ΠΛΗΡΟΦΟΡΙΚΗ Ι (Python)
Ενότητα 6

Συμβολοσειρές (Strings)

Οι συμβολοσειρές ουσιαστικά αποτελούν αλληλουχίες χαρακτήρων. Η Python έχει τη δυνατότητα να χειρίζεται τις συμβολοσειρές με αυτόν τον τρόπο (ως αλληλουχίες χαρακτήρων) και διαθέτει συγκεκριμένες μεθόδους για τη διαχείρισή τους.

Η προσπέλαση στα στοιχεία μιας συμβολοσειράς (δηλαδή στους επιμέρους χαρακτήρες της) μπορεί να γίνει είτε με έναν βρόχο `for`, είτε με τη χρήση δεικτών. Με βρόχο `for`:

```
for μεταβλητή in συμβολοσειρά:
    εντολή
    εντολή
    ...
```

όπου σε κάθε επανάληψη εκχωρείται στη *μεταβλητή* από ένας χαρακτήρας της συμβολοσειράς. Π.χ.,

```
>>> text = 'Hello'
>>> for ch in text:
      print(ch)

H
e
l
l
o
```

Οι συμβολοσειρές είναι μια αμετάβλητη (immutable) δομή δεδομένων, επομένως τα στοιχεία τους δεν μπορούν να τροποποιηθούν. Αυτό φαίνεται π.χ. στο παρακάτω παράδειγμα:

```
>>> text = 'Hello'
>>> for ch in text:
      ch = 'x'
>>> print(text)
Hello
```

δηλαδή η εντολή `ch = 'x'` μέσα στο `for` δεν άλλαξε κάποιο από τα στοιχεία της συμβολοσειράς.

Παράδειγμα:

- Πρόγραμμα που μετράει τις εμφανίσεις κάποιου χαρακτήρα σε μια συμβολοσειρά:

```
def main():
    text = input('Πληκτρολόγησε ένα κείμενο: ')
    t = input('Δώσε έναν χαρακτήρα: ')
    count = 0
    for ch in text:
        if ch==t:
```

```
            count += 1
    print('Ο χαρακτήρας', t, 'εμφανίζεται', count, 'φορές στο κείμενο.')
```

Με τη χρήση δεικτών, η προσπέλαση στα στοιχεία μιας συμβολοσειράς μπορεί να γίνει όπως και στις λίστες:

```
συμβολοσειρά[δείκτης]
```

όπου και εδώ, η αρίθμηση των δεικτών ξεκινάει από το 0, ενώ υπάρχουν και οι ίδιες δυνατότητες χρήσης του τελεστή τεμαχισμού : όπως και στις λίστες. Επειδή όμως πρόκειται για αμετάβλητο τύπο δεδομένων, οποιαδήποτε εντολή εκχώρησης τιμής σε κάποιο στοιχείο, προκαλεί σφάλμα. Π.χ.,

```
>>> text = 'Hello'
>>> text[1]
'e'
>>> text[1:3]
'el'
>>> text[1:10]
'ello'
>>> text[1] = 'x'
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    text[1] = 'x'
TypeError: 'str' object does not support item assignment
```

Για τον υπολογισμό του πλήθους των χαρακτήρων μιας συμβολοσειράς, χρησιμοποιείται η συνάρτηση len. Π.χ.,

```
>>> text = 'Hello there'
>>> index = 0
>>> while index < len(text):
      print(text[index], end = ' ')
      index += 1

H e l l o   t h e r e
```

Υπάρχει επίσης και εδώ, όπως και στις λίστες, ο τελεστής συνένωσης +, καθώς και ο +=:

```
συμβολοσειρά_1 = συμβολοσειρά_2 + συμβολοσειρά_3
```

ή

```
συμβολοσειρά_1 += συμβολοσειρά_2
```

Επειδή όμως οι συμβολοσειρές είναι αμετάβλητος τύπος, εντολές της μορφής:

```
συμβολοσειρά_1 = συμβολοσειρά_1 + συμβολοσειρά_2
```

ή

```
συμβολοσειρά_1 += συμβολοσειρά_2
```

που φαινομενικά τροποποιούν τη συμβολοσειρά_1, δεν την τροποποιούν, αλλά δημιουργούν *νέα* συμβολοσειρά στην οποία αναφέρεται πλέον το όνομα συμβολοσειρά_1.

Τελεστές `in` και `not in`

Με τον τελεστή `in` μπορεί να ελεγχθεί εάν μια συμβολοσειρά περιέχεται σε κάποια άλλη συμβολοσειρά, ενώ με τον τελεστή `not in` εάν δεν περιέχεται. Π.χ.,

```
>>> text = 'Hello there'
>>> 'hello' in text
False
>>> 'there' in text
True
>>> 'Hello' not in text
False
```

Μέθοδοι για συμβολοσειρές

Η Python περιέχει πολλές μεθόδους που χρησιμοποιούνται για τη διαχείριση συμβολοσειρών. Οι πιο σημαντικές για τον *έλεγχο συμβολοσειρών* είναι οι εξής:

- `isalnum()` → Επιστέφει `True` αν η συμβολοσειρά περιέχει μόνο γράμματα ή αριθμητικά ψηφία.
- `isalpha()` → Επιστέφει `True` αν η συμβολοσειρά περιέχει μόνο γράμματα.
- `isdigit()` → Επιστέφει `True` αν η συμβολοσειρά περιέχει μόνο αριθμητικά ψηφία.
- `isspace()` → Επιστέφει `True` αν η συμβολοσειρά περιέχει μόνο λευκούς χαρακτήρες (δηλαδή κενά, `\n` και `\t`).
- `islower()` → Επιστέφει `True` αν όλα τα γράμματα της συμβοσειράς είναι πεζά.
- `isupper()` → Επιστέφει `True` αν όλα τα γράμματα της συμβοσειράς είναι κεφαλαία.

Οι πιο σημαντικές μέθοδοι για τη *δημιουργία τροποποιημένων εκδόσεων συμβολοσειρών* είναι οι εξής:

- `lower()` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας μετατρέψει όλα τα γράμματα σε πεζά.
- `upper()` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας μετατρέψει όλα τα γράμματα σε κεφαλαία.
- `strip()` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας αφαιρέσει όλους τους αρχικούς και τελικούς λευκούς χαρακτήρες (δηλαδή από την αρχή και από το τέλος).
- `strip(char)` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας αφαιρέσει όλες τις αρχικές και τελικές εμφανίσεις της συμβολοσειράς `char`.
- `lstrip()` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας αφαιρέσει όλους τους αρχικούς λευκούς χαρακτήρες, δηλαδή όσους βρίσκονται στην αρχή της συμβολοσειράς.
- `lstrip(`*`char`*`)` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας αφαιρέσει όλες τις αρχικές εμφανίσεις της συμβολοσειράς *`char`*, δηλαδή αυτές που βρίσκονται στην αρχή της συμβολοσειράς.
- `rstrip()` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας αφαιρέσει όλους τους τελικούς λευκούς χαρακτήρες, δηλαδή όσους βρίσκονται στο τέλος της συμβολοσειράς.
- `rstrip(`*`char`*`)` → Επιστέφει ένα αντίγραφο της συμβολοσειράς έχοντας αφαιρέσει όλες τις τελικές εμφανίσεις της συμβολοσειράς *`char`*, δηλαδή αυτές που βρίσκονται στο τέλος της συμβολοσειράς.

Οι πιο σημαντικές μέθοδοι για την *αναζήτηση και την αντικατάσταση υπο-συμβολοσειρών σε συμβολοσειρές* είναι οι εξής:

- `find(`*`substring`*`)` → Επιστέφει τον δείκτη της συμβολοσειράς από τον οποίο ξεκινάει η υπο-συμβολοσειρά *`substring`* στην πρώτη της εμφάνιση μέσα στη συμβολοσειρά.
- `startswith(`*`substring`*`)` → Επιστέφει `True` αν η συμβολοσειρά αρχίζει με την υπο-συμβολοσειρά *`substring`*.
- `endswith(`*`substring`*`)` → Επιστέφει `True` αν η συμβολοσειρά τελειώνει με την υπο-συμβολοσειρά *`substring`*.
- `replace(`*`old, new`*`)` → Επιστέφει ένα αντίγραφο της συμβολοσειράς στο οποίο όλες οι εμφανίσεις της υπο-συμβολοσειράς *`old`* έχουν αντικατασταθεί από την υπο-συμβολοσειρά *`new`*.

Άλλα χαρακτηριστικά των συμβολοσειρών

Ο τελεστής επανάληψης * λειτουργεί στις συμβολοσειρές όπως ακριβώς και στις λίστες, δηλαδή μπορεί να χρησιμοποιηθεί για να επαναλάβει μια συμβολοσειρά κάποιον συγκεκριμένο αριθμό φορών, ως εξής:

```
συμβολοσειρά * πλήθος_αντιγράφων
```

Π.χ.,

```
>>> my_str1 = 'a' * 10
>>> my_str1
'aaaaaaaaaa'
>>> my_str2 = 'Hello'*3
>>> my_str2
'HelloHelloHello'
```

Μια συμβολοσειρά μπορεί να διαχωριστεί στις λέξεις που την απαρτίζουν με τη χρήση της μεθόδου `split`. Η μέθοδος αυτή επιστρέφει μια ***λίστα*** με στοιχεία τις λέξεις που απαρτίζουν τη συμβολοσειρά με την οποία καλείται. Εξ ορισμού, η `split` χρησιμοποιεί ως διαχωριστικά λέξεων τα κενά. Μπορεί όμως να οριστεί και διαφορετικός χαρακτήρας διαχωρισμού, στέλνοντας τον επιθυμητό χαρακτήρα ως όρισμα στη μέθοδο. Π.χ.,

```
>>> my_str = 'Hello there!'
>>> words = my_str.split()
>>> words
['Hello', 'there!']
>>> date_str = '17/12/2015'
>>> date = date_str.split('/')
>>> date
['17', '12', '2015']
```

Κωδικοποίηση συμβολοσειρών

Η Python χρησιμοποιεί την *κωδικοποίηση* Unicode για την αναπαράσταση χαρακτήρων. Κάθε χαρακτήρας αντιστοιχεί σε ένα ακέραιο *i* (*0 <= i <= 1114111).*

Υπάρχουν 2 συναρτήσεις μετατροπής ανάμεσα σε χαρακτήρες και τους ακέραιους κώδικες που τους αντιστοιχούν.

- Χαρακτήρας σε ακέραιο κώδικα: η συνάρτηση `ord()`
  ```
  >>> print(ord('A'), ord('B'), ord('7'), ord('8'),ord('\n'))
  65 66 55 56 10
  ```
- Ακέραιος κώδικας σε χαρακτήρα: η συνάρτηση `chr()`
  ```
  >>> print(chr(67), chr(54), chr(37)
  C 6 %
  ```

Οι συνηθέστεροι ορατοί χαρακτήρες είναι ανάμεσα στο 32 και στο 128 και μπορούμε να τους εμφανίσουμε με το παρακάτω πρόγραμμα:

```
print(end='  ')
for i in range(32, 129):
    print(chr(i), end=' ')
    if i%10 == 0:
        print()
```

και εμφανίζει

```
   ! " # $ % & ' (
) * + , - . / 0 1 2
3 4 5 6 7 8 9 : ; <
= > ? @ A B C D E F
G H I J K L M N O P
Q R S T U V W X Y Z
[ \ ] ^ _ ` a b c d
e f g h i j k l m n
o p q r s t u v w x
y z { | } ~
```

Οι συμβολοσειρές συγκρίνονται *λεξικογραφικά* χαρακτήρα-προς-χαρακτήρα σύμφωνα με τον κώδικα των στοιχείων τους. Η σύγκριση σταματάει στην πρώτη θέση που διαφοροποιούνται οι αντίστοιχοι χαρακτήρες των 2 συμβολοσειρών, ή στο τέλος της "κοντύτερης" από αυτές.

Χρησιμοποιούνται οι γνωστοί τελεστές σύγκρισης (`==`, `!=`, `<`, `<=`, `>`, `>=`)