ΜΕΜ104 Γλώσσα Προγραμματισμού Ι

Μιχάλης Πλεξουσάκης, Ιωάννης Λιλής

12 Ιανουαρίου 2021

Μαθηματικά και Εφαρμοσμένα Μαθηματικά

Περιεχόμενα

Σύντομη επανάληψη της Python

Βασικοί τύποι

Οι βασικοί τύποι της Python είναι οι `int`, `float`, `boolean`, `string` και `bytes`.

`int` 182 0 -831 0o742 (octal) 0x5f1a4 (hexadecimal)

Τα ψηφία στο δεκαεξαδικό σύστημα είναι τα 0–9 και $a = 10$, $b = 11$, $c = 12, d = 13, e = 14, f = 15$.

`float` 14.96 0.0 -4.87 1.35e−6 (δηλ. 1.35×10^{-6})

`bool` `True` `False`

`str` ‘One’ “Two” ”I\'m OK” ’First line.\nSecond line.’

Ονόματα

Ονόματα μεταβλητών, συναρτήσεων, κλάσεων, modules...

Ο πρώτος χαρακτήρας του ονόματος μπορεί να είναι ένας από τους `a..zA..Z_` και οι υπόλοιποι μεταξύ των

`a..zA..Z_0...9`

Πεζά και κεφαλαία νοούνται ως διαφορετικοί χαρακτήρες ενώ δεν μπορούν να χρησιμοποιηθούν ως ονόματα οι δεσμευμένες λέξεις της Python.

`my_name` `x_min` `f32` `__valid_but_long`

~~`1_num`~~ ~~`@name`~~ ~~`first-time`~~

Αναθέσεις

Ο τελεστής ανάθεσης είναι ο `=`. Η εντολή

όνομα `=` *τιμή*

αντιστοιχεί ένα *όνομα* με μια *τιμή*.

```
n = 2
x = 1 + 0.2 * 1.32e-7
x = y = z = 10
a, b = 1, 2;  a, b = b, a
p = None
```

`x += 2` (ισοδύναμο με `x = x + 2`)

```
x -= 3;  x *= 4;  x /= 3.2;  n //= 10
```

Ακολουθίες

Λίστες

`list [3, 8, 1] ['xyz', 3.4, 8] ["one"] []`

Πλειάδες (οι παρενθέσεις είναι προαιρετικές)

`tuple (4, 9, 2) 9, 'zzz', -2.3 (1,) ()`

Συμβολοσειρές

`str "one string" "one with escaped\tchars"`

Θυμόμαστε ότι οι πλειάδες και οι συμβολοσειρές είναι *μη μεταλλάξιμες*, δηλαδή δεν μπορούμε να αλλάξουμε τις τιμές των στοιχείων τους μετά τη δημιουργία της πλειάδας.

Μετατροπές μεταξύ τύπων

`int("32")` $\longrightarrow$ 15
`int("3f", 16)` $\longrightarrow$ 63 (δεύτερη παράμετρος είναι η βάση του αριθμ. συστήματος)

`float("-3.3e5")` $\longrightarrow$ -330000.0
`round(378.227, 2)` $\longrightarrow$ 378.23 (στρογυλλοποίηση σε 2 δεκαδικά ψηφία)

`str(x)` $\longrightarrow$ `"..."` (συμβολοσειρά αναπαράστασης του `x` για εκτύπωση)

`bool(x)` $\longrightarrow$ `False` αν `x` είναι `None` ή κενή ακολουθία ή έχει την τιμή `False`. Σε διαφορετική περίπτωση είναι `True`.

Μετατροπές μεταξύ τύπων

`list("abc")` $\longrightarrow$ `["a", "b", "c"]`

Η μέθοδος `join`:
διαχωριστικό και ακολουθία από `str` $\longrightarrow$ συναρμολογημένη συμβολοσειρά

`":".join(["12", "01", "2021"])` $\longrightarrow$ `"12:01:2021"`

Η μέθοδος `split`:
`"words spaces".split()` $\longrightarrow$ `["words", "spaces"]`
`"2, 3, 9, 4".split(",")` $\longrightarrow$ `["2", "3", "9", "4"]`

Συμπερίληψη λίστας: μετατροπή ακολουθίας κάποιου τύπου σε λίστα με στοιχεία κάποιου άλλου τύπου:
`[int(x) for x in ('1', '-2')]` $\longrightarrow$ `[1, -2]`

Απαρίθμηση στοιχείων ακολουθίας

Αν `lst = [10, 20, 30, 40, 50]` τότε `len(lst) = 5` και

`lst[0]` $\longrightarrow$ 10 `lst[1]` $\longrightarrow$ 20 `lst[4]` $\longrightarrow$ 50
`lst[-1]` $\longrightarrow$ 50 `lst[-2]` $\longrightarrow$ 40

Υπο-ακολουθίες με το συντακτικό `[start:end:step]`

`lst[2:-1]` $\longrightarrow$ `[30, 40]` `lst[:-2]` $\longrightarrow$ `[10, 20, 30]`

`lst[2:]` $\longrightarrow$ `[30, 40, 50]`

`lst[::-1]` $\longrightarrow$ `[50, 40, 30, 20, 10]`

`lst[::2]` $\longrightarrow$ `[10, 30, 50]`

`lst[-1:1:-1]` $\longrightarrow$ `[50, 40, 30]`

Όταν το `step` παραλείπεται νοείται ως 1

Αριθμητικοί τελεστές και προτεραιότητα

Οι αριθμητικοί τελεστές είναι: `+ - * / // % **`
Προτεραιότητα: χαμηλή $\longrightarrow$ υψηλή

`abs(-4.8)` $\longrightarrow$ 4.8

`round(3.14, 1)` $\longrightarrow$ 3.1

`pow(2, 5)` $\longrightarrow$ 32

Θυμόμαστε ότι:

- Ο τελεστής `/` παράγει αποτέλεσμα τύπου `float`
- `//` είναι ο τελεστής της ακέραιας διαίρεσης
- `%` είναι ο τελεστής υπολοίπου
- Η προσεταιριστικότητα του τελεστή `**` είναι από δεξιά προς τ' αριστερά

Τελεστές σύγκρισης και λογικοί τελεστές

Τελεστές σύγκρισης: `< > <= >= == !=`

`bool(3 <= 4)` $\longrightarrow$ `True`
`bool(-2 < 3 > 0)` $\longrightarrow$ `True`

Προτεραιότητα των λογικών τελεστών (υψηλότερη προς χαμηλότερη): `not`, `and`, `or`

```
leap=year%4==0 and year%100!=0 or year%400==0
```

είναι ισοδύναμο με

```
leap=(year%4==0 and year%100!=0) or year%400==0
```

Η μαθηματική βιβλιοθήκη

```
import math
math.sin( math.pi / 4)
math.sqrt(123)
math.ceil(7.6)
math.floor(8.4)

import math as m
m.exp(1.0)

from math import sin, cos, pi
sin(pi/6) + cos(pi/3)
```

Τα ορίσματα των τριγωνομετρικών συναρτήσεων σε ακτίνια!

Εντολή συνθήκης

`if` *λογική συνθήκη*`:`
⟶ | *εντολές*

μπορεί να ακολουθείτε από μια ή περισσότερες εντολές `elif` και ένα μόνο `else`

```
if age <= 10:
  state = "Kid"
elif age > 65:
  state = "Retired"
else:
  state = "Active"
```

`if bool(x) == True:` είναι ισοδύναμο με το `if x:`

`if bool(x) == False:` είναι ισοδύναμε με το `if not x:`

Εντολή επανάληψης while

`while` *λογική συνθήκη*`:`
$\longrightarrow$ | *εντολές*

Η εντολή `continue` μεταφέρει τη ροή στην επόμενη επανάληψη ενώ η εντολή `break` μεταφέρει τη ροή στην πρώτη εντολή που καολουθεί την εντολή επανάληψης.

```
s = 0
i = 1
while i <= 10:
  s = s + i**2
  i = i + 1
```

Εντολή επανάληψης for

`for` *μεταβλητή* `in` *ακολουθία*`:`
⟶ | *εντολές*

Η εντολή `continue` μεταφέρει τη ροή στην επόμενη επανάληψη ενώ η εντολή `break` μεταφέρει τη ροή στην πρώτη εντολή που καολουθεί την εντολή επανάληψης.

```
s = 'some string'
k = 0
for c in s:
  if c == 's':
    k += 1

print("Found", k, "\'s\' in the string", s)
```

Εντολή επανάληψης for

Επανάληψη με χρήση δεικτών:

```
lst = [3, 12, 8.9, 17.4, 5]
big = []

for i in range(len(lst)):
  v = lst[i]
  if v > 9:
    big.append(v)
    lst[i] = 9

print("New list:", lst)
print("Discarded values:", big)
```

Ακολουθίες ακεραίων για την επανάληψη for

`range(start, end, step)` Μόνος υποχρεωτικός δείκτης είναι ο `end`. Αν το `start` δεν υπάρχει νοείται ως μηδέν. Αν ο δείκτης `step` δεν υπάρχει νοείται ως 1.

`range(len(seq))` $\longrightarrow$ ακολουθία δεικτών των τιμών της ακολουθίας `seq`

`range(4)` $\longrightarrow$ 0 1 2 3

`range(3, 14, 4)` $\longrightarrow$ 3 7 11

`range(20, 5, -5)` $\longrightarrow$ 20 15 10

Θυμόμαστε ότι μπορούμε να προσπελάσουμε και τους δείκτες και τις τιμές των στοιχείων με την εντολή:
`for idx, val in enumerate(lst):`

Πράξεις σε ακολουθίες

`len(seq)` $\longrightarrow$ αριθμός στοιχείων της `seq`

`min(seq)` `max(seq)` `sum(seq)` $\longrightarrow$ ελάχιστο, μέγιστο στοιχείο, άθροισμα στοιχείων

`sorted(seq)` $\longrightarrow$ ταξινομημένη λίστα των στοιχείων της `seq`

`sorted('mama')` $\longrightarrow$ `['a', 'a', 'm', 'm']`

`sorted(seq, reverse=True)` $\longrightarrow$ ταξινομημένη με φθίνουσα σειρά λίστα των στοιχείων της `seq`

`val in seq` $\longrightarrow$ `True` αν η `val` είναι στοιχείο της `seq`

`val not in seq` $\longrightarrow$ `True` αν η `val` δεν είναι στοιχείο της ακολουθίας `seq`

`seq1 + seq2` $\longrightarrow$ συνένωση ακολουθιών

`n * seq` $\longrightarrow$ επανάληψη των στοιχείων της `seq` n φορές

Πράξεις σε ακολουθίες

`reversed(seq)` $\longrightarrow$ reversed iterator

```
lst = [10, 20, 30, 40]
for e in reversed(lst):
  print(e)
```

`seq.index(val)` $\longrightarrow$ δείκτης πρώτης εμφάνισης της τιμής `val` στην ακολουθία `seq`

`seq.count(val)` $\longrightarrow$ αριθμός εμφανίσεων της τιμής `val` στην ακολουθία `seq`

`zip(seq1, seq2, ...)` $\longrightarrow$ iterator on tuples

```
name = ['Mary', 'John', 'Anna']
id = [9889, 7342, 8666]; age = [19, 21, 20]
for t in zip(name, id, age): print(t)
```

Μέθοδοι σε λίστες

`lst.append(val)` $\longrightarrow$ προσθήκη στο τέλος της λίστας

`lst.extend(seq)` $\longrightarrow$ προσθήκη ακολουθίας στο τέλος

`lst.insert(idx, val)` $\longrightarrow$ εισαγωγή στοιχείου `val` στη θέση `idx`

`lst.remove(val)` $\longrightarrow$ διαγραφή πρώτου στοιχείου με τιμή `val`

`lst.pop()` $\longrightarrow$ διαγραφή και επιστροφή τελευταίου στοιχείου

`lst.pop(idx)` $\longrightarrow$ διαγραφή και επιστροφή του στοιχείου με δείκτη `idx`

`lst.sort()` $\longrightarrow$ επιτόπια ταξινόμηση των στοιχείων της `lst`

`lst.reverse()` $\longrightarrow$ επιτόπια αντιστροφή των στοιχείων

Μέθοδοι σε συμβολοσειρές

`s.strip()` $\longrightarrow$ αποκοπή λευκών χαρακτήρων

`s.count(substring)` $\longrightarrow$ αριθμός εμφανίσεων της συμβολοσειράς `substring`

`s.index(substring)` $\longrightarrow$ δείκτης πρώτης εμφάνισης της συμβολοσειράς `substring`. Μήνυμα λάθους αν το `substring` δεν υπάρχει στη συμβολοσειρά `s`.

`s.find(substring)` $\longrightarrow$ δείκτης πρώτης εμφάνισης της συμβολοσειράς `substring`, -1 αν δεν εμφανίζεται στη συμβολοσειρά `s`

`s.lower()` `s.upper()` `s.title()` `s.capitalize()`
Μετατροπή σε πεζά, κεφαλαία, πρώτου χαρακτήρα κάθε λέξης σε κεφαλαίο, πρώτου χαρακτήρα σε κεφαλαία, αντίστοιχα

Συναρτήσεις

`def` *όνομα*`(`*τυπικές παράμετροι*`):`
⟶ | *εντολές*
⟶ | `return val`

Η εντολή `return` επιστρέφει την `val` ως τιμή της συνάρτησης.
Αν δεν εμφανίζεται, η συνάρτηση επιστρέφει `None`.

```
def common(L, M):
  K = []
  for e in L:
    if e in M:
      K.append(e)
  return K
L = [9, 12, 4, 15];  M = [3, 8, 4, 9]
K = common(L, M)
```