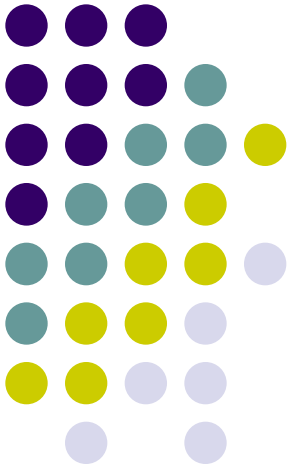
Προηγμένα στοιχεία γλώσσας προγραμματισμού

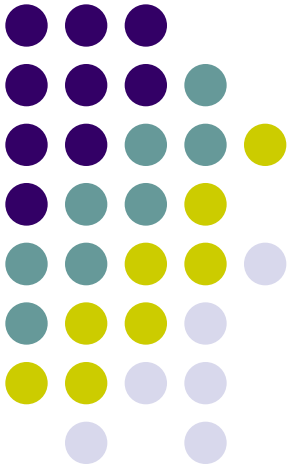
Γ’ Πληροφορική

Συναρτήσεις

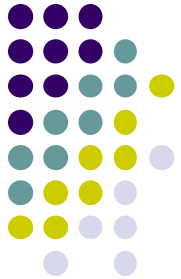


Προηγμένα στοιχεία γλώσσας προγραμματισμού

Ιδιότητες υποπρογραμμάτων
Εμβέλεια μεταβλητών
Αρθρώματα
Πακέτα



Συναρτήσεις

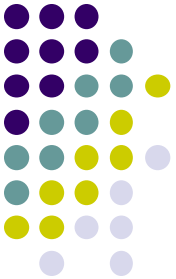


- Η Python παρέχει ένα τύπο μόνο υποπρογραμμάτων, τις **συναρτήσεις** τις οποίες θεωρεί ως αντικείμενα.
- Μπορούμε να δημιουργήσουμε τις δικές μας συναρτήσεις ή να χρησιμοποούμε και να ενσωματώνουμε στο πρόγραμμα έτοιμες βιβλιοθήκες.

Ορισμός και κλήση Συναρτήσης

- Ο ορισμός μιας συνάρτησης περιλαμβάνει το όνομα της και τις παραμέτρους εισόδου ακι καλείται από σημεία του προγράμματος μέσω της λειτουργίας που ονομάζεται **κλήση(**calling) της συνάρτησης.
- Ορίζεται με τη λέξη κλειδί **def**, που την ακολουθεί ένα όνομα το οποίο την ταυτοποιεί και ένα ζευγάρι παρενθέσεων που μπορούν να περιέχουν μερικά ονόματα μεταβλητών, ενώ η δήλωση τελειώνει με διπλή τελεία (:)
- Κάτω από τη γραμμή αυτή τοποθετούνται, σε εσοχή οι εντολές που καθορίζουν τη λειτουργία της συνάρτησης.

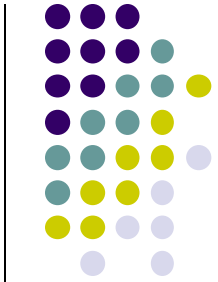
Παράδειγμα ορισμού συνάρτησης χωρίς χρήση παραμέτρων



```
def fun_name():
    print 'hello'
# Τέλος της συνάρτησης
fun_name() # κλήση της συνάρτησης
fun_name() # κι άλλη κλήση της συνάρτησης
```

Σημείωση: Μια συνάρτηση πρέπει να έχει οριστεί πριν χρησιμοποιηθεί.

Παράδειγμα ορισμού συνάρτησης με χρήση δυο παραμέτρων



```
def find_sum(par1, par2):
    result=par1+par2
    return result
print find_sum(3, 4)
```

Η συνάρτηση αυτή επιστρέφει το άθροισμα των τιμών των δύο παραμέτρων.

Η συνάρτηση δέχεται ορίσματα και επιστρέφει ένα αποτέλεσμα. Το αποτέλεσμα ονομάζεται **επιστρεφόμενη τιμή** (return value)

Παράδειγμα συνάρτησης που καλεί άλλη συνάρτηση

```
def fun_name():
    print 'hello'
def epanalave_name():
    fun_name()
    fun_name()
    # Τέλος της συνάρτησης
epanalave_name()
```

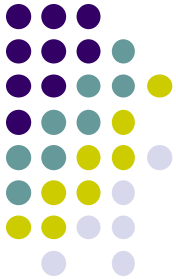
Γενική Δομή Συνάρτησης

Σύνταξη

```
def  όνομα_συνάρτησης ([{παράμετροι}]):
        εντολές
        [return <αποτέλεσμα>]
```

- Οι εντολές της συνάρτησης έχουν υποχρεωτικά εσοχή (πχ. 4 κενά). Αν δεν υπάρχει η εσοχή, το πρόγραμμα δεν εκτελείται και εμφανίζονται μηνύματα λάθους.
- Οι αγκύλες [ ] σημαίνουν πως ότι περικλείεται μέσα σε αυτές είναι προαιρετικό.

Κατηγορίες Συναρτήσεων (1)



- Οι συνάρτησεις μπορούν να κατηγοριοποιηθούν με πολλούς τρόπους:

Α) αυτές που δεν τροποποιούν το αντικείμενο στο ποίο εφαρμόζονται:

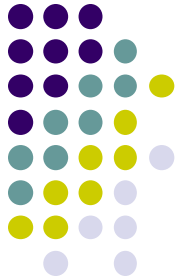
```
>>> a='Python'
>>> print a.upper()
PYTHON
>>> print a
Python
```

Κατηγορίες Συναρτήσεων (2)

Β) εκείνες που μπορούν να αλλάξουν το αντικείμενο στο οποίο καλούνται:

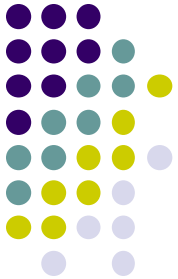
```
>>> b=[‘a’, ‘b’, ‘c’, ‘d’]
>>> b.reverse()
>>> print b
[‘d’, ‘c’, ‘b’, ‘a’]
```

Κατηγορίες Συναρτήσεων (3)



Μια άλλη κατηγορία συναρτήσεων είναι:
Α) αυτές οι οποίες, όταν κληθούν, επιστρέφουν αποτέλεσμα (κάποια τιμή)
Β) εκείνες που δεν επιστρέφουν κάποια τιμή (κενές/void συναρτήσεις), αλλά εκτελούν ενέργειες μέσω των εντολών τους. Οι κενές αυτές συναρτήσεις μπορεί να εμφανίζουν αποτέλεσμα στην οθόνη ή να έχουν κάποιο άλλο αποτέλεσμα, αλλά δεν επιστρέφουν κάποια τιμή.

Μεταβλητές και παράμετροι



- Η Python παρέχει ένα τύπο μόνο υποπρογραμμάτων, τις **συναρτήσεις** τις οποίες θεωρεί ως αντικείμενα.
- Μπορούμε να δημιουργήσουμε τις δικές μας συναρτήσεις ή να χρησιμοποούμε και να ενσωματώνουμε στο πρόγραμμα έτοιμες βιβλιοθήκες.

Άσκηση 3α- Παράδειγμα

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει δυο ακεραίους
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται και να εμφανίζεται άθροισμα.

Άσκηση 3α - Λύση

```
def add_sum(n1,n2):
    sum=n1+n2
    print 'Το άθροισμα είναι: ',sum

num1=input('Δώσε πρώτο ακέραιο: ')
num2=input('Δώσε δεύτερο ακέραιο: ')
add_sum(num1,num2)
```

Παράμετροι

Ορίσματα

Άσκηση 3β- Παράδειγμα

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει δυο ακεραίους
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται το άθροισμα.
- Γ) Να εμφανίζεται το άθροισμα στο κύριο πρόγραμμα

Άσκηση 3β - Λύση

```
def add_sum(n1,n2):
    sum=n1+n2
    return sum


num1=input('Δώσε πρώτο ακέραιο: ')
num2=input('Δώσε δεύτερο ακέραιο: ')
print 'Το άθροισμα είναι:
    ',add_sum(num1,num2)
```

Άσκηση 4- Παράδειγμα

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει δυο ακεραίους
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται το γινόμενο.
- Γ) Να εμφανίζεται το γινόμενο στο κύριο πρόγραμμα

Άσκηση 4 - Λύση

```
def gin(n1,n2):
    p=n1*n2
    return p

num1=input('Δώσε πρώτο ακέραιο: ')
num2=input('Δώσε δεύτερο ακέραιο: ')
print 'Το γινόμενο είναι: ', gin(num1,num2)
```

Άσκηση 5

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει δυο ακεραίους
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται ο μεγαλύτερος.
- Γ) Να εμφανίζεται ο μεγαλύτερος στο κύριο πρόγραμμα

Άσκηση 5 - Λύση

```
def meg(n1,n2):
    max=n1
    if n2 >max:
       max=n2
    return max

num1=input('Δώσε πρώτο ακέραιο: ')
num2=input('Δώσε δεύτερο ακέραιο: ')
print 'Ο μεγαλύτερος είναι: ', meg(num1,num2)
```

Άσκηση 6

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει δυο ακεραίους
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται ο μικρότερος.
- Γ) Να εμφανίζεται ο μικρότερος στο κύριο πρόγραμμα

Άσκηση 6 - Λύση

```
def mik(n1,n2):
    min=n1
    if n2 <min:
       min=n2
    return min

num1=input('Δώσε πρώτο ακέραιο: ')
num2=input('Δώσε δεύτερο ακέραιο: ')
print 'Ο μικρότερος είναι: ', mik(num1,num2)
```

Άσκηση 7

- Να γραφεί πρόγραμμα το οποίο:
- Με τη βοήθεια μιας συνάρτησης να εμφανίζεται 3 φορές η λέξη «Python».

Άσκηση 7 - Λύση

```
def printPython3():
    for i in range(3):
        print 'Python'

printPython3()
```

Άσκηση 8

- Να γραφεί πρόγραμμα το οποίο:
- Να διαβάζει το πλήθος Ν που θέλουμε να εμφανίζεται η λέξη «Python».
- Με τη βοήθεια μιας συνάρτησης να εμφανίζεται Ν φορές η λέξη «Python».

Άσκηση 8 - Λύση

```
def printPython(n):
    for i in range(n):
        print 'Python'

num=input('Δώσε πλήθος ν: ')
printPython(num)
```

Άσκηση 9

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει την πλευρά ενός τετραγώνου
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται το εμβαδό του τετραγώνου.
- Γ) Να εμφανίζεται το εμβαδό στο κύριο πρόγραμμα.

Άσκηση 9 - Λύση

```
def embtetr(a):
    emvado=a*a
    return emvado

num=input('Δώσε πλευρά τετραγώνου: ')
print 'Το εμβαδό του τετραγώνου είναι: ',
    embtetr(num)
```

Άσκηση 10

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει την ακτίνα ενός κύκλου
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται το εμβαδό του κύκλου.
- Γ) Να εμφανίζεται το εμβαδό στο κύριο πρόγραμμα.

Άσκηση 10 - Λύση

```
def embcircle(r):
    emvado=3.14*r*r
    return emvado

num=input('Δώσε ακτίνα κύκλου: ')
print 'Το εμβαδό του κύκλου είναι: ',
    embcircle(num)
```

Άσκηση 11

- Να γραφεί πρόγραμμα το οποίο:
- Α) Να διαβάζει τις πλευρές ενός ορθογωνίου
- Β) Με τη βοήθεια μιας συνάρτησης με παραμέτρους να υπολογίζεται το εμβαδό του ορθογωνίου.
- Γ) Να εμφανίζεται το εμβαδό στο κύριο πρόγραμμα.

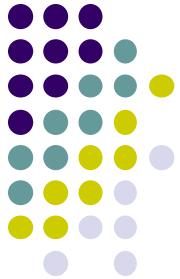
Άσκηση 11 - Λύση

```
def emborth(a,b):
    emvado=a*b
    return emvado


num1=input('Δώσε μήκος ορθογωνίου: ')
num2=input('Δώσε πλάτος ορθογωνίου: ')
print 'Το εμβαδό του ορθογωνίου είναι: ',
    emborth(num1,num2)
```

Άσκηση 11 - Λύση

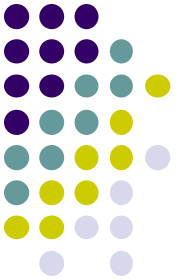
```
def emborth(a,b):
    emvado=a*b
    return emvado


num1=input('Δώσε μήκος ορθογωνίου: ')
num2=input('Δώσε πλάτος ορθογωνίου: ')
print 'Το εμβαδό του ορθογωνίου είναι: ',
    emborth(num1,num2)
```

Εμβέλεια των μεταβλητών



Όλες οι μεταβλητές σε ένα πρόγραμμα δεν μπορούν να είναι προσπελάσιμες από όλα τα μέρη του προγράμματος.
Η ***εμβέλεια (scope) μιας μεταβλητής*** αναφέρεται στο τμήμα του προγράμματος που μπορεί αυτή να έχει πρόσβαση.

Εμβέλεια των μεταβλητών (2)

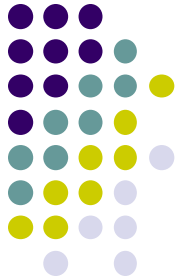


Απεριόριστη εμβέλεια: Όλες οι μεταβλητές είναι ορατές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος. Αυτού του τύπου οι μεταβλητές χαρακτηρίζονται ως ***καθολικές (global)***. Το μειονέκτημα είναι ότι περιορίζεται έτσι η ανεξαρτησία των υποπρογραμμάτων.

✔ **Περιορισμένη εμβέλεια**: Αυτές οι μεταβλητές είναι ***τοπικές (local)***, ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν.

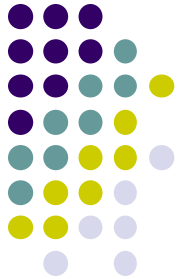
.

Καθολικές και Τοπικές μεταβλητές



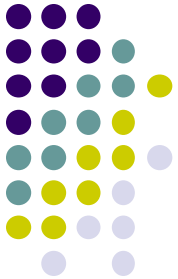
Οι μεταβλητές που έχουν οριστεί μέσα στο σώμα της συνάρτησης, έχουν ***τοπική εμβέλεια***, ενώ αυτές που ορίστηκαν έξω από αυτό έχουν ***καθολική εμβέλεια***.
Αυτό σημαίνει ότι οι ***τοπικές μεταβλητές*** μπορούν να προσπελαστούν μόνο μέσα στη συνάρτηση στην οποία δηλώθηκαν, ενώ οι ***καθολικές μεταβλητές*** μπορεί να είναι προσβάσιμες από όλες τις συναρτήσεις.

Παράδειγμα με Καθολικές και Τοπικές μεταβλητές



```
total = 0       # καθολική μεταβλητή

def sum(a1,a2):
        total = a1 + a2 # τοπική μεταβλητή
        print "Μέσα στη συνάρτηση η τοπική total", total
        return total

sum(10, 20)
print "Έξω  από τη συνάρτηση η καθολική total", total
```

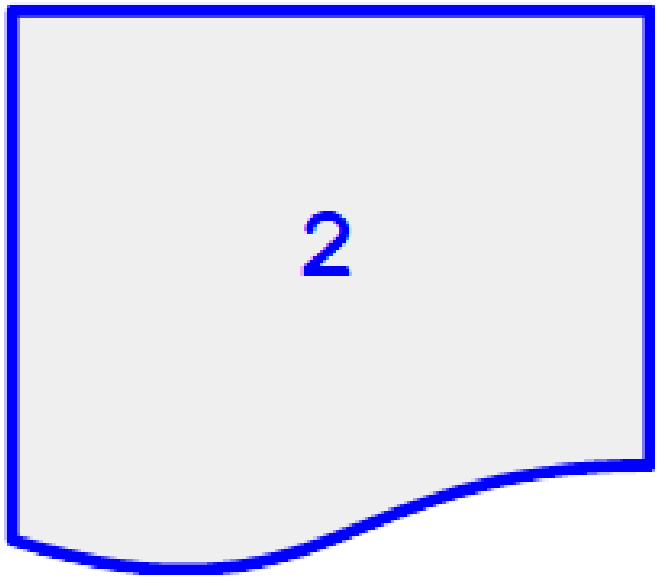
Χρήση της εντολής global



- Οι μεταβλητές που δηλώνονται στο κύριο μέρος του προγράμματος - έξω από τις συναρτήσεις είναι ***καθολικές μεταβλητές (global)***.
- Εάν θέλουμε μέσα σε μια συνάρτηση να αλλάξουμε την τιμή μιας καθολικής μεταβλητής τότε πρέπει να δηλώσουμε μέσα στην συνάρτηση ότι η μεταβλητή αυτή είναι καθολική. Αυτό γίνεται με τη χρήση της εντολής **global**:

Χρήση της εντολής global (Παράδειγμα)

```
x = 50
def func():
        global x
        x = 2
func()
print x
```



Παράδειγμα 1 με Καθολικές και Τοπικές μεταβλητές

```
x = 5

def fun1():
        x = 10
        print x

def fun2():
        print x

fun1()
fun2()
```

```
10
5
```

Παράδειγμα 1 (Εξήγηση)

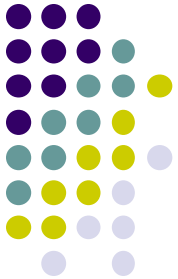
```
x = 5

def fun1():
        x = 10
        print x

def fun2():
        print x

fun1()
fun2()
```

```
10
5
```

Στην **fun1()** η μεταβλητή **x** είναι τοπική.

Στην **fun2()** ζητάμε να τυπωθεί η τιμή μιας μεταβλητής η οποία *δεν έχει πάρει τιμή μέσα στη συνάρτηση*. Στην περίπτωση αυτή, η Python κοιτάζει να δει αν υπάρχουν καθολικές μεταβλητές με το όνομα **x**. Αν τη βρει, την τυπώνει. Αν δεν την βρεί, εμφανίζει μήνυμα λάθους.

Παράδειγμα 2 με Καθολικές και Τοπικές μεταβλητές



```
x = 5

def fun1():
        global x
        x = 10
        print x

def fun2():
        print x

fun1()
fun2()
```

```
10
10
```

Παράδειγμα 2 (Εξήγηση)

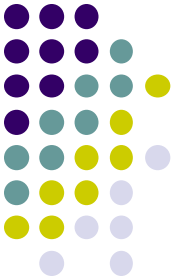
```
x = 5

def fun1():
        global x
        x = 10
        print x

def fun2():
        print x

fun1()
fun2()
```

```
10
10
```

Στην **fun1()** η μεταβλητή **x** γίνεται καθολική με την εντολή **global x**.

Στην **fun2()** ζητάμε να τυπωθεί η τιμή μιας μεταβλητής η οποία *δεν έχει πάρει τιμή μέσα στη συνάρτηση*. Στην περίπτωση αυτή, η Python κοιτάζει να δει αν υπάρχουν καθολικές μεταβλητές με το όνομα **x**. Αν τη βρει, την τυπώνει. Αν δεν την βρει, εμφανίζει μήνυμα λάθους.

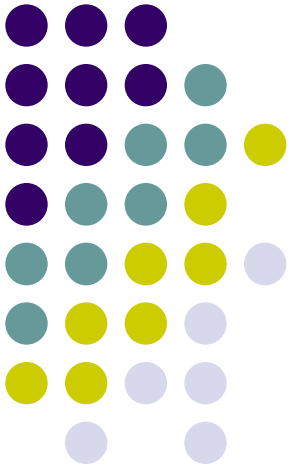
Καθολικές και Τοπικές μεταβλητές Συμπεράσματα



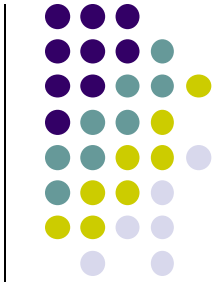
- Αν μέσα σε μια συνάρτηση εκχωρηθεί σε οποιαδήποτε μεταβλητή μια τιμή, η μεταβλητή αυτή είναι τοπική και η τιμή της ισχύει μόνο για αυτή τη συνάρτηση, εκτός αν δηλωθεί ρητά διαφορετικά.
- Αν μέσα σε μια συνάρτηση απλώς εκτυπώνεται η τιμή μιας μεταβλητής που δεν υπάρχει τοπικά, τότε γίνεται αναζήτηση σε οποιαδήποτε δυνατή εμβέλεια, για παράδειγμα καθολική.
- Αν μέσα σε μια συνάρτηση θέλουμε να αλλάξουμε την τιμή μιας καθολικής μεταβλητής τότε, η ορθότερη πρακτική είναι να επιστρέψουμε (**return**) την τιμή και να την λάβει με την εντολή εκχώρησης. Σε εξαιρετικές περιπτώσεις, μπορούμε να χρησιμοποιήσουμε την εντολή **global**.

Προηγμένα στοιχεία γλώσσας προγραμματισμού

Αρθρώματα

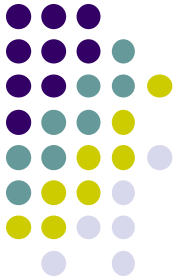


Αρθρώματα (modules)



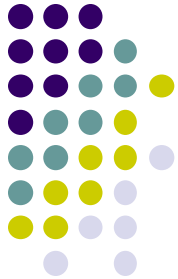
- **Άρθρωμα** είναι ένα αρχείο (με επέκταση **.py**) που περιέχει ***συναρτήσεις***, ***κλάσεις*** και ***μεταβλητές***. Είναι ένα ***αντικείμενο***, όπως όλα, στη γλώσσα Python.
- Για να χρησιμοποιήσουμε ένα άρθρωμα στο πρόγραμμά μας, θα πρέπει να το ***"εισάγουμε"*** σε αυτό. Αυτό γίνεται με τη δήλωση **import**.

Η δήλωση import (1)



- **import module1[, module2[,... moduleN]]**
- Π.χ.
  import random, math

Η δήλωση import (2)

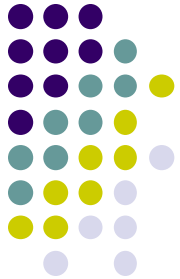


- **from modname import name1[, name2[,...nameN]]**
- Δίνει τη δυνατότητα εισαγωγής συγκεκριμένων χαρακτηριστικών - δυνατοτήτων του module, χωρίς να εισάγει όλο το module
- π.χ. **from math import sqrt**

- **from modname import ***

Εισάγει όλο το περιεχόμενο του module. Δεν ενδείκνυται η συχνή χρήση του!

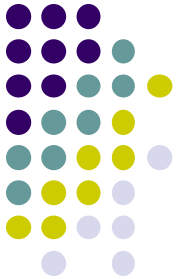
Συμβολισμός με τελεία



- Για να έχουμε πρόσβαση σε μια από τις συναρτήσεις, θα πρέπει να προσδιορίσουμε το όνομα του module και το όνομα της συνάρτησης, χωρισμένα με μία τελεία. Αυτή η μορφή ονομάζεται ***συμβολισμός με τελεία (dot notation)***.
- π.χ.
- **random.randint(1,10)**

που επιστρέφει έναν τυχαίο ακέραιο από το 1 έως και το 10.

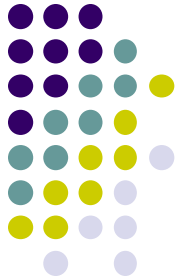
Συναρτήσεις σε πρόγραμμα



Σε ένα πρόγραμμα μπορούν να συνυπάρχουν τρεις κατηγορίες συναρτήσεων:

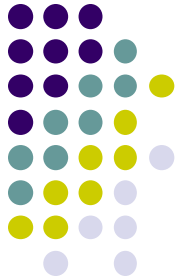
- ***ενσωματωμένες*** στο περιβάλλον (built-in) που είναι πάντα διαθέσιμες για χρήση (όπως η **abs**)
- που περιέχονται σε ***εξωτερικά module***, τα οποία πρέπει πρώτα να εισαχθούν (με την **import**)
- που ***ορίζονται από τον προγραμματιστή*** (με το **def**)

Συναρτήσεις σε πρόγραμμα (Παράδειγμα)



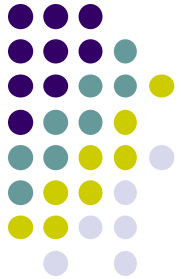
```
from math import sqrt
def cube(x):
        return x * x * x


print abs(-1) # ενσωματωμένη
print cube(5) # ορισμένη από τον προγραμματιστή
print sqrt(81) # από το module math
```

Πρότυπη Βιβλιοθήκη (standard library)



Μια ***βιβλιοθήκη (library)*** -σε οποιαδήποτε γλώσσα προγραμματισμού- είναι μια συλλογή εργαλείων που μπορεί να έχουν γραφτεί και από άλλους προγραμματιστές, προκειμένου να εκτελούνται συγκεκριμένες λειτουργίες.
Οι βιβλιοθήκες είναι πολύ σημαντικές στον προγραμματισμό, γιατί μας δίνουν τη δυνατότητα να χρησιμοποιούμε τα εργαλεία που περιλαμβάνονται σε αυτές.

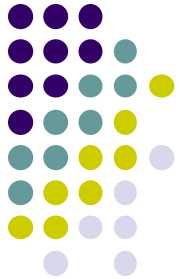
Πρότυπη Βιβλιοθήκη (standard library) (2)



Η ***πρότυπη βιβλιοθήκη*** της Python περιέχει έναν τεράστιο αριθμό χρήσιμων module και είναι μέρος κάθε πρότυπης εγκατάστασης Python.

- Πολλά προβλήματα μπορούν να λυθούν γρήγορα, αν αξιοποιηθεί το εύρος των δυνατοτήτων που έχουν οι βιβλιοθήκες αυτές.
- Περιλαμβάνει τμήματα για προγραμματισμό γραφικών (**Tkinter**), αριθμητική επεξεργασία, web συνδεσιμότητα, βάσεις δεδομένων (**Anydbm**), Βιοπληροφορική (**Biopython**) κ.ά.
- Στην Python, μπορούν να χρησιμοποιηθούν βιβλιοθήκες από πολλές άλλες γλώσσες προγραμματισμού...

Πρότυπη Βιβλιοθήκη Παραδείγματα (math module)



Μαθηματική μονάδα λογισμικού (**math module)**

A) >>> import math

>>> print math.pi

3.14159265359

B) Εναλλακτικά, μπορούμε να εισάγουμε ένα αντικείμενο από μια μονάδα ως εξής:

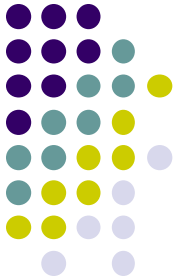
>>> from math import pi

Τώρα μπορούμε να έχουμε απευθείας πρόσβαση στη **pi** χωρίς το συμβολισμό της τελείας.

>>> print pi

>>> 3.14159265359

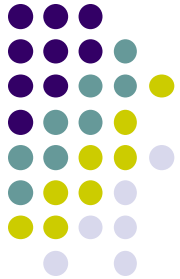
Πρότυπη Βιβλιοθήκη Παράδειγμα (random module)



Μονάδα λογισμικού random (**random module)**
Διαθέτει εργαλεία δημιουργίας ψευδοτυχαίων αριθμών:

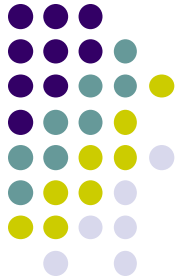
```
>>> import random
# τυχαίος ακέραιος στο διάστημα range(6)
>>> random.randrange(6)
5
# τυχαίος κινητής υποδιαστολής (float)
>>> random.random()
0.17970987693370618 6
```

Πακέτα (packages)



Τα ***πακέτα*** είναι ένα εργαλείο για την ιεραρχική οργάνωση των modules.
Η Python παρέχει ένα απλό σύστημα δημιουργίας πακέτων, ως επέκταση του μηχανισμού των αρθρωμάτων.
Κάθε κατάλογος με ένα **__init__.py** αρχείο αναφέρεται ως ένα Python πακέτο...

Δραστηριότητα 1



```
name = "black "

def f():
        name = "white"
        print " Μέσα στη συνάρτηση ", name

print " Έξω από τη συνάρτηση ", name
f()
```

Δραστηριότητα 2

```
name = "black "

def f():
        print " Μέσα στη συνάρτηση ", name

print " Έξω από τη συνάρτηση ", name
f()
```

Δραστηριότητα 3

```
name = "black "

def f():
        global name
        name = "white"
        print " Μέσα στη συνάρτηση ", name

print " Έξω από τη συνάρτηση ", name
f()
print " Έξω από τη συνάρτηση ", name
```

Άσκηση 12

Μια εταιρία κινητής τηλεφωνίας χρεώνει τους πελάτες της, ως προς τον όγκο κατανάλωσης δεδομένων, κλιμακωτά, σύμφωνα με τον παρακάτω πίνακα:

Όγκος Κατανάλωσης σε MB	Χρέωση ανά MB
Τα πρώτα 100	0,03 ευρώ
Από 101 μέχρι και 500	0,02 ευρώ
Περισσότερα από 500	0,01 ευρώ

Αν η χρέωση είναι μεγαλύτερη από πενήντα (50) ευρώ, τότε ο πελάτης δικαιούται έκπτωση 10%.

Να γράψετε πρόγραμμα σε γλώσσα προγραμματισμού Python, το οποίο:

1. Να διαβάζει από το πληκτρολόγιο τον όγκο των δεδομένων σε MB που κατανάλωσε ο πελάτης.

2. Να καλεί συνάρτηση, η οποία να δέχεται τον όγκο των δεδομένων που κατανάλωσε ο πελάτης και να επιστρέφει τη χρέωση που προκύπτει, χωρίς την έκπτωση, από τον παραπάνω πίνακα. Αναπτύξτε τη συνάρτηση που χρειάζεται για το σκοπό αυτόν.

3. Να εμφανίζει τη χρέωση του πελάτη που προκύπτει από τη συνάρτηση.

4. Σε περίπτωση που ο πελάτης δικαιούται έκπτωση, να υπολογίζει και να εμφανίζει το ποσό της έκπτωσης, καθώς και την τελική χρέωση του πελάτη. Διαφορετικά, να εμφανίζει το μήνυμα «Δεν δικαιούστε έκπτωση».

Άσκηση 12 - Λύση

```
def xreosi(n):
    if n<=100:
        x=n*0.03
    elif n<=500:
        x=100*0.03+(n–100)*0.02
    else:
        x=100*0.03+400*0.02 +(n–500)*0.01
    return x
d=input('Δώστε  όγκο δεδομένων: ')
poso=xreosi(d)
print 'Η χρέωση είναι: ' , poso
if  xreosi>50:
    ekpt=poso*0.1
    print  'Η έκπτωση είναι:  ', ekpt
    print  'Η τελική χρέωση είναι: ', poso-ekpt
else: print  'Δεν δικαιούστε έκπτωση'
```

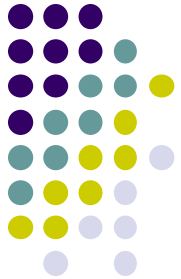
Συναρτήσεις και Λίστες (1)

```
def changeme(mylist):
    mylist.append([1, 2, 3, 4])
    print "Τιμές μέσα στη συνάρτηση:", mylist
     return
mylist=[10, 20, 30]
changeme(mylist)
print " Τιμές έξω από τη συνάρτηση: ", mylist
```

Αποτέλεσμα:

Τιμές μέσα στη συνάρτηση: [10,20,30,[1,2,3,4]]

Τιμές έξω από τη συνάρτηση:[10,20,30,[1,2,3,4]]



Συναρτήσεις και Λίστες (2)

```
def changeme(mylist):
    mylist=[1, 2, 3, 4]
    print "Τιμές μέσα στη συνάρτηση:", mylist
     return

mylist=[10, 20, 30]
changeme(mylist)
print " Τιμές έξω από τη συνάρτηση: ", mylist
```

Αποτέλεσμα:

```
Τιμές μέσα στη συνάρτηση: [1, 2, 3, 4]
Τιμές έξω από τη συνάρτηση: [10, 20, 30]
```