Λίστες στην Python

Υλοποίηση μιας ιδέας από τη ζωή

ΧΡΗΣΤΟΣ ΜΑΒΟΓΛΟΥ

Προαπαιτούμενες γνώσεις:

1. Γνώση βασικών τύπων δεδομένων

- **integer** 5
- **float** 5.0
- **string** “Hello”
- **Boolean** True

2. Γνώση βασικών εντολών και σύνταξης της Python

```
x=input()
y=x*2
print y
```

3. Χρήση της range στη δομή επανάληψης for

```
for i in range(1,11):
    print i
```

Τι είναι μια λίστα;

Από τις μεταβλητές στις λίστες

Ένα δεδομένο → Πολλά δεδομένα ενοποιημένα

Ορισμός λίστας

Οι λίστες είναι δομές δεδομένων που επιτρέπουν την αποθήκευση πολλών δεδομένων διαφόρων τύπων (Integer, Float, String, Boolean).

Ευελιξία των λιστών (Mutable Δομή Δεδομένων)

Οι λίστες προσφέρουν ευελιξία, καθώς μπορείτε να προσθέτετε ή να αφαιρείτε στοιχεία εύκολα, προσαρμόζοντας τη δομή τους.

Διαχείριση δεδομένων

Η χρήση λιστών διευκολύνει τη διαχείριση μεγάλου όγκου δεδομένων, επιτρέποντας την οργανωμένη αποθήκευση και πρόσβαση.

Προηγούμενες γνώσεις

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

Η συνάρτηση **range( )** δημιουργεί μια λίστα ακεραίων αριθμών.

Μπορεί η range να δημιουργήσει λίστα αριθμών τύπου float ;

Δημιουργία λίστας

1. Δημιουργία κενής λίστας

```
l1=[]
```

2. Δημιουργία λίστας δεδομένων

```
l2=["Red","Yellow","Green"]
```

Γιατί να δημιουργήσουμε μια κενή λίστα ;

Ασκήσεις

1. Πώς μπορούμε να δημιουργήσουμε με το όνομα n1 τη λίστα: [5,2,10];

2. Πώς μπορούμε να δημιουργήσουμε με το όνομα n2 τη λίστα: [0,1,2,3,4];

3. Πώς μπορούμε να δημιουργήσουμε την παραπάνω λίστα n2 με τη συνάρτηση range ;

4. Πώς μπορούμε να δημιουργήσουμε με το όνομα n3 με τη συνάρτηση range τη λίστα : [1,3,5,7,9];

Ασκήσεις

1. Πώς μπορούμε να δημιουργήσουμε με το όνομα n1 τη λίστα: [5,2,10];

   n1=[5,2,10]

2. Πώς μπορούμε να δημιουργήσουμε με το όνομα n2 τη λίστα: [0,1,2,3,4];

   n2=[0,1,2,3,4]

3. Πώς μπορούμε να δημιουργήσουμε την παραπάνω λίστα n2 με τη συνάρτηση range ;

   n2=range(5) ή n2=range(0,5,1)

4. Πώς μπορούμε να δημιουργήσουμε με το όνομα n3 με τη συνάρτηση range τη λίστα : [1,3,5,7,9];

   n3=range(1,10,2)

Τεστ 1: Διαφάνειες 2-7 (Βασικές έννοιες και δημιουργία λιστών)

1. **Τι τύπους δεδομένων μπορεί να περιέχει μια λίστα στην Python 2;**
   a) Μόνο ακέραιους αριθμούς
   b) Μόνο αριθμητικά δεδομένα (int, float)
   c) Δεδομένα διαφορετικών τύπων (int, float, string, boolean)
   d) Μόνο συμβολοσειρές

2. **Πώς δηλώνουμε μια κενή λίστα στην Python 2;**
   a) `list = {}`
   b) `list = ()`
   c) `list = []`
   d) `list = None`

3. **Τι θα επιστρέψει η `range(1,10,2)` στην Python 2;**
   a) `[1, 10, 2]`
   b) `[1, 3, 5, 7, 9]`
   c) `(1, 3, 5, 7, 9)`
   d) `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`

Τεστ 1: Διαφάνειες 2-7 (Βασικές έννοιες και δημιουργία λιστών)

4. **Ποιο είναι το αποτέλεσμα του `range(5)` στην Python 2;**
   a) `[0, 1, 2, 3, 4]`
   b) `(0, 1, 2, 3, 4)`
   c) `{0, 1, 2, 3, 4}`
   d) `[1, 2, 3, 4, 5]`

5. **Γιατί να δημιουργήσουμε μια κενή λίστα πριν τη χρήση της;**
   a) Για να προκαλέσουμε σφάλμα
   b) Για να εισάγουμε δεδομένα αργότερα
   c) Για να βελτιώσουμε την ταχύτητα εκτέλεσης του προγράμματος
   d) Για να έχουμε πιο αναγνώσιμο κώδικα

Δείκτες (indexing)

0	1	2	3	4
List_1=[΄Πορτοκάλι΄, ΄Μήλο΄, ΄Μπανάνα΄, ΄Αχλάδι΄, ΄Ροδάκινο΄]				
-5	-4	-3	-2	-1

Αφού δημιουργήσετε την προηγούμενη λίστα, τρέξτε τις εντολές:

```
print List_1[2]

print List_1[0]

print List_1[-1]

print List_1
```

Τι θα συμβεί αν εκτελέσουμε τις εντολές :

```
print List_1[10]
print List_1[-8]
```

Αριθμητικές Πράξεις με λίστες

1. Πρόσθεση Λιστών

```
>>> l1=[5,3,1]
>>> l2=[2,4,6]
>>> l3=l1+l2
>>> print l3
[5, 3, 1, 2, 4, 6]
```

2. Πολλαπλασιασμός λίστας με θετικό ακέραιο

```
>>> l1=['Monday','Tuesday']
>>> l2=l1*3
>>> print l2
['Monday', 'Tuesday', 'Monday', 'Tuesday', 'Monday', 'Tuesday']
```

Στις δύο παραπάνω πράξεις ισχύει η αντιμεταθετική ιδιότητα ;
Τι θα εμφανίσει η εντολή: print l1*0 ;

Slicing (διαμέριση) No 1

```
>>> List_1=['Orange','Apple','Banana','Pear','Peach','Grapes']
>>> List_2=List_1[1:4]
>>> print List_2
['Apple', 'Banana', 'Pear']
```

Το slicing (διαμέριση) μιας λίστας δημιουργεί μια νέα λίστα, παρόμοια με τη δημιουργία λίστας με τη συνάρτηση range. Η μόνη διαφορά είναι πως ο τελεστής της διαμέρισης είναι **:** η άνω και κάτω τελεία.

Το **List_1[a:b]** δημιουργεί μια νέα λίστα από τα δεδομένα της λίστας List_1,
ξεκινώντας από το στοιχείο με δείκτη a: List_1[a]
μέχρι το στοιχείο με δείκτη b, χωρίς όμως το στοιχείο List_1[b].

```
>>> List_1=['Orange','Apple','Banana','Pear','Peach','Grapes']
>>> List_2=List_1[0:5:2]
>>> print List_2
['Orange', 'Banana', 'Peach']
```

Ποιο είναι το αποτέλεσμα του παρακάτω κώδικα;

```
List_1 = ['Orange', 'Apple', 'Banana', 'Pear', 'Peach', 'Grapes']
print List_1[0:10]
print List_1[-1:-10:-2]
```

Slicing (διαμέριση) No 2

Γενικός Κανόνας

List_1[a:b:c]

Δημιουργεί μια λίστα από την List_1 με το πρώτο στοιχείο να βρίσκεται στη θέση a,
με βήμα c
και με όλες τις τιμές μέχρι αυτές που βρίσκονται μία θέση πριν το στοιχείο της θέσης b

- a είναι η αρχική θέση (συμπεριλαμβάνεται)
- b είναι η τελική θέση (δεν συμπεριλαμβάνεται)
- c είναι το βήμα (όταν παραλείπεται, θεωρείται 1, Προσοχή: δεν μπορεί να είναι 0)

Ποιο είναι το αποτέλεσμα του παρακάτω κώδικα;

```
List_1 = ['Orange', 'Apple', 'Banana', 'Pear', 'Peach', 'Grapes']
print List_1[0:5]
print List_1[-1:5:2]
print List_1[-1:3:-1]
```

Slicing (διαμέριση) No 3

Βήμα c>0

Όταν το c παραλείπεται θεωρείστε πως c=1

Τι συμβαίνει όταν παραλείπουμε το a ή το b σε μια διαμέριση λίστας List_1[a:b:c] ;

```
>>> List_1=['Orange', 'Apple', 'Banana', 'Pear', 'Peach', 'Grapes']
>>> List_2=List_1[:3]
>>> print List_2
['Orange', 'Apple', 'Banana']
```

- Όταν παραλείπουμε το a τότε η διαμέριση ξεκινάει από την αρχή της λίστας (αν το c είναι θετικό) ή από το τέλος (αν το c είναι αρνητικό)
- Όταν παραλείπουμε το b τότε η διαμέριση σταματάει στο τέλος της λίστας (αν το c είναι θετικό) ή στην αρχή (αν το c είναι αρνητικό)

```
>>> List_1=['Orange', 'Apple', 'Banana', 'Pear', 'Peach', 'Grapes']
>>> List_2=List_1[1:]
>>> print List_2
['Apple', 'Banana', 'Pear', 'Peach', 'Grapes']
```

```
>>> List_3=List_1[:5:2]
>>> print List_3
['Orange', 'Banana', 'Peach']
```

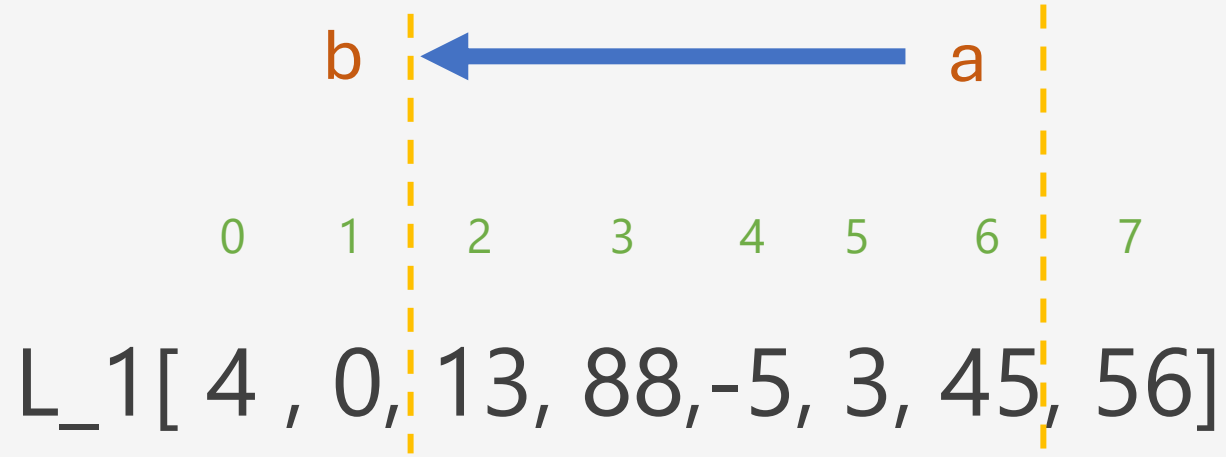
Ποιο είναι το αποτέλεσμα του παρακάτω κώδικα;

```
List_1 = ['Orange', 'Apple', 'Banana', 'Pear', 'Peach', 'Grapes']
List_2=List_1[::]
print List_2
```

Slicing (διαμέριση) No 4

Βήμα c<0

Τι συμβαίνει όταν το βήμα είναι αρνητικό c<0 σε μια διαμέριση λίστας List_1[a:b:c] ;



L_1[6:1:-1] → [45,3,-5,88,13] (το στοιχείο στη θέση b δεν περιλαμβάνεται)

a<b και c<0 η λίστα που δημιουργείται είναι κενή [ ]

Slicing (διαμέριση) No 5

Ότι ισχύει στη διαμέριση (slicing) των λιστών, το ίδιο ισχύει και στη δομή δεδομένων string.

```
#Φτιάξε ένα πρόγραμμα
#που να διαβάζει ένα string
#και να εμφανίζει αν είναι παλίνδρομο
#δηλαδή αν διαβάζεται το ίδιο
#από τα αριστερά προς τα δεξιά
#και από τα δεξιά προς τα αριστερά
s=raw_input("Δώσε μου μία φράση να ελέγξω αν είναι παλίνδρομη: ")
if s==s[::-1]:
      print "Είναι παλίνδρομη"
else:
      print "Δεν είναι παλίνδρομη"
```

```
Δώσε μου μία φράση να ελέγξω αν είναι παλίνδρομη: ΝΙΨΟΝΑΝΟΜΗΜΑΤΑΜΗΜΟΝΑΝΟΨΙΝ
Είναι παλίνδρομη
```

Οτιδήποτε ισχύει στη διαμέριση (slicing) για τις λίστες, ισχύει και για τις συμβολοσειρές (string).

Τεστ 2: Διαφάνειες 10-14 (Δείκτες, αριθμητικές πράξεις, slicing)

1. **Τι θα εκτυπώσει η `print List_1[2]` αν `List_1 = ['Πορτοκάλι', 'Μήλο', 'Μπανάνα', 'Αχλάδι', 'Ροδάκινο']`;**
   a) `'Μήλο'`
   b) `'Μπανάνα'`
   c) `'Αχλάδι'`
   d) `'Πορτοκάλι'`

2. **Τι θα συμβεί αν επιχειρήσουμε `print List_1[10]` όταν η λίστα έχει 5 στοιχεία;**
   a) Θα εμφανιστεί `'None'`
   b) Θα εμφανιστεί `' '` (κενή συμβολοσειρά)
   c) Θα προκληθεί `IndexError`
   d) Θα επιστρέψει το τελευταίο στοιχείο

3. **Ποιο είναι το αποτέλεσμα του `List_1 + List_2` αν και οι δύο μεταβλητές είναι λίστες;**
   a) Προσθέτει αριθμητικά τα αντίστοιχα στοιχεία των λιστών
   b) Συνενώνει τις δύο λίστες σε μία
   c) Πολλαπλασιάζει τα στοιχεία των λιστών
   d) Δημιουργεί μια νέα λίστα με κενά στοιχεία

Τεστ 2: Διαφάνειες 10-14 (Δείκτες, αριθμητικές πράξεις, slicing)

4. **Τι κάνει το `List_1[1:4]` όταν `List_1 = ['A', 'B', 'C', 'D', 'E'];`**
   a) `['A', 'B', 'C']`
   b) `['B', 'C', 'D']`
   c) `['B', 'C', 'D', 'E']`
   d) `['A', 'B', 'C', 'D', 'E']`

5. **Τι θα επιστρέψει η `print List_1[-1:-10:-2]` αν `List_1 = ['A', 'B', 'C', 'D', 'E', 'F'];`**
   a) `['F', 'D', 'B']`
   b) `['E', 'C', 'A']`
   c) `['F', 'D']`
   d) Σφάλμα εκτέλεσης

Μέθοδοι append, insert, pop – Τροποποίηση λίστας

Μια λίστα, που έχει ήδη δημιουργηθεί, μπορεί να τροποποιηθεί.
Οι μέθοδοι που θα μάθουμε είναι:

- append προσθέτει ένα στοιχείο στο τέλος της λίστας
- insert προσθέτει ένα στοιχείο σε μια επιλεγμένη θέση της λίστας
- pop αφαιρεί ένα στοιχείο της λίστας

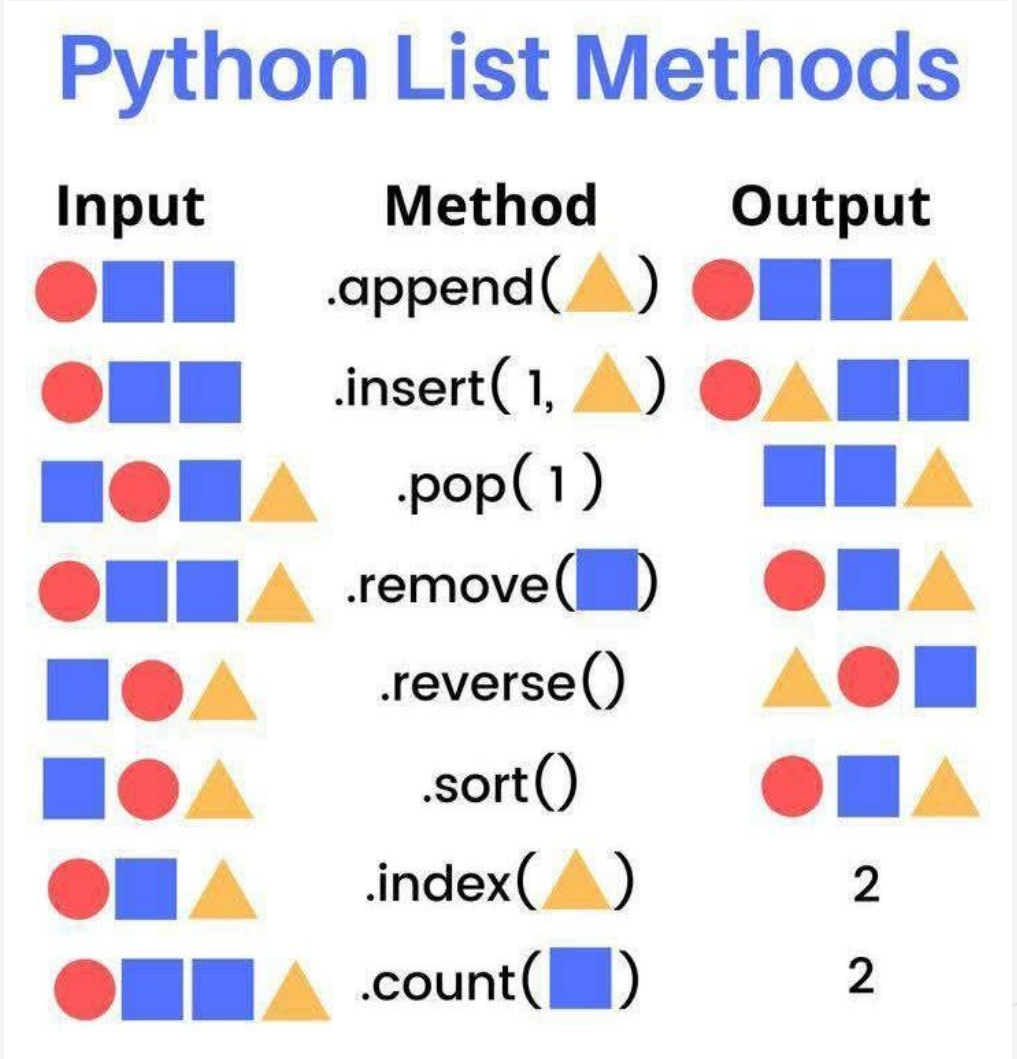
Υπάρχουν και άλλες μέθοδοι για την τροποποίηση λίστας.
Εμείς θα μάθουμε μόνο αυτές τις τρεις.

Μία χρήσιμη συνάρτηση για τη διαχείριση μια λίστας είναι η len(), η οποία επιστρέφει το μέγεθος της λίστας.

```
>>> List_1=[1,4,7,2]
>>> a=len(List_1)
>>> print a
4
```

Γίνεται μια λίστα να περιέχει δεδομένα διαφορετικού τύπου;
π.χ.
List_1 = ['Orange', 55, True, 33.4]

Μέθοδοι σε λίστες στην Python 3 (γιατί οι γλώσσες προγραμματισμού εξελίσσονται)



Python List Methods

Input	Method	Output
[10, 20, 30]	.append(40)	[10, 20, 30, 40]
[10, 20, 30]	.extend([40])	[10, 20, 30, 40]
[10, 20, 30]	.insert(2, 40)	[10, 20, 40, 30]
[10, 20, 30, 20]	.count(20)	2
[10, 20, 30]	.clear()	[ ]
[10, 20, 30, 40]	.index(40)	3
[10, 20, 30, 40]	.remove(10)	[20, 30, 40]
[10, 20, 30, 40]	.pop(2)	[10, 20, 40]
[30, 20, 10]	.reverse()	[10, 20, 30]
[30, 10, 40, 20]	.sort()	[10, 20, 30, 40]
[10, 20, 30]	.copy()	[10, 20, 30]

Μέθοδοι σε λίστες στην Python 3 (No 2)

List Functions

- append(): Adds an element to the end of the list.
- extend(): Adds elements from another list to the end of the list.
- insert(): Inserts an element at a specified position.
- remove(): Removes the first occurrence of an element.
- pop(): Removes and returns an element at a specified index (default is last).
- clear(): Removes all elements from the list.
- index(): Returns the index of the first occurrence of an element.
- count(): Returns the number of occurrences of a specified element.
- sort(): Sorts the list in ascending order (or by a specified key).
- reverse(): Reverses the order of the list.
- copy(): Returns a shallow copy of the list.

Μέθοδος append

προσθέτει ένα στοιχείο στο τέλος της λίστας

[10, 20, 30] .append(40) [10, 20, 30, 40]

```
>>> list_1=[15,13,16,12,18]
>>> list_1.append(19)
>>> print list_1
[15, 13, 16, 12, 18, 19]
```

```
>>> list_1=[15,13,16,12,18]
>>> a=input()
17
>>> list_1.append(a)
>>> print list_1
[15, 13, 16, 12, 18, 17]
```

Μέθοδος append No 2

```
#Φτιάξε ένα πρόγραμμα
#που να διαβάζει 10 αριθμούς
#και να βρίσκει πόσοι από αυτούς
#είναι μεγαλύτεροι από τον μέσο όρο
list_1=[]
sum=0
for i in range(10):
    x=input()
    list_1.append(x)
    sum+=x
mo=sum/10.0
count=0
for i in range(10):
    if list_1[i]>mo:
        count+=1
print 'Είναι μεγαλύτεροι από τον μέσο όρο: ',count
```

```
14
15
17
13
12
18
19
15
16
15
Είναι μεγαλύτεροι από τον μέσο όρο:  4
```

Είναι ένα παράδειγμα που χρειάζεται: 1) να αποθηκεύσουμε όλα τα δεδομένα μας 2) να προσπελάσουμε όλα τα δεδομένα μας

Γι' αυτόν τον σκοπό χρειαζόμαστε Δομές Δεδομένων (Data Structures) στον Προγραμματισμό.

Μέθοδος insert

προσθέτει ένα στοιχείο σε μια επιλεγμένη θέση της λίστας

[10, 20, 30] .insert(2, 40) [10, 20, 40, 30]

```
>>> list_1=[1,3,5,7,9]
>>> list_1.insert(2,4)
>>> print list_1
[1, 3, 4, 5, 7, 9]
```

Η εντολή:

list_1.insert(2,4)

εισαγάγει στη θέση [2] της λίστας list_1

την τιμή: 4

Μέθοδος insert No 2

```
#Διάβασε ένα αριθμό και εισήγαγέ τον
#σε συγκεκριμένη λίστα ταξινομημένων αριθμών
#ώστε η τελική λίστα που προκύπτει να είναι ταξινομημένη
list_1=[1,4,6,9,12,14,17,18]
print list_1
x=input('Δώσε έναν αριθμό: ')
length=len(list_1)
ind=0
for i in range(length):
    if x>list_1[i]:
        ind=i+1  #αφού το x>list[i] το τοποθετώ στην επόμενη θέση
list_1.insert(ind,x)
print 'Η νέα ταξινομημένη λίστα είναι: ',list_1
```

1η λύση με δομή for

```
[1, 4, 6, 9, 12, 14, 17, 18]
Δώσε έναν αριθμό: 10
Η νέα ταξινομημένη λίστα είναι:  [1, 4, 6, 9, 10, 12, 14, 17, 18]
```

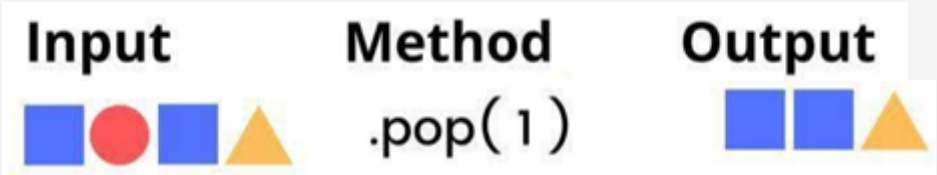
Μέθοδος insert No 3

```
#Διάβασε ένα αριθμό και εισήγαγέ τον
#σε συγκεκριμένη λίστα ταξινομημένων αριθμών
#ώστε η τελική λίστα που προκύπτει να είναι ταξινομημένη
list_1=[1,4,6,9,12,14,17,18]
print list_1
x=input('Δώσε έναν αριθμό: ')
length=len(list_1)
ind=0
found=False
while found ==False and ind<length:
    if x>list_1[ind]:
        ind+=1
    else:
        found=True
list_1.insert(ind,x)
print 'Η νέα ταξινομημένη λίστα είναι: ',list_1
```

2η λύση με δομή while

```
[1, 4, 6, 9, 12, 14, 17, 18]
Δώσε έναν αριθμό: 10
Η νέα ταξινομημένη λίστα είναι:  [1, 4, 6, 9, 10, 12, 14, 17, 18]
```

Μέθοδος pop



[10, 20, 30, 40] .pop(2) [10, 20, 40]

```
>>> list_1=[1,6,3,7,5,9]
>>> list_1.pop(4)
5
>>> print list_1
[1, 6, 3, 7, 9]
```

```
>>> print list_1
[1, 6, 3, 7]
>>> list_1.pop(-2)
3
>>> print list_1
[1, 6, 7]
```

.pop(i)

Αφαιρεί το στοιχείο της λίστας με δείκτη i

```
>>> print list_1
[1, 6, 3, 7, 9]
>>> list_1.pop()
9
>>> print list_1
[1, 6, 3, 7]
```

.pop( )

Αφαιρεί το τελευταίο στοιχείο της λίστας

Ο δείκτης σε μια μέθοδο μπορεί να είναι μηδέν (πρώτο στοιχείο), θετικός ή αρνητικός (μέτρηση ανάποδα)

Τεστ 3: Διαφάνειες 17-23 (Μέθοδοι λιστών)

1. **Ποια μέθοδος προσθέτει αυτόματα ένα στοιχείο στο τέλος της λίστας;**
   a) `insert()`
   b) `append()`
   c) `extend()`
   d) `push()`

2. **Ποιο από τα παρακάτω αφαιρεί το τελευταίο στοιχείο της λίστας;**
   a) `pop()`
   b) `remove()`
   c) `del list[-1]`
   d) `clear()`

3. **Τι επιστρέφει η `len(list_1);`**
   a) Το άθροισμα των αριθμητικών στοιχείων της λίστας
   b) Τον αριθμό των στοιχείων της λίστας
   c) Τον δείκτη του τελευταίου στοιχείου
   d) Μία νέα λίστα με το μέγεθος των στοιχείων

Τεστ 3: Διαφάνειες 17-23 (Μέθοδοι λιστών)

4. **Πώς λειτουργεί η μέθοδος pop();**
   a) Διαγράφει το πρώτο στοιχείο της λίστας
   b) Διαγράφει το τελευταίο στοιχείο της λίστας
   c) Διαγράφει όλα τα στοιχεία της λίστας
   d) Αφού δεν έχει δείκτη, δεν κάνει τίποτα

5. **Ποια από τις παρακάτω εντολές εισάγει το στοιχείο 5 στη θέση 2 της λίστας `list_1`;**
   a) `list_1.append(2,5)`
   b) `list_1.insert(2,5)`
   c) `list_1[2] = 5`
   d) `list_1.append(5,2)`

Άσκηση: Φτιάξτε μια συνάρτηση που να δέχεται μία λίστα και να επιστρέφει την αντεστραμμένη της

Λύση No 1

Σκέψη: Η προσπέλαση των στοιχείων της λίστας γίνεται αντίστροφα. Χρήση της μεθόδου append.

```
def rev(l1):
    length=len(l1)
    l2=[]
    for i in range(length-1,-1,-1):
        l2.append(l1[i])
    return l2

li=[1,5,3,6,4,7,43]
print li
print rev(li)
```

```
[1, 5, 3, 6, 4, 7, 43]
[43, 7, 4, 6, 3, 5, 1]
```

Γιατί στην 4η γραμμή η πρώτη παράμετρος του range είναι το length-1 ;

Άσκηση: Μια δοσμένη λίστα ακεραίων [1,5,2,-15,3,7] να την μετατρέψετε σε λίστα πραγματικών αριθμών.

```
L_1=[1,5,2,-15,3,7]
print L_1
for i in range(len(L_1)):
    L_1[i]=float(L_1[i])
print L_1
```

```
#δημιουργία διαφορετικής λίστας
L_1=[1,5,2,-15,3,7]
print L_1
L_2=[]
for i in L_1:
    L_2.append(float(i))
print L_2
```

```
[1, 5, 2, -15, 3, 7]
[1.0, 5.0, 2.0, -15.0, 3.0, 7.0]
```

Άσκηση: Φτιάξτε μια συνάρτηση που να δέχεται μία λίστα και να επιστρέφει την αντεστραμμένη της

Λύση No 2

Σκέψη: Η προσπέλαση των στοιχείων της λίστας γίνεται από την αρχή προς το τέλος.
Κάθε νέο στοιχείο εισαγάγεται στην αρχή της λίστας. Χρήση της μεθόδου insert.

```
def rev(l1):
    l2=[]
    for i in l1:
        l2.insert(0,i)
    return l2

li=[1,5,3,6,4,7,43]
print li
print rev(li)
```

```
[1, 5, 3, 6, 4, 7, 43]
[43, 7, 4, 6, 3, 5, 1]
```

Άσκηση: Φτιάξτε μια συνάρτηση που να δέχεται μία λίστα και να επιστρέφει την αντεστραμμένη της

Λύση No 3

Σκέψη: Χρήση της διαμέρισης (slicing) της λίστας
Δεν χρησιμοποιείται μέθοδος.

```
def rev(l1):
    l2=l1[::-1]
    return l2

li=[1,5,3,6,4,7,43]
print li
print rev(li)
```

```
[1, 5, 3, 6, 4, 7, 43]
[43, 7, 4, 6, 3, 5, 1]
```

Σύγκριση των λύσεων της άσκησης

Άσκηση: Φτιάξτε μια συνάρτηση που να δέχεται μία λίστα και να επιστρέφει την αντεστραμμένη της

Λύση No 1

```
def rev(l1):
    length=len(l1)
    l2=[]
    for i in range(length-1,-1,-1):
        l2.append(l1[i])
    return l2

li=[1,5,3,6,4,7,43]
print li
print rev(li)
```

Λύση No 2

```
def rev(l1):
    l2=[]
    for i in l1:
        l2.insert(0,i)
    return l2

li=[1,5,3,6,4,7,43]
print li
print rev(li)
```

Λύση No 3

```
def rev(l1):
    l2=l1[::-1]
    return l2

li=[1,5,3,6,4,7,43]
print li
print rev(li)
```

```
[1, 5, 3, 6, 4, 7, 43]
[43, 7, 4, 6, 3, 5, 1]
```

Τεστ Πολλαπλής Επιλογής – Λίστες στην Python

1. Τι χαρακτηρίζει μια λίστα στην Python;

a) Είναι αμετάβλητη (immutable)

b) Μπορεί να περιέχει μόνο αριθμούς

c) Είναι μια δομή δεδομένων που αποθηκεύει πολλά δεδομένα οργανωμένα

2. Ποιος από τους παρακάτω τρόπους δημιουργεί μια λίστα με αριθμούς από το 1 έως το 5;

a) `numbers = [1,2,3,4,5]`

b) `numbers = range(1,5)`

c) `numbers = {1,2,3,4,5}`

3. Ποια από τις παρακάτω εντολές επιστρέφει το πρώτο στοιχείο της λίστας `my_list`;

a) `my_list[1]`

b) `my_list[-1]`

c) `my_list[0]`

4. Ποιο θα είναι το αποτέλεσμα της `print my_list[10]` αν `my_list` έχει 5 στοιχεία;

a) Θα εμφανιστεί `None`

b) Θα προκληθεί `IndexError`

c) Θα εμφανιστεί το τελευταίο στοιχείο

Τεστ Πολλαπλής Επιλογής – Λίστες στην Python

5. Ποιο από τα παρακάτω ΔΕΝ είναι μέθοδος λίστας στην Python;

a) `append()`

b) `insert()`

c) `removeAll()`

6. Τι επιστρέφει η `len(my_list)` σε μια λίστα;

a) Το άθροισμα των αριθμητικών στοιχείων της λίστας

b) Τον αριθμό των στοιχείων της λίστας

c) Τον δείκτη του τελευταίου στοιχείου

7. Ποιο από τα παρακάτω θα επιστρέψει `['B', 'C', 'D']` αν `my_list = ['A', 'B', 'C', 'D', 'E']`?

a) `my_list[1:4]`

b) `my_list[0:3]`

c) `my_list[2:5]`

8. Ποιο θα είναι το αποτέλεσμα του `print my_list[-1:-10:-2]` αν `my_list = ['A', 'B', 'C', 'D', 'E', 'F']`?

a) `['F', 'D', 'B']`

b) `['E', 'C', 'A']`

c) `['F', 'D']`

Τεστ Πολλαπλής Επιλογής – Λίστες στην Python

9. Ποια εντολή εισάγει το στοιχείο `5` στη θέση `2` της λίστας `my_list`;

a) `my_list.append(2,5)`

b) `my_list.insert(2,5)`

c) `my_list[2] = 5`

10. Τι επιστρέφει η `my_list.pop()` αν δεν δοθεί δείκτης;

a) Διαγράφει το πρώτο στοιχείο της λίστας

b) Διαγράφει το τελευταίο στοιχείο της λίστας

c) Δεν κάνει τίποτα

11. Πώς μπορούμε να προσθέσουμε το στοιχείο `10` στο τέλος της λίστας `my_list`;

a) `my_list.add(10)`

b) `my_list.insert(-1, 10)`

c) `my_list.append(10)`

12. Τι θα εμφανίσει η `print my_list[1:4]` αν `my_list = [10, 20, 30, 40, 50]`;

a) `[20, 30, 40]`

b) `[10, 20, 30]`

c) `[30, 40, 50]`

Τεστ Πολλαπλής Επιλογής – Λίστες στην Python

13. Ποια εντολή αφαιρεί το στοιχείο στη θέση `2` της λίστας `my_list`

a) `my_list.pop(2)`

b) `my_list.remove(2)`

c) `my_list.delete(2)`

14. Ποια εντολή αντιστρέφει τα στοιχεία της λίστας `my_list`;

a) `my_list.revese()`

b) `my_list[::-1]`

c) `my_list. append()`

15. Τι θα εμφανίσει η `print my_list[-1]` αν `my_list = ['a', 'b', 'c', 'd']`;

a) `a`

b) `d`

c) `c`

16. Ποια από τις παρακάτω εντολές προσθέτει το στοιχείο `10` στη δεύτερη θέση (θέση 1) της λίστας `my_list`;

a) `my_list.append(1, 10)`

b) `my_list.insert(1, 10)`

c) `my_list.pop(1, 10)`

Τεστ Πολλαπλής Επιλογής – Λίστες στην Python

17. Ποια από τις παρακάτω εντολές αφαιρεί το δεύτερο στοιχείο (θέση 1) της λίστας `my_list` και το επιστρέφει;

a) `my_list.append(1)`
b) `my_list.insert(1)`
c) `my_list.pop(1)`

18. Τι θα επιστρέψει η `print len my_list` αν `my_list = [10, 20, 30, 40, 50]`;

a) `4`
b) `5`
c) `6`

19. Ποια από τις παρακάτω εντολές δημιουργεί μια κενή λίστα;

a) `my_list = {}`
b) `my_list = ()`
c) `my_list = []`

20. Ποιο είναι το αποτέλεσμα της `print [1, 2, 3] * 2`;

a) `[1, 2, 3, 1, 2, 3]`
b) `[1, 2, 3, 6]`
c) `[2, 4, 6]`

Λίστες στην Python

Υλοποίηση μιας ιδέας από τη ζωή

ΧΡΗΣΤΟΣ ΜΑΒΟΓΛΟΥ