Πλειάδες

Κεφάλαιο 10

Python για Όλους

www.py4e.com

Οι Πλειάδες είναι σαν τις Λίστες

Οι Πλειάδες είναι άλλο ένα είδος διατεταγμένης ακολουθίας που λειτουργεί σαν μια λίστα - έχουν στοιχεία που αντιστοιχούν σε έναν αύξων αριθμό, ξεκινώντας από το 0

```
>>> x = ('Glenn', 'Sally', 'Joseph')
>>> print(x[2])
Joseph
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for iter in y:
...     print(iter)
...
1
9
2
>>>
```

αλλά ... οι Πλειάδες είναι «αμετάβλητες»

Σε αντίθεση με μια λίστα, αφής στιγμής δημιουργήσετε μια πλειάδα, δεν μπορείτε να αλλάξετε τα περιεχόμενά της - παρόμοια με μια συμβολοσειρά

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>>[9, 8, 6]
>>>
```

```
>>> y = 'ABC'
>>> y[2] = 'D'
Traceback:'str'
object does
not support item
Assignment
>>>
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback:'tuple'
object does
not support item
Assignment
>>>
```

Πράγματα που δεν κάνουμε με Πλειάδες

```
>>> x = (3, 2, 1)
>>> x.sort()
Traceback:
AttributeError: 'tuple' object has no attribute 'sort'
>>> x.append(5)
Traceback:
AttributeError: 'tuple' object has no attribute 'append'
>>> x.reverse()
Traceback:
AttributeError: 'tuple' object has no attribute 'reverse'
>>>
```

Μια ιστορία Δύο Ακολουθιών

```
>>> l = list()
>>> dir(l)
['append', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']

>>> t = tuple()
>>> dir(t)
['count', 'index']
```

Οι Πλειάδες είναι Πιο Αποτελεσματικές

- Δεδομένου ότι η Python δεν χρειάζεται να κατασκευάσει δομές πλειάδων, που να είναι τροποποιήσιμες, είναι απλούστερες ως προς τη χρήση και πιο αποτελεσματικές ως προς τις επιδόσεις της μνήμης από τις λίστες
- Έτσι, στο πρόγραμμά μας όταν δημιουργούμε «προσωρινές μεταβλητές» προτιμούμε πλειάδες από λίστες

Πλειάδες και Εκχώρηση

- Μπορούμε να βάλουμε και μια πλειάδα στην αριστερή πλευρά μιας εντολής εκχώρησης
- Μπορούμε ακόμη και να παραλείψουμε τις παρενθέσεις

```
>>> (x, y) = (4, 'Φώτης')
>>> print(y)
Φώτης
>>> (a, b) = (99, 98)
>>> print(a)
99
```

Πλειάδες και Λεξικά

Η μέθοδος items()
στα λεξικά
επιστρέφει μια λίστα
πλειάδων της
μορφής (κλειδί, τιμή)

```
>>> d = dict()
>>> d['csev'] = 2
>>> d['cwen'] = 4
>>> for (k,v) in d.items():
...     print(k, v)
...
csev 2
cwen 4
>>> tups = d.items()
>>> print(tups)
dict_items([('csev', 2), ('cwen', 4)])
```

Οι Πλειάδες είναι Συγκρίσιμες

Οι συγκριτικοί τελεστές λειτουργούν στις πλειάδες και σε άλλες ακολουθίες. Εάν το πρώτο στοιχείο είναι ίσο, η Python συνεχίζει στο επόμενο στοιχείο και ούτω καθεξής, μέχρι να βρει στοιχεία που διαφέρουν.

```
>>> (0, 1, 2) < (5, 1, 2)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
>>> ( 'Γιώργος', 'Σάββας' ) < ('Γιώργος', 'Σοφία')
True
>>> ( 'Γιώργος', 'Σάββας') > ('Αγνή', 'Σοφία')
True
```

Ταξινομόντας Λίστες Πλειάδων

- Μπορούμε να εκμεταλλευτούμε τη δυνατότητα ταξινόμησης μιας λίστας πλειάδων για να λάβουμε μια ταξινομημένη έκδοση ενός λεξικού
- Αρχικά ταξινομούμε το λεξικό κατά το κλειδί χρησιμοποιώντας τη μέθοδο items() και έπειτα τη συνάρτηση sorted()

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22), ('b', 1)])
>>> sorted(d.items())
[('a', 10), ('b', 1), ('c', 22)]
```

Χρησιμοποιώντας την sorted()

Μπορούμε να το κάνουμε ακόμα πιο άμεσα χρησιμοποιώντας την ενσωματωμένη συνάρτηση sorted που δέχεται μια ακολουθία ως παράμετρο και επιστρέφει μια ταξινομημένη ακολουθία

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = sorted(d.items())
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> for k, v in sorted(d.items()):
...     print(k, v)
...
a 10
b 1
c 22
```

Ταξινόμηση κατά Τιμή Αντί για Κλειδί

- Αν μπορούσαμε να δημιουργήσουμε μια λίστα πλειάδων της μορφής (τιμή, κλειδί) θα μπορούσαμε να ταξινομήσουμε κατά τιμή
- Το επιτυγχάνουμε αυτό με έναν βρόχο for που δημιουργεί μια λίστα πλειάδων

```
>>> c = {'a':10, 'b':1, 'c':22}
>>> tmp = list()
>>> for k, v in c.items() :
...     tmp.append( (v, k) )
...
>>> print(tmp)
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> tmp = sorted(tmp, reverse=True)
>>> print(tmp)
[(22, 'c'), (10, 'a'), (1, 'b')]
```

Οι 10 ποιο συχνές λέξεις

```
fhand = open('romeo.txt’)
πλήθος = {}
for γραμμή in fhand:
    λέξεις = γραμμή.split()
    for λέξη in lexeis :
        πλήθος[λέξη] = πλήθος.get(λέξη, 0 ) + 1

lst = []
for κλειδί, τιμή in πλήθος.items():
    νεαπλ = (τιμή, κλειδί)
    lst.append(νεαπλ)

lst = sorted(lst, reverse=True)

for τιμή, κλειδί in lst[:10] :
    print(κλειδί, τιμή)
```

Ακόμα και πιο Σύντομη Έκδοση

```
>>> c = {'a':10, 'b':1, 'c':22}

>>> print( sorted( [ (v,k) for k,v in c.items() ] ) )

[(1, 'b'), (10, 'a'), (22, 'c')]
```

Η κατανόηση της λίστας δημιουργεί μια δυναμική λίστα. Σε αυτή την περίπτωση, φτιάχνουμε μια λίστα με αντεστραμμένες τις πλειάδες και στη συνέχεια την ταξινομούμε.

http://wiki.python.org/moin/HowTo/Sorting

Σύνοψη

- Σύνταξη Πλειάδων
- Αμετάβλητη
- Συγκρισιμότητα
- Ταξινόμηση

- Πλειάδες σε εντολές εκχώρησης
- Ταξινόμηση λεξικών είτε κατά κλειδί είτε κατά τιμή

Ευχαριστίες / Συνεισφορές

Αρχική ανάπτυξη : Charles Severance, University of Michigan School of Information

Απόδοση στα Ελληνικά: Κιουρτίδου Δ. Κωνσταντία

… Εισαγάγετε νέους Μεταφραστές και άτομα που έχουν συνεισφέρει εδώ

Συνέχεια…