Λίστες

Σειρά στοιχείων

Μία σειρά στοιχείων (αριθμών, συμβολοσειρών κλπ) μπορούν να οργανωθούν σε μία δομή η οποία ονομάζεται *λίστα*. Η σειρά των στοιχείων ορίζεται από ορθογώνιες παρενθέσεις και τα στοιχεία χωρίζονται μεταξύ τους με κόμμα. Παραδείγματα:

```
>>> rgb = ['red','green','blue']
>>> cities = ['Agios','Heraklion','Rethymno','Chania']
>>> digits = [0,1,2,3,4,5,6,7,8,9]
```

Ο αριθμός των στοιχείων μίας λίστας μπορεί να βρεθεί με χρήση της συνάρτησης `len`:

```
>>> len(digits)
10
```

Μπορούμε να ανακτήσουμε ένα από τα στοιχεία της λίστας χρησιμοποιώντας τον δείκτη του. Η δεικτοδότηση είναι ακριβώς όπως στις συμβολοσειρές.

```
>>> rgb[1]
'green'
>>> rgb[-1]
'blue'
```

Τεμαχισμός (slicing): Μπορούμε να ανακτήσουμε ένα τμήμα της λίστας χρησιμοποιώντας ένα εύρος δεικτών. Το αποτέλεσμα είναι μία νέα λίστα.

```
>>> digits[1:3]
```

```
[1,2]
>>> digits[7:]
[7,8,9]
```

Βλέπουμε ότι η σύνταξη είναι όμοια με αυτή που χρησιμοποιήσαμε στον τεμαχισμό συμβολοσειρών.

Βασικές λειτουργίες

Μπορούμε να χρησιμοποιήσουμε το σύμβολο της πρόσθεσης (+) για λίστες. Από την πρόσθεση δύο λιστών προκύπτει μία μέα λίστα η οποία περιέχει όλα τα στοιχεία των δύο λιστών. Για παράδειγμα:

```
>>> cmyk = ['cyan','magenta','yellow','black']
>>> colors = rgb + cmyk
>>> colors
['red','green','blue','cyan','magenta','yellow','black']
```

Μπορούμε να χρησιμοποιήσουμε το σύμβολο του πολλαπλασιασμού (*) μεταξύ ακεραίου *n* και λίστας *L*:

```
>>> 2*rbg
['red','green','blue','red','green','blue']
>>> 2*[0,1]
[0,1,0,1]
```

Η `n*L` δίνει μία νέα λίστα η οποία περιέχει *n* φορές τα στοιχεία της *L* σε διαδοχική σειρά.

Μέθοδοι

Σε μία υπάρχουσα λίστα μπορούμε να προσθέσουμε ένα στοιχείο. Για παράδειγμα:

```
>>> colours.append('white')
>>> colors
['red','green','blue','cyan','magenta','yellow','black','white']
```

L.append(e): Έχουμε χρησιμοποιήσει μία *μέθοδο* η οποία ονομάζεται `append` και αυτή επέδρασε στη λίστα `colors`. Προστέθηκε ένα στοιχείο (`'white'`) στην λίστα. Το όρισμα `e` της μεθόδου `append` προστίθεται ως τελευταίο στοιχείο στη λίστα `L` και η υπάρχουσα λίστα μεγαλώνει κατά ένα στοιχείο.

Η γενική σύνταξη για μία μέθοδο είναι `λιστα.μεθοδος(ορισμα)`. Ας δούμε τώρα μερικές χρήσιμες μεθόδους για λίστες.

L.remove(e): Αφαιρεί το στοιχείο `e` από την λίστα `L`. Για παράδειγμα:

```
>>> colors.remove('white')
>>> colors
['red', 'green', 'blue', 'cyan', 'magenta', 'yellow', 'black']
```

L.pop(): Αφαιρεί το τελευταίο στοιχείο λίστας `L` (και το επιστρέφει). Επίσης μπορούμε να αφαιρέσουμε στοιχείο με συγκεκριμένο δείκτη.

```
>>> colors.pop()
'white'
>>> colors
['red','green','blue','cyan','magenta','yellow', 'black']]

>>> colors.pop(1)
'green'
>>> colors
['red','blue','cyan','magenta','yellow', 'black']
```

Μπορούμε να ελέγξουμε αν ένα στοιχείο υπάρχει σε λίστα:

```
>>> 'white' in colors
False
```

Θα μπορούσαμε να προσθέσουμε στην λίστα στοιχεία εφόσον αυτά δεν υπάρχουν ήδη:

```
>>> if 'white' not in colors: colors.append('white')
```

Παράδειγμα.

```
# Create lists with the countries of the continents
Europe = ['greece','italy','france','norway','denmark','the
netherlands']
Asia = ['china','india','korea','nepal','thailand','iran']
America = ['USA','perou','canada','mexico']

# ask the user for the name of a country
country = input('Give the name of a country: ')

# Initialize an auxiliary variable
newCountry = ''

# check whether the given country is included in a list
if country in Europe:
        print(country,'is in Europe')
elif country in Asia:
        print(country,'is in Asia')
elif country in America:
        print(country,'is in America')
else:
        newCountry = input('Does this country belong to Europe, Asia
or America? ')


# Add the new country to one of the lists
if newCountry == 'Europe':
        Europe.append(country)
elif newCountry == 'Asia':
        Asia.append(country)
elif newCountry == 'America':
        America.append(country)

# Print the content of the lists
print(Europe)
print(Asia)
print(America)
```

Παρατήρηση. Μία λίστα είναι δυνατόν να μεταβληθεί και αυτό μπορεί να συμβεί, π.χ., με μία μέθοδο. Λέμε ότι οι λίστες είναι *μεταλλάξιμες*.

Παράδειγμα. Εισάγετε μία ημερομηνία (ημέρα, μήνας, έτος) και βρείτε (και τυπώστε) την ημερομηνία της επόμενης ημέρας.

```
import sys      # Εισάγει εντολές και συναρτήσεις του interpreter

months31 = [1,3,5,7,8,10,12]          # Μήνες με 31 ημέρες
months30 = [4,6,9,11]                 # Μήνες με 30 ημέρες

# Εισάγουμε ημερομηνία
day   = int(input("Give day: "))
month = int(input("Give month: "))
year  = int(input("Give year: "))

if 1 <= month <= 12:                  # Ο μήνας πρέπει να είναι από 1 έως
12
   print("You entered the date {}-{}-{}".format(day,month,year))
else:
   print("There is no such month")
   sys.exit()                         # Τερματίζεται η εκτέλεση του
προγράμματος

if month in months31:                 # Βρίσκουμε την τελευταία ημέρα του
μήνα
   lastday = 31
elif month in months30:
   lastday = 30
elif month == 2:
   lastday = 28
else:                                  # Θα συμβεί αν δεν έχουμε προβλέψει
πόσες ημέρες έχει ο μήνας
   print("Month has no days!")
   sys.exit()

# Βρίσκουμε την επόμενη ημερομηνία: ημέρα, μήνα, έτος
if 0 < day < lastday:
   day = day + 1
elif day == lastday:
   day = 1
```

```
        if month == 12:
            month = 1
            year = year + 1
        else:
            month = month + 1
    else:
        print("There is no such day!")
        sys.exit()

    # Τυπώνουμε το αποτέλεσμα
    print("Next date is {}-{}-{}".format(day,month,year))
```

Άσκηση. Εισάγετε μία ημερομηνία (ημέρα, μήνας, έτος) και βρείτε (και τυπώστε) την ημερομηνία της επόμενης ημέρας. Λάβετε υπόψιν την περίπτωση να είναι το έτος δίσεκτο.

Άλλες μέθοδοι

L.index(e). Επιστρέφει το δείκτη της πρώτης παρουσίας του στοιχείου `e` στη λίστα `L`.

L.count(e). Επιστρέφει τον αριθμό φορών που εμφανίζεται το στοιχείο `e` στη λίστα `L`.

L.reverse(). Αναστρέφει τη σειρά των στοιχείων στη λίστα `L`.

L.sort(). Ταξινομεί τα στοιχεία της λίστας `L` σε αύξουσα σειρά.

L.insert(i,e). Το στοιχείο `e` τοποθετείται στη θέση `i` της λίστας `L`.

Παράδειγμα. Ας δημιουργήσουμε μία λίστα, μετά θα αντιστρέψουμε τη σειρά των στοιχείων της.

```
>>> L = ['a','b','c','d','e','f']
>>> L.reverse()
>>> L
['f', 'e', 'd', 'c', 'b', 'a']
```

Παρατήρηση. Δείτε ότι η εντολή `L.reverse` (καθώς και αρκετές άλλες μέθοδοι: `sort`, `insert` κλπ) μεταλλάσσει την ίδια τη λίστα (και δεν δημιουργεί μία καινούρια για να βάλει το αποτέλεσμα της μεθόδου).

Παράδειγμα. Ας δημιουργήσουμε μία λίστα για την οποία μετά θα ανακατέψουμε τα στοιχεία της με τυχαίο τρόπο και τέλος θα εντοπίσουμε ένα συγκεκριμένο στοιχείο της.

```
import random
L = ['a','b','c','d','e','f']
random.shuffle(L)
print(L.index('a'))
```

Ειδικότερα είδη λιστών

Παρατήρηση. Μία λίστα μπορεί να περιέχει στοιχεία διαφορετικών τύπων:

```
>>> exampleList = ['father',30,'son',3]
```

Παρατήρηση. Μία λίστα μπορεί να περιέχει λίστες:

```
>>> exampleList = [[1,2,3],['a','b','c']]
```

Παράδειγμα. (Άρτιες μεταθέσεις τριών αριθμών.) Ξεκινούμε με μία άδεια λίστα `[]`. Ακολούθως, της επεκτείνουμε βάζοντας μία-μία τις άρτιες μεταθέσεις τριών αριθμών. Κάθε μία από τις μεταθέσεις είναι λίστα τριών αριθμών.

```
>>> permute = []
>>> permute.append([1,2,3])
>>> permute.append([2,3,1])
>>> permute.append([3,1,2])
>>> permute
[[1, 2, 3], [2, 3, 1], [3, 1, 2]]
```

Παράδειγμα.

Θα δούμε την πρόσβαση στα στοιχεία μίας λίστας `L1` η οποία είναι στοιχείο άλλης λίστας `L`. Αυτό μπορεί να γίνει με δύο τρόπους: είτε μέσω της αρχικής λίστας `L1` είτε μέσω του αντίστοιχου στοιχείου της μεγαλύτερης λίστας `L`.

```
>>> L1 = ['a','b','c']
>>> L2 = [1,2,3]
>>> L = [L1,L2]
>>> L
[['a', 'b', 'c'], [1, 2, 3]]
L1.append('d')
>>> L
[['a', 'b', 'c', 'd'], [1, 2, 3]]
>>> L[0].append('e')
>>> L
[['a', 'b', 'c', 'd', 'e'], [1, 2, 3]]
>>> L1
['a', 'b', 'c', 'd', 'e']
```

Ακολουθία range()

Η εντολή `range(n,m)` δημιουργεί μία ακολουθία ακεραίων από τον *n* έως τον *m-1*.

```
>>> range(1,10)
range(1, 10)
```

Μπορούμε να δημιουργήσουμε μία λίστα που περιέχει την ακολουθία αριθμών της `range`:

```
>>> list(range(1,10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Παραδείγματα. (α) Αν παραλήψουμε το πρώτο όρισμα τότε αυτό θεωρείται ότι είναι το 0. (β) Ως τρίτο όρισμα μπορούμε να βάλουμε το βήμα για την ακολουθία των παραγόμενων αριθμών.

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(0,10,2))
```

```
[0, 2, 4, 6, 8]
```

Η γενική μορφή της συνάρτησης είναι `range(start,end,step)` και κατασκευάζει την αριθμητική πρόοδο `start, start+step, start+2*step,...,start+n*step`.

- Αν `step>0` τότε στην παραπάνω ακολουθία ο `start+n*step` είναι ο μεγαλύτερος ακέραιος μικρότερος από τον `end`.
- Αν `step<0` τότε ο `start+n*step` είναι ο μικρότερος ακέραιος μεγαλύτερος από τον `end`.
- Αν το `start` παραληφθεί τότε `start=0`. Αν το `step` παραληφθεί τότε `step=1`.

Μελέτη

Βιβλιογραφία

1. J.V. Guttag, *Υπολογισμοί και προγραμματισμός με την python* (Κεφάλαιο 5).
2. Δημήτριος Καρολίδης, *Μαθαίνετε εύκολα python* (Παράγραφοι 4.1, 4.2) (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