Python - 7. Προηγμένα Στοιχεία Γλώσσας Προγραμματισμού

Υποπρογράμματα

Ένα υποπρόγραμμα είναι ένα κομμάτι προγράμματος που έχει γραφεί ξεχωριστά από το υπόλοιπο πρόγραμμα και επιτελεί ένα αυτόνομο έργο.

Κάθε υποπρόγραμμα πρέπει να έχει τα παρακάτω **βασικά χαρακτηριστικά**:

α) Έχει μόνο ένα σημείο εισόδου από το οποίο δέχεται τα δεδομένα του.

β) Το υποπρόγραμμα το οποίο καλεί ένα άλλο υποπρόγραμμα σταματάει την εκτέλεσή του όσο εκτελείται το καλούμενο υποπρόγραμμα. Μόνο ένα υποπρόγραμμα μπορεί να εκτελείται σε μια χρονική στιγμή.

γ) Ο έλεγχος επιστρέφει στο υποπρόγραμμα το οποίο καλεί, όταν το καλούμενο υποπρόγραμμα σταματήσει να εκτελείται.

Καλές πρακτικές

- Πριν ξεκινήσουμε να γράφουμε ένα πρόγραμμα, μελετάμε πώς αυτό μπορεί να αναλυθεί σε **επιμέρους τμήματα** και αποφασίζουμε για τα αντίστοιχα υποπρογράμματα.
- Εξετάζουμε αν κάποια υποπρογράμματα, τα οποία έχουμε ήδη γράψει ή υπάρχουν σε **έτοιμες βιβλιοθήκες** προγραμμάτων, μπορούν να χρησιμοποιηθούν, για να κερδίσουμε χρόνο και να αποφύγουμε λάθη.
- Προσπαθούμε κάθε υποπρόγραμμα **να είναι όσο το δυνατόν πιο ανεξάρτητο από τα άλλα**. Αυτό μας προφυλάσσει από λάθη στο πρόγραμμά μας και επιτρέπει τη χρήση του σε άλλα προγράμματα αργότερα.

Η Python παρέχει ένα μόνο τύπο υποπρογραμμάτων, τις ***συναρτήσεις***, τις οποίες τις θεωρεί ως ***αντικείμενα***.

Συναρτήσεις

Ένα υποπρόγραμμα είναι ένα κομμάτι προγράμματος που έχει γραφεί ξεχωριστά από το υπόλοιπο πρόγραμμα και επιτελεί ένα αυτόνομο έργο.

Ο ορισμός μιας συνάρτησης περιλαμβάνει το όνομά της και τις παραμέτρους εισόδου και 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 ζεύγος παρενθέσεων που μπορούν να περιέχουν 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.

```
def όνομα(παράμετροι):
  εντολές
```

Παράδειγμα

```
def add2(par1, par2):
  result = par1 + par2
  return result
```

Η συνάρτηση αυτή επιστρέφει το άθροισμα των τιμών των δύο παραμέτρων: par1 και par2.

Κλήση συνάρτησης

Μια συνάρτηση μπορεί να καλείται από διάφορα σημεία του κύριου προγράμματος ή και μέσα από μια άλλη συνάρτηση, γράφοντας το όνομά της και τις κατάλληλες παραμέτρους μέσα σε παρενθέσεις.

Παράδειγμα 1ο

```
>>> type(45)
<type 'int'>
```

Παράδειγμα 2ο

```
>>> add2(1,2)
3
```

Κατηγορίες συναρτήσεων

α) αυτές που δεν τροποποιούν το αντικείμενο στο οποίο εφαρμόζονται:

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

β) αυτές που αλλάζουν το αντικείμενο στο οποίο καλούνται να ενεργήσουν:

```
>>> b = ['A', 'B', 'C', 'D']
>>> b.reverse()
>>> print b
['D', 'C', 'B', 'A']
```

γ) αυτές που επιστρέφουν αποτέλεσμα:

```
>>> def add2(par1, par2):
  result = par1 + par2
  return result
>>> print add2(1,2)
3
```

δ) αυτές που δεν επιστρέφουν κάποια τιμή αλλά εκτελούν ενέργειες μέσω των εντολών τους:

Πιθανή τέτοια συνάρτηση η... print

Μεταβλητές και παράμετροι

Ένα **υποπρόγραμμα** αποτελεί ένα ανεξάρτητο τμήμα προγράμματος που μπορεί να καλείται από οποιοδήποτε σημείο του προγράμματος ή άλλων υποπρογραμμάτων. Δέχεται τιμές από το τμήμα προγράμματος που το καλεί και μετά την εκτέλεση των εντολών του επιστρέφει σε αυτό νέες τιμές.

Οι τιμές αυτές που μεταβιβάζονται από το ένα υποπρόγραμμα στο άλλο λέγονται ***παράμετροι*** και γενικά σε γλώσσες προγραμματισμού διακρίνονται σε *παραμέτρους Εισόδου* και *παραμέτρους Εξόδου*.

Μια συνάρτηση μπορεί να δεχθεί παραμέτρους, οι οποίες είναι τιμές που δίνονται / εισέρχονται στη συνάρτηση, έτσι ώστε αυτή να μπορεί να λειτουργήσει αξιοποιώντας τες.

Οι **παράμετροι** μοιάζουν με τις μεταβλητές, καθορίζονται μέσα στο ζευγάρι των παρενθέσεων, στον ορισμό της συνάρτησης και διαχωρίζονται με κόμμα, ενώ οι τιμές αυτών των μεταβλητών ορίζονται, όταν καλούμε τη συνάρτηση.

Οι τιμές με τις οποίες καλούμε μία συνάρτηση να εκτελεστεί λέγονται **ορίσματα**.

Παράδειγμα 1ο

```
def max2(x1,x2):
  if x1>x2:
    m=x1
  else:
    m=x2
  return m
```

Παράμετροι είναι τα x1 και x2

Παράδειγμα 2ο

```
def max2(x1,x2):
  if x1>x2:
    m=x1
  else:
    m=x2
  return m
```

Μέσα στη συνάρτηση, τα ορίσματα εκχωρούνται σε μεταβλητές οι οποίες ονομάζονται παράμετροι.

```
>>> print max2(4,5)
5
```

Εδώ, τα ορίσματα είναι σταθερές τιμές

```
>>> a=7
>>> b=2
>>> print max2(a,b)
7
```

Εδώ, τα ορίσματα είναι μεταβλητές

Παράδειγμα 3ο

Μέσα στη συνάρτηση, τα ορίσματα εκχωρούνται σε μεταβλητές οι οποίες ονομάζονται παράμετροι.

Συνάρτηση που παίρνει ένα όρισμα και το τυπώνει δύο φορές:

```
def print_twice(x):
  print x
  print x
```

Το x μπορεί να είναι οτιδήποτε μπορεί να εμφανιστεί: int, float, str, ...

```
>>> print_twice("Spam")
Spam
Spam
>>> print_twice(25)
25
25
>>> print_twice(math.pi)
3.14159265359
3.14159265359
```

Παράδειγμα 4ο

Όταν οι παράμετροι περνάνε με αναφορά, αν αλλάξουμε μια παράμετρο μέσα στη συνάρτηση, η αλλαγή είναι μόνιμη και μετά την κλήση της συνάρτησης.

Συνάρτηση που προσθέτει το στοιχείο 45 σε λίστα:

```
def changeme(mylist):
  mylist.append(45)
  print "Μέσα στη συνάρτηση: ", mylist
  return
```

Παράδειγμα 5ο

Οι παράμετροι στις συναρτήσεις είναι τοπικές.

Αλλάζοντας την τιμή μιας μεταβλητής-παραμέτρου μέσα στη συνάρτηση, η αλλαγή αυτή εφαρμόζεται μόνο στην "περιοχή" της συνάρτησης και δεν αλλάζει την τιμή της μεταβλητής-ορίσματος στο πρόγραμμα απ' όπου κλήθηκε!

Ας δούμε ένα παράδειγμα:

```
def changeme(mylist):
  print "[α]", mylist
  mylist = [1,2,3]
  print "[β]", mylist
  return
mylist = [10,20,30]
changeme(mylist)
print "[γ]", mylist
```

το οποίο θα τυπώσει:

```
[α] [10, 20, 30]
[β] [1, 2, 3]
[γ] [10, 20, 30]
```

Παράδειγμα 6ο

Με το παρακάτω παράδειγμα, ίσως, είναι εμφανέστερο ότι οι μεταβλητές - παράμετροι εντός των συναρτήσεων είναι τοπικές:

```
def fun(x):
  print "[α]", x
  x = 1
  print "[β]", x
  return
x = 2
fun(x)
print "[γ]", x
```

το οποίο θα τυπώσει:

```
[α] 2
[β] 1
[γ] 2
```

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.

✖ 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

```
total = 0 # καθολική μεταβλητή
def sum(a1,a2):
  total = a1 + a2 # τοπική μεταβλητή
  print "[α]", total
  return total
sum(10, 20)
print "[β] ", total
```

θα τυπώσει:

```
[α] 30
[β]  0
```

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:

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

θα τυπώσει:

```
2
```

Παράδειγμα 1ο

```
x = 5
def fun1():
  x = 10
  print x
def fun2():
  print x
fun1()
fun2()
```

θα τυπώσει:

```
10
5
```

Παράδειγμα 2ο

```
x = 5
def fun1():
  global x
  x = 10
  print x
def fun2():
  print x
fun1()
fun2()
```

θα τυπώσει:

```
10
10
```

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.

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

import module1[, module2[,... moduleN]]

π.χ.

```
import random, math
```

from modname import name1[, name2[,... nameN]]

Δίνει τη δυνατότητα εισαγωγής συγκεκριμένων χαρακτηριστικών - δυνατοτήτων του module, χωρίς να εισάγει όλο το module

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)

Παράδειγμα:

```
from math import sqrt
def cube(x):
  return x * x * x
# ενσωματωμένη
print abs(-1)
# ορισμένη από τον προγραμματιστή
print cube(5)
# από το module math
print sqrt(81)
```

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.

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...

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...