Σημειώσεις

Προηγούμενο Επόμενο

Πλειάδες (tuples)

Ομαδοποίηση στοιχείων

Μία σειρά στοιχείων (αριθμών, συμβολοσειρών κλπ) μπορούν να ομαδοποιηθούν κάτω από ένα όνομα (μίας μεταβλητής) σε μία δομή η οποία ονομάζεται *πλειάδα*. Τα στοιχεία της πλειάδας γράφονται μέσα σε παρένθεση και χωρίζονται με κόμμα. Παραδείγματα:

```
>>> info = ('Nick',22)
>>> cities = ('Agios','Heraklion','Rethymno','Chania')
```

Ένας ακόμα τρόπος να ορίσουμε μία πλειάδα είναι να δώσουμε πολλές τιμές σε μία μεταβλητή, οπότε αυτή είναι μία πλειάδα:

```
>>> t1 = 1,2,'three'
>>> print(t1)
(1, 2, 'three')
>>> type(t1)
<class 'tuple'>
```

Δεικτοδότηση: Μπορούμε να ανακτήσουμε ένα από τα στοιχεία της πλειάδας χρησιμοποιώντας τον δείκτη του. Η δεικτοδότηση είναι ακριβώς όπως στις συμβολοσειρές και στις λίστες.

```
>>> t1[1]
2
>>> t1[-1]
'three'
```

Τεμαχισμός: Μπορούμε να ανακτήσουμε ένα τμήμα της πλειάδας χρησιμοποιώντας ένα εύρος δεικτών. Το αποτέλεσμα είναι μία νέα πλειάδα. (Η δεικτοδότηση είναι ακριβώς

όπως στις συμβολοσειρές και στις λίστες.)

```
>>> t1[0:2]
(1, 2)
```

Βασικές λειτουργίες

Μπορούμε να χρησιμοποιήσουμε τα σύμβολα της πρόσθεσης (+) για να ενώσουμε δύο πλειάδες:

```
>>> t1 = (1,2,'three')
>>> t2 = ('a','b')
>>> t1 + t2
(1, 2, 'three', 'a', 'b')
```

καθώς επίσης και του πολλαπλασιασμού (*) (για να πάρουμε μία σειρά αντιγράφων μίας πλειάδας):

```
>>> 2*t1
(1, 2, 'three', 1, 2, 'three')
```

Μία πλειάδα μπορεί να έχει ως στοιχεία πλειάδες.

```
>>> t  = (t1,t2)
>>> print(t)
((1, 2, 'three'), ('a', 'b'))
```

Ειδικές περιπτώσεις πλειάδων

Έχουμε την κενή πλειάδα.

```
>>> t = ()
```

Έχουμε την πλειάδα με ένα στοιχείο (προσέξτε ότι το κόμμα στο τέλος της πλειάδας είναι αναγκαίο!).

```
>>> t = (1,)
>>> 3*t
(1, 1, 1)
```

Παράδειγμα. Βρείτε τους διαιρέτες ενός ακεραίου.

```
def findDivisors(n):
    divisors = ()
    for i in range(1,n+1):
        if n%i == 0: divisors = divisors + (i,)
    return divisors

x = 20
result = findDivisors(x)
print(result)
```

Μη-μεταλλαξιμότητα

Οι πλειάδες δεν είναι μεταλλάξιμες: Δεν μπορούμε να αλλάξουμε ένα (ή ορισμένα) από τα στοιχεία μίας πλειάδας.

Θυμηθείτε ότι και οι συμβολοσειρές είναι μη-μεταλλάξιμος τύπος. Αντιθέτως, οι λίστες είναι μεταλλάξιμος τύπος, δηλαδή, μπορούμε να προσθέσουμε, αφαιρέσουμε ή να αλλάξουμε στοιχεία μίας λίστας.

Δεν μπορούμε να αλλάξουμε την τιμή στοιχείου πλειάδας, αλλά μπορούμε να σβήσουμε μία πλειάδα με την εντολή `del`.

```
>>> t = (1,2,3)
>>> print(t)
(1, 2, 3)
>>> del t
>>> print(t)
Traceback (most recent call last):
  File "", line 1, in
NameError: name 't' is not defined
```

Πολλαπλές αναθέσεις τιμών

Μπορούμε να μεταφέρουμε τις τιμές μίας πλειάδας μήκους *n* σε ισάριθμες μεταβλητές (αυτό ονομάζεται unpacking).

```
>>> t = (1, 4, 8)
>>> x,y,z = t
>>> print(x)
>>> 1
>>> print(x,y,z)
1 4 8
```

Μπορούμε να δώσουμε τιμές σε σειρά μεταβλητών με μία εντολή (για το σκοπό αυτόν χρησιμοποιούμε με άμεσο ή έμεσο τρόπο πλειάδες).

```
>>> x,y,z = (1, 4, 8)
>>> print(x)
>>> 1
>>> print(x,y,z)
1 4 8
```

Συναρτήσεις οι οποίες επιστρέφουν πλειάδες

Στην περίπτωση που μίας συνάρτηση επιστρέφει περισσότερα από ένα αποτελέσματα αυτά επιστρέφονται ως πλειάδα (μπαίνουν στην εντολή `return` ως πλειάδα).

Παράδειγμα. Σώμα μάζας m εκτοξεύεται προς τα επάνω με ταχύτητα v_0. Βρείτε την ταχύτητα και θέση του για διαδοχικούς χρόνους.

```
def fallingParticle(t):
    global v0,g
    velocity = v0 - g*t
    position = v0*t - 0.5*g*t**2
    return velocity,position

v0 = 20.0
g = 10.0
time = 0.0
while time < 2.0:
    v,y = fallingParticle(time)
```

```
print(time,v,y)
time += 0.2
```

Συναρτήσεις για πλειάδες

- `len(t1)`: δίνει τον αριθμό στοιχείων πλειάδας.
- `max(t1), min(t1)`: δίνουν το μέγιστο και ελάχιστο στοιχείο πλειάδας (όταν αυτά είναι συγκρίσιμα μεταξύ τους).
- `tuple`: μετατρέπει σε πλειάδα (π.χ., `L=[1,2,3]; t=tuple(L)`).

Άσκηση. Κατασκευάστε μία πλειάδα η οποία θα περιέχει την ακολουθία αριθμών Fibonacci $F_i, i = 1, \ldots, n$ για έναν θετικό ακέραιο n (την οποία θα εισάγετε όταν εκτελείται το πρόγραμμα).

Μελέτη

Βιβλιογραφία

1. J.V. Guttag, *Υπολογισμοί και προγραμματισμός με την python* (Παράγραφος 5.1).
2. Δημήτριος Καρολίδης, *Μαθαίνετε εύκολα python* (Παράγραφος 4.3) (Εκδόσεις Καρολίδη, 2016).
3. Κ. Μαγκούτης, Χ. Νικολάου, *Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με Python*, (Αποθετήριο "Κάλλιπος", 2016) - Κεφάλαιο 6. Συμβολοσειρές, λίστες, πλειάδες, λεξικά.

Προηγούμενο Επόμενο