Λίστες στην Python - Μέρος 1ο

Μιχάλης Δρακόπουλος

Απρίλιος 2020

1 ΛΙΣΤΕΣ - Μέρος Α

Είναι *αλληλουχίες* αντικειμένων. Τα αντικείμενα-*στοχεία* μιας λίστας είναι *διατεταγμένα* με βάση τη θέση τους από αριστερά προς τα δεξιά.

Παράδειγμα *άμεσης* δημιουργίας λίστας

[1]:
```
[82, 21, 34, 46, 39]
```

[1]:
```
[82, 21, 34, 46, 39]
```

[2]:
```
L = [82, 21, 34, 46, 39]
```

[3]:
```
L
```

[3]:
```
[82, 21, 34, 46, 39]
```

Αντιστοιχούν στα arrays που συναντάμε σε άλλες γλώσσες προγραμματισμού με τη διαφορά ότι τα στοιχεία μιας λίστας μπορεί να είναι *διαφορετικού τύπου*.

[4]:
```
L = [42, 'The Life of Brian', 3.14159]
L
```

[4]:
```
[42, 'The Life of Brian', 3.14159]
```

Το όνομα του τύπου λίστας στην Python είναι `list`

[5]:
```
type(L)
```

[5]:
```
list
```

Η *συνάρτηση* `list` που αντιστοιχεί στο όνομα του τύπου χρησιμοποείται για να *κατασκευάσει* λίστες, μετατρέποντας αντικείμενα άλλων τύπων δεδομένων σε λίστες. Τα αντικείμενα που θα μετατραπούν πρέπει να τηρούν το πρωτόκολλο επανάληψης, π.χ. `str`, `range` και όχι `int` ή `float`!

[6]:
```
list('abcdef')
```

[6]:
```
['a', 'b', 'c', 'd', 'e', 'f']
```

[7]:
```
list(range(1, 9, 3))
```

[7]:
```
[1, 4, 7]
```

[8]:
```
list(123)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-8-1c325c17b3d1> in <module> ----> 1 list(123)


TypeError: 'int' object is not iterable
```

1.1 Πράξεις με λίστες

Δημιουργούν *νέα* λίστα:

- με συνένωση (ο τελεστής +)

[9]:
```
M = [1, 2, 3]
M + [10, 20]
```

[9]:
```
[1, 2, 3, 10, 20]
```

Οι όροι του τελεστή + πρέπει να είναι λίστες!

[10]:
```
M = [1, 2, 3]
```

[11]:
```
M + 50
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-11-da7dd51c84c2> in <module> ----> 1 M + 50


TypeError: can only concatenate list (not "int") to list
```

[12]:
```
M + [50]
```

[12]:
```
[1, 2, 3, 50]
```

- με επανάληψη (ο τελεστής `*`)

[13]:
```
P = [True]*10
P
```

[13]:
```
[True, True, True, True, True, True, True, True, True, True]
```

1.2 Διαχείριση στοιχείων με δείκτες

Πλήθος στοιχείων - μήκος λίστας: η συνάρτηση `len`

[14]:
```
len([10, 20, 12, 14, 100, -1])
```

[14]:
```
6
```

Το πρώτο στοιχείο βρίσκεται στη θέση `0`

[15]:
```
L = list(range(10, 20))
L
```

[15]:
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

[16]:
```
L[0]
```

[16]:
```
10
```

Το τελευταίο στοιχείο βρίσκεται στη θέση `len(L) - 1`

[17]:
```
len(L)
```

[17]:
```
10
```

[18]:
```
L[9]
```

[18]:
```
19
```

Οι δείκτες μπορεί να είναι και αρνητικοί (μετράνε από το τέλος προς την αρχή):

[19]:
```
L[-1], L[-3]
```

[19]:
```
(19, 17)
```

Ενας αρνητικός δείκτης στην ουσία *προστίθεται* στο μήκος της λίστας και προκύπτει ο αντίστοιχός του θετικός δείκτης.

[20]:
```
L[-1], L[-2]
```

[20]:
```
(19, 18)
```

[21]:
```
L[len(L)-1], L[len(L)-2]
```

[21]:
```
(19, 18)
```

Δείκτες εκτός των ορίων `0` και `len(L)-1` δίνουν σφάλμα

[22]:
```
L[100]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-22-ce4c37e013d3> in <module> ----> 1 L[100]


IndexError: list index out of range
```

[23]:
```
L[-50]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-23-fceee65b55e6> in <module> ----> 1 L[-50]


IndexError: list index out of range
```

1.3 Τεμαχισμός λίστας

Με τον τελεστή `:` μπορούμε να αποκόψουμε υπο-λίστες μιας λίστας, όπως ακριβώς και στις συμβολοσειρές. Για μια λίστα `S`:

[25]:
```
S = list(range(10,20))
S
```

[25]:
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

- `S[I:J]` Υπο-λίστα από τον δείκτη `I` έως **και** τον δείκτη `J-1` (η θέση `J` δεν συμπεριλαμβάνεται στην υπό-λίστα)

[26]:
```
S[1:4]
```

[26]: `[11, 12, 13]`

[27]:
```
S[5:-1]
```

[27]: `[15, 16, 17, 18]`

- `S[I:J:K]` Υπο-λίστα από `I` έως `J-1` με βήμα `K`

[28]:
```
S[1:6:2]
```

[28]: `[11, 13, 15]`

[29]:
```
S[5:1:-1]
```

[29]: `[15, 14, 13, 12]`

- Παράλειψη του `I` ισοδυναμεί με `I = 0`

[30]:
```
S[:4]
```

[30]: `[10, 11, 12, 13]`

- Παράλειψη του `J` ισοδυναμεί με `J = len(S)`

[31]:
```
S[2:]
```

[31]: `[12, 13, 14, 15, 16, 17, 18, 19]`

[32]:
```
S[1::2]
```

[32]: `[11, 13, 15, 17, 19]`

- Παράλειψη `I` **και** `J`: ολόκληρη η λίστα

[33]:
```
S[:]
```

[33]: `[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]`

- Αντιστροφή όλων των στοιχείων, `K < 0`

[34]:
```
S[::-1]
```

[34]: `[19, 18, 17, 16, 15, 14, 13, 12, 11, 10]`

- Αν `I` **ή** `J` εκτός ορίων, παίρνουμε την μέγιστη δυνατή υπο-λίστα που προκύπτει. Δεν είναι σφάλμα!!!

[35]:
```
S[6:100]
```

[35]: `[16, 17, 18, 19]`

[36]:
```
S[-5:], S[len(S)-5:]
```

[36]:
```
([15, 16, 17, 18, 19], [15, 16, 17, 18, 19])
```

[37]:
```
S[-20:], S[len(S)-20:]
```

[37]:
```
([10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
 [10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

- Αν τα I, J, K δεν επιτρέπουν τον σχηματισμό υπο-λίστας παίρνουμε την *κενή* λίστα

[38]:
```
S[5:5], S[5:1], S[2:5:-1]
```

[38]:
```
([], [], [])
```

1.4 Λίστες με στοιχεία αλληλουχίες - Πίνακες

Οι λίστες μπορεί να περιέχουν ως στοιχεία δεδομένα που είναι και αυτά αλληλουχίες (π.χ. συμβολοσειρές ή άλλες λίστες)

[39]:
```
L = [42, 'The Meaning of Life', [21, 30, 34, 82, 92]]
```

Το μήκος της L είναι 3:

[40]:
```
len(L)
```

[40]:
```
3
```

Περιέχει 3 στοιχεία:

[41]:
```
L[0]
```

[41]:
```
42
```

[42]:
```
L[1]
```

[42]:
```
'The Meaning of Life'
```

[43]:
```
L[2]
```

[43]:
```
[21, 30, 34, 82, 92]
```

Καθένα από αυτά τα στοιχεία μπορούμε να το διαχειριστούμε σύμφωνα με τον τύπο του:

- να βρούμε το μήκος του

[44]:
```
len(L[1]), len(L[2])
```

[44]:
```
(19, 5)
```

- να επιλέξουμε τα στοιχεία του

[45]:
```
L
```

[45]:
```
[42, 'The Meaning of Life', [21, 30, 34, 82, 92]]
```

[46]:
```
L[1][4]
```

[46]:
```
'M'
```

[47]:
```
L[2][0]
```

[47]:
```
21
```

- να το ”τεμαχίσουμε”

[48]:
```
L[2][1:]
```

[48]:
```
[30, 34, 82, 92]
```

[49]:
```
L[1][::-1]
```

[49]:
```
'efiL fo gninaeM ehT'
```

Αν δεν χρησιμοποιήσουμε κάποια εξειδικευμένη βιβλιοθήκη της Python, μπορούμε με αυτόν τον τρόπο να δουλέψουμε με *πίνακες* 2 διαστάσεων

[50]:
```
M = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
```

[51]:
```
M
```

[51]:
```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[52]:
```
M[1]
```

[52]:
```
[4, 5, 6]
```

[53]:
```
M[2][1]
```

[53]:
```
8
```

1.5 Τροποποίηση στοιχείων λίστας

Οι λίστες σε αντίθεση με τις συμβολοσειρές είναι *μεταβλητοί* τύποι: τα στοιχεία τους μπορούν να μεταβληθούν

[54]:
```
L = list(range(10,20))
L
```

[54]:
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

[55]:
```
L[0] = -99
L[1] = 'spam'
L
```

[55]:
```
[-99, 'spam', 12, 13, 14, 15, 16, 17, 18, 19]
```

Εκτός από εκχώρηση σε κάποια θέση (δείκτη), μπορούμε να αντικαταστήσουμε "τεμάχια" μιας λίστας με άλλα αντικείμενα. Τα αντικείμενα που θα εκχωρηθούν πρέπει να τηρούν το *πρωτόκολλο επανάληψης* (π.χ. `str`, `list`, `range`)

- αντικατάσταση

[56]:
```
L[2:4] = [-1000, 1000]
L
```

[56]:
```
[-99, 'spam', -1000, 1000, 14, 15, 16, 17, 18, 19]
```

[57]:
```
L[3:5] = [42]*4
L
```

[57]:
```
[-99, 'spam', -1000, 42, 42, 42, 42, 15, 16, 17, 18, 19]
```

- διαγραφή

[58]:
```
L[1:2] = []
L
```

[58]:
```
[-99, -1000, 42, 42, 42, 42, 15, 16, 17, 18, 19]
```

[59]:
```
L[0] = []
L
```

[59]:
```
[[], -1000, 42, 42, 42, 42, 15, 16, 17, 18, 19]
```

[60]:
```
L[0:1] = []
L
```

[60]:
```
[-1000, 42, 42, 42, 42, 15, 16, 17, 18, 19]
```

[61]:
```
L[1:5] = []
L
```

[61]: `[-1000, 15, 16, 17, 18, 19]`

- παρεμβολή στοιχείων

[62]:
```
L
```

[62]: `[-1000, 15, 16, 17, 18, 19]`

[63]:
```
L[2:2] = [3, 1, 4, 1, 5]
L
```

[63]: `[-1000, 15, 3, 1, 4, 1, 5, 16, 17, 18, 19]`

Ο μηχανισμός εκχώρησης σε τεμάχια απαρτίζεται από τα εξής βήματα 1. Το αντικείμενο δεξιά από το = αποθηκεύεται προσωρινά στη μνήμη 2. Το τεμάχιο αριστερά από το = διαγράφεται 3. Το αποθηκευμένο αντικείμενο εκχωρείται στη θέση του διαγραμμένου τεμαχίου.

[64]:
```
L
```

[64]: `[-1000, 15, 3, 1, 4, 1, 5, 16, 17, 18, 19]`

[65]:
```
L[1:7] = L[2:5]
```

[66]:
```
L
```

[66]: `[-1000, 3, 1, 4, 16, 17, 18, 19]`