MEM104 Γλώσσα Προγραμματισμού Ι

Μιχάλης Πλεξουσάκης, Ιωάννης Λιλής

2-6 Νοεμβρίου 2020

Μαθηματικά και Εφαρμοσμένα Μαθηματικά

Περιεχόμενα

Συναρτήσεις

Συναρτήσεις

Οι *συναρτήσεις* (functions) είναι ένας μηχανισμός της Python (και πολλών άλλων γλωσσών) που επιτρέπει:

- Την επαναχρησιμοποίηση και γενίκευση κώδικα
- Τον κατακερματισμό μεγάλων υπολογιστικών καθηκόντων σε μικρότερα
- Την απόκρυψη των λεπομερειών μιας λειτουργίας από μέρη του κώδικα που δεν τις χρειάζονται

Η Python, παρέχει τρόπους ορισμού και χρήσης συναρτήσεων, όπως τις ενσωματωμένες συναρτήσεις `abs`, `len`, `print`, κλπ.

Συναρτήσεις

Στην Python, ο ορισμός μιας συνάρτησης έχει τη μορφή:

`def` *όνομα συνάρτησης*`(` *λίστα τυπικών παραμέτρων* `):`
    εντολές συνάρτησης

Θα μπορούσαμε, για παράδειγμα, να γράψουμε

```
def maxx(x, y):
    if x > y:
        return x
    else:
        return y
```

για να ορίσουμε μια συνάρτηση με το όνομα `maxx` που υπολογίζει τον μεγαλύτερο δύο αριθμών.

Συναρτήσεις

- Η `def` είναι λέξη-κλειδί που ενημερώνει την Python ότι ακολουθεί ορισμός συνάρτησης.
- Το όνομα της συνάρτησης ακολουθεί τους κανόνες ονοματοδοσίας των μεταβλητών.
- Όταν χρησιμοποιείται η συνάρτηση, οι τυπικές παράμετροι συνδέονται με τις πραγματικές παραμέτρους ή *ορίσματα* της *κλήσης* της συνάρτησης. Για παράδειγμα, η κλήση `maxx(3, 4)` συνδέει το `x` με το 3 και το `y` με το 4.
- Όλες οι εντολές της Python μπορούν να εμφανίζονται στο σώμα μιας συνάρτησης. Η εντολή `return` μπορεί να εμφανίζεται μόνο μέσα στο σώμα μιας συνάρτησης.

Συναρτήσεις

- Η κλήση μιας συνάρτησης είναι μια παράσταση και όπως όλες οι παραστάσεις έχει τιμή. Αυτή είναι η τιμή η οποία επιστρέφεται (`return`-ed) από τη συνάρτηση που καλείται.
- Η εκτέλεση μιας εντολής `return` τερματίζει την κλήση μιας συνάρτησης.

Κατά την κλήση μιας συνάρτησης συμβαίνουν τα ακόλουθα:

1. Υπολογίζονται οι παραστάσεις που απαρτίζουν τις πραγματικές παραμέτρους και οι τυπικές παράμετροι συνδέονται με αυτές τις τιμές. Για παράδειγμα, κατά την κλήση `maxx(3+4, z)` η τιμή 7 συνδέεται με την τυπική παράμετρο `x` και η τιμή της μεταβλητής `z` συνδέεται με την τυπική παράμετρο `y`.

Συναρτήσεις

2. Η εκτέλεση του προγράμματος μεταφέρεται από το σημείο κλήσης στην πρώτη εντολή στο σώμα της συνάρτησης.
3. Οι εντολές στο σώμα της συνάρτησης εκτελούνται μέχρι να κληθεί μια εντολή `return`. Η τιμή της παράστασης ακριβώς δίπλα από την εντολή `return` γίνεται η τιμή της κλήσης της συνάρτησης. Αν δεν υπάρχει παράσταση μετά τη λέξη `return` η τιμή της κλήσης είναι `None`.
4. Η τιμή `None` επιστρέφεται όταν εξαντληθούν οι εντολές στο σώμα της συνάρτησης, χωρίς προηγούμενη εκτέλεση κάποιας εντολής `return`.
5. Η ροή εκτέλεσης του προγράμματος συνεχίζεται με την εντολής μετά την κλήση της συνάρτησης.

Συναρτήσεις

Η σύνδεση των τυπικών παραμέτρων με τις πραγματικές παραμέτρους μπορεί να γίνει με δύο τρόπους:

- Κατά τη λεγόμενη *θεσιακή* (positional) μέθοδο, η πρώτη τυπική παράμετρος συνδέεται με την πρώτη πραγματική παράμετρο, η δέυτερη τυπική παράμετρος συνδέεται με την δεύτερη πραγματική παράμετρο, κ.λ.π.
- Κατά τη δεύτερη μέθοδο, οι τυπικές παράμετροι συνδέονται με τις πραγματικές με χρήση του ονόματος της τυπικής παραμέτρου.

Συναρτήσεις

Η συνάρτηση `printName` απαιτεί τα `firstName` και `lastName` να είναι συμβολοσειρές και η τυπική παράμετρος `rev` μια λογική τιμή. Αν `rev == False` εκτυπώνει `firstName lastName`, διαφορετικά εκτυπώνει `lastName, firstName`:

```
def printName(firstName, lastName, rev):
    if rev:
        print(lastName + ', ' + firstName)
    else:
        print(firstName, lastName)
```

Παρατηρήστε ότι δεν εμφανίζεται η εντολή `retrun` άρα η συνάρτηση επιστρέφει κατά την κλήσης της την τιμή `None`.

Συναρτήσεις

Όλες οι παρακάτω κλήσεις της συνάρτησης `printName` θα τυπώσουν ακριβώς το ίδιο μήνυμα:

```
printName('Mickey', 'Mouse', False)
printName('Mickey', 'Mouse', rev = False)
printName(firstName = 'Mickey', lastName = 'Mouse', rev = False)
printName('Mickey', lastName = 'Mouse', rev = False)
printName(lastName = 'Mouse', firstName = 'Mickey', rev = False)
```

Όμως η κληση

```
printName('Mickey', lastName = 'Mouse', False)
```

θα παράξει ένα μήνυμα σφάλματος επειδή το ζεύγος *όνομα τυπικής παραμέτρου = όρισμα* δεν μπορεί να εμφανίζεται πριν από ένα όρισμα (χωρίς το όνομα της τυπικής παραμέτρου).

Συναρτήσεις

Το συντακτικό *όνομα τυπικής παραμέτρου = όρισμα* χρησιμοποιείται συνήθως με τις *προεπιλεγμένες τιμές παραμέτρων*. Θα μπορούσαμε να γράψουμε

```
def printName(firstName, lastName, rev= False):
    if rev:
        print(lastName + ', ' + firstName)
    else:
        print(firstName, lastName)
```

και τότε οι κλήσεις

```
printName('Mickey', 'Mouse')
printName('Mickey', 'Mouse', rev = True)
printName('Mickey', 'Mouse', True)
```

Συναρτήσεις

θα παράξουν τα

```
Mickey Mouse
Mouse, Mickey
Mouse, Mickey
```

Συναρτήσεις

Εμβέλεια ονομάτων: Κάθε συνάρτηση ορίζει το δικό της *χώρο ονομάτων* (name space). Θα μπορούσαμε λοιπόν να χρησιμοποιήσουμε το ίδιο όνομα για μια τυπική παράμετρο και ένα όρισμα μιας συνάρτησης.

```
def f(x):
    y = 1
    x = x + y
    print('x = ', x)
    return x

x = 3;  y = 2
z = f(x)
print('x = ', x, 'y = ',  y,  'z = ', z)
```

θα εκτυπώσει

```
x = 4
x = 3 y = 2 z = 4
```

Μαθηματικές συναρτήσεις

Η Python παρέχει υλοποιήσεις πολλών μαθηματικών συναρτήσεων στη βιβλιοθήκη `math`. Για να χρησιμοποιήσουμε τις συναρτήσεις από τη βιβλιοθήκη αυτή πρέπει πρώτα να ενημερώσουμε την Python με την εντολή `import math`. Για παράδειγμα,

```
>>> import math
>>> math.exp(1)
2.718281828459045
```

Παρατηρήστε ότι η λέξη `math` εμφανίζεται ως πρόθεμα της συνάρτησης με το όνομα `exp`, της εκθετικής συνάρτησης. Το όρισμα των περισσοτέρων μαθηματικών συναρτήσεων είναι μετατρέπεται στον τύπο `float`.

Μαθηματικές συναρτήσεις

Μπορούμε επίσης να συντομεύσουμε κάπως την κλήση των μαθηματικών συναρτήσεων γράφοντας

```
>>> import math as m
```

και καλώντας, για παράδειγμα, τη συνάρτηση τετραγωνική ρίζα ως

```
>>> m.sqrt(3.0)
1.7320508075688772
```

Θα μπορούσαμε ακόμα να γράψουμε

```
>>> from math import sin, cos
>>> sin(1) + cos(1)
1.3817732906760363
```

για τη χρήση μικρού αριθμού μαθηματικών συναρτήσεων και χωρίς το πρόθεμα `math`.

Μαθηματικές συναρτήσεις

Μαθηματικές σταθερές. Η Python ορίζει τις μαθηματικές σταθερές `math.pi`, `math.e` και `math.tau` με την τελευταία ίση με 2π:

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.e
2.718281828459045
>>> math.tau
6.283185307179586
```

Ορίζονται ακόμα οι σταθερές `math.inf` (για το $+\infty$) και `math.nan` (‘Not A Number’).

Αριθμο-θεωρητικές συναρτήσεις

- `math.gcd(a, b)`. Ο μέγιστος κοινός διαιρέτης των ακεραίων a και b.
- `math.factrorial(n)`. Ο αριθμός $n!$.
- `math.comb(n, k)`. Ο αριθμός των συνδυασμών n αντικειμένων ανά k.
- `math.perm(n, k)`. Ο αριθμός των συνδυασμών n αντικειμένων ανά k όπου η διάταξη έχει σημασία.
- `math.floor(x)`. Ο μεγαλύτερος ακέραιος μικρότερος ή ίσος του x.
- `math.ceil(x)`. Ο μικρότερος ακέραιος μεγαλύτερος ή ίσος του x.
- `math.trunc(x)`. Το ακέραιο μέρος του x.

Δυνάμεις, εκθετικές, λογαριθμικές συναρτήσεις

- `math.sqrt(x)`. Η τετραγωνική ρίζα του x.
- `math.pow(x, y)`. Ο αριθμός x^y. Προτιμήστε τον τελεστή δύναμης `**` για τον υπολογισμό ακέραιων δυνάμεων.
- `math.exp(x)`. Ο αριθμός e^x.
- `math.log(x)`. Ο φυσικός λογάριθμός του x.
- `math.log10(x)`. Ο δεκαδικός λογάριθμος του x.
- `math.log2(x)`. Ο λογάριθμος του x με βάση το 2.
- `math.log(x, `*base*`)`. Ο λογάριθμος του x με βάση τον αριθμό *base*.

Τριγωνομετρικές συναρτήσεις

Οι τριγωνομέτρικές συναρτήσεις της Python είναι οι `sin`, `cos` και `tan`, το όρισμα των οποίων είναι σε ακτίνια. Έτσι, αν, για παράδειγμα θέλουμε να υπολογίσουμε το ημίτονο των 30° θα πρέπει να γράψουμε

```
math.sin(30 * math.pi / 180)
```

Η Python παρέχει τις συναρτήσεις `math.degrees(x)` για μετατροπή από ακτίνια σε μοίρες και `math.radians(x)` για την αντίστροφη μετατροπή.

Τριγωνομετρικές συναρτήσεις

Οι αντίστροφες τριγωνομετρικές συναρτήσεις που ορίζονται στη μαθηματική βιβλιοθήκη είναι οι `asin(x)`, `acos(x)` και `atan(x)`, οι οποίες επιστρέφουν γωνίες σε ακτίνια.

Ορίζεται ακόμα η συνάρτηση `atan2(y, x)` η οποία επιστρέφει τη γωνία που σχηματίζει η ευθεία που ενώνει τα σημεία (x, y) και $(0, 0)$ με τον θετικό x-ημιάξονα.

Αναδρομή

Αναδρομικές συναρτήσεις

Μια συνάρτηση λέγεται *αναδρομική* αν καλεί τον εαυτό της κατά τη διάρκεια της εκτέλεσής της. Για παράδειγμα,

```
def f(n):
    if n <= 1:
        return 0
    else:
        return (n-1) + f(n-1)
```

Κατ' αρχάς, η κλήση `f(1)` θα επιστρέψει την τιμή 0 και η κλήση `f(2)` θα επιστρέψει `1 + f(1)`, δηλαδή `1`. Ανάλογα, η κλήση `f(3)` θα επιστρέψει `3` και η κλήση `f(10)` θα επιστρέψει `45`.

Ποιά θα μπορούσε να είναι η χρησιμότητα αυτής της συνάρτησης;

Αναδρομικές συναρτήσεις

Πρόβλημα. Σε ένα πάρτυ με n καλεσμένους, ο καθένας σφίγγει το χέρι όλων των άλλων. Πόσες συνολικά χειραψίες θα γίνουν;

Κάθε καλεσμένος σφίγγει το χέρι των υπολοίπων $n-1$. Θα γίνουν λοιπόν ακριβώς $n(n-1)/2$ χειραψίες.

Εναλλακτικά, έστω $f(n)$ ο αριθμός τψν χειραψιών σε ένα πάρτυ με n καλεσμένους. Προφανώς $f(1) = 0$. Αν ένας από τους καλεσμένους σφίξει το χέρι των υπολοίπων, τότε θα γίνουν $n-1$ χειραψίες και το πρόβλημα ανάγεται στο να βρούμε τον αριθμό των χειραψιών σε ένα πάρτυ με $n-1$ καλεσμένους, δηλαδή,

$$f(1) = 0, \qquad f(n) = (n-1) + f(n-1), \quad n \geq 1 \tag{1}$$

Αναδρομικές συναρτήσεις

Αυτόν ακριβώς τον τρόπο σκέψης υλοποιεί η συνάρτηση

```
def f(n):
     if n <= 1:
         return 0
    else:
        return (n-1) + f(n-1)
```

Εύκολα επιβεβαιώνουμε ότι υπολογίζει το σωστό αποτέλεσμα για κάθε $n \geq 2$. Για παράδειγμα, η κλήση `f(100)` επιστρέφει `4950` και $100 \cdot 99/2 = 4950$.

Εξάλλου, εύκολα δείχνει κανείς με επαγωγή ότι η λύση της αναδρομικής σχέσης (1) είναι όντως $n(n-1)/2$ για $n \geq 1$.

Αναδρομικές συναρτήσεις

Άλλο παράδειγμα αναδρομικά ορισμένης συνάρτησης είναι η συνάρτηση $n!$ (n παραγοντικό, n factorial) που ορίζεται από τις σχέσεις $0! = 1$ και $n! = n(n-1)!$ για $n \geq 1$.

Η αναδρομική υλοποίησή της είναι, φυσικά, η

```
def fact(n):
    if n <= 0:
        return 1
    else:
        return n * fact(n-1)
```

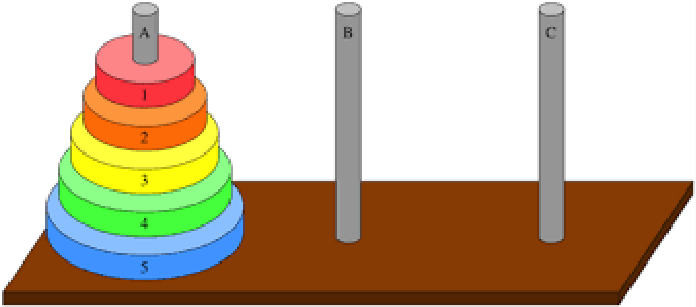
Εύκολα μπορούμε να δείξουμε (με επαγωγή) ότι $n! = 1 \cdot 2 \cdots n$ και επομένως η μή αναδρομική υλοποίηση της συνάρτησης παραγοντικό είναι επίσης προφανής. (ποιά είναι;)

Αναδρομικές συναρτήσεις

Οι πύργοι του Hanoi. Ένα puzzle κατά το οποίο ένας αριθμός δίσκων με διαφορετικά μεγέθη πρέπει να μετακινηθούν από τον ένα στύλο στον άλλο ακολουθώντας τους εξής κανόνες:

- Ένας μόνο δίσκος μετακινείται κάθε φορά
- Κάθε κίνηση συνίσταται στη μετακίνηση του πρώτου από επάνω δίσκου σε έναν άδειο στύλο ή πάνω σε ένα μεγαλύτερο δίσκο.
- Ένας δίσκος μπορεί να τοποθετηθεί μόνο επάνω σε δίσκο με μεγαλύτερη ακτίνα.

Αναδρομικές συναρτήσεις



Αναδρομικές συναρτήσεις

Μια στρατηγική για τη λύση αυτού του προβλήματος με n δίσκους θα μπορούσε να είναι η ακόλουθη:

- Μεταφέρουμε $n-1$ δίσκους από τον στύλο A στον στύλο B (χρησιμοποιώντας, αν χρειάζεται, τον στύλο C)
- Μεταφέρουμε τον τελευταίο δίσκο από τον στύλο A στον στύλο C
- Μεταφέρουμε $n-1$ δίσκους από τον στύλο B στον στύλο C (χρησιμοποιώντας, αν χρειάζεται, τον στύλο A)

Για παράδειγμα, αν $n=3$, η μεταφορά των δίσκων από τον στύλο A στον στύλο C μπορεί να γίνει με τις κινήσεις:

$$A \to C,\ A \to B,\ C \to B,\ A \to C,\ B \to A, B \to C,\ A \to C$$

Αναδρομικές συναρτήσεις

Η υλοποίηση αυτής της στρατηγικής είναι εύκολη:

```
def hanoi(n, start, finish, using):
    if n == 0:
        return

#   Move n-1 disks from A to B, using peg C

    hanoi(n-1, from, using, finish)

#   Move a disk from A to C

    print('Move a disk from', start, 'to', finish)

#   Move n-1 disks from B to C, using peg A

    hanoi(n-1, using, finish, from)
```

Αναδρομικές συναρτήσεις

Η κλήση της συνάρτησης `hanoi(3, 'A', 'C', 'B')` θα τυπώσει τις επτά κινήσεις που απαιτούνται:

```
Move a disk from A to C
Move a disk from A to B
Move a disk from C to B
Move a disk from A to C
Move a disk from B to A
Move a disk from B to C
Move a disk from A to C
```

Ο ενδιαφερόμενος αναγνώστης θα πρέπει να αναρωτηθεί αν αυτές είναι οι ελάχιστες κινήσεις που απαιτούνται…

Αναδρομικές συναρτήσεις

Η αναδρομική συνάρτηση που φαίνεται παρακάτω επιστρέφει `True` αν η συμβολοσειρά που δίνεται ως όρισμα είναι παλινδρομική, διαφορετικά επιστρέφει `False`:

```
def isPalindrome(s):
    if len(s) <= 1:
        return True
    else:
        return s[0] == s[-1] and isPalindrome(s[1:-1])
```

Αναδρομικές συναρτήσεις

Η ακολουθία του Fibonacci ορίζεται από τις σχέσεις $f_0 = 0$, $f_1 = 1$ και

$$f_n = f_{n-1} + f_{n-2}, \quad n \geq 2.$$

Μια αναδρομική υλοποίηση του υπολογισμού των όρων της ακολουθίας θα μπορούσε να είναι:

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```