Ακολουθίες

Έχουμε δει μέχρι τώρα διάφορες ακολουθιακές δομές δεδομένων. Αυτές είναι οι (α) συμβολοσειρές, (β) λίστες (γ) πλειάδες (δ) `range` και ονομάζονται συλλογικά <<ακολουθίες>>.

Ας θεωρήσουμε ότι στο όνομα μεταβλητής `seq` έχουμε είτε μία συμβολοσειρά, είτε μία πλειάδα, είτε μία λίστα. Τότε μπορούμε να χρησιμοποιήσουμε τις ακόλουθες εντολές.

- `seq[i]`: i-οστό στοιχείο ακολουθίας.
- `seq[`αρχη`:`τελος`:`βημα`]`: τεμαχισμός ακολουθίας.
- `seq1 + seq2`: συνένωση ακολουθιών.
- `n*seq`: επαναλαμβανόμενη ακολουθία *n* φορές.
- στοιχειο `in seq`: ελέγχει αν το στοιχειο ανήκει στην `seq` (boolean).
- στοιχειο `not in seq`: ελέγχει αν το στοιχειο δεν ανήκει στην `seq` (boolean).
- `for` μεταβλητη `in seq`: διατρέχει τα στοιχεία ακολουθίας.

Επίσης, έχουμε τις ακόλουθες συναρτήσεις οι οποίες παίρνουν ως όρισμα μία ακολουθία

- `len(seq)`: ο αριθμός των στοιχείων ακολουθίας.
- `min(seq)`: το ελάχιστο στοιχείο ακολουθίας.
- `max(seq)`: το μέγιστο στοιχείο ακολουθίας.

Σε μία ακολουθία μπορούμε να εξασκήσουμε ορισμένες λειτουργίες:

- `seq.index(`στοιχειο`)`: η θέση της πρώτης εμφάνισης του <<στοιχείο>> στην ακολουθία.
- `seq.count(`στοιχειο`)`: αριθμός εμφανίσεων του <<στοιχείο>> στην ακολουθία.

Συμβολοσειρές και μέθοδοι

Αν *s* είναι μία συμβολοσειρά, έχουμε τις μεθόδους:

- `s.lower(), s.upper()`: μετατρέπει όλους τους χαρακτήρες σε πεζούς και σε κεφαλαίους αντίστοιχα.

- `s.capitalize()`: μετατρέπει σε κεφαλαίο τον πρώτο χαρακτήρα της συμβολοσειράς (αν αυτός είναι γράμμα).
- `s.replace(old,new)`: αντικαθιστά στην *s* την συμβολοσειρά *old* με την συμβολοσειρά *new*.

Παράδειγμα.

```
>>> s = "I like python"
>>> s1 = s.replace("like","love")
>>> print(s1)
I love python
>>> s2 = s1.upper()
>>> print(s2)
I LOVE PYTHON
```

Ερώτηση κατανόησης. Η συμβολοσειρά *s* προποποιείται ή όχι μετά την εφορμογή της μεθόδου `s.upper()`;

Επίσης, έχουμε τις μεθόδους:

- `s.split(d)`: χωρίζει την *s* όπου υπάρχει το *d* και παράγει μία λίστα με το αποτέλεσμα.
- `s.partition(d)`: χωρίζει την *s* εκεί που θα βρει την συμβολοσειρά *d* και επιστρέφει τρεις συμβολοσειρές: το κομμάτι της *s* πριν την *d*, την *d* και το κομμάτι της *s* μετά την *d*.

Παράδειγμα. Θα διαβάσουμε σειρά αριθμών από το πληκτρολόγιο και θα τους καταχωρήσουμε σε μία λίστα ως floats.

```
>>> line = input("Give numbers separated by comma: ")
Give numbers separated by comma: 1.2,  2.3,  3.14, 6
>>> strList = line.split(',')
>>> strList
['1.2', '  2.3', '  3.14', ' 6']
>>> numList = []
>>> for e in strList:
        numList.append(float(e))

>>> numList
```

```
[1.2, 2.3, 3.14, 6.0]
```

Ερώτηση. Πώς θα τροποποιούσατε το παραπάνω πρόγραμμα ώστε να διαβάζει μία σειρά αριθμών οι οποίοι διαχωρίζονται μόνο με κενά μεταξύ τους;

Παράδειγμα. Διαβάστε μία έκφραση η οποία παριστάνει άθροισμα, εκτελέστε τις πράξεις και δώστε το αποτέλεσμα.

```
>>> s = input("Give an expression for a sum: ")
Give an expression for a sum: 1.2+4.5+6.4+14.8
>>> L = s.split("+")
>>> print(L)
['1.2', '4.5', '6.4', '14.8']
>>> summ = 0
>>> for e in L:
...     summ += float(e)
...
>>> print(summ)
26.900000000000002
```

Έχουμε επίσης τις μεθόδους:

- `s.index(d)`: θέση του στοιχείου d στην s.
- `s.find(d)`: θέση του στοιχείου d στην s.

Παράδειγμα.

```
>>> s = "abcdefghijk"
>>> s.find("c")
2
>>> s.index("c")
2
>>> s.find("z")
-1
>>> s.index("z")
Traceback (most recent call last):
  File "", line 1, in
```

```
ValueError: substring not found
```

Έχουμε επίσης την μέθοδο:

- `ch.join(s)`: μεταξύ των διαδοχικών χαρακτήρων της συμβολοσειράς *s* θα παρεμβληθεί η συμβολοσειρά *ch*.
- `ch.join(L)`: μεταξύ των διαδοχικών στοιχείων της λίστας συμβολοσειρών *L* θα παρεμβληθεί η συμβολοσειρά *ch*.

Η μέθοδος επιστρέφει μία συμβολοσειρά.

Παράδειγμα.

```
>>> s = '12345'
>>> '-'.join(s)
'1-2-3-4-5'
>>> L = ['31','12','2017']
>>> '-'.join(L)
'31-12-2017'
>>> L = ['Once','Upon','A','Time']
>>> ' '.join(L)
'Once Upon A Time'
>>> ''.join(L)
'OnceUponATime'
```

Έχουμε επίσης τις μεθόδους:

- `s.isalpha()`: ελέχγει αν όλοι οι χαρακτήρες σε ένα string είναι γράμματα.
- `s.isdigit()`: ελέχγει αν όλοι οι χαρακτήρες σε ένα string είναι αριθμητικά ψηφία.

Επίσης τις μεθόδους (χρήσιμες για το formatting των strings που θα δούμε αργότερα):

- `s.strip()`: αφαιρεί τους κενούς χαρακτήρες από την αρχή και το τέλος της *s*.
- `s.lstrip(), s.rstrip()`: αφαιρεί τους κενούς χαρακτήρες από την αρχή (αριστερά) ή από το τέλος (δεξιά) της *s* αντίστοιχα.
- `s.center(`*πλατος*`), s.ljust(`*πλατος*`[,`*χαρακτήρας*`]), s.rjust(`*πλατος*`[,`*χαρακτήρας*`])`
- `s.endwith(`συμβολοσειρα`), s.startwith(`συμβολοσειρα`)`

Λίστες: μέθοδοι και επεξεργασία.

List comprehension (Υπολογιζόμενη λίστα). Έχουμε την ακόλουθη μέθοδο δημιουργίας μιας λίστας.

```
comprehensionList = [εκφραση for μεταβλητη in ακολουθια]
```

```
>>> squares = [i**2 for i in range(11)]
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Παράδειγμα. Χρειαστήκαμε σε προηγούμενο παράδειγμα να μετατρέψουμε λίστα strings σε floats. Για να μετατρέψουμε την λίστα των strings σε λίστα floats μπορούμε να χρησιμοποιήσουμε υπολογιζόμενη λίστα (list comprehension):

```
numList = [float(e) for e in strList]
```

Συνάρτηση `map`. Για να μετατρέψουμε τα στοιχεία λίστας `strList` από strings σε floats μπορούμε επίσης να χρησιμοποιήσουμε την συνάρτηση `map`:

```
numList = map(float,strList)
list(numList)
```

Μελέτη

Βιβλιογραφία

1. Guru99, Python Strings: Replace, Join, Split, Reverse.
2. J.V. Guttag, *Υπολογισμοί και προγραμματισμός με την python* (Κεφάλαιο 5).
3. Δημήτριος Καρολίδης, *Μαθαίνετε εύκολα python* (Παράγραφοι 4.1, 4.2) (Εκδόσεις Καρολίδη, 2016).
4. Κ. Μαγκούτης, Χ. Νικολάου, *Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με Python*, (Αποθετήριο "Κάλλιπος", 2016) - Κεφάλαιο 6. Συμβολοσειρές, λίστες, πλειάδες, λεξικά.