Πλειάδες (Tuples)

Μιχάλης Δρακόπουλος

Απρίλιος 2020

ΠΛΕΙΑΔΕΣ

Οι *πλειάδες (tuples)* είναι *αλληλουχίες* αντικειμένων *διατεταγμένων* ως προς τη θέση τους από αριστερά προς τα δεξιά, όπως και οι λίστες.

Το όνομα του τύπου πλειάδας στην Python είναι `tuple`.

Δημιουργία

- άμεσα

[1]:
```
(21, 34, 82, 30, 92)
```

[1]:
```
(21, 34, 82, 30, 92)
```

Τα στοιχεία τους μπορεί να είναι *διαφορετικού* τύπου:

[2]:
```
T = (2.71, 'spam', 42, [10, 3, 4])
T
```

[2]:
```
(2.71, 'spam', 42, [10, 3, 4])
```

Οπου δεν υπάρχει σύγχυση, οι πλειάδες μπορούν να δημιουργηθούν και *χωρίς παρενθέσεις*:

[3]:
```
P = 5, 11.2
type(P)
```

[3]:
```
tuple
```

Η *κενή* πλειάδα:

[4]:
```
Empty_tuple = ()
type(Empty_tuple)
```

[4]:
```
tuple
```

Πλειάδα με ένα μόνο στοιχείο:

[5]:
```
s = (10)
type(s)
```

[5]:
```
int
```

Το κόμμα *απαραίτητο*:

[6]:
```
One_element_tuple = (10,)
type(One_element_tuple)
```

[6]:
```
tuple
```

- με μετατροπή άλλων τύπων (που τηρούν το πρωτόκολλο επανάληψης) με τη συνάρτηση `tuple`

[7]:
```
tuple('abcdef')
```

[7]:
```
('a', 'b', 'c', 'd', 'e', 'f')
```

Διαχείριση με δείκτες - Τεμάχισμός

Οπως και στους άλλους τύπους αλληλουχίας: `str`, `list`

[8]:
```
T
```

[8]:
```
(2.71, 'spam', 42, [10, 3, 4])
```

[9]:
```
T[0], T[1][2], T[-1][1]
```

[9]:
```
(2.71, 'a', 3)
```

[10]:
```
T[1:-1], T[:]
```

[10]:
```
(('spam', 42), (2.71, 'spam', 42, [10, 3, 4]))
```

Σε αντίθεση με τις λίστες είναι **αμετάβλητοι** τύποι: τα στοιχεία τους δεν μεταβάλλονται ως προς την τιμή ή το πλήθος τους.

[11]:
```
T[0] = 3.14151
```

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

TypeError                                 Traceback (most recent call last)

<ipython-input-11-993010621326> in <module>() ----> 1 T[0] = 3.14151


TypeError: 'tuple' object does not support item assignment
```

Πράξεις

Δημιουργούν πάντα **νέα** αντικείμενα!

- Επανάληψη με τον τελεστή *

[12]:
```
('abc', 4)*3
```

[12]:
```
('abc', 4, 'abc', 4, 'abc', 4)
```

- Συνένωση με τον τελεστή +

[13]:
```
T1 = (10, 20, 30)
T1 + (-99, 99)
```

[13]:
```
(10, 20, 30, -99, 99)
```

Μπορούμε να **"αλλάξουμε"** στοιχεία μιας πλειάδας, δημιουργώντας με τεμαχισμό μια **νέα** πλειάδα:

[14]:
```
T
```

[14]:
```
(2.71, 'spam', 42, [10, 3, 4])
```

(ας κρατήσουμε ένα αντίγραφο της T)

[15]:
```
T1 = T
T1
```

[15]:
```
(2.71, 'spam', 42, [10, 3, 4])
```

[16]:
```
T1 is T
```

[16]:
```
True
```

– αλλαγή του spam σε ham (με "επεισοδιακό" τρόπο ;-)

[17]:
```
T = T[0] + 'ham' + T[2:]
```

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

TypeError                                 Traceback (most recent call last)

<ipython-input-17-a5121416c4a0> in <module>() ----> 1 T = T[0] + 'ham' + T[2:]


TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

[18]:
```
T = T[0:1] + 'ham' + T[2:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-727d7afcb1c5> in <module>() ----> 1 T = T[0:1] + 'ham' + T[2:]

TypeError: can only concatenate tuple (not "str") to tuple
```

[19]:
```
T = T[0:1] + ('ham') + T[2:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-9bd2f56f75e4> in <module>()----> 1 T = T[0:1] + ('ham') + T[2:]

TypeError: can only concatenate tuple (not "str") to tuple
```

[20]:
```
T = T[0:1] + ('ham',) + T[2:]
T
```

[20]:
```
(2.71, 'ham', 42, [10, 3, 4])
```

Δημιουργήθηκε νέο αντικείμενο. Το αντίγραφο στο `T1` "δείχνει" στο αρχικό αντικείμενο:

[21]:
```
T1 is T
```

[21]:
```
False
```

[22]:
```
T1
```

[22]:
```
(2.71, 'spam', 42, [10, 3, 4])
```

- Ελεγχος μέλους: ο τελεστής `in`

[23]:
```
'ham' in T
```

[23]:
```
True
```

[24]:
```
'spam' in T
```

[24]:
```
False
```

Συναρτήσεις για πλειάδες

Ιδιες με εκείνες για λίστες (με τους ίδιους περιορισμούς):

[25]:
```
P = (2000, 21, 34, 82, 30, 92)
P
```

[25]:
```
(2000, 21, 34, 82, 30, 92)
```

[26]:
```
len(P), sum(P), min(P), max(P),
```

[26]:
```
(6, 2259, 21, 2000)
```

[27]:
```
L = sorted(P)
L
```

[27]:
```
[21, 30, 34, 82, 92, 2000]
```

Η `sorted` επιστρέφει *λίστα*. Αν θέλουμε μια (νέα) διατεταγμένη πλειάδα:

[28]:
```
SP = tuple(sorted(P))
SP
```

[28]:
```
(21, 30, 34, 82, 92, 2000)
```

Μέθοδοι για πλειάδες

[29]:
```
dir(tuple)
```

[29]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
```

```
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

Μας ενδιαφέρουν (προς το παρόν) οι εξής 2:

[30]:
```
dir(tuple)[-2:]
```

[30]:
```
['count', 'index']
```

Παραδείγματα χρήσης

[31]:
```
(15, 'abc', 1, 'abc', 4).count('abc')
```

[31]:
```
2
```

[32]:
```
T = tuple(range(10, 20))
T
```

[32]:
```
(10, 11, 12, 13, 14, 15, 16, 17, 18, 19)
```

[33]:
```
T.index(13)
```

[33]:
```
3
```

Επαναληπτική επεξεργασία στοιχείων πλειάδας

Ισχύουν ό,τι και για τις λίστες

- χωρίς δείκτες

[34]:
```
p = 1
for x in T:
    p *= x
print(p)
```

```
335221286400
```

- με δείκτες

[35]:
```
for i in range(len(T)):
    print(T[i], end = ', ')
```

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

Γιατί πλειάδες

- Εχουν μαθηματική υπόσταση (π.χ. διατεταγμένα ζεύγη, συντεταγμένες)
- Παρέχουν ασφάλεια από αλλαγές μέσω κοινών αναφορών
- Χρησιμοποιούνται ως στοιχεία άλλων τύπων δεδομένων (όπως θα δούμε) που προϋποθέτουν *immutability* (σύνολα, λεξικά)
- Κώδικας με πλειάδες μπορεί να έχει μεγαλύτερη ταχύτητα εκτέλεσης απ' ότι αντίστοιχος κώδικας με λίστες.