Συμβολοσειρές (Strings)

Μπορούμε να γράψουμε ακολουθίες χαρακτήρων, τα όρια των οποίων καθορίζονται από απλές (') ή διπλές (") αποστρόφους. Αυτές θα τις ονομάζουμε συμβολοσειρές (strings). Ορίζουμε μεταβλητές οι οποίες περιέχουν συμβολοσειρές.

```
>>> a = 'Nick'
>>> b = 'Papadakis'
```

Παραδείγματα. Μπορούμε να εφαρμόσουμε σε συμβολοσειρές τους τελεστές πρόσθεσης (+) και πολλπλασιασμού (*).

```
>>> onoma = a + b
>>> print(onoma)
NickPapadakis
>>> onoma = a + " " + b
>>> print(onoma)
Nick Papadakis
>>> 2*a
'NickNick'
>>> year_in = 2016
>>> print(onoma,"came to Crete in",year_in)
Nick Papadakis came to Crete in 2016
```

Μπορούμε να βρούμε το μήκος της συμβολοσειράς.

```
>>> len(onoma)
14
```

Κάθε χαρακτήρας σε μία συμβολοσειρά έχει αριθμό θέσης (δείκτη, index). Η αρίθμηση ξεκινάει από το 0 και ο τελευταίος χαρακτήρας μιας συμβολοσειράς με μήκος n έχει δείκτη n-1.

```
>>> len(onoma)
14
>>> onoma[0]
'N'
>>> onoma[13]
's'
>>> onoma[1]
'i'
```

Τεμαχισμός (Slicing): επιλέγουμε ένα τμήμα της συμβολοσειράς. Ο συμβολισμός onoma[start:end] αναφέρεται σε ένα τμήμα της αρχικής συμβολοσειράς, ειδικότερα στους χαρακτήρες της συμβολοσειράς onoma που βρίσκονται στις θέσεις start, start+1,..., end-1.

```
>>> onoma[0:4]
'Nick'
>>> onoma[1:4]
'ick'
```

Η θέση κάθε χαρακτήρα θέσης μπορεί επίσης να καθορισθεί μετρώντας από το τέλος της συμβολοσειράς (string). Αν το μήκος της συμβολοσειράς είναι n τότε η αρίθμηση ξεκινάει από το -n. Η συνολική συμβολοσειρά μπορεί να αναφερθεί ως s[-n:].

```
>>> s = 'Nick'
>>> s[-4:-1]
'Nic'
>>> s[-4:]
'Nick'
```

Μπορούμε να γράψουμε [start:end:step] για να αναφερθούμε στις θέσεις μεταξύ start και end-1 με βήμα step.

```
>>> s1 = 'Take'
>>> s2 = 'Make'
>>> s3 = 'Fake'
```

```
>>> s = s1 + s2 + s3
>>> s
'TakeMakeFake'
>>> len(s)
12
>>> s[0:12:4]
'TMF'
```

Είσοδος δεδομένων.

Η είσοδος δεδομένων (από το πληκτρολόγιο) γίνεται με τη συνάρτηση `input`. Το (προαιρετικό) όρισμα της συνάρτησης τυπώνεται στην οθόνη κατά την εκτέλεση του προγράμματος ως μια προτροπή (prompt) στο χρήστη να εισαγάγει δεδομένα. Οτιδήποτε εισάγει ο χρήστης αποθηκεύεται ως ακολουθία χαρακτήρων και εκχωρείται στη μεταβλητή αριστερά του ίσον (=).

```
>>> name = input('What is your name? ')
What is your name? Barack Obama
>>> print('Hello,', name)
Hello, Barack Obama
>>> age = input('How old are you? ')
How old are you? 42
>>> print(age)
42
>>> type(age)
<class 'str'>
```

Παρατηρήστε ότι ο τύπος της μεταβλητής *name* αλλά και της *age* είναι `str`, δηλαδή ακολουθία χαρακτήρων (και όχι `int` που ίσως να περίμενε κάποιος). Μπορούμε όμως να μετατρέψουμε τη μεταβλητή *age* σε ακέραιο γράφοντας `int(age)` και με αυτό τον τρόπο αυτή μπορεί να χρησιμοποιηθεί σε αριθμητικές εκφράσεις:

```
>>> age = input('How old are you? ')
How old are you? 42
>>> my_age = 25
>>> print('You are', int(age)-my_age, 'years older than I am!')
You are 17 years older than I am!
```

Μπορούμε επίσης να περάσουμε το αποτέλεσμα της `input` απ' ευθείας σε μία άλλη συνάρτηση, π.χ., στην `float`.

```
>>> price = float(input('How much for the apples? '))
How much for the apples? 1.79
>>> type(price)
<class 'float'>
>>> kilos = 4.5
>>> cost = price * kilos
```

Η python διαθέτει την εντολή `eval` η οποία μετατρέπει το αποτέλεσμα της `input` στον τύπο που αναμένουμε:

```
>>> x = eval(input("Give a number: "))
Give a number: 1.1
>>> print(x)
1.1
```

Η εντολή `eval` αναγνωρίζει επίσης και αριθμητικές πράξεις:

```
>>> x = eval(input("Give a number: "))
Give a number: 1+2.4
>>> print(x)
3.4

>>> x = eval(input("Give a number: "))
Give a number: 3/4
>>> print(x)
0.75
```

Μελέτη

Βιβλιογραφία

1. Δ. Καρολίδης, *Μαθαίνετε εύκολα python*.