Δώστε την πρώτη πλευρά (σε εκατοστά): 5
Δώστε την δεύτερη πλευρά (σε εκατοστά): 5
Το σχήμα είναι τετράγωνο
Περίμετρος τετραγώνου = 20 εκατοστά
Εμβαδόν τετραγώνου = 25 τετραγωνικά εκατοστά
```

Άσκηση 8

Να γραφούν, για κάθε μία από τις παρακάτω περιπτώσεις, οι εντολές επιλογής (if) που ελέγχουν αν η μεταβλητή Χ είναι:

- θετικός αριθμός διαφορετικός του 100
- διψήφιος αριθμός
- θετικός αριθμός (μεγαλύτερος από το μηδέν) και έχει το πολύ 3 ψηφία (δηλ είναι το πολύ 3ψήφιος αριθμός)

Κάθε φορά να εμφανίζετε μήνυμα για να ελέγξετε ότι γράψατε σωστά την εντολή επιλογής.

Άσκηση 9

Γράψτε 2 τιμές των μεταβλητών a, b έτσι ώστε τα δύο τμήματα προγράμματος να έχουν **διαφορετικό** αποτέλεσμα. Εξηγήστε την απάντησή σας.

1ο τμήμα:

```
if (a>0 or b>0):
        print (a,b)
```

2ο τμήμα:

```
if (a>0):
        print (a,b)
if (b>0):
        print (a,b)
```

Άσκηση 10

Να γράψετε πρόγραμμα που να διαβάζει (input) 3 αριθμούς και να τους εμφανίζει (print) σε φθίνουσα σειρά, από τον μεγαλύτερο προς τον μικρότερο.

Παράδειγμα εκτέλεσης:

```
Δώσε 3 αριθμούς: 3, 5, 1
Σε φθίνουσα σειρά : 5, 3, 1
```

Άσκηση 11

Να δημιουργήσετε πρόγραμμα που να διαβάζει 2 αριθμούς και να εμφανίζει: το άθροισμά τους, το γινόμενό τους, την διαφορά τους (αφαίρεση) - σε περίπτωση που η διαφορά είναι αρνητικός αριθμός να την κάνει θετικό αριθμό (απόλυτη τιμή) και, τέλος, να εμφανίζει το πηλίκο τους (διαίρεση). Προσοχή! Δεν γίνεται διαίρεση με το μηδέν, σε αυτήν την περίπτωση να εμφανίζει το μήνυμα 'Αδύνατη η διαίρεση με το μηδέν')

Άσκηση 12

Να γράψετε πρόγραμμα που θα διαβάζει 2 αριθμούς και έναν χαρακτήρα (βάλτε str μπροστά από το input) από τους +, -, *, / και θα εκτελεί την πράξη που αντιστοιχεί στον χαρακτήρα. Προσοχή στη διαίρεση, **δεν γίνεται διαίρεση με το μηδέν**. Σε περίπτωση που ο 2ος αριθμός είναι μηδέν να εμφανίζεται το μήνυμα ‘Διαίρεση αδύνατη' .

Παράδειγμα εκτέλεσης:

```
Δώστε 1ο αριθμό: 5
Δώστε 2ο αριθμό: 7
Δώστε πράξη (+, -, /, *): +
5 + 7 = 12
```

Δομές επανάληψης (for και while)

Συχνά σε ένα πρόγραμμα, μια ομάδα εντολών είναι αναγκαίο να εκτελείται περισσότερες από μία φορές.
Υπάρχουν δύο τύποι επαναλήψεων:

- Με προκαθορισμένο πλήθος επαναλήψεων, 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.
- Με μη προκαθορισμένο πλήθος επαναλήψεων, 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 και να υπολογίζεται ο μέσος όρος τους.

Δομή επανάληψης for (Για…)

Στην Python, χρησιμοποιούμε την δομή for για να εκτελεστεί ένα τμήμα του κώδικα για έναν καθορισμένο αριθμό επαναλήψεων. Στην δομή for (Για…) χρησιμοποιείται η συνάρτηση range() για τον καθορισμό του αριθμού των επαναλήψεων.

Παράδειγμα:

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

Δομή επανάληψης for (Για…)

Η **range()** είναι μια ενσωματωμένη συνάρτηση της γλώσσας Python, η οποία χρησιμοποιείται για τον καθορισμό του αριθμού των επαναλήψεων που θα εκτελεστούν σε ένα βρόχο επανάληψης for (for loop).

Η δομή της έχει τη μορφή **range (start, end, step)**, όπου start, end, step είναι ακέραιοι αριθμοί. Τα start και step δεν είναι υποχρεωτικά, 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 while (Όσο…)

Η δομή while (Όσο <συνθήκη> επανάλαβε) χρησιμοποιείται για μη προκαθορισμένο αριθμό επαναλήψεων.

Χρησιμοποιούμε την δομή while για να εκτελεστεί ένα τμήμα του κώδικα υπό συνθήκη και μάλιστα, όσο αυτή είναι αληθής.

Ο έλεγχος της συνθήκης πραγματοποιείται σε κάθε επανάληψη (και στην αρχική), πριν από την εκτέλεση των εντολών του βρόχου. Αυτό σημαίνει ότι υπάρχει περίπτωση να μην εκτελεστούν οι εντολές του βρόχου.

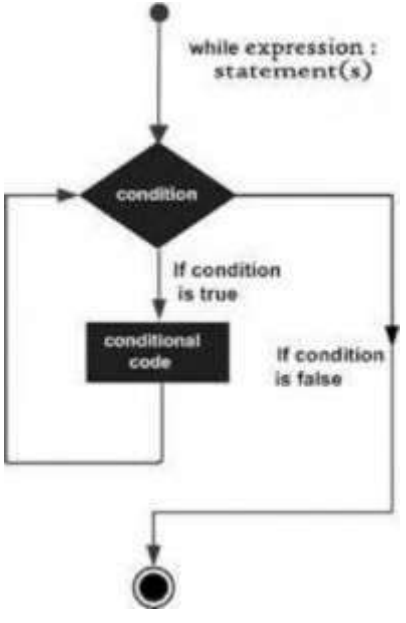
Παράδειγμα:

```
Αρχικοποίηση τιμής μεταβλητής
while όνομα_μεταβλητής <συνθήκη>:
      Εντολή_1
      Εντολή_2
       ….
      Εντολή_κ
```

Δομή Επανάληψης while (Όσο…)

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 δεν θα τερματίζεται.

Σημείωση 2: πριν το βρόχο while θα πρέπει να δώσουμε μία αρχική τιμή στη μεταβλητή που ελέγχει τη συνθήκη του βρόχου, ώστε ανάλογα με την τιμή να εκτελεστεί ή όχι ο βρόχος.



Άσκηση 13

Να δημιουργήσετε πρόγραμμα που να εμφανίζει όλους τους ζυγούς 3ψήφιους ακεραίους: 100, 102, …., 998

Άσκηση 14

Να δημιουργήσετε πρόγραμμα που να διαβάζει 2 αριθμούς και να εμφανίζει όλους τους ακεραίους ανάμεσα στον μικρότερο και τον μεγαλύτερο αριθμό.

Άσκηση 15

Να δημιουργήσετε πρόγραμμα που να διαβάζει 5 αριθμούς από το πληκτρολόγιο και να εμφανίζει τον μεγαλύτερο και τον μικρότερο από αυτούς.
Υπόδειξη:
Βάλτε τον 1ο αριθμό στην μεταβλητή max.
Κάθε επόμενο να τον συγκρίνετε με τον max και αν είναι μικρότερος να βάζετε στον max τον νέο αριθμό.
Το ίδιο κάνετε και με τον μικρότερο (min).
Η εύρεση min και max να γίνει με μια επανάληψη.

Άσκηση 16

Να διαβαστεί ένας ακέραιος αριθμός και να δημιουργηθεί το παρακάτω σχήμα αν δοθεί ο αριθμός 5, αν δοθεί το 10 θα έχει 10 «επίπεδα» κ.ο.κ.:

```
*
**
***
****
*****
```

Υπόδειξη: θα χρειαστούν 2 επαναλήψεις, η 1η θα αλλάζει γραμμή και η 2η θα τυπώνει “*”.

Χρησιμοποιήστε:

Για αλλαγή γραμμής την εντολή: `print(‘’)`

Για εμφάνιση αστεριών σε μια γραμμή: `print(‘*’ * a)` όπου a πόσα αστέρια θέλετε να τυπώσετε

Άσκηση 17

Χρησιμοποιώντας τα ψηφία x, y και z, από μια φορά το καθένα, ένας μαθητής βρήκε 6 τριψήφιους αριθμούς.

Να γραφεί πρόγραμμα που να διαβάζει (input) 3 ψηφία και να εμφανίζει (print) τους 6 αριθμούς που προκύπτουν καθώς και το άθροισμά τους.

Προσοχή : κάθε νέος αριθμός που φτιάχνετε να αποθηκεύεται σε 1 μεταβλητή.

Παράδειγμα εκτέλεσης:

```
Δώστε 1ο ψηφίο : 4
Δώστε 2ο ψηφίο : 3
Δώστε 3ο ψηφίο : 1
431, 413, 341, 314, 143, 134
Άθροισμα = 1776
```

Άσκηση 18

Εκτύπωση γραμμάτων λέξης

Να γράψετε πρόγραμμα που διαβάζει μία λέξη και εμφανίζει όλα τα γράμματά της το ένα κάτω από το άλλο. Επίσης, να ελέγχει αν το πρώτο γράμμα είναι το ίδιο με το τελευταίο.
Σε αυτήν την περίπτωση να εμφανίζει το μήνυμα «**OK**» αλλιώς «**NO**».
Θα χρειαστείτε την συνάρτηση len() που επιστρέφει το πλήθος των στοιχείων μιας συμβολοσειράς (string).

Παράδειγμα εκτέλεσης :

```
Δώστε λέξη: ΣΥΡΟΣ

Σ Υ Ρ Ο Σ
OK
```

Άσκηση 19

Ένα παιχνίδι με την προπαίδεια

Φτιάξτε ένα πρόγραμμα που:

α) ζητά από τον χρήστη πόσες φορές θέλει να παίξει

β) "δημιουργεί" 2 τυχαίους ακέραιους αριθμούς (11-19) και τους εμφανίζει στην οθόνη

γ) ζητά από τον χρήστη να πληκτρολογήσει το γινόμενό τους

Σε περίπτωση που ο χρήστης απαντήσει με τον σωστό αριθμό εμφανίζεται μήνυμα επιβράβευσης, ενώ σε αντίθετη περίπτωση εμφανίζεται το σωστό αποτέλεσμα.

Όλα τα παραπάνω βήματα επαναλαμβάνονται για όσες φορές επιθυμεί ο χρήστης να παίξει (δες ερώτημα α).

Επιπλέον, προγραμματίστε να ζητείται αρχικά το επίπεδο δυσκολίας, ως εξής:1-εύκολο, 2-δύσκολο. Η δυσκολία θα είναι ότι σε κάθε επίπεδο θα δημιουργούνται διαφορετικού εύρους αριθμοί (δες ερώτημα β).

Άσκηση 19

Στο τέλος θα εμφανίζονται το πλήθος σωστών και το πλήθος όλων των ερωτήσεων.
Παράδειγμα εκτέλεσης :

```
Δώσε επίπεδο δυσκολίας (1: εύκολο, 2: δύσκολο): 2
Πόσες φορές θέλεις να παίξεις; 18
12 x 11 = 6473
λάθος, η σωστή απάντηση είναι 132
.....................
Σκορ: 9/18
Ποσοστό σωστών απαντήσεων: 50%
```

Το ποσοστό είναι ένα κλάσμα: αριθμητής είναι το "πόσα" και παρονομαστής το "στα πόσα" επί 100.

Για εμφάνιση: Προσπάθειας / Σύνολο προσπαθειών χρησιμοποιήστε την λογική:
`print(str(count) + "/" + str(total))`

Άσκηση 20

Παιχνίδι Τυχερό 7

Να γράψετε πρόγραμμα που για 10 γύρους εμφανίζει 3 τυχαίους μονοψήφιους αριθμούς. Σε περίπτωση που 2 από αυτούς είναι το 7 τότε κερδίζεται ένας πόντος. Σε περίπτωση που τύχουν 3 7άρια τότε εμφανίζεται το μήνυμα «ΕΙΣΑΙ ΠΟΛΥ ΤΥΧΕΡΟΣ/Η» και τερματίζεται το παιχνίδι.

Κάθε φορά που τελειώνει ένας γύρος, περιμένει να πατηθεί ένα πλήκτρο ώστε να συνεχιστεί το παιχνίδι.

Στο τέλος, εμφανίζεται το πλήθος των πόντων καθώς και το ποσοστό επιτυχίας, δηλ πόσες φορές έτυχαν 2 7άρια στο σύνολο των γύρων που παίχτηκε το παιχνίδι.

Άσκηση 20

Παράδειγμα εκτέλεσης :

```
Λοιπόν ξεκινάμε πάτησε ένα πλήκτρο για να αρχίσει το παιχνίδι
7 8 9
Δυστυχώς δεν πέτυχες 2 7άρια
Πάτησε ένα πλήκτρο για τον επόμενο γύρο
7 8 7
Μπράβο πέτυχες 2 7άρια

…….
```

Αποτελέσματα

```
Σύνολο πόντων: 5

Ποσοστό: 5/10, 50%
```

Δομές Δεδομένων – Λίστα (List)

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 = [ ]
```

Λίστα (List)

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
```

Λίστα (List)

Απαρίθμηση στοιχείων λίστας - Δομή for

Η δομή for είναι μια δομή 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 μπορούμε να χρησιμοποιήσουμε στην for

Παράδειγμα:

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

Πλήθος στοιχείων - Συνάρτηση len

Δίνει το πλήθος των στοιχείων μιας λίστας.

Παράδειγμα:

```
if len(nums) == 0:
      print(“Η λίστα είναι κενή”)
```

Λίστα (List)

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‘)
```

Λίστα (List)

Τεμαχισμός λίστας (List Slicing)

Δημιουργεί μια νέα λίστα που αντιστοιχεί σε «τεμαχισμένο» τμήμα της αρχικής. 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, 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 2 έως και 4 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
```

Λίστα (List)

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
```

Λίστα (List)

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)
```

Λίστα (List)

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.
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('Γεωργία')
```

Λίστα (List)

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)
```

Λίστα (List)

Αντιστροφή στοιχείων
Μέθοδος reverse()

Παράδειγμα:
```
nums.reverse()
```

Αντιστροφή μπορεί να γίνει και με τεμαχισμό:
Παράδειγμα:
```
Nums[::-1]
```

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

Παράδειγμα:
```
random.shuffle(nums)
```

Λίστα (List)

Επιλογή ενός τυχαίου στοιχείου
Συνάρτηση choice() της βιβλιοθήκης random
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() της βιβλιοθήκης random
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)
```

Άσκηση 21

Εμφάνιση στοιχείων λίστας

Να γράψετε πρόγραμμα που να εμφανίζει όλα τα στοιχεία μια λίστας από το πρώτο μέχρι και το τελευταίο και κατόπιν αντίστροφα.

Παράδειγμα εκτέλεσης :

```
L=[“anna”, “kostas”, “efi”, “panagiotis”]

anna kostas efi panagiotis

panagiotis efi kostas anna
```

Άσκηση 22

Πρόγραμμα - Λεξικό

α) να βάλετε στην λίστα WORDS_ENG 20 αγγλικές λέξεις και στην λίστα WORDS_GR τις 20 ελληνικές λέξεις που αντιστοιχούν. (όλες με κεφαλαία)

β) το πρόγραμμα θα εμφανίζει μια τυχαία αγγλική λέξη και θα ζητά από τον χρήστη να πληκτρολογήσει την μετάφρασή της στα ελληνικά.

Παράδειγμα:
αν διαβάσει την λέξη «SUMMER» θα πρέπει να πληκτρολογηθεί η λέξη «ΚΑΛΟΚΑΙΡΙ».

Το πρόγραμμα θα τερματίζεται αν δοθεί ως λέξη το ”*”.
Στο τέλος θα εμφανίζει πόσες λέξεις «παίξατε», πόσες βρήκατε σωστά και πόσες λάθος.

Συναρτήσεις (Functions)

Οι συναρτήσεις είναι κομμάτια κώδικα που γράφονται μία φορά και χρησιμοποιούνται πολλές.
Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να το εκτελούμε καλώντας το όνομά τους, οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε. Αυτή η διαδικασία ονομάζεται κλήση (call) της συνάρτησης.
Η Python έχει αρκετές ενσωματωμένες συναρτήσεις, όπως οι int, range, len και άλλες

Ορίζουμε μια συνάρτηση χρησιμοποιώντας τη λέξη def, από το define που σημαίνει ορίζω. Στη συνέχεια ακολουθεί ένα όνομα της συνάρτησης. Μετά προσθέτουμε ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν μερικά ονόματα μεταβλητών και η γραμμή τελειώνει με άνω-κάτω τελεία (:).

```
def <όνομα συνάρτησης> ( [ {λίστα παραμέτρων} ] ):
    εντολές
  [ return <αποτέλεσμα> ]
```

Συναρτήσεις (Functions)

Οι αγκύλες [ ] σημαίνουν, πως, ότι περικλείεται μέσα σε αυτές, είναι προαιρετικό. Η λίστα των παραμέτρων μπορεί να είναι κενή. Μια συνάρτηση δεν είναι υποχρεωτικό να επιστρέφει κάποια τιμή.

Παραδείγματα:

```
def add(arg1, arg2):
    result = arg1+arg2
    return result
def times3(arg):
    ginomeno = 3*arg
    return ginomeno
```

```
>>> add(10, 18)
28
>>> add(10, 18.5)
28.5
>>> times3(10)
30
```

Συναρτήσεις (Functions)

Μπορούμε να συνδυάσουμε την κλήση συναρτήσεων, το αποτέλεσμα μιας συνάρτησης μπορεί να αποτελέσει τα δεδομένα εισόδου μιας άλλης.

Παρατηρούμε ότι έχουμε ορίσει μία συνάρτηση, η οποία δέχεται όλους τους τύπους των ορισμάτων και η **λειτουργία της αλλάζει δυναμικά, ανάλογα** με τα ορίσματα, αν δοθούν αριθμοί τους προσθέτει, ενώ αν δοθούν αλφαριθμητικά, τα συνενώνει.

```
>>> times3(2.5)
7.5
>>> times3('python')
'python python python'
>>> times3( times3( 9 ') )
'9 9 9 9 9 9 9 9 9 '
>>> times3(add(add('ab','ba'),' '))
'abba  abba  abba'
```

Συναρτήσεις (Functions)

Κατηγορίες συναρτήσεων

Οι συναρτήσεις μπορούν να κατηγοριοποιηθούν με πολλούς τρόπους. Μια πρώτη κατηγορία συναρτήσεων είναι:

α) αυτές οι οποίες δεν τροποποιούν το αντικείμενο στο οποίο εφαρμόζονται, όπως:

```
>>> a = 'Python'
>>> print a.upper( )
PYTHON
>>> print a
Python
```

β) αυτές που μπορούν να τροποποιήσουν το αντικείμενο στο οποίο καλούνται, όπως:

```
>>> b = [ 'a' , 'b' , 'c' , 'd' ]
>>> print b.append('e')
None
>>> print b
['a', 'b', 'c', 'd', 'e']
```

Συναρτήσεις (Functions)

Παράμετροι συναρτήσεων

Μια συνάρτηση δέχεται δεδομένα μέσω των παραμέτρων και επιστρέφει τα αποτελέσματα μέσω άλλων ή και των ίδιων παραμέτρων στο πρόγραμμα ή σε άλλη συνάρτηση.

Οι παράμετροι καθορίζονται μέσα στις παρενθέσεις στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα. Όταν καλούμε τη συνάρτηση, περνάμε και τις τιμές με τον ίδιο τρόπο, οι οποίες **ονομάζονται ορίσματα**. Τα ορίσματα δεν είναι υποχρεωτικά, δηλ. μπορεί να φτιάξουμε μια συνάρτηση χωρίς ορίσματα π.χ. η συνάρτηση help() που εμφανίζει ένα συγκεκριμένο κείμενο.

```
def ginomeno (a,b):
    x = a * b
    return x
print ginomeno(5,10)
```

Συναρτήσεις (Functions)

Σημείωση: Η μεταβίβαση παραμέτρων στην Python λειτουργεί με τέτοιο τρόπο, ώστε οποιαδήποτε αλλαγή στις παραμέτρους εντός της συνάρτησης **δεν έχει καμία επίδραση** στα ορίσματα-μεταβλητές που έχουν οριστεί εκτός της συνάρτησης, όπως φαίνεται παρακάτω:

```
def increment(a, b):
    a = a + 1
    b = b + 1
    return a+b
# τέλος συνάρτησης
z = 1
w = 2
```

```
q = increment( z, w )
print z, w, q
1 2 5
```

Τα ορίσματα z, w δεν αλλάζουν τιμή, παρόλο που οι αντίστοιχοι παράμετροι αυξάνονται εντός της συνάρτησης increment, όπως φαίνεται και από την τιμή που επιστρέφει η συνάρτηση και καταχωρείται στην q.

Άσκηση 23

Να φτιάξετε μια συνάρτηση (function) με όνομα max3 που δέχεται ως παραμέτρους 3 αριθμούς και μας επιστρέφει (return) τον μεγαλύτερο από αυτούς.

Να γράψετε ένα πρόγραμμα που να διαβάζει 3 αριθμούς και χρησιμοποιώντας την συνάρτηση max3 εμφανίζει τον μεγαλύτερο.

Άσκηση 24

Να φτιάξετε μια συνάρτηση με όνομα compute_grade η οποία δέχεται έναν βαθμό ως παράμετρο και επιστρέφει την αντίστοιχη αξιολόγηση ως string με βάση τον ακόλουθο πίνακα:

Βαθμός	Αξιολόγηση
>= 18.1	Άριστα
>= 16.1	Λίαν καλώς
>= 13.1	Καλώς
>= 09.5	Σχεδόν καλώς
>= 05.1	Ανεπαρκώς
<= 05.0	Κακώς

Να γράψετε ένα πρόγραμμα που να διαβάζει μια βαθμολογία μεταξύ 0.0 και 20.0. Εάν η βαθμολογία είναι εκτός των ορίων, να εκτυπώνει μήνυμα σφάλματος. Εάν η βαθμολογία είναι μεταξύ 0.0 και 20.0, να εκτυπώνει μια αξιολόγηση χρησιμοποιώντας την συνάρτηση compute_grade.

Επεξεργασία Αρχείων

Ένα αρχείο κειμένου μπορεί να θεωρηθεί ως μια ακολουθία γραμμών

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/

Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772

Επεξεργασία Αρχείων

Ο Χαρακτήρας νέα γραμμή (newline)

Χρησιμοποιούμε έναν ειδικό χαρακτήρα που ονομάζεται «νέα γραμμή» για να υποδείξουμε πότε τελειώνει μια γραμμή. Συμβολίζουμε τη νέα γραμμή με \n. Η νέα γραμμή είναι ένας χαρακτήρας - όχι δύο

Παράδειγμα:

```
>>> stuff = 'Hello\nWorld!’
>>> stuff
'Hello\nWorld!'
>>> print(stuff)
Hello
World!
>>> stuff = 'X\nY'
>>> print(stuff)
X
Y
>>> len(stuff)
3
```

Επεξεργασία Αρχείων

Ένα αρχείο κειμένου έχει τον χαρακτήρα “νέας γραμμής” στο τέλος κάθε γραμμής

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008\n
Return-Path: <postmaster@collab.sakaiproject.org>\n
Date: Sat, 5 Jan 2008 09:12:18 -0500\n
To: source@collab.sakaiproject.org\n
From: stephen.marquard@uct.ac.za\n
Subject: [sakai] svn commit: r39772 - content/branches/\n
\n
Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772\n

Επεξεργασία Αρχείων

Άνοιγμα αρχείου – Η συνάρτηση open()

```
handle = open(όνομα_αρχείου, λειτουργία)
```

Επιστρέφει έναν περιγραφέα που χρησιμοποιείται για να χειριστούμε το αρχείο.
Η λειτουργία είναι “r” για να διαβάσουμε το αρχείο και “w” για να γράψουμε στο αρχείο

Παράδειγμα:

```
fhand = open('mbox.txt', 'r')
```

Επεξεργασία Αρχείων

Ανάγνωση Αρχείων

Ένα ανοιχτό αρχείο μπορεί να αντιμετωπιστεί ως μια ακολουθία συμβολοσειρών όπου κάθε γραμμή στο αρχείο είναι μια συμβολοσειρά

Μπορούμε να χρησιμοποιήσουμε την εντολή for για να προσπελάσουμε μια ακολουθία

Παράδειγμα:

```
xfile = open('mbox.txt', 'r')
for color in xfile:
    print(color)
```

Επεξεργασία Αρχείων

Μέτρηση Γραμμών ενός Αρχείου
Ανοίξτε ένα αρχείο μόνο για ανάγνωση
Χρησιμοποιήστε μια επανάληψη for για να διαβάσετε κάθε γραμμή
Μετρήστε τις γραμμές και εκτυπώστε το πλήθος των γραμμών

Παράδειγμα:

```
xfile = open('mbox.txt’, ‘r’)
πλήθος = 0
for line in xfile:
    πλήθος = πλήθος + 1
print('Πλήθος Γραμμών:', πλήθος)
```

Επεξεργασία Αρχείων

Διάβασμα *Ολόκληρου* Αρχείου – Η μέθοδος read()
Μπορούμε να διαβάσουμε ολόκληρο το αρχείο (όλους τους χαρακτήρες μαζί με τις νέες γραμμές) σε μια συμβολοσειρά

Παράδειγμα:

```
>>> fhand = open('mbox-short.txt', ‘r’)
>>> inp = fhand.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
From stephen.marquar
```

Επεξεργασία Αρχείων

Γράψιμο σε αρχεία – Η λειτουργία "w"

Για να γράψουμε ένα αρχείο, πρέπει να το ανοίξουμε με τη λειτουργία "w" ως δεύτερη παράμετρο:

Παράδειγμα:

```
>>> fout = open('output.txt', 'w')
>>> print(fout)
<_io.TextIOWrapper name='output.txt' mode='w' encoding='cp1252'>
```

Επεξεργασία Αρχείων

Γράψιμο σε αρχεία – Η μέθοδος write()

Εάν το αρχείο υπάρχει ήδη, το άνοιγμα του σε λειτουργία εγγραφής διαγράφει τα προηγούμενα δεδομένα και τα αντικαθιστά με τα καινούρια, οπότε πρέπει να είμαστε προσεκτικοί! Εάν το αρχείο δεν υπάρχει, δημιουργείται ένα νέο.

Η μέθοδος write του αντικειμένου χειρισμού αρχείου τοποθετεί δεδομένα στο αρχείο, επιστρέφοντας τον αριθμό των χαρακτήρων που γράφτηκαν.

Παράδειγμα:

```
>>> γραμμή1 = "Αυτό το κλαδί, είναι\n"
>>> fout.write(γραμμή1)
21
```

Επεξεργασία Αρχείων

Κλείσιμο αρχείων – Η μέθοδος close()

Όταν ολοκληρώσουμε την επεξεργασία, πρέπει να κλείσουμε το αρχείο για να βεβαιωθούμε ότι και το τελευταίο bit δεδομένων είναι γραμμένο στο δίσκο.

Παράδειγμα:

```
>>> fout.close()
```

Άσκηση 25

Να γράψετε ένα πρόγραμμα που να διαβάζει ένα αρχείο και να εκτυπώνει τα περιεχόμενα του αρχείου (γραμμή προς γραμμή) όλα με κεφαλαία.

Παράδειγμα εκτέλεσης:

```
Δώστε το όνομα του αρχείου: mbox-short.txt
FROM STEPHEN.MARQUARD@UCT.AC.ZA SAT JAN  5 09:14:16 2008
RETURN-PATH: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>
RECEIVED: FROM MURDER (MAIL.UMICH.EDU [141.211.14.90])
     BY FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA;
     SAT, 05 JAN 2008 09:14:16 -0500
```

Άσκηση 26

α) Να γράψετε πρόγραμμα που ανοίγει το αρχείο workfile.txt για γράψιμο και γράφει αυτό:

line 0
line 1
line 2
line 3
line 4

β) Να γράψετε πρόγραμμα που διαβάζει από το πληκτρολόγιο 5 ονόματα, ανοίγει το αρχείο names.txt για γράψιμο και γράφει σε αυτό τα ονόματα.

Θα χρειαστείτε την εντολή `f = open('workfile.txt', 'w')` και την αντίστοιχη εντολή για το αρχείο names.txt

Μην ξεχάσετε να κλείσετε το αρχείο στο τέλος με την εντολή `f.close()`

Άσκηση 27

α) Να γράψετε πρόγραμμα που να διαβάζει όλες τις γραμμές του αρχείου colors.txt και να τις γράφει στην λίστα L. Το αρχείο colors.txt θα το φτιάξετε από το σημειωματάριο των windows και μέσα θα αυτό θα γράψετε σε κάθε γραμμή ένα χρώμα. Μπορείς να γράψετε όσο χρώματα θέλετε.

β) Να εμφανίζει το μήκος κάθε γραμμής χωρίς το '\n' που το ακολουθεί και

γ) έπειτα να εμφανίζει τις γραμμές σε αντίστροφη σειρά