ΣΥΝΑΡΤΗΣΕΙΣ

Τι είναι οι Συναρτήσεις

- Πρόκειται για μικρά κομμάτια κώδικα τα οποία είναι κλεισμένα κάτω από κάποιο όνομα. Το όνομα συνοδεύεται πάντα από ένα ζευγάρι παρενθέσεων βλ. **range()**.

- Η **κλήση** μιας συνάρτησης γίνεται μέσα από το κυρίως πρόγραμμα με απλή αναφορά στο όνομά της. Κάθε φορά που μια συνάρτηση καλείται ο υπολογιστής εκτελεί τον κώδικα που βρίσκεται κλεισμένος στο εσωτερικό της.

- Υπάρχουν φορές που ο κώδικας αυτός κάνει κάποιους υπολογισμούς και καταλήγει σε ένα αποτέλεσμα. **ΠΡΟΣΟΧΗ!** Το αποτέλεσμα αυτό επιστρέφεται στη συνέχεια πίσω στο κυρίως πρόγραμμα στο σημείο που βρισκόταν το όνομα της συνάρτησης (η κλήση στη συνάρτηση) αντικαθιστώντας το όνομα με το αποτέλεσμα.

Τι είναι οι Συναρτήσεις - Ορίσματα

- **ΠΡΟΣΟΧΗ!** Πολλές φορές θα χρειαστεί να δώσουμε στη συνάρτηση κάποια δεδομένα με τα οποία θα δουλέψει.

- Για παράδειγμα στην περίπτωση της **range()** που παράγει μία λίστα τιμών, θα πρέπει να της δώσουμε το νούμερο εκκίνησης, το νούμερο τερματισμού και το βήμα προκειμένου να καταλάβει τι είδους λίστα επιθυμούμε να παράγει. Τα δεδομένα αυτά τα «**περνάμε**» στη συνάρτηση (της τα δίνουμε) **κατά την κλήση** τοποθετώντας τα μέσα στις παρενθέσεις που συνοδεύουν το όνομά της και χωρισμένα μεταξύ τους με κόμματα π.χ. **range(1,20,2)**

- Τα δεδομένα αυτά λέγονται **ορίσματα** (**arguments**).

Τι είναι οι Συναρτήσεις

- Οι συναρτήσεις είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. 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.

- Οι συναρτήσεις μπορεί να είναι είτε **έτοιμες** από τη γλώσσα προγραμματισμού, είτε να **δημιουργούνται από εμάς**.

ΒΑΣΙΚΕΣ ΕΝΣΩΜΑΤΩΜΕΝΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Διαθέσιμες ανά πάσα στιγμή

Συνάρτηση float()

- Η **float()** μετατρέπει ακεραίους σε δεκαδικούς αριθμούς.

```
>>> print float(10)
10.0
```

Προσέξτε ότι η συνάρτηση float(10) καλείται περνώντας της τον αριθμό 10 για να τον μετατρέψει σε πραγματικό αριθμό. Η συνάρτηση παράγει το 10.0 το οποίο επιστρέφει στο πρόγραμμα στο σημείο που βρισκόταν η κλήση αντικαθιστώντας αυτή με το αποτέλεσμα.
Έτσι η εντολή **print float(10)** μετά την εκτέλεση της συνάρτησης παίρνει τη μορφή **print 10.0** η οποία θα μας εμφανίσει στην οθόνη τον πραγματικό αριθμό **10.0**

Χρήσεις :

```
>>> print 3/2
1
>>> print 3/float(2)
1.5
```

Συνάρτηση float()

- Επίσης η **float()** μπορεί να δεχτεί σαν όρισμα ένα αλφαριθμητικό - string αποτελούμενο από αριθμούς με τελεία το οποίο μετατρέπει σε πραγματικό αριθμό.

```
>>> float("123.45")
123.45

>>> float("123")
123.0
```

Συνάρτηση int()

- Η **int()** δέχεται οποιαδήποτε αριθμητική τιμή και τη μετατρέπει σε ακέραιο κόβοντας τα δεκαδικά ψηφία, αν υπάρχουν.

```
>>> int(123.456)
123
```

Συνάρτηση int()

- Αν θέλουμε οπωσδήποτε να διαβάσουμε από το πληκτρολόγιο έναν ακέραιο αριθμό με τη συνάρτηση **input()**, θα πρέπει να χρησιμοποιούμε και τη συνάρτηση **int(**) με τον τρόπο που φαίνεται πιο κάτω.

```
α=int(input('Δώσε ακέραιο αριθμό :'))
```

Συνάρτηση int()

- Η **input()** είναι κι αυτή μια από τις έτοιμες συναρτήσεις που διαθέτει η Python. Την καλούμε περνώντας της μέσα στις παρενθέσεις ένα string. Η συνάρτηση ξεκινά εμφανίζοντας το string στην οθόνη του χρήστη. Το string περιέχει κείμενο που τον πληροφορεί για το είδος του δεδομένου που πρέπει να δώσει στην παρούσα στιγμή. Στο παράδειγμά μας του ζητά να πληκτρολογήσει έναν ακέραιο αριθμό. Μόλις ο χρήστης δώσει τον αριθμό η συνάρτηση τον παίρνει και τον αποδίδει πίσω στο κυρίως πρόγραμμα στο σημείο που βρισκόταν η κλήση αντικαθιστώντας την.

- Αν ο χρήστης πληκτρολογήσει **40.5** αυτό το νούμερο θα επιστραφεί από την συνάρτηση μέσα στις παρενθέσεις της **int()** στη θέση της κλήσης - **input('Δώσε ακέραιο αριθμό :')** – και έτσι χωρίς να το αντιληφθούμε η εντολή **a=int(input('Δώσε ακέραιο αριθμό :'))** θα πάρει τη μορφή **a=int(40.5).**

- Ο υπολογιστής στη συνέχεια θα καλέσει την συνάρτηση **int(40.5)** για να μετατρέψει σε ακέραιο τον αριθμό **40.5**. Η συνάρτηση κόβει τα δεκαδικά ψηφία και επιστρέφει το **40** στη θέση της κλήσης. Η εντολή παίρνει την μορφή **a=40**. Ο υπολογιστής δεσμεύει μια θέση στη μνήμη την ονομάζει **a** και καταχωρεί μέσα το **40**.

- Μ' αυτό τον τρόπο ο αριθμός που παραλαμβάνεται από το χρήστη μέσω της **input()** οδηγείται στην **int()** για να μετατραπεί σε ακέραιο και στη συνέχεια καταχωρείται στη μεταβλητή που αναφέρεται στα αριστερά της έκφρασης.

Συνάρτηση int()

- Η **int()** μπορεί να δεχτεί και ένα string αποτελούμενο από αριθμούς δίχως τελεία προκειμένου να το μετατρέψει σε ακέραιο αριθμό.

```
>>> print int("34")
34
```

Συνάρτηση str()

- Η **str()** δέχεται οποιαδήποτε αριθμητική τιμή και την μετατρέπει σε συμβολοσειρά.

```
>>> str(40.5)
'40.5'
```

Συνάρτηση abs()

- Η **abs()** επιστρέφει την απόλυτη τιμή ενός αριθμού.

```
>>> abs(-45)
45
```

Συνάρτηση pow()

- Η **pow(a,b)** επιστρέφει την δύναμη του a υψωμένη στην b.

```
>>> pow(2,3)
8
```

2^3

Συνάρτηση divmod()

- Η **divmod(x,y)** επιστρέφει το ακέραιο πηλίκο και το ακέραιο υπόλοιπο της διαίρεσης x/y.

```
>>> divmod(10,3)
(3, 1)
```

Συνάρτηση round()

Για στρογγυλοποίηση αποτελεσμάτων μπορώ να χρησιμοποιώ την ενσωματωμένη συνάρτηση **round()**. Αυτή δέχεται δύο παραμέτρους τον αριθμό που πρέπει να στρογγυλοποιηθεί και το πλήθος των δεκαδικών ψηφίων.

Έτσι αν έχω τον αριθμό **12.789** και θέλω να τον εμφανίσω στρογγυλοποιημένο με ακρίβεια 2 δεκαδικών ψηφίων καλώ την round ως εξής:

```
>>> round(12.789,2)
12.79
```

Για στρογγυλοποίηση με ακρίβεια 1 δεκαδικού ψηφίου:

```
>>> round(12.789,1)
12.8
```

Για στρογγυλοποίηση δίχως δεκαδικά ψηφία:

```
>>> round(12.789,0)
13.0
```

ΕΞΩΤΕΡΙΚΕΣ ΒΙΒΛΙΟΘΗΚΕΣ

Εξωτερικές βιβλιοθήκες

- Εκτός από τις ενσωματωμένες συναρτήσεις υπάρχει πληθώρα εξωτερικών βιβλιοθηκών με συλλογή από έτοιμες συναρτήσεις όπως η Μαθηματική βιβλιοθήκη & οι Βιβλιοθήκες γραφικών.
- **ΠΡΟΣΟΧΗ!** Οι βιβλιοθήκες πρέπει πρώτα να εισαχθούν στο πρόγραμμά μας μέσω της εντολής **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 βιβλιοθήκης και το όνομα της συνάρτησης, χωρισμένα με μια τελεία, μορφή που ονομάζεται *συμβολισμός με τελεία* ***(dot notation)***.

Βιβλιοθήκη math – Συνάρτηση sqrt() - Σταθερά pi


```
>>> import math
>>> riza = math.sqrt(4)
>>> print riza
2.0
>>> x = math.pi
>>> print x
3.14159265359
```

Βιβλιοθήκη random – 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 χρησιμοποιούνται κατά κόρον, είναι η **randint** και η **randrange**, που επιστρέφουν τυχαίους ακέραιους αριθμούς εντός κάποιων ορίων.

Τα όρια στην **randrange** ακολουθούν την ίδια λογική με την **range** της Python, ενώ η **randint** συμπεριλαμβάνει και το άνω άκρο.

Βιβλιοθήκη random – Παραγωγή τυχαίων αριθμών


```
import random
number = random.randint(1, 10)
        # επιστρέφει έναν τυχαίο αριθμό στο διάστημα [1, 10]

number = random.randrange(1,10)
        # επιστρέφει έναν τυχαίο αριθμό στο διάστημα [1, 9]

number = random.randrange(10)
        # επιστρέφει έναν τυχαίο αριθμό στο διάστημα [0, 9]
```

ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Καλές Πρακτικές

Καλές Πρακτικές

- Δίνουμε ένα χαρακτηριστικό τίτλο στο πρόγραμμα με τη μορφή σχολίων, τα οποία ξεκινάνε με το σύμβολο #. Αυτό, παρότι δεν είναι αναγκαίο, είναι ιδιαίτερα χρήσιμο.
- 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.
- Επιλέγουμε τους κατάλληλους τελεστές.
- Να χρησιμοποιούμε τα ίδια εισαγωγικά (μονά εισαγωγικά με μονά, διπλά εισαγωγικά με διπλά) για μια συμβολοσειρά.
- Προσθέτουμε, όπου κρίνουμε χρήσιμο, επεξηγηματικά σχόλια μέσα στον κώδικα.
- Δίνουμε ονόματα μεταβλητών που έχουν σχέση με τη χρήση τους.

ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΤΙΣ ΔΙΚΕΣ ΜΑΣ ΣΥΝΑΡΤΗΣΕΙΣ

Δημιουργώντας δικές μας Συναρτήσεις

- Για να ορίσουμε μια δική μας συνάρτηση χρησιμοποιούμε τη χαρακτηριστική λέξη **def**, ακολουθεί ένα όνομα που ταυτοποιεί την εκάστοτε συνάρτηση και ένα ζευγάρι παρενθέσεων που μπορούν να περικλείουν ονόματα μεταβλητών, ενώ η γραμμή τελειώνει με άνω και κάτω τελεία (:). Κάτω από τη γραμμή αυτή τοποθετούνται, σε εσοχή, οι εντολές που καθορίζουν τη λειτουργία της συνάρτησης.

```
# ορισμός συνάρτησης
def hello() :
        print 'Hello World!'

# κυρίως πρόγραμμα
hello()     # κλήση στη συνάρτηση με αναφορά στο όνομά της
```

Κλήση Συνάρτησης μέσα από Συνάρτηση


```
# ορισμός συνάρτησης
def  hello() :
       print  'Hello  World!'

# ορισμός συνάρτησης
def  epanelave_hello() :
       hello()
       hello()

# ορισμός συνάρτησης
def  epanelave_4fores() :
       epanelave_hello()
       epanelave_hello()

# κυρίως πρόγραμμα
epanelave_4fores()     # κλήση στη συνάρτηση με αναφορά στο όνομά της
```

Ο ρόλος των Εσοχών

- Οι εσοχές στην αρχή των εντολών είναι πολύ σημαντικές στην Python. Ο κενός χώρος πριν από μια εντολή και γενικότερα η στοίχιση των εντολών, δεν είναι μόνο θέμα αισθητικής, όπως σε άλλες γλώσσες, αλλά θέμα ουσίας που μπορεί να αλλάξει το αποτέλεσμα του προγράμματος, όπως φαίνεται παρακάτω:

```
def welcome():
    name=raw_input("Give your name: ")
    print "Welcome",name

welcome()
```

Έξοδος:

```
Give your name: Peter
Welcome Peter
```

```
def welcome():
    name=raw_input("Give your name: ")
print "Welcome",name

welcome()
```

Έξοδος:

```
Traceback (most recent call last):
  File "C:/Python27/test.py", line 4, in <module>
    print "Welcome",name
NameError: name 'name' is not defined
```

*** Εδώ η print έχει βγει εκτός συνάρτησης παίζοντας το ρόλο της πρώτης εντολής του κυρίως προγράμματος. Ο υπολογιστής ξεκινά με την εκτέλεσή της, η μεταβλητή name όμως δεν έχει οριστεί και γι αυτό προκύπτει και το σχετικό σφάλμα.**

Ορίσματα

- Τις περισσότερες φορές θα χρειαστεί να δημιουργήσουμε συναρτήσεις τις οποίες θα πρέπει να καλούμε για να μας κάνουν κάποια δουλειά, αλλά ταυτόχρονα θα πρέπει να τους **«περνάμε»** - να τους δίνουμε - κάποια δεδομένα τα οποία θα υποβάλουν σε επεξεργασία. Δηλ. θα χρειαστεί να δημιουργήσουμε συναρτήσεις που να συμπεριφέρονται όπως οι **float()**, **range()** κλπ. που είδαμε πιο πριν.
- Στην περίπτωση της **float()** η οποία μετατρέπει έναν αριθμό σε πραγματικό θα πρέπει να της δώσουμε τον ακέραιο αριθμό που θέλουμε να μας μετατρέψει σε πραγματικό. Στην περίπτωση της **range()** που παράγει μία λίστα τιμών, θα πρέπει να της δώσουμε το νούμερο εκκίνησης, το νούμερο τερματισμού και το βήμα προκειμένου να καταλάβει τι είδους λίστα επιθυμούμε να παράγει.
- Τα δεδομένα τα «**περνάμε**» στη συνάρτηση (της τα δίνουμε) **κατά την κλήση** τοποθετώντας τα μέσα στις παρενθέσεις που συνοδεύουν το όνομά της π.χ. **float(32)**. Αν η συνάρτηση πρέπει να πάρει περισσότερα από ένα δεδομένα όπως στην περίπτωση της range() τότε τα τοποθετούμε μέσα στην παρένθεση χωρισμένα μεταξύ τους με κόμματα. π.χ. **range(1,20,2)**
- Τα δεδομένα αυτά λέγονται **ορίσματα** (**arguments**).

Παράμετροι Συναρτήσεων

- Η συνάρτηση απ' την πλευρά της θα πρέπει να πάρει αυτά τα δεδομένα και να τα φυλάξει προσωρινά στη μνήμη. Η φύλαξη γίνεται σε μεταβλητές. Θα χρειαστούν τόσες μεταβλητές όσα τα δεδομένα που θα της δοθούν. Οι μεταβλητές αυτές πρέπει να έχουν ονόματα.
- Τα **ονόματα** των **μεταβλητών** που θα φιλοξενήσουν τα δεδομένα που θα πάρει η συνάρτηση από το κυρίως πρόγραμμα λέγονται **παράμετροι**. Κατά τον **ορισμό** της συνάρτησης οι παράμετροι τοποθετούνται μέσα στις παρενθέσεις που ακολουθούν το όνομά της, χωρισμένες μεταξύ τους με κόμματα.
- Κατά την **κλήση** της συνάρτησης και μέσα στις παρενθέσεις τοποθετούνται χωρισμένα με κόμμα τα δεδομένα **(ορίσματα)** που πρέπει να περαστούν στη συνάρτηση. Πρώτο τοποθετείται το όρισμα που αντιστοιχεί στην πρώτη παράμετρο, στη συνέχεια το όρισμα που αντιστοιχεί στη δεύτερη παράμετρο κοκ.

Επιστροφή τιμής μέσω συνάρτησης – Εντολή Return

- Αν η συνάρτηση χρειάζεται να επιστρέψει κάποιο αποτέλεσμα πίσω στο κυρίως πρόγραμμα αυτό γίνεται με χρήση της εντολής **return**. Η συνάρτηση καλείται, εκτελείται και το αποτέλεσμα επιστρέφεται πίσω στο κυρίως πρόγραμμα αντικαθιστώντας την κλήση της συνάρτησης. Προκειμένου να μη χαθεί το αποτέλεσμα θα πρέπει να φυλαχθεί σε μία μεταβλητή. Γι' αυτό συνήθως χρησιμοποιείται στο κυρίως πρόγραμμα μια έκφραση της μορφής: **<όνομα μεταβλητής> = <κλήση συνάρτησης>**

Συναρτήσεις & Μνήμη

- Ο **παράμετροι** είναι τα **ονόματα των μεταβλητών** που θα φιλοξενήσουν τις τιμές που θα πάρει η συνάρτηση από το κυρίως πρόγραμμα.
- Οι μεταβλητές αυτές **δημιουργούνται** στη μνήμη **τη στιγμή** που **καλείται** η συνάρτηση και γεμίζουν με τις τιμές που περνάνε στη συνάρτηση όταν αυτή καλείται.
- **ΠΡΟΣΟΧΗ!** Ο κώδικας της συνάρτησης κάθεται ανενεργός. Αν η συνάρτηση δεν κληθεί ο κώδικάς της δεν πρόκειται να εκτελεστεί. Μόλις η συνάρτηση κληθεί τότε σε μια άλλη περιοχή της μνήμης **διαφορετική** από αυτή του κυρίως προγράμματος δεσμεύονται σταδιακά θέσεις αρχικά για τις παραμέτρους και στη συνέχεια για άλλες μεταβλητές που πιθανόν να απαιτεί ο κώδικας της συνάρτησης.
- **ΠΡΟΣΟΧΗ!** Μόλις ολοκληρωθεί η εκτέλεση του κώδικα της συνάρτησης ο χώρος μνήμης που είχε δεσμευθεί για τις παραμέτρους και άλλες μεταβλητές που πιθανόν αυτή να χρειάστηκε αποδεσμεύεται και όλες οι μεταβλητές, παράμετροι κλπ. διαγράφονται.

Γενική Μορφή Συνάρτησης


```
# ορισμός συνάρτησης

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


#κυρίως πρόγραμμα

[<όνομα μεταβλητής> = ]<όνομα συνάρτησης> ( [ {λίστα ορισμάτων} ] )
```

Οι αγκύλες σημαίνουν πως ότι περικλείεται μέσα σ αυτές είναι προαιρετικό. Η λίστα των παραμέτρων μπορεί να είναι κενή. Μία συνάρτηση δεν είναι υποχρεωτικό να επιστρέφει κάποια τιμή.

Άσκηση

- Να γραφτεί πρόγραμμα που α) διαβάζει δύο τυχαίους αριθμούς που δίνονται από το πληκτρολόγιο β) καλεί μία συνάρτηση με το όνομα **add()** η οποία βρίσκει το άθροισμά τους και το επιστρέφει πίσω στο κυρίως πρόγραμμα και γ) εμφανίζει το αποτέλεσμα στην οθόνη του χρήστη.

Άσκηση

- **Ερώτημα Α.** Διάβασμα δύο τυχαίων αριθμών που δίνονται από το πληκτρολόγιο

```
# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
```

Άσκηση

- **Ερώτημα Β.** Δημιουργία συνάρτησης add()
- Η συνάρτηση πρέπει να παίρνει από το κυρίως πρόγραμμα δύο αριθμούς και στη συνέχεια να τους προσθέτει. Άρα θα χρειαστεί δύο μεταβλητές για να φυλάξει αυτούς τους αριθμούς - δύο παραμέτρους - έστω **x**, **y** τα ονόματά τους. Ακολουθεί ο **ορισμός** της συνάρτησης μαζί με τις παραμέτρους τοποθετημένες μέσα στις παρενθέσεις που ακολουθούν το όνομά της και χωρισμένες μεταξύ τους με κόμματα.
- Η συνάρτηση θα χρειαστεί μία επιπλέον μεταβλητή την **s** για να φυλάξει το άθροισμα των 2 τιμών που έχουν φυλαχτεί στις **x** & **y**. Τέλος το περιεχόμενο της **s** αποδίδεται πίσω στο κυρίως πρόγραμμα μέσω της εντολής **return**.

```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
```

Άσκηση

- **Ερώτημα Β.** Δημιουργία συνάρτησης add()
- Για να μπορέσει να εκτελεστεί ο κώδικας της συνάρτησης θα πρέπει να την καλέσουμε μέσα από το κυρίως πρόγραμμα. Η κλήση γίνεται μέσα από το κυρίως πρόγραμμα με αναφορά στο όνομά της ενώ μέσα στις παρενθέσεις θα πρέπει να δώσουμε τα δύο νούμερα που θέλουμε να μας προσθέσει. Τα νούμερα έχουν φυλαχτεί στις μεταβλητές a, b οπότε θα πρέπει να κάνουμε αναφορά στα ονόματα των μεταβλητών. Η κλήση λοιπόν θα γίνει ως εξής: **add(a,b)**

```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)      # κλήση της συνάρτησης
```

Άσκηση

- **Ερώτημα Β.** Δημιουργία συνάρτησης add()
- Προσέξτε ότι το νούμερο που επιστρέφει πίσω η συνάρτηση θα αντικαταστήσει την κλήση της δηλ. την έκφραση **add(a,b)** στο κυρίως πρόγραμμα. Το νούμερο αυτό δεν πρέπει να χαθεί γι αυτό οδηγείται προς φύλαξη σε μια νέα μεταβλητή με το όνομα **Sum**.

```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)      # κλήση της συνάρτησης
```

Άσκηση

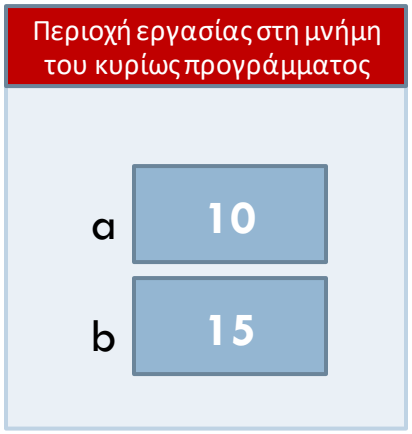
- **Ερώτημα Γ.** Εμφάνιση αποτελέσματος.
- Ακολουθεί το τελικό πρόγραμμα

```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)      # κλήση της συνάρτησης
print Sum
```

Άσκηση – Διεργασίες στη Μνήμη

- Ας δούμε τις διεργασίες που λαμβάνουν χώρα στη μνήμη κατά την εκτέλεση του προγράμματος. Ο υπολογιστής ξεκινά εκτελώντας τις δύο πρώτες εντολές του κυρίως προγράμματος. Ζητούνται δύο αριθμοί από το χρήστη. Έστω ότι αυτός δίνει το 10 και το 15. Ο υπολογιστής τους παίρνει και τους φυλά στη μνήμη σε δύο μεταβλητές με τα ονόματα a, b αντίστοιχα.

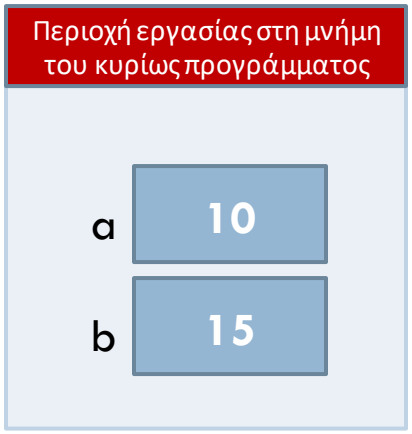
```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```



Άσκηση – Διεργασίες στη Μνήμη

- Στη συνέχεια ο υπολογιστής βλέπει την έκφραση **Sum=add(a,b)** . Δεξιά απ' τον τελεστή καταχώρησης βλέπει την κλήση στη συνάρτηση add() . Επίσης βλέπει ότι μέσα στην παρένθεση υπάρχουν δύο δεδομένα που πρέπει να σταλούν στη συνάρτηση - τα περιεχόμενα των μεταβλητών a και b. Η έκφραση παίρνει τη μορφή **Sum=add(10,15)**.

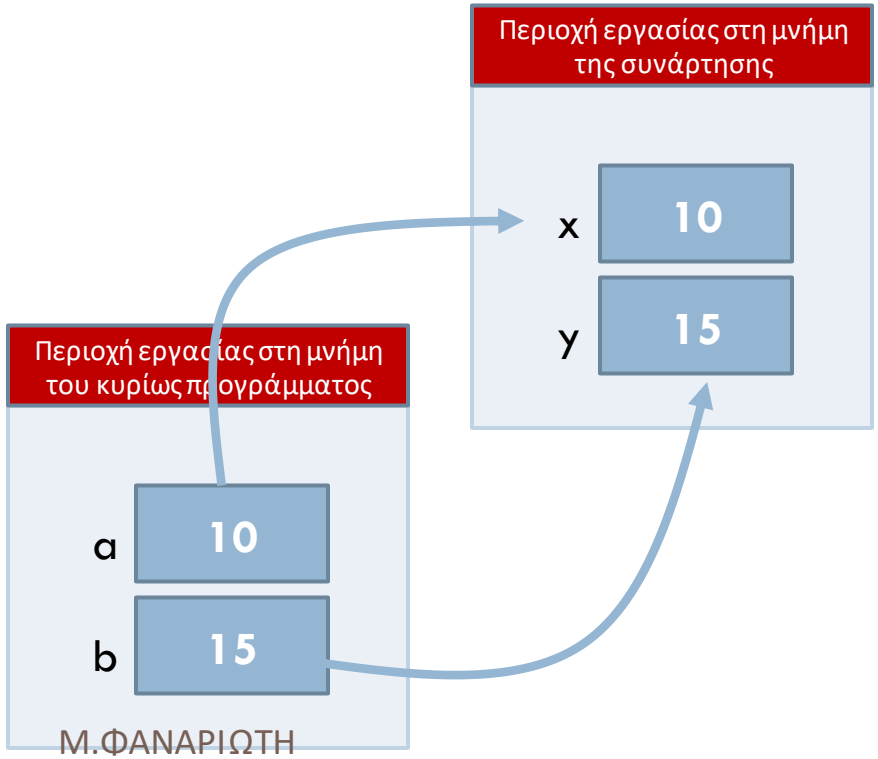
```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```



Άσκηση – Διεργασίες στη Μνήμη

- Ο υπολογιστής εντοπίζει τον ορισμό της συνάρτησης μέσα στον κώδικα και σε **μία νέα περιοχή μέσα στη μνήμη** δημιουργεί αρχικά τις μεταβλητές **x,y** τις οποίες γεμίζει με τους αριθμούς 10 και 15 αντίστοιχα.

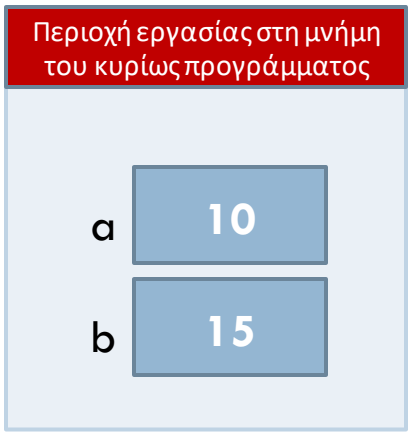
```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```

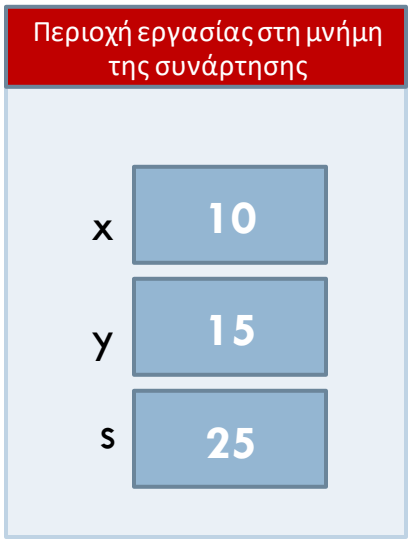


Άσκηση – Διεργασίες στη Μνήμη

- Στη συνέχεια ο υπολογιστής μπαίνει μέσα στην εσοχή και εκτελεί τον κώδικα της συνάρτησης δημιουργώντας μια νέα μεταβλητή **s** στην οποία φυλά το άθροισμα των δύο αριθμών.

```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```

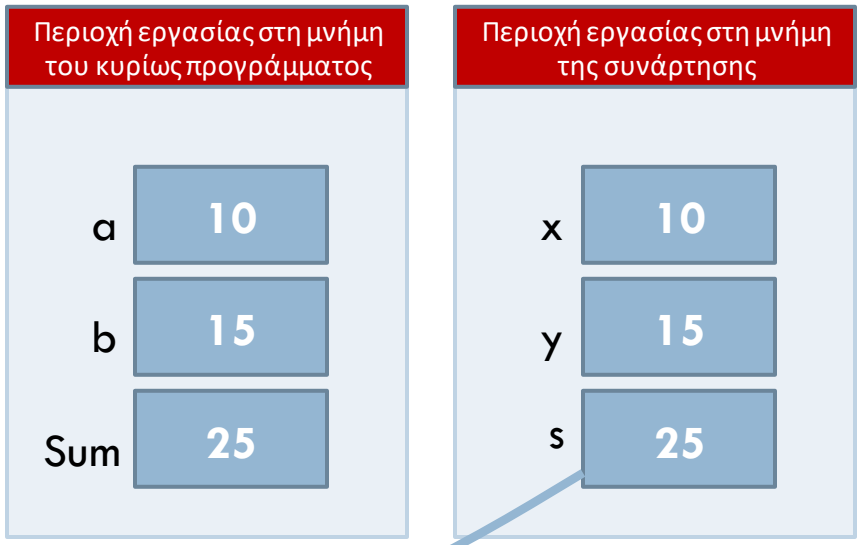




Άσκηση – Διεργασίες στη Μνήμη

- Τέλος εκτελείται η εντολή **return** η οποία επιστρέφει πίσω στο κυρίως πρόγραμμα το περιεχόμενο της μεταβλητής s. Το νούμερο 25 επιστρέφεται στο σημείο που βρισκόταν η κλήση και έτσι η εντολή **Sum=add(a,b)** παίρνει την μορφή **Sum=25**. Το νούμερο καταχωρείται σε μια νέα μεταβλητή με το όνομα **Sum** στην περιοχή εργασίας του κυρίως προγράμματος.

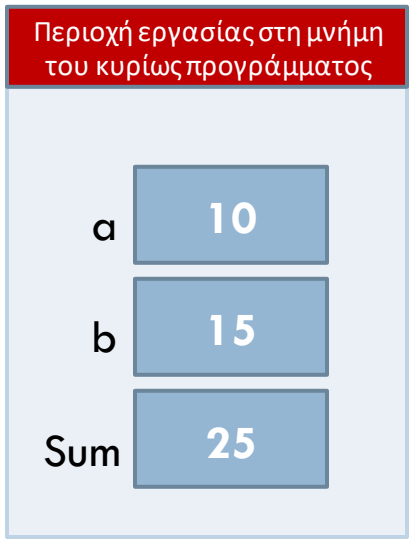
```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```

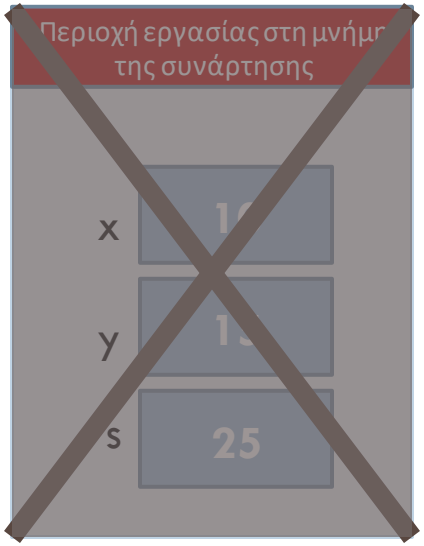


Άσκηση – Διεργασίες στη Μνήμη

- Ταυτόχρονα με το τέλος της συνάρτησης η περιοχή εργασίας στη μνήμη που αυτή χρησιμοποιούσε απελευθερώνεται. Ακολουθεί η εμφάνιση του αποτελέσματος.

```
# ορισμός συνάρτησης
def add(x,y):
    s=x+y
    return s

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```

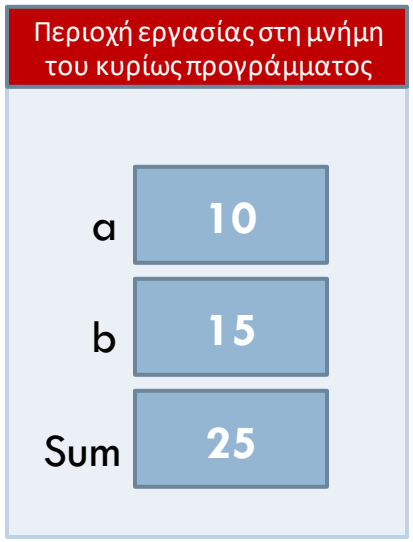


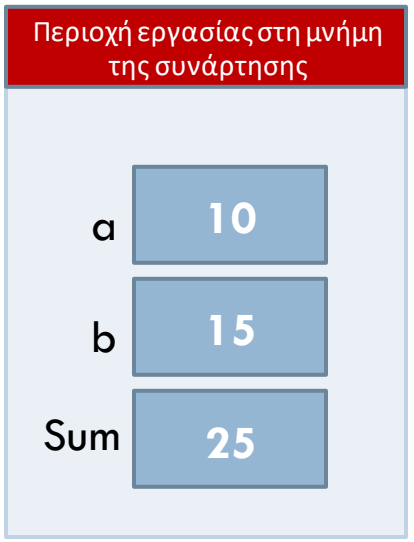


Διεργασίες στη Μνήμη – Τοπικές Μεταβλητές

- **ΠΡΟΣΟΧΗ! Επειδή ο χώρος εργασίας της συνάρτησης στη μνήμη είναι διαφορετικός από αυτόν του κυρίως προγράμματος, γι' αυτό παράμετροι της συνάρτησης και μεταβλητές του κυρίως προγράμματος που φυλάνε τα δεδομένα που θα της δοθούν, μπορούν να έχουν τα ίδια ονόματα χωρίς να δημιουργείται κάποιο μπέρδεμα.** Επίσης **μεταβλητές που χρησιμοποιεί η συνάρτηση** μπορούν να έχουν **το ίδιο όνομα** με **μεταβλητές που χρησιμοποιεί το κυρίως πρόγραμμα** αφού πρόκειται για διαφορετικές μεταβλητές εφόσον βρίσκονται σε διαφορετικές περιοχές της μνήμης. Αυτές οι μεταβλητές λέγονται **τοπικές μεταβλητές** και **μπορεί να τις δει και να τις επεξεργαστεί μόνο η συνάρτηση**.

```
# ορισμός συνάρτησης
def add(a,b) :
    Sum=a+b
    return Sum

# κυρίως πρόγραμμα
a=input("Give the first number: ")
b=input("Give the second number: ")
Sum=add(a,b)
print Sum
```





Καθολικές και Τοπικές Μεταβλητές

- Οι μεταβλητές της συνάρτησης λέγονται **τοπικές** και μπορούν να διαβαστούν και να τροποποιηθούν μόνο από τη συνάρτηση στην οποία ανήκουν. Οι μεταβλητές του κυρίως προγράμματος λέγονται **καθολικές** και **μπορούν να διαβαστούν από όλες τις συναρτήσεις όχι όμως να τροποποιηθούν**.

Καθολικές και Τοπικές Μεταβλητές

- **Πρόγραμμα:** Να γραφτεί πρόγραμμα που α) Διαβάζει το ακαθάριστο εισόδημα ενός υπαλλήλου β) Καλεί συνάρτηση με το όνομα foros() και παράμετρο το ακαθάριστο εισόδημα. Η συνάρτηση υπολογίζει και επιστρέφει στο κυρίως πρόγραμμα το φόρο που αναλογεί στον υπάλληλο κάνοντας χρήση ενός συντελεστή κρατήσεων που έχει οριστεί κατά την έναρξη του προγράμματος στο 0.2 και έχει εκχωρηθεί σε μια μεταβλητή με το όνομα k. γ) εμφανίζει στην οθόνη το καθαρό εισόδημα του υπαλλήλου.

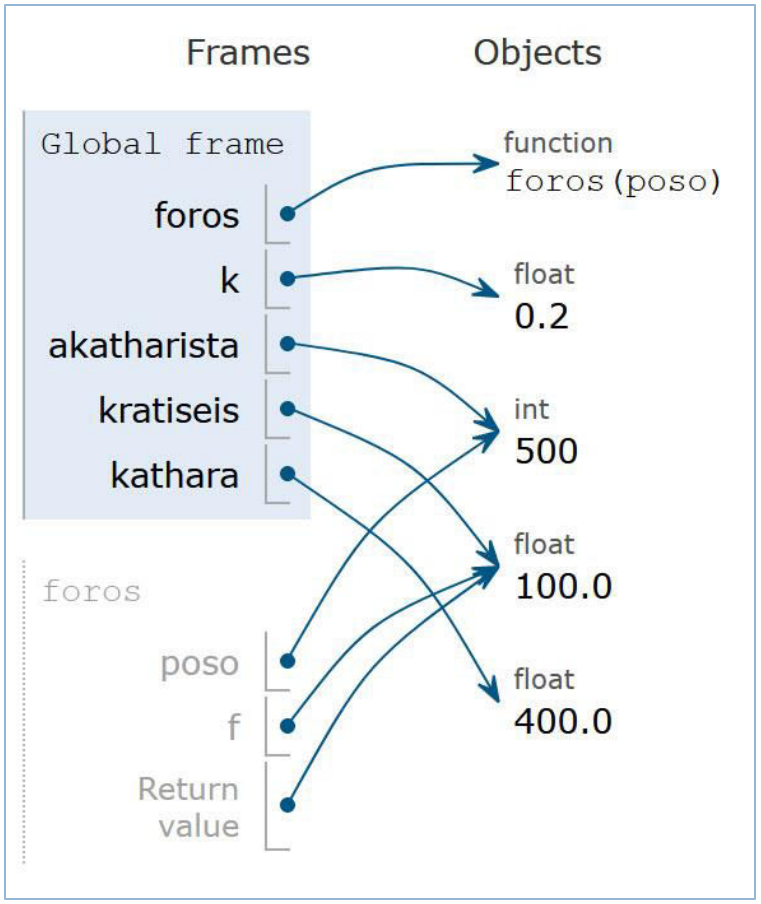
Καθολικές και Τοπικές Μεταβλητές

Πρόγραμμα

```
def foros(poso):
    f=k*poso      #αναφορά στην καθολική μεταβλητή k
    return f


k=0.2      #συντελεστής κρατήσεων
akatharista=input("Ακαθάριστο Εισόδημα:")
kratiseis=foros(akatharista)
kathara=akatharista-kratiseis
print "Καθαρό Εισόδημα",kathara
```

Καθολικές και Τοπικές Μεταβλητές

Δίπλα φαίνονται οι δομές στη μνήμη του υπολογιστή κατά την εκτέλεση του προηγούμενου προγράμματος στο Python Tutor



Εντολή global

- Όπως είδαμε οι μεταβλητές του κυρίως προγράμματος μπορούν να διαβαστούν από όλες τις συναρτήσεις όχι όμως να τροποποιηθούν. **Μπορούμε να δώσουμε τη δυνατότητα σε μια συνάρτηση να τροποποιήσει μια καθολική μεταβλητή** χρησιμοποιώντας την **εντολή global** όπως φαίνεται στη συνέχεια:

Εντολή global

Παράδειγμα

```
x=50

def func():
    global x
    print 'Το x είναι',x
    x=2
    print 'Το καθολικό x άλλαξε σε',x

func()
print 'Η τιμή του x είναι',x
```

- Έξοδος:

```
Το x είναι 50
Το καθολικό x άλλαξε σε 2
Η τιμή του x είναι 2
```

Χρήση της εντολής global

- **Παράδειγμα επίδειξης χειρισμού της Python των μεταβλητών τοπικής και καθολικής εμβέλειας.** Έστω ότι έχουμε το εξής σενάριο:

global2.py

```
def func1():
   myGlobal=42

def func2():
   print myGlobal

myGlobal=5
func1()
func2()
```

Τυπώνει, όχι την τιμή 42, αλλά το 5.

Χρήση της εντολής global

- Αν βέβαια προσθέσουμε μία δήλωση 'global' στη func1(), όπως στο παρακάτω παράδειγμα, τότε η func2() θα τυπώσει 42.

global2.py

```
def func1():
    global myGlobal
    myGlobal=42

def func2():
    print myGlobal

myGlobal=5
func1()
func2()
```

Τυπώνει 42