Συναρτήσεις

Μιχάλης Δρακόπουλος

Μάϊος 2020

ΣΥΝΑΡΤΗΣΕΙΣ

Είδη συναρτήσεων στην Python

- **Ενσωματωμένες στη γλώσσα (built-in)** π.χ. `len`, `sorted`
- **Ορισμένες σε modules της γλώσσας (συναρτήσεις βιβλιοθήκης)** π.χ. `math.sqrt`, `random.random`
- **Ορισμένες από τον χρήστη (user defined)**

Ορισμός συναρτήσεων

```
def name(parameter1, parameter2, ..., parameterN):
    statements
    return value
```

Κλήση συναρτήσεων

```
result = name(arg1, arg2, ..., argN)
```

Παραδείγματα

Χωρίς παραμέτρους και χωρίς επιστρεφόμενη τιμή

[1]:
```
def show_menu():
    print('1. Play game')
    print('2. View High Scores')
    print('3. Quit')
```

[2]:
```
show_menu()
```

```
1. Play game
2. View High Scores
3. Quit
```

Με παραμέτρους και μία επιστρεφόμενη τιμή

[3]:
```
def times(x, y):
    return x*y
```

[4]:
```
times(5, 10)
```

[4]:
```
50
```

Αλλά και:

[5]:
```
times('spam', 4)
```

[5]:
```
'spamspamspamspam'
```

[6]:
```
times(3, [10, 20, 30])
```

[6]:
```
[10, 20, 30, 10, 20, 30, 10, 20, 30]
```

Αυτό είναι μια περίπτωση **πολυμορφισμού**: η συμπεριφορά της συνάρτησης εξαρτάται από τον τύπο των ορισμάτων με τα οποία την καλούμε! Η ίδια συνάρτηση έχει *πολλές μορφές* και κάνει πολλαπλασιασμό αριθμών αν την καλέσουμε με αριθμούς ή επανάληψη για δεδομένα που υποστηρίζουν τον τελεστή επανάληψης *

Εκχώρηση επιστρεφόμενης τιμής

[7]:
```
def plus(x, y):
    return x + y
```

[8]:
```
z = plus(1, 1)
z
```

[8]:
```
2
```

Πολλαπλά σημεία εξόδου (πολλές `return`) Η συνάρτηση θα τερματίσει στην πρώτη `return` που θα εκτελεστεί στην πορεία εκτέλεσης

[9]:
```
def sign(x):
    if x > 0:
        return '+'
    elif x < 0:
        return '-'
    else:
        return '0'
```

[10]:
```
sign(-5)
```

[10]:
```
'-'
```

Συναρτήσεις χωρίς `return` Τερματίζουν όταν εκτελεστεί η τελευταία εντολή τους. *Επιστρέφουν* την ειδική τιμή `None`. Η `print` είναι μια τέτοια συνάρτηση.

[11]:
```
res = print('Hello')
print(res)
```

```
Hello
None
```

Πολλές επιστρεφόμενες τιμές Μέγιστο και ελάχιστο στοιχείο και μέσος όρος μιας αλληλουχίας αριθμών

[12]:
```
def everything(data):
    m = min(data)
    M = max(data)
    mo = sum(data)/len(data)
    return m, M, mo
```

[13]:
```
items = [21, 34, 82, 30, 92, 105]
a, b, c = everything(items)
print('Minimum element: ', a)
print('Maximum element: ', b)
print('Average value: ', c)
```

```
Minimum element:  21
Maximum element:  105
Average value:  60.666666666666664
```

Η συνάρτηση `everything` ουσιαστικά επιστρέφει **μία** τιμή: μία **πλειάδα**

ΠΑΡΕΝΘΕΣΗ: Εκχώρηση στοιχείων αλληλουχίας σε μεταβλητές

- Τα στοιχεία μιας πλειάδας μπορούν να εκχωρηθούν (διαχωριστούν) σε ανεξάρτητες μεταβλητές

[14]:
```
T = (10, 20, 5)
A, B, C = T
print(A, B, C)
```

```
10 20 5
```

Αυτός είναι ο λόγος που μας επιτρέπει να επιστρέψουμε “πολλά πράγματα” με τη `return`: επιστρέφουμε μια πλειάδα!

Μια εφαρμογή: εναλλαγή τιμών 2 μεταβλητών Σε γλώσσες προγραμματισμού χωρίς αντίστοιχο τύπο πλειάδας:

[15]:
```
a = 99
b = 22
temp = a
a = b
b = temp
print('a = ', a, 'b = ', b)
```

```
a =  22 b =  99
```

Με πλειάδες

[16]:
```
a = 100
b = 999
a, b = b, a
print('a = ', a, 'b = ', b)
```

```
a =  999 b =  100
```

Πέρασμα τιμών στα ορίσματα

Κατά την κλήση μιας συνάρτησης, τα ορίσματα είναι **αναφορές** σε αντικείμενα της Python.

- Αν τα ορίσματα είναι αντικείμενα **αμετάβλητου** τύπου (αριθμοί, συμβολοσειρές, κλπ), τυχόν αλλαγές τους μέσα στη συνάρτηση **δεν επηρεάζουν** τις τιμές κλήσης.
- Αν τα ορίσματα είναι αντικείμενα **μετάβλητου** τύπου (λίστες, λεξικά, κλπ), τυχόν αλλαγές τους μέσα στη συνάρτηση **επηρεάζουν** τις τιμές κλήσης.

[17]:
```python
def func(x, L1, L2):

    print('\n*** In func before changes ***')
    print(x)
    print(L1)
    print(L2)

    x = 2*x
    L1 = L1*2
    for i in range(len(L2)):
        L2[i] = L2[i]**2

    print('\n*** In func after changes ***')
    print(x)
    print(L1)
    print(L2)
```

[18]:
```python
t = 500
P1 = [11, 12, 13]
P2 = [10, 20, 30]
func(t, P1, P2)
print('\n*** What has happened ***')
print(t)
print(P1)
print(P2)
```

```
*** In func before changes ***
500
[11, 12, 13]
[10, 20, 30]

*** In func after changes ***
1000
[11, 12, 13, 11, 12, 13]
[100, 400, 900]

*** What has happened ***
```

```
500
[11, 12, 13]
[100, 400, 900]
```

Οι συναρτήσεις ως αντικείμενα

Οι συναρτήσεις στην Python είναι **αντικείμενα**, που δημιουργούνται με την εντολή `def`

- Μπορούν να εκχωρηθούν σε μεταβλητές

[19]:
```
a = times(2, 5)
print(a)
mult = times
b = mult(20, 50)
print(b)
```

```
10
1000
```

- Μπορούν να είναι ορίσματα άλλων συναρτήσεων

Παράδειγμα: Προσέγγιση παραγώγου (μαθηματικής) συνάρτησης σε σημείο x

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

Γράφουμε μια γενικευμένη συνάρτηση (Python)

[20]:
```
def deriv(f, x, h):
    return (f(x+h) - f(x))/h
```

την οποία μπορούμε να καλέσουμε για να υπολογίσουμε προσεγγίσεις παραγώγων συναρτήσεων f μιας μεταβλητής. Για κάθε (μαθηματική) συνάρτηση της οποίας θέλουμε να υπολογίσουμε την παράγωγο γράφουμε κατάλληλη συνάρτηση Python.

$$g(x) = 2x^2 - 3x + 5$$

[21]:
```
def g(x):
    return 2*x**2 - 3*x + 5
```

Η παράγωγος $g'(1) = 1$ προσεγγίζεται με ακρίβεια $h = 0.01$ καλώντας τη συνάρτηση `deriv`:

[22]:
```
fd = deriv(g, 1, 0.01)
print(fd)
```

```
1.019999999999932
```

Μπορούμε να χρησιμοποιήσουμε και μαθηματικές συναρτήσεις από το module `math` της Python. Για την παράγωγο της $\sqrt{x}$ στο $x = 4$ (πραγματική τιμή $1/4$), η υπολογιστική προσέγγιση με ακρίβεια 10^{-4} είναι:

[23]:
```
import math
sqd = deriv(math.sqrt, 4, 0.0001)
print(sqd)
```

```
0.2499984375203823
```

Συναρτήσεις με προεπιλεγμένες (default) τιμές παραμέτρων.

Κάποιες παράμετροι μιας συνάρτησης μπορεί να έχουν προεπιλεγμένες τιμές. Στην περίπτωση αυτή, μπορούμε να μην αντιστοιχίσουμε ορίσματα στις παραμέτρους αυτές όταν καλέσουμε τη συνάρτηση. Για παράδειγμα στην `deriv` η παράμετρος `h` θα μπορούσε να έχει μια default τιμή ακρίβειας, έστω $h = 10^{-4}$:

[24]:
```
def deriv(f, x, h=1e-4):
    return (f(x+h) - f(x))/h
```

Η τιμή αυτή χρησιμοποιείται όταν καλέσουμε την `fdev` με 2 ορίσματα:

[25]:
```
deriv(g, 1)
```

[25]:
```
1.000199999996454
```

Αν φυσικά θέλουμε, μπορούμε να δώσουμε οποιαδήποτε τιμή στην `h` παρακάμπτοντας την default:

[26]:
```
deriv(g, 1, 1e-10)
```

[26]:
```
1.000000082740371
```

Προεπιλεγμένες τιμές συναντάμε και σε πολλές ενσωματωμένες συναρτήσεις της Python:

[27]:
```
help(sum)
```

```
Help on built-in function sum in module builtins:

sum(iterable, start=0, /)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

[28]:
```
sum([1, 2, 3])
```

[28]:
```
6
```

[29]:
```
sum([1, 2, 3], 1000)
```

[29]:
```
1006
```

Μια συνάρτηση μπορεί να έχει περισσότερες από μια παραμέτρους με προεπιλεγμένες τιμές. Οι παράμετροι με προεπιλεγμένες τιμές **πρέπει** να είναι **ορισμένες στο τέλος** της λίστας παραμέτρων.

Κλήση με ονόματα παραμέτρων

Η αντιστοιχία παραμέτρων (ορισμός συνάρτησης) - ορισμάτων (κλήση συνάρτησης) γίνεται συνήθως με βάση τη θέση τους.

Βολή από αρχικό ύψος h_0, με αρχική ταχύτητα v_0 και γωνία βολής θ. Συνάρτηση που υπολογίζει τις συντεταγμένες x, y στο χρόνο t.

[30]:
```
import math

def coord(t, v0, theta = math.pi/4, h0 = 0):
    g = 9.81
    vx = v0*math.cos(theta)
    vy = v0*math.sin(theta)
    x = h0 + vx*t
    y = h0 + vy*t - 0.5*g*t**2
    if y < 0:
        x = (vy + math.sqrt(vy**2 + 2*g*h0))*vx/g
        y = 0
    return x, y
```

[31]:
```
coord(0.6, 5.385)
```

[31]:
```
(2.284662010013735, 0.5188620100137347)
```

Στην Python:

- Μπορούμε να καλέσουμε τη συνάρτηση με τα ονόματα των παραμέτρων της

[32]:
```
coord(t = 0.8, v0 = 6)
```

[32]:
```
(3.3941125496954285, 0.2549125496954274)
```

- Υπάρχει η δυνατότητα να καλέσουμε τη συνάρτηση με διαφορετική σειρά ορισμάτων, **εφόσον** γνωρίζουμε τα ονόματα των παραμέτρων τους.

[33]:
```
coord(v0 = 5.2, h0 = 10, t = 0.6, theta = 1.2)
```

[33]:
```
(11.130556193967221, 11.142161948217746)
```

[34]:
```
coord(v0 = 6, t = 0.4)
```

[34]:
```
(1.6970562748477143, 0.9122562748477139)
```

- Μπορούμε να συνδυάσουμε ορίσματα θέσης με ορίσματα ονόματος, με την **προϋπόθεση** ότι προηγούνται τα ορίσματα θέσης κατά την κλήση.

[35]:
```
coord(0.9, h0 = 10, v0 = 4.5)
```

[35]:
```
(12.863782463805517, 8.890732463805517)
```

- Μπορούμε να παραλείψουμε *ενδιάμεσες* παραμέτρους που έχουν όμως default τιμές.

[36]:
```
coord(h0 = 20, v0 = 6, t = 0.4)
```

[36]:
```
(21.697056274847714, 20.912256274847714)
```

Η κλήση με ονόματα παραμέτρων είναι χρήσιμη όταν:

- Θέλουμε να διαχειριστούμε τις παραμέτρους με κάποιο μνημονικό τρόπο, χωρίς να μας ενδιαφέρει η σειρά τους.
- Θέλουμε να παραλείψουμε κάποιες παραμέτρους, οι οποίες όμως πρέπει να έχουν default τιμές.

Modules

- Οι συναρτήσεις γράφονται σε αρχεία.

- Το σύνολο των συναρτήσεων ενός αρχείου, μαζί με άλλες εντολές που περιέχονται σε αυτό, αποτελούν ένα module.
- Ενα module περιέχει συνήθως:
  - Τις συναρτήσεις που χρειάζεται ένα πρόγραμμα για να τρέξει.
  - Συναρτήσεις με κοινό αντικείμενο και έχει την έννοια μιας βιβλιοθήκης (όπως π.χ. το module `math`)
- Ενα module μπορεί να κάνει `import` άλλα modules
- Ενα module είναι ένας **χώρος ονομάτων (namespace)** τα οποία είναι μοναδικά για το συγκεκριμένο module και διαχωρίζονται από ίδια ονόματα σε άλλα modules.

Η συνάρτηση `main` Η εκτέλεση ενός προγράμματος ξεκινάει από μια βασική *κύρια* συνάρτηση που *παραδοσιακά* αλλά *όχι υποχρεωτικά* ονομάζεται `main`.

Παράδειγμα δημιουργίας και χρήσης module

Εστω το αρχείο `calculus.py` που περιέχει, μεταξύ άλλων, τη παραπάνω συνάρτηση `deriv` και μια συνάρτηση `main`. Αυτό είναι ένα module.

[37]:
```
def deriv(f, x, h = 1e-4):
     return (f(x+h) - f(x))/h

def p(x):
    y = x**2 - 5*x + 2
    return y

def main():
    x = float(input('X ='))
    y = p(x)
    dp = deriv(p, x)
    print('P(', x, ')=', y)
    print('DP(',x, ')=', dp)

main()
```

```
X = 10
```

```
P( 10.0 )= 52.0
DP( 10.0 )= 15.000099999866734
```

Οι μεταβλητές `x` και `y` που ορίζονται στη συνάρτηση `p` είναι **τοπικές μεταβλητές** για τη `p` και είναι διαφορετικές από τις αντίστοιχες τοπικές μεταβλητές της `main`.

Καθολικές μεταβλητές

Θέλουμε να μετατρέψουμε το παραπάνω πρόγραμμα ώστε να δουλεύει με πολυώνυμα οποιουδήποτε βαθμού και οποιουσδήποτε συντελεστές. Οι συντελεστές για το πολυώνυμο

$$P(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0$$

δίνονται στη λίστα `coeff` $= [c_n, c_{n-1}, \ldots, c_1, c_0]$

[38]:
```
def p(x, coeff):
    value = 0
    xpower = 1
    for c in coeff[::-1]:
        value += c*xpower
        xpower *= x
    return value
```

[39]:
```
C = [1, -5, 2]
p(10, C)
```

[39]: 52

Το **πρόβλημα** τώρα είναι ότι δεν μπορούμε να χρησιμοποιήσουμε τη `deriv` η οποία περιμένει η συνάρτηση `f` να έχει *μία μόνο παράμετρο!*

Η **λύση** είναι να βγάλουμε έξω την `coeff`, να την κάνουμε δηλαδή **καθολική** μεταβλητή!

> Οι **τοπικές** μεταβλητές δεν είναι ορατές εκτός της συνάρτησης. Αντίθετα οι **καθολικές** μεταβλητές, ορίζονται εκτός των ορίων της συνάρτησης και είναι *ορατές* και μέσα στη συνάρτηση.
>
> Συνήθως γράφονται με ΚΕΦΑΛΑΙΑ γράμματα για να ξεχωρίζουν από τις τοπικές μεταβλητές.

Το module `calculus.py` στη γενική του μορφή γίνεται:

[40]:
```
def deriv(f, x, h=1e-4):
     return (f(x+h) - f(x))/h

COEFF = []

def p(x):
    value = 0
    xpower = 1
    for c in COEFF[::-1]:
        value += c*xpower
        xpower *= x
```

```
    return value

def main():
    n = int(input('Degree of polynomial? '))
    global COEFF
    COEFF.clear()
    for i in range(n,-1,-1):
        message = 'C' + str(i) + '? '
        c = float(input(message))
        COEFF.append(c)
    x = float(input('X ='))
    y = p(x)
    dp = deriv(p, x)
    print('P(', x, ')=', y)
    print('DP(',x, ')=', dp)

main()
```

```
Degree of polynomial?  4
C4?  1
C3?  -3
C2?  12
C1?  5
C0?  22
X = 14

P( 14.0 )= 32628.0
DP( 14.0 )= 9553.106200473849
```

Οι συναρτήσεις p και `deriv` μπορούν να χρησιμοποιηθούν και σε άλλα modules. Για να γίνει αυτό θα πρέπει να γράψουμε στο άλλο module:

```
import calculus
```

και οι p, `deriv` και η καθολική μεταβλητη `COEFF` θα είναι ορατές ως `calculus.p`, `calculus.deriv` και `calculus.COEFF` αντίστοιχα.