Μορφοποίηση εξόδου

Έχουμε δει ότι όταν τυπώνουμε μεταβλητές, π.χ., `print(s,a,i)` δεν έχουμε ιδιαίτερο έλεγχο στη μορφή με την οποία θα τυπωθούν οι τιμές των μεταβλητών. Με την εντολή `print` οι τιμές των μεταβλητών τυπώνονται σε πεδία τόσης έκτασης όση ακριβώς χρειάζεται για να χωρέσουν.

Πολύ συχνά όμως η μορφή αυτή της εκτύπωσης δεν είναι αυτή που επιθυμούμε. Για παράδειγμα, αν `a=3.000000000000001` τότε είναι πιθανό ότι θα επιθυμούσαμε να τυπωθεί το *3.0* και όχι τα παραπάνω 16 ψηφία. Υπάρχουν και πολλές άλλες περιπτώσεις που θα θέλαμε να έχουμε έλεγχο στην μορφή της εκτύπωσης, π.χ., θα θέλαμε να υπάρχουν ορισμένος αριθμός κενών μεταξύ δύο αριθμών, θα θέλαμε να τυπώνονται νούμερα ή λέξεις σε διαδοχικές σειρές κατά ευθυγραμμισμένο τρόπο κλπ.

Άσκηση κατανόησης. Δοκιμάστε να τυπώσετε διαδοχικούς αριθμούς: `for i in range(10): print(i, i*0.1)`. Θα παρατηρήσετε ότι είναι απαραίτητο να βελτιώσουμε την μορφή της εκτύπωσης.

Η python δίνει την δυνατότητα μορφοποίησης της εκτύπωσης αριθμών (ή οτιδήποτε άλλης μεταβλητής). Χειρίζεται τη μορφοποίηση μετατρέποντάς τις τιμές των μεταβλητών πρώτα σε συμβολοσειρές και φέρνοντας τις συμβολοσειρές στην μορφή ακριβώς που θα θέλαμε θα δούμε το εκτυπωμένο αποτέλεσμα.

Η συμβολοσειρά την οποία θέλουμε τελικά να τυπώσουμε διαμορφώνεται με την μέθοδο `format`. Για παράδειγμα, θα μπορούσαμε να βάλουμε έναν ακέραιο *i=5* ανάμεσα σε δύο κενά ως εξής:

```
>>> i = 5
>>> " {} ".format(i)
' 5 '
```

Η συμβολοσειρά `' {} '` περιέχει δύο κενά στα άκρα, ενώ ενδιάμεσα υπάρχει θέση για την τιμή μίας μεταβλητής. Με την μέθοδο `format` εισήχθει η τιμή της μεταβλητής και έγινε η τελική διαμόρφωση.

Εκτύπωση ακεραίων. Μπορούμε να τυπώσουμε έναν ακέραιο σε πεδίο καθορισμένο μήκους. Δηλώνουμε ότι αναμένουμε εκτύπωση ακεραίου με τον διακριτικό χαρακτήρα `d`. Θα τυπώσουμε διαδοχικά ακεραίους σε πεδίο πέντε θέσεων (με ευθυγράμμιση στα δεξιά του πεδίου):

```
for i in range(0,20,4):
        s = '{:5d}'.format(i)
        print(s)
```

ή, σε πιο συμπαγή μορφή:

```
for i in range(0,20,4):
        print('{:5d}'.format(i))
```

Παράδειγμα. Το παρακάτω πρόγραμμα τυπώνει δύο ακεραίους σε κάθε γραμμή σε πεδία πλάτους 5 και 7 χαρακτήρων αντίστοιχα. Μεταξύ τους αφήνονται δύο κενές θέσεις.

```
for i in range(0,20,4):
        print('{:5d}  {:7d}'.format(i,i**2))
```

Παράδειγμα. Δείτε επίσης το αποτέλεσμα του προγράμματος:

```
for i in range(10):
        print('{:5d} - {:7d}'.format(i,i**2))
```

Εκτύπωση floats. Δηλώνουμε ότι αναμένουμε εκτύπωση float με τον διακριτικό χαρακτήρα `f`. Επίσης, χρησιμοποιούμε τον διακριτικό χαρακτήρα `e` για floats με επιστημονικό συμβολισμό (scientific notation).

Παράδειγμα. Θα τυπώσουμε έναν float διατηρώντας έξι δεκαδικά ψηφία:

```
>>> '{:f}'.format(3.141592653589793)
```

```
'3.141593'
```

Παράδειγμα. Θα τυπώσουμε έναν float σε πεδίο έξι θέσεων διατηρώντας δύο δεκαδικά ψηφία:

```
>>> '{:6.2f}'.format(3.141592653589793)
'  3.14'
```

Για την εκτύπωση ενός float χρειαζόμαστε αρκετές περισσότερες θέσεις από το πλήθος των δεκαδικών του ψηφίων. Χρειαζόμαστε τουλάχιστον μία θέση για το ακέραιο μέρος, μία θέση για την υποδιαστολή και πιθανόν μία θέση για το πρόσημο (εφόσον πρόκειται για αρνητικό αριθμό).

Παράδειγμα. Μπορούμε να προκαλέσουμε την εκτύπωση του προσήμου ενός αριθμού.

```
>>> a = 42.5
>>> '{:+6.2f}'.format(a)
'+42.50'
>>> a = -42.5
>>> '{:+6.2f}'.format(a)
'-42.50'
```

Παράδειγμα. Δηλώνουμε την εκτύπωση float σε scientific notiation με τον διακριτικό χαρακτήρα `e`. Θα τυπώσουμε την ταχύτητα του φωτός.

```
>>> c = 299792458
>>> '{:11.4e}'.format(c)
' 2.9979e+08'
>>> '{:11.4e} m/sec'.format(c)
' 2.9979e+08 m/sec'
```

Εκτύπωση strings. Δηλώνουμε την εκτύπωση string με τον διακριτικό χαρακτήρα `s`. Μπορούμε όμως και να παραλήψουμε τον διακριτικό χαρακτήρα.

Μπορούμε να τυπώσουμε μία συμβολοσειρά σε πεδίο μεγαλύτερο από το μήκος της, ή να την περικόψουμε κατά την εκτύπωση.

Παράδειγμα. Θα τυπώσουμε συμβολοσειρά με αριστερή (default) ή δεξιά ευθυγράμμιση ή στην μέση του πεδίου.

```
>>> '{:15s}'.format('abrakatabra')
'abrakatabra    '
>>> '{:15}'.format('abrakatabra')
'abrakatabra    '
>>> '{:>15}'.format('abrakatabra')
'    abrakatabra'
>>> '{:^15}'.format('abrakatabra')
'  abrakatabra  '
```

Παράδειγμα. Μπορούμε να επιλέξουμε έναν χαρακτήρα ο οποίος θα γεμίσει το επιπλέον πεδίο.

```
>>> '{0:-<6}{1:-<6}{2:-<6}'.format('abra','kat','abra')
'abra--kat---abra--'
```

Παράδειγμα. Μπορούμε να περικόψουμε μία υπερβολικά μεγάλη συμβολοσειρά.

```
>>> '{:.7}'.format('abrakatabra')
'abrakat'
```

Παράδειγμα. Μπροστά από το : μπορούμε να βάλουμε έναν αριθμό ο οποίος αντιστοιχεί στην θέση της μεταβλητής την οποία θέλουμε να τυπώσουμε.

```
>>> a = 1
>>> b = 2
>>> '{0:5d} {1:5d}'.format(a,b)
'    1     2'
>>> '{1:5d} {0:5d}'.format(a,b)
'    2     1'
```

```
>>> '{0:}{1:}{0:}'.format('abra','kat')
'abrakatabra'
```

Περίληψη. Η γενική μορφή δήλωσης για την εκτύπωση μίας μεταβλητής είναι

```
{[position]:[flags][width][.precision][type]}
```

Άσκηση. Γράψτε μία συνάρτηση η οποία θα τυπώσει τα στοιχεία λίστας σε διαδοχικές γραμμές. Υποθέτουμε ότι όλα τα στοιχεία της λίστας είναι του ίδιου τύπου.

Άσκηση. Τυπώστε μεταβλητή float δίνονται το εύρος πεδίου και τον αριθμό δεκαδικών ψηφίων μέσα στην `format`.

```
'{:{width}.{prec}f}'.format(2.7182, width=5, prec=2)
```

Βελτιώστε την συνάρτηση που φτιάξατε στην προηγούμενη άσκηση χρησιμοποιώντας την μέθοδο που είδατε σε αυτή την άσκηση.

Μελέτη

Βιβλιογραφία

1. PyFormat.
2. Python course, Formatted Output (για python 2).
3. J.V. Guttag, *Υπολογισμοί και προγραμματισμός με την python*.
4. Δημήτριος Καρολίδης, *Μαθαίνετε εύκολα python* (Εκδόσεις Καρολίδη, 2016).
5. Κ. Μαγκούτης, Χ. Νικολάου, *Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με Python*, (Αποθετήριο "Κάλλιπος", 2016).

Προηγούμενο Επόμενο