8.2 Python2 - Λίστες

από το βιβλίο μας

Η λίστα είναι μια διατεταγμένη ακολουθία αντικειμένων, όχι απαραίτητα του ίδιου τύπου και αποτελεί τη βασική δομή δεδομένων της Python. Η λίστα, σε αντίθεση με τη συμβολοσειρά, είναι μια δυναμική δομή στην οποία μπορούμε να προσθέτουμε ή να αφαιρούμε στοιχεία (mutable). Κάθε αντικείμενο της λίστας χαρακτηρίζεται από ένα μοναδικό αύξοντα αριθμό, ο οποίος ορίζει τη θέση του στη λίστα, ενώ η προσπέλαση στα στοιχεία της λίστας γίνεται όπως στις συμβολοσειρές, με το όνομα της λίστας και τον αύξοντα αριθμό του αντικείμενου μέσα σε αγκύλες.

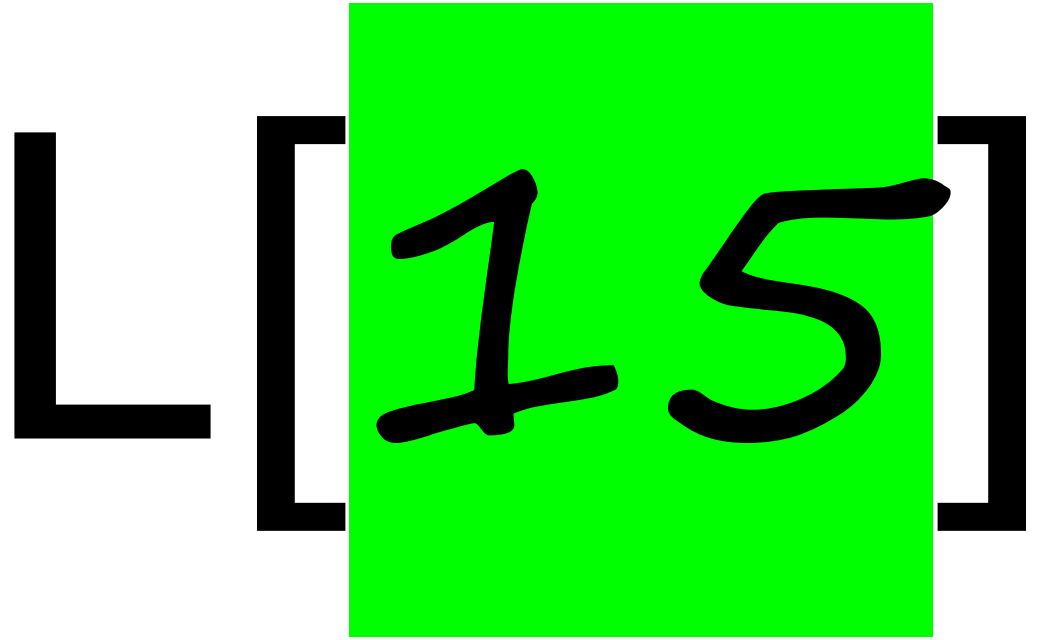
Δομή παρουσίασης

- Βασικά χαρακτηριστικά
- Λειτουργίες λίστας
- Συνηθισμένα κομμάτια κώδικα
- Παραδείγματα βιβλίου

Λίστα: Βασικά χαρακτηριστικά

- Διατεταγμένη ακολουθία αντικειμένων
  - >>> L[0], L[1], L[2], ...
- Όχι απαραίτητα του ίδιο τύπου
  - >>> L = [1, 2.0, "Hello!"]
- Δυναμική δομή (μεταβαλλόμενη – mutable)
  - >>> id(L)
  - 140371624912048
  - >>> L.append("kati akoma")
  - >>> id(L)
  - 140371624912048
- Αρίθμηση από το 0

Επιλογή ενός στοιχείου της L



L: λίστα, 15: ο δείκτης

Λειτουργίες Λίστας

- πρόσθεση αντικειμένων σε λίστα, πρόσθεση λιστών
- in, not in, len()
- list(“hello”)
- κομμάτιασμα λιστών (list slicing)
- range() : γρήγορη δημιουργία λιστών με ακέραιους
- συναρτήσεις που έχει κάθε λίστα
  - L.append(), L.insert(), L.pop()

δημιουργία λίστας

L = []

```
L = ["cpu","ram",3]
```

πρόσθεση αντικειμένων σε λίστα

```
>>> L = []
>>> L.append(1)
>>> L.insert(0,"a")
>>> L = L + [False] # both iterable, new object
>>> L += [True]     # both iterable, same object
>>> L
```

+ vs +=

+

```
>>> id(L)
140630764913600
>>> L = L + [False]
>>> id(L)
140630764631696
```

+=

```
>>> id(L)
140630764913600
>>> L += [False]
>>> id(L)
140630764913600
```

... πρόσθεση στο τέλος της λίστας ..

iterable

```
>>> L = []
>>> L.append(1)
>>> L += 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

Ποιά είναι iterables ?

- for ... in ...:

Ποιά είναι iterables ?

- Strings
  - `for letter in sentence:`
- Files:
  - `for line in f:`
- Lists:
  - `for l in L:`

iterable

```
>>> L = []
>>> L.append(1)
>>> L += 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

iterable

```
>>> L = []
>>> L.append(1)
>>> L += [2,5] #ένα ένα τα στοιχεία
               #στο τέλος της L
>>> L
[1, 2, 5]
```

iterable

```
>>> L = []
>>> L += “Hello”
```

iterable

```
>>> L = []
>>> L +=["Hello"]
>>> L
['H', 'e', 'l', 'l', 'o']
```

`in`, **`not`** `in` **και** `len`

- Πολύ παρόμοια με τα strings

```
>>> fruits = [‘apple’, ‘orange’]
>>> len(fruits)
2
>>> print fruits[0]
apple
>>>‘apple’ in fruits
True
>>> powers = [2, 4, 8, 16]
```

in, **not** in **και** len

```
>>> fib = [3, 5, 8, 13, 21]
>>> fib + powers
[3, 5, 8, 13, 21, 2, 4, 8, 16]
>>> powers + fruits
[2, 4, 8, 16, ‘apple’, ‘orange’]
>>> fib = fib + [ fib[3] + fib[4] ]
>>> print fib
[3, 5, 8, 13, 21, 34]
```

`list()`

- μετατροπή ενός iterable σε λίστα.

L = list(“hello”)

- “Hello”
  - Είναι “iterable” (ακολουθιακή δομή)
  - Παίρνω ένα ένα τα στοιχεία του και τα βάζω σε λίστα.
  - Το αποτέλεσμα είναι μία λίστα
  - με ένα ένα τα στοιχεία του iterable

L = list(“hello”)

```
>>> L
['H', 'e', 'l', 'l', 'o']
```

L = list(f)

```
>>> f = open(“test.txt”)
>>> L = list( f )
>>> L
???????
```

κομμάτιασμα λιστών (list slicing)

- Ακριβώς το ίδιο με τα Strings
- Δημιουργείται νέα λίστα με το κομμάτιασμα

```
K = L[_____:_____:_____]
```

κομμάτιασμα λιστών (list slicing)

```
Κ = L[από:μέχρι:βήμα]
```

αν το βήμα λείπει, υπονοείται το 1

κομμάτιασμα λιστών (list slicing)

```
Κ = L[από:μέχρι:βήμα]
```

αν το από λείπει, υπονοείται από την αρχή

κομμάτιασμα λιστών (list slicing)

```
Κ = L[από:μέχρι:βήμα]
```

αν το μέχρι λείπει, υπονοείται μέχρι το τέλος

```
>>> word = “zanneio gymnasio”


>>> word[ :7 ]
‘zanneio’
>>> word[ 8: ]
‘zanneio’
>>> word[ : ]
‘zanneio gymnasio’
```

```
>>> word[ 3:11 ]
'neio gym'
>>> word[ 0:7 ]
'zanneio'
>>> word[ 8:len(word) ]
'gymnasio'
>>> word[ 0:len(word) ]
'zanneio gymnasio'
```

```
>>> fibonacci = [5,8,13,21,34]
>>> fib = fibonacci[ : ]        # δημιουργεί αντίγραφο
```

Προσοχή! Δεν μπορεί να γίνει το ίδιο με τις συμβολοσειρές(strings).

Δηλαδή η εντολή word[:] δεν δημιουργεί αντίγραφο της συμβολοσειράς word.

Αυτό μπορείτε να το διαπιστώσετε με χρήση της συνάρτησης id που δείχνει τη διεύθυνση στη μνήμη του κάθε αντικειμένου.

range()

- γρήγορη δημιουργία λιστών με ακέραιους

```
1) range(_____)
   πχ range(3) -> [0,1,2]
2) range(_____, _____)
   πχ range(0,2) -> [0,1]
3) range(_____, _____, _____)
   πχ range(0,6,2) -> [0,2,4]
```

συναρτήσεις που έχει κάθε λίστα

```
>>> a, b = ["letters"], ["numbers"]
>>> a.append("a")                    #προσθέτει στο τέλος
>>> b.append(1)
>>> a
['letters', 'a']
>>> b
['numbers', 1]
>>>
```

συναρτήσεις που έχει κάθε λίστα

```
>>> b
['numbers', 100, 1]
>>> b.insert(1,"W")
>>> b
['numbers', 'W', 100, 1]
>>>
```

συναρτήσεις που έχει κάθε λίστα

```
>>> b
['numbers', 100, 1]
>>> b.insert(10,"θέση δέκα;;;")
>>> b
['numbers', 100, 1, 'θέση δέκα;;;']
>>>
```

συναρτήσεις που έχει κάθε λίστα

```
>>> a
['letters', 'z', 'a']
>>> a.pop() #από το τέλος, βγάζει ένα
'a'
>>> l = a.pop()
>>> print l
'z'
>>> a
['letters']
```

συναρτήσεις που έχει κάθε λίστα

```
>>> a
['letters', 'z', 'a']
>>> l = a.pop(1)
>>> print l
'z'
>>> a
['letters', 'a']
```

a **vs** b

- Και τα δύο λίστες *(ανήκουν στην ίδια κλάση)*
- Έχουν ακριβώς τις ίδιες συναρτήσεις
  - a.append(), b.append() ....
- Διαφέρουν ως προς τα δεδομένα που έχουν
  - άλλο το a[0], άλλο το b[0]

ίδιες λειτουργίες,

διαφορετικά δεδομένα

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας List με for

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας List με for

```
for item in List :
    <Εντολές Επεξεργασίας του αντικειμένου item>
```

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας L με for

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for item in L :
        print item,
6 28 496 8128
```

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας L με for με χρήση δείκτη

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for index in [ 0, 1, 2, 3 ] :
        print L[index],
6 28 496 8128
```

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας L με for με χρήση δείκτη

```
>>> L = [ 6, 28, 496, 8128 ]
>>> for index in range(0,4) :
        print L[index],
6 28 496 8128
```

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας με while (αναγκαστικά) με χρήση δείκτη

Κώδικας

- ένα ένα τα στοιχεία μιας λίστας με while (αναγκαστικά) με χρήση δείκτη

```
>>> a = ["apple", "lemon", "cinamon"]
>>> i = 0
>>> while i<len(a):
...     print a[i] ,
...     i+=1
apple lemon cinamon
```

Συνηθισμένα προβλήματα λιστών

1)Εμφάνιση στοιχείων λίστας

2)Άθροισμα όλων των στοιχείων μιας λίστας

3)Θέση Μέγιστης/Ελάχιστης τιμής μιας λίστας

4)Επιλογή στοιχείων μιας λίστας (διαχωρισμός)

5)Ενοποίηση ταξινομημένων λιστών σε μία λίστα

Εμφάνιση στοιχείων λίστας

```
L = [1, 2, 3, 4, 5, 6]
for number in L :
    print number
```

Άθροισμα όλων των στοιχείων μιας λίστας

```
sum = 0.0                          # το sum είναι πραγματικός (float)
for number in L :
    sum = sum + number
average = sum / len( L )           # δεν θα γίνει ακέραια διαίρεση
print average
```

Θέση Μέγιστης/Ελάχιστης τιμής μιας λίστας

```
maximum = L[0]
for number in L :
    if number > maximum :
        maximum = number
print maximum
```

Θέση Μέγιστης/Ελάχιστης τιμής μιας λίστας

```
import random


L = []
for i in range(10):
    n = random.randint(0,100)
    L.append( n )

#η συνέχεια στην επόμενη διαφάνεια
```

Θέση Μέγιστης/Ελάχιστης τιμής μιας λίστας

```
#συνέχεια από προηγούμενη διαφάνεια

i = 0
while i<=len(L):
    print L[i]



print "η θέση της μέγιστης τιμής είναι"
```

Επιλογή στοιχείων μιας λίστας (διαχωρισμός)

```
positives = [ ]
negatives = [ ]
for number in numbers :      # για κάθε αριθμό της λίστας
    if number > 0 :              # αν είναι θετικός
        positives.append( number )
        # πρόσθεσέ τον στη λίστα με τους θετικούς
    else :
        negatives.append( number )
        # αλλιώς στη λίστα με τους αρνητικούς
print positives
print negatives
```

Ενοποίηση ταξινομημένων λιστών σε μία λίστα

```
def merge( A, B ) :
    L = [ ]
    # όσο οι δυο λίστες έχουν στοιχεία
    while A != [ ] and B != [ ] :
        if A[0] < B[0] : # Αν το 1ο στοιχείο της A είναι το μικρότερο
            L.append( A.pop(0) )        # το μεταφέρουμε στο τέλος της L
        else :
            L.append( B.pop(0) )
            # αλλιώς μεταφέρουμε το πρώτο στοιχείο της B στην L
    return L + A + B
    # στο τέλος προσθέτουμε τα στοιχεία που έχουν μείνει
```

Δύσκολο πρόβλημα βιβλίου

- Παράδειγμα 4 - Ρέστα

Παράδειγμα 4. Ρέστα

Καλούμαστε να σχεδιάσουμε το λογισμικό μιας ταμειακής μηχανής, το οποίο θα διαβάζει το ποσό που έδωσε ο πελάτης και το κόστος των αγορών του και θα εμφανίζει το ελάχιστο πλήθος κερμάτων ή χαρτονομισμάτων που θα δοθούν για ρέστα. Θεωρήστε ότι όλες οι τιμές είναι σε ακέραια πολλαπλάσια του ευρώ:

σκέψη

- κόστος : 124
- εμείς δώσαμε : 150
- ρέστα = 150 – 124 = 26 (αυτά που δώσαμε – κόστος)
- `νομίσματα = [100, 50, 20, 10, 5, 2, 1]`
- Ξεκινάμε από το μεγαλύτερο:
  - τα 100
    - Πως θα βρούμε πόσα 100 είναι στα ρέστα;
      - ρέστα / 100
      - ότι περισσέψει,
      - θα το δώσω με το επόμενο νόμισμα

ονομάζουμε τις οντότητες

- κόστος : 124 **#κόστος**
- εμείς δώσαμε : 150 **#έδωσε**
- ρέστα = 150 – 124 = 26 **#ρέστα**
- `νομίσματα = [100, 50, 20, 10, 5, 2, 1]` **#νομίσματα**
- **#πλήθος_νομισμάτων : πόσα κέρματα ή χαρτονομίσματα έδωσε**
- Ξεκινάμε από το μεγαλύτερο: **#για κάθε νόμισμα**
  - τα 100
    - Πως θα βρούμε πόσα 100 είναι στα ρέστα;
      - ρέστα / 100 **#ρέστα/νόμισμα**
      - **#πλήθος_νομισμάτων += ρέστα/νόμισμα**
      - ότι περισσέψει, **#ρέστα = ρέστα % νόμισμα**
      - θα το δώσω με το επόμενο νόμισμα

ονομάζουμε τις οντότητες

- `νομίσματα = [100, 50, 20, 10, 5, 2, 1]` **`#νομίσματα`**
- κόστος : 124 **`#κόστος`** `= input( “Δώσε το κόστος των αγορών” )`
- εμείς δώσαμε : 150 **`#έδωσε`** `= input( “Δώσε το κόστος των αγορών” )`
- ρέστα = 150 – 124 = 26 **`#ρέστα`** `= έδωσε – κόστος`
- **`#πλήθος_νομισμάτων : πόσα κέρματα ή χαρτονομίσματα έδωσε`** `= 0`
- Ξεκινάμε από το μεγαλύτερο: `#for` **`νόμισμα`** `in νομίσματα :`
  - **`#πλήθος_νομισμάτων += ρέστα/νόμισμα`**
  - ότι περισσέψει, **`#ρέστα = ρέστα % νόμισμα`**

```
νομίσματα = [100, 50, 20, 10, 5, 2, 1]
κόστος = input( “Δώσε το κόστος των αγορών” )
έδωσε = input( “Δώσε το ποσό πληρωμής” )
ρεστα = έδωσε - κόστος
πλήθος_νομισμάτων = 0
for νόμισμα in νομίσματα :
  πλήθος_νομισμάτων += ( ρεστα / νόμισμα )
  ρεστα = ρεστα % νόμισμα
print counter
```