Διάλεξη 5η

Λίστες

Δεδομένα

Δεδομένα είναι τα στοιχεία από τα οποία μπορούν να προκύψουν πληροφορίες. Ένα πρόγραμμα επεξεργάζεται δεδομένα και παράγει πληροφορίες.

- Ο τρόπος οργάνωσης των δεδομένων είναι από τα βασικότερα στοιχεία του προγράμματος.
- Η εξέλιξη στον τομέα της ανάπτυξης λογισμικού έδωσε τη δυνατότητα να ενσωματώνουμε στα δεδομένα και τις μεθόδους επεξεργασίας τους.
- Αυτή η τεχνική ονομάζεται αντικειμενοστρεφής προγραμματισμός.

Πίνακας

- Μία από τις βασικές λειτουργίες ενός προγράμματος είναι ο χειρισμός των δεδομένων , η ανάκτηση, η επεξεργασία και η αποθήκευσή τους.

  Σε προηγούμενα κεφάλαια έγινε παρουσίαση αποθήκευσης και τρόπων ανάκτησης δεδομένων σε μεταβλητές.

  Όσο όμως η ποσότητα των δεδομένων μεγαλώνει, η διαχείρισή τους γίνεται δυσκολότερη.
- Η ανάγκη αυτή δημιούργησε σε όλες τις γλώσσες προγραμματισμού μια πιο σύνθετη δομή που ονομάζεται **πίνακας** (array).

Λίστα αντί πίνακα

- Στην Python δομή πίνακα με τη σημασία που έχει σε άλλες γλώσσες δεν υπάρχει.
- Η βασική δομή της Python που μοιάζει με τον πίνακα, είναι η δομή της **λίστας**.
- Η λίστα είναι **μια ακολουθία στοιχείων**, και τα στοιχεία **αποθηκεύονται σε θέσεις μνήμης** από την ίδια τη γλώσσα, χωρίς να απαιτείται ιδιαίτερη προγραμματιστική τεχνική.
- Οι λίστες είναι από τις πιο σημαντικές δομές δεδομένων.

Λίστα

- Μία λίστα είναι μια διατεταγμένη ακολουθία αντικειμένων.
- Τα αντικείμενα μπορούν να είναι οποιουδήποτε τύπου. Δεν χρειάζεται να είναι όλα του ιδίου τύπου.
- Το μήκος της λίστας δεν είναι προκαθορισμένο
- Είναι δυναμική, το μέγεθός της μεταβάλλεται κατά τη διάρκεια εκτέλεσης ενός προγράμματος.
- **Διαθέτει σειρά από ενσωματωμένες λειτουργίες – μεθόδους**.

Παραδείγματα από Λίστες:

- >>>list1= [1,1.3,2,2,2,4,5,-7,0,-3]
- >>>list2=[“kozani”,2024,” ΠΔΜ”]
- >>>list3=[“yes”,[1,2,”no”]]

```
>>> type(list1),type(list2),type(list3)
(<class 'list'>, <class 'list'>, <class 'list'>)
```

Δημιουργία νέας Λίστας

Υπάρχουν πολλοί τρόποι για να δημιουργηθεί μία λίστα. Το κυριότερο όμως, είναι ότι **μπορεί να γίνει σε οποιοδήποτε σημείο του κώδικα και όχι απαραίτητα στην αρχή.**

1ος ΤΡΟΠΟΣ:

- Για τη δημιουργία μίας νέας λίστας χρησιμοποιούμε το όνομα της λίστας, το σύμβολο = και τις αγκύλες [ ] για αρχή και τέλος της λίστας. >>>list1=[1,2]
- Τα στοιχεία της λίστας χωρίζονται μεταξύ τους με κόμμα , ενώ τα αλφαριθμητικά στοιχεία μπαίνουν σε εισαγωγικά. >>>list2=[‘k’,’p’,’3,4,3,’k’]
- Μία λίστα μπορεί να είναι κενή >>>list3=[]
- Μία λίστα μπορεί να περιλαμβάνει και άλλη λίστα (θα δούμε ένθετες λίστες). list4=[1,2,’k’,[‘k’,2],1,’p’]

2ος ΤΡΟΠΟΣ:

Με την εντολή list(a), σε μία ακολουθία (a)

>>> a=(1,2,'d',4,5,[1,2],'o') και >>> a

(1, 2, 'd', 4, 5, [1, 2], 'o')

list(a) μας δίνει [1, 2, 'd', 4, 5, [1, 2], 'o']

Δημιουργία – Εμφάνιση νέας Λίστας

3ος ΤΡΟΠΟΣ:
Μία λίστα μπορεί να δημιουργηθεί μέσω μιας επαναληπτικής διαδικασίας.

```
>>> x=[]
>>> for i in range(10):
        x+=[i]
```

Εμφάνιση Λίστας

- Για την εμφάνιση των στοιχείων μιας λίστας [x], μπορεί απλά να δοθεί το όνομά της στην κονσόλα της γλώσσας (Idle)

```
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]      ή >>> [x]

[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]
```

- ή μέσα σε κάποιο σενάριο, (κώδικα), μέσω της εντολής print

```
k=[10,-5,0,5,10]
print(k)
[10,-5,0,5,10]
```

Δημιουργία – Εμφάνιση νέας Λίστας

- Για την εμφάνιση πολλών μεμονωμένων στοιχείων, μπορεί να γίνει χρήση της επανάληψης for.

  - k=[-10,-5,0,5,10]
    for **item** in k:
        print(item)

```
-10
-5
0
5
10
```

  - ενώ print(item,end=' ')

```
10 -5 0 5 10
```

  - ή k=[-10,-5,0,5,10],
    for **item** in k:
        print(item)

```
[-10, -5, 0, 5, 10]
```

Προσπέλαση Λίστας

Από τη στιγμή που θα δημιουργηθεί μια λίστα, **κάθε στοιχείο της είναι διαθέσιμο** μέσω της θέσης που έχει στη λίστα. Η αρίθμηση μιας λίστας ξεκινάει από το **0**.

- Αν μια λίστα περιέχει **n** στοιχεία, το 1ο βρίσκεται στη θέση 0, ενώ το τελευταίο στη , θέση **n-1**.
- Για πρόσβαση σε ένα στοιχείο θα πρέπει να χρησιμοποιηθεί **δείκτης** που θα αναφέρει τη θέση.

```
>>> x=[3,'k',0,2,'p',3]
>>> x
    [3, 'k', 0, 2, 'p', 3]
```

ΘΕΣΕΙΣ	**0**	**1**	**2**	**3**	**4**	**5**
Λίστα **x**	3	K	0	2	P	3
Αρνητική αρίθμηση	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

```
>>> x[5]   (=x[-1])
3
>>> x[2]   (=x[-4])
0
>>> x[0]   (=x[-6])
3
>>> x[1]   (=x[-5])
'k'
```

- Η προσπέλαση στα στοιχεία μιας λίστας μπορεί να γίνει δίνοντας αρνητικές τιμές στον δείκτη.
- Σε αυτή την περίπτωση η καταμέτρηση **ξεκινάει** από το **-1** και αναφέρεται στο **τελευταίο στοιχείο της λίστας**

```
>>> x[-1]
3
>>> x[-2]
'p'
>>> x[-6]
3
>>> x[0]
3
```

Εδώ βλέπουμε ότι το στοιχείο x[-6] με αρνητική δεικτοδότηση, είναι το ίδιο με το στοιχείο x[0] = 3.

Γενικά το 1ο στοιχείο μιας λίστας k είναι το κ[0], ενώ τελευταίο το κ[-1]

Προσπέλαση Λίστας

- Προσπέλαση με δεικτοδότηση
  - >>> x=['a','b','c']
    >>> for i in range(3):
    print(x[i])

    a
    b
    c
- Προσπέλαση με τη χρήση της Len
  - >>>clubs=['ΠΑΟΚ','ΟΦΙ','ΓΙΟΥΒΕΝΤΟΥΣ']
    >>> for i in range(len(clubs)):
    print(clubs[i],end=' ')

    [ΠΑΟΚ ΟΦΙ ΓΟΥΒΕΝΤΟΥΣ]
- Αντίστροφη προσπέλαση

  >>>clubs=['ΠΑΟΚ','ΟΦΙ','ΓΙΟΥΒΕΝΤΟΥΣ']

  >>>for i in range(-1,-len(clubs)-1,-1):
  print(clubs[i],end=' ')

ΓΟΥΒΕΝΤΟΥΣ ΟΦΙ ΠΑΟΚ

Η range σε αυτήν την περίπτωση θα παράγει τιμές από το -1 έως και το -3

Το ίδιο αποτέλεσμα θα είχαν και οι επόμενες εντολές:(Κατάτμηση)

>>> **clubs[::-1] #(:=όλα)**

['ΓΙΟΥΒΕΝΤΟΥΣ', 'ΟΦΙ', 'ΠΑΟΚ']

>>> **print(clubs[::-1])**

['ΓΙΟΥΒΕΝΤΟΥΣ', 'ΟΦΙ', 'ΠΑΟΚ']

Τεμάχια

Ένα τεμάχιο μιας λίστας, είναι ένα μέρος της λίστας με άνω και κάτω τελεία.

List1[m:n]	Από το στοιχείο με δείκτη m μέχρι το n-1
List[:]	Όλα τα στοιχεία της λίστας ή αλλιώς List
List[m:]	Από το στοιχείο με δείκτη m έως το τέλος της λίστας
List[:m]	Από την αρχή της λίστας έως το στοιχείο με δείκτη m-1
ΠΑΡΑΔΕΙΓΜΑΤΑ LIST1=[‘a’,’b’,’c’,’d’,’e’,’f’]	
List1[1:3]	['b', 'c']
List1[-4:-2]	['c', 'd']
List1[:4]	['a', 'b', 'c', 'd']
List1[:]	['a', 'b', 'c', 'd', 'e', 'f']
del List1[1:3]	['a', 'd', 'e', 'f']
List1[1:3][1]	'c'

ΠΡΑΞΕΙΣ ΜΕΤΑΞΥ ΛΙΣΤΩΝ

Ισχύουν οι πράξεις που ισχύουν σε συμβολοσειρές.

- Πρόσθεση ανάμεσα σε λίστες

```
>>>list1=['Sun',16,2.7]
>>>list2=[7,8,9]
>>>list=list1+list2
>>>list
['Sun',16,2.7,7,8,9]
```

- Μία λίστα μπορεί να επαναλαμβάνει τα στοιχεία της

```
>>>list1=['Sun',16,2.7]2
>>>list1*2
['Sun',16,2.7,'Sun',16,2.7]
```

- Αναφορά σε στοιχεία της λίστας

```
>>>list1[0]
'Sun'
>>>list1[2]
2.7
>>>
```

Επεξεργασία λίστας – αλλαγή προσήμου (1ος τρόπος)

Παράδειγμα 1ο: Δίδεται λίστα που περιέχει αριθμούς. Να γίνει αλλαγή του πρόσημου όπου χρειάζεται, ώστε η λίστα να περιέχει μόνο θετικούς αριθμούς.

```
numbers=[2,-4,1,-3,4,2,0,-12]
for i in range(len(numbers)):
   numbers[i]=abs(numbers[i])
print(numbers)
```

```
[2, 4, 1, 3, 4, 2, 0, 12]
```

Επεξεργασία λίστας – αλλαγή προσήμου (2ος τρόπος)

```
numbers=[2,-4,1,-3,4,2,0,-12]
for i in range(len(numbers)):
   if numbers[i]>=0:
      numbers[i]=numbers[i]
   else:
      numbers[i]=-numbers[i]
print(numbers)
```

```
[2, 4, 1, 3, 4, 2, 0, 12]
```

Εισαγωγή στοιχείων σε λίστα

Οι μέθοδοι που επιτρέπουν την εισαγωγή στοιχείων σε μία λιστα είναι οι **append,insert και extend.**

- Η μέθοδος append

Με τη μέθοδο αυτή εισάγουμε νέα στοιχεία σε μια λίστα, αυξάνοντας ταυτόχρονα το μέγεθός της.

Γενική μορφή:

λίστα.append(αντικείμενο)

Με τη μέθοδο αυτή, εισάγουμε νέο στοιχείο στην τελευταία θέση της λίστας.

```
>>>list1=[1,3,5]
>>>list1.append(7)
>>>list1.append(0)
>>>list1
[1,3,5,7,0]
>>>list1.append(['a','b','c'])
>>>list1
[1,3,5,7,0, ['a','b','c']]
>>>list1[5]  #εμφάνιση του στοιχείο τη θέσης 5
['a','b','c']
```

Εισαγωγή στοιχείων σε λίστα

- Η μέθοδος append

Παράδειγμα 2ο: Από μια λίστα βαθμών θα πρέπει να δημιουργηθεί νέα λίστα, που θα περιέχει όσους είναι κάτω από 5.

```
grades=[2,10,9,4,1,6,6,1,0]
k=[]        # ορίζω μία κενή λίστα k
for item in grades:
    if item<5:
        k.append(item)
print(k)
```

```
[2, 4, 1, 1, 0]
```

Παράδειγμα 2b: Σας ζητούνται αριθμοί για να φτιάξετε μία λίστα 5 στοιχείων, η οποία θα εμφανιστεί στην οθόνη

```
a=[]
for i in range(5):
    x=int(input('dose arithmo '))
    a.append(x)
print(a)
```

```
[1, 2, 3, 4, 5]
```

Β Τρόπος (αντίστροφα-οριζόντια):

```
for i in range(4,-1,-1): #αντικαθιστώ την print(a)
    print(a[i],end=' ')     #με αυτές τις δύο γραμμές
```

```
5 4 3 2 1
```

Γ Τρόπος (αντίστροφα - κάθετα):

```
for i in range(4,-1,-1):     #το ίδιο όπως πάνω
    print(a[i])
```

```
5
4
3
2
1
```

Εισαγωγή στοιχείων σε λίστα

- Η μέθοδος insert

  Η μέθοδος αυτή χρησιμοποιείται, όταν η εισαγωγή στοιχείων πρέπει να γίνει σε συγκεκριμένη θέση μέσα σε μία λίστα

Γενική μορφή:

λίστα.insert(δείκτης,αντικείμενο)

```
>>>list1=[1,3,5]
>>>list1.insert(2,'a')
>>>list1
[1,3,'a',5]
>>>list1.insert(2,[0,0])
>>>list1
[1,3,[0,0],'a',5]
```

Εισαγωγή στοιχείων σε λίστα

- Η μέθοδος extend (το ίδιο κάνει ο τελεστής +)

  η μέθοδος extend() χρησιμοποιείται για να προσθέσει τα στοιχεία μιας λίστας, στο τέλος μιας υπάρχουσας λίστας.

Γενική μορφή:

λίστα.extend(αντικείμενο)

```
list1=[1,3,5]
list2=[6,7,8]
list1.extend(list2)
print(list1)
```

[1, 3, 5, 6, 7, 8]

Διαφορές με τη μέθοδο append():

- Η extend() προσθέτει τα στοιχεία στην λίστα.
- Η append() προσθέτει ολόκληρη τη λίστα ως ένα στοιχείο.

```
list1 = [1, 2, 3]
list2 = [4, 5]
list1.append(list2)
```

[1, 3, 5, [6, 7, 8]]

Διαγραφή στοιχείων από λίστα

- Η συνάρτηση del

Η συνάρτηση αυτή διαγράφει εντελώς το αντικείμενο.

>>>a=[1,2,3,4]

>>>del a ή del[a]

διαγράφει εντελώς το αντικείμενο a

ενώ ή

>>> del a[:] ή a[:]=[] ή a=[]

αδειάζει τα περιεχόμενα της λίστας a

>>>a=[]

>>>a

[]

>>>del a[1]

>>>a

[1,3,4]

>>>del a[1:2]

>>>a

[1,4]

Διαγραφή στοιχείων από λίστα

- Η μέθοδος remove

Με τη μέθοδο remove, είναι δυνατή η διαγραφή συγκεκριμένου στοιχείου μιας λίστας με **βάση την τιμή του**.

Γενική μορφή: λίστα.remove(τιμή)

Αν υπάρχει πολλές φορές η τιμή, τότε θα αφαιρεθεί η πρώτη τιμή. Αν δεν υπάρχει, τότε προκαλείται σφάλμα.

```
>>>a=[1,2,3,4]
>>>a.remove(3)
[1,2,4]
>>>a.remove(5)
ValueError:list.remove(x) x not in list
```

Διαγραφή στοιχείων από λίστα

- Η μέθοδος clear (όπως η del)

Από την έκδοση 3.3 και μετά.

Γενική μορφή: λίστα.clear()

```
>>>a=[1,2,3]
>>>del a[:]   ή
>>>a.clear()      ή
>>>a=[]
>>>a
```

(σε όλες τις παραπάνω περιπτώσεις)

[] Αποτέλεσμα

Εξαγωγή στοιχείων από λίστα

- Η μέθοδος pop

Η εξαγωγή στοιχείου από λίστα γίνεται **με βάση το δείκτη** στοιχείου

Στην οθόνη επιστρέφει το στοιχείο που εξήχθη

Γενική μορφή: λίστα.pop(δείκτης)

- Εξαγωγή του τελευταίου στοιχείου της λίστας

```
>>>a=[1,2,3,4]
>>>a.pop()    4
```

(b=a.pop(), το επιστρέφει στη μεταβλητή b)

```
>>>b          4
```

- Εξαγωγή συγκεκριμένου στοιχείου (χρησιμοποιούμε δείκτη θέσης)

```
>>>a=[1,2,3,4]
>>>b=a.pop(2)
>>>b          3
```

Αναδιάταξη λίστας

Μία λίστα μπορεί να τροποποιηθεί όχι μόνο σε επίπεδο μεμονωμένου στοιχείου, αλλά και ολόκληρη. Αυτό μπορεί να γίνει με δύο μεθόδους

- Αντιστροφή λίστας, μέθοδος reverse

```
>>>a=[1,2,3,4]
>>>a.reverse()
>>>a
[4,3,2,1]
```

Όπως θα μπορούσε να γίνει και με εντολή δεικτοδότησης

```
>>>b=a[::-1]
>>>b          [4,3,2,1]
```

- Ταξινόμηση λίστας, μέθοδος sort

```
>>>a=[2,9,-1,5,0]
>>>a.sort() #προαιρετικά(key=f,reverse=True)
>>>a          [-1,0,2,5,9]
```

Οι Μέθοδοι split και join

Η μέθοδος split μετατρέπει μια συμβολοσειρά σε λίστα δευτερευουσών συμβολοσειρών, μέσω ενός διαχωριστή

```
>>>text=(’Αυτό είναι ένα παράδειγμα κειμένου που θέλω να διαχωρίσω’)
>>>words = text.split(" ") #διαχωρίζει με διαχωριστή το κενό
print(words)
['Αυτό', 'είναι', 'ένα', 'παράδειγμα', 'κειμένου', 'που', 'θέλω', 'να', 'διαχωρίσω.']
```

Η μέθοδος join μετατρέπει μια λίστα συμβολοσειρών σε μια συμβολοσειρά.

```
>>> words = ['Αυτό', 'είναι', 'ένα', 'παράδειγμα', 'κειμένου', 'που', 'θέλω', 'να', 'διαχωρίσω.']
>>> text = " ".join(words)
>>> print(text)
Αυτό είναι ένα παράδειγμα κειμένου που θέλω να διαχωρίσω.
>>>
```

Ταξινόμηση λίστας

Παράδειγμα 3ο: Δίνεται μία λίστα με αριθμούς. Να τους ταξινομήσετε με βάση την απόλυτη τιμή τους. Στο όρισμα key θα δοθεί η τιμή της συνάρτησης abs

```
a=[2,-2,4,-7,3,1,0,7]
a.sort(key=abs)
print(a)
```

```
[0, 1, 2, -2, 3, 4, -7, 7]
```

αν

```
a.sort(reverse=True) #αντίστροφη ταξινόμηση
print(a)
```

```
[7, 4, 3, 2, 1, 0, -2, -7]
```

ενώ

```
a.sort(key=abs,reverse=True)
```

```
[-7, 7, 4, 3, 2, -2, 1, 0]
```

Ταξινόμηση λίστας με τη χρήση Sum

Παράδειγμα 4ο: Δίνεται μια λίστα που περιέχει ένθετες λίστες. Να ταξινομήσετε το περιεχόμενό της με βάση το άθροισμα των ένθετων λιστών.

```
b=[[2,3],[1,3],[-5,-6],[-3,2]]
b.sort(key=sum)
print(b)
```

```
[[-5, -6], [-3, 2], [1, 3], [2, 3]]
```

- Οι συναρτήσεις max,min

Για να επιστρέψει τη μέγιστη ή ελάχιστη τιμή μιας λίστας πρέπει τα στοιχεία της να είναι του ίδιου τύπου

nums=[3,7,4,2,9,3,-1,11]

>>> max(nums),min(nums) (11,-1)

- Οι συμβολοσειρές ταξινομούνται αλφαβητικά

  >>>a=['g','k','a','l']

  >>>max(a),min(a)

  ('l','a')

- Η συνάρτηση len, μετρά το πλήθος των στοιχείων

  len(nums) 8

- Στις λογικές τιμές False<True
- Η συνάρτηση **list(),** μετατρέπει ακολουθία σε λίστα

a='university' list(a)

['u', 'n', 'i', 'v', 'e', 'r', 's', 'i', 't', 'y']

Ταξινόμηση λίστας από min προς max

Παράδειγμα 4b: Να υπολογιστούν max, min, ΜΟ μιας ακολουθίας μη αρνητικών αλλά πρώτα να αποθηκεύονται οι αριθμοί σε λίστα

Με -1 να σταματώ να δίνω άλλους αριθμούς

```
#Γέμισμα λίστας
k1=[]
num=eval(input('dose arithmo -1 for stop :'))
while num !=-1: #έλεγχος για -1
   k1.append(num)
   num=eval(input('dose arithmo -1 for stop :'))
mo=sum(k1)/len(k1)
k1.sort()
print(k1,'min είναι :',k1[0],'max είναι :', k1[-1])
print('mesos oros: {0:.2f}'.format(mo))
```

Συγκρίσεις αναζήτηση και εντοπισμός

- Κάνοντας συγκρίσεις

Δύο λίστες μπορούν να συγκριθούν μεταξύ τους με τη χρήση των σχεσιακών τελεστών (<,> κ.λ.π)

Η σύγκριση γίνεται στοιχείο προς στοιχείο.

Τα στοιχεία προς σύγκριση θα πρέπει να είναι του ίδιου τύπου, ενώ τα στοιχεία κάθε λίστας, μπορεί να είναι διαφορετικού τύπου.

Έστω για παράδειγμα οι λίστες:

>>>a=[1,3,'a'], c=[0,3,'a']

>>>b=[0,3,'b'], d=[0,3,'b']

>>>a>b

True (Γιατί 1>0 , ο έλεγχος σταματά)

>>>c>b

False (Γιατί 'a'<'b'), ενώ

>>>a=[1,2]

>>>b=[0,'c']

>>>a>b

True

- Ο τελεστής in

Για τον έλεγχο ύπαρξης μιας τιμής μέσα σε μια λίστα χρησιμοποιούμε τον τελεστή in (x in y).

```
>>>k=5
>>>a=[3,5,7]
>>>k in a, 1 in a
```

(True, False)

Παράδειγμα 5ο: Να γράψετε ένα πρόγραμμα, που θα διαβάζει ένα γράμμα και θα εμφανίζει αν είναι φωνήεν ή σύμφωνο.

```
f=['α','ε','η','ι','ο','υ','ω']
  while True:          #(Με έλεγχο εισόδου)
  a=int(input('Δώσε ο για να σταματήσω, έναν αριθμό για να συνεχίσω ?'))
  if a==0:
      break
  x=input('Δώσε ένα γράμμα : ? ')
  if x in f:
      print('Είναι φωνήεν')
  else:
      print('Είναι σύμφωνο')
```

- Η μέθοδος count

Εντοπίζει μία τιμή μέσα σε μία λίστα και επιστρέφει αριθμό.

Σύνταξη:μεταβλητή=λίστα.count(x)

Παράδειγμα 6ο: Δίνεται λίστα με βαθμούς γραπτών. Να γραφεί πρόγραμμα που θα εμφανίζει πόσα γραπτά μηδενίστηκαν, καθώς και τα άριστα γραπτά.

Έλεγχος γραπτών που μηδενίστηκαν

```
grades=[10,7,6,0,3,4,10,9,0]
zeros=grades.count(0)
print(Μηδενικά γραπτά είναι : ',zeros)
```

Έλεγχος Άριστων γραπτών

```
g=grades.count(10)
if g!=0:
   print('Υπήρχαν ',g, ' γραπτά με 10')
else:
   print('Δεν υπήρχαν άριστα γραπτά')
```

```
Μηδενικά γραπτά είναι :  2
**********************************************
Υπήρχαν  2  γραπτά με 10
```

Εντοπισμός στοιχείου με την index

Πολύ συχνά εκτός του εντοπισμό ενός στοιχείου σε μια λίστα, είναι απαραίτητο να βρεθεί και η θέση του μέσα σ' αυτή. Αυτό γίνεται με τη μέθοδο index.

Σύνταξη:
μεταβλητή=λίστα.index(αντικείμενο)

Παραδείγματα:

```
>>>a=[1,3,5,7,3,5,3,5]
>>>b=a.index(7) ή a.index(7)
>>>b
3
>>>a.index(3)
1
```

(Μας δίνει την πρώτη θέση του '3')

Μπορούμε με την count να βρούμε αν υπάρχουν και άλλα '3' στη λίστα).

Ένθετες λίστες

Μία λίστα μπορεί να έχει για στοιχεία ακόμα και άλλες λίστες

>>>list1=[1,2,['a','b','c'],[1,2],3]

>>>list1[2]

['a','b','c'] (Το 2ο στοιχείο της list1 είναι μία λίστα)

>>>list1[3][1] (Το 1ο στοιχείο της 2ης λίστας)

2

Πίνακες

Στην Python οι πίνακες είναι στην πραγματικότητα λίστες μέσα σε άλλες λίστες και προσπελάζονται μέσω δύο δεικτών αν είναι δύο διαστάσεων ή η δεικτών αν είναι η διαστάσεων.

```
>>>pinakas=[[1,2,3],[4,5,6],[7,8,9]]
>>>pinakas
[[1,2,3], [4,5,6], [7,8,9]]
```

- Εκτύπωσε όλα τα στοιχεία του 1ου πίνακα

```
>>> for k in range(3):
        print(pinakas[1][k],end=' ')
4 5 6
```

- Εκτύπωσε τα μεσαία στοιχεία όλων των πινάκων του πίνακα

```
>>> for k in range(3):
            print(pinakas[k][1],end=' ')
2 5 8
```

- Εκτύπωσε από τον 1ο πίνακα το 1ο στοιχείο, από τον 2ο το 2ο, και από τον 3ο το 3ο από τον πίνακα

```
>>> for k in range(3):
            print(pinakas[k][k],end=' ')
1 5 9
```

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΝΑΚΕΦΑΛΑΙΩΣΗΣ

Παράδειγμα 7ο : Να γραφεί πρόγραμμα που θα διαβάζει μια λίστα 10 θέσεων με ακέραιους αριθμούς και θα δημιουργεί μια λίστα 5 θέσεων, στην οποία θα καταχωρηθούν τα στοιχεία της πρώτης λίστας που βρίσκονται σε περιττές θέσεις.

```
a=[]
b=[]
print('ΔΗΜΙΟΥΡΓΙΑ ΛΙΣΤΑΣ με for 10 στοιχείων')
print()
for i in range(0,10):
   print('dose to {0:d}o stoixeio :'.format(i+1),end='')
   x=int(input())
   a.append(x)
print(a)
for i in range(0,5):
   b.append(a[2*i])
print(b)
```

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΝΑΚΕΦΑΛΑΙΩΣΗΣ

Παράδειγμα 8ο erg1o: Μια εταιρία catering απασχολεί 15 άτομα. Να γράψετε πρόγραμμα το οποίο:
Διαβάζει το ονοματεπώνυμο των υπαλλήλων και το αποθηκεύει σε μια λίστα
Διαβάζει τον μηνιαίο μισθό κάθε υπαλλήλου (500<=μισθός<=2000) και τους τοποθετεί σε λίστα
Υπολογίζει και τυπώνει τον μέσο μηνιαίο μισθό
Υπολογίζει και τυπώνει τα ονόματα και το πλήθος των υπαλλήλων που αμείβονται με μηνιαίο μισθό μεγαλύτερου του μέσου όρου.

```
sals=[]
for i in range(5):
   name = input('Δώσε το όνομα :')
   while True:
      sal = float(input('Δώσε το μισθό του (500<=Μισθός<=2000) = :'))
        if 500<=sal<=2000:
                break#τερματίζει το βρόχο while  εκτελείται μόνο αν η συνθήκη if
                                                                    #είναι ψευδής
        print("Έδωσες ποσό εκτός ορίων, ξαναδώσε μισθό = : ')
        names.append(name)
       sals.append(sal)
mm=sum(sals)/len(sals)
print('Ο Μέσος μηνιαίος μισθός είναι {0:6.2f} ευρώ'.format(mm))
print('')
pl=0
for i in range(5):
   if (sals[i])>=mm:
      print("Όνομα : ',names[i],'Μισθός : ',sals[i])
      pl+=1
print ('Πλήθος ατόμων με μισθό μεγαλύτερο του μέσου όρου =: ',pl)
```

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΝΑΚΕΦΑΛΑΙΩΣΗΣ

Παράδειγμα 9ο: Δίνεται η ακόλουθη διεύθυνση:
k=('Ιερά Οδός 75 11235 Αθήνα'). Να μετατραπεί σε πέντε συμβολοσειρές.

```
k=('Ιερά Οδός 75 11235 Αθήνα')
lexeis=k.split()
for index in lexeis:
   print(index)
```

```
Ιερά
Οδός
75
11235
Αθήνα
```

Παράδειγμα 10ο: Να δημιουργηθεί μία αυξανόμενη λίστα η οποία θα γεμίζει με αριθμούς που δίνουμε από το πληκτρολόγιο, μέχρι να δώσουμε τον αριθμό -1

```
l1=[]
num=eval(input('dose arithmo -1 for stop :'))
while num !=-1:
   l1.append(num)
   num=eval(input('dose arithmo -1 for stop :'))
print(l1)
```

Η συνάρτηση zip

Η zip δέχεται δύο συλλογές δεδομένων και αποδίδει σε κάθε επανάληψη ένα ζεύγος τιμών, το ένα το αντλεί από την πρώτη και το δεύτερο από τη δεύτερη συλλογή δεδομένων

Παράδειγμα 10ο b: Από 2 λίστες που έχουν τα ονόματα και τους βαθμούς φοιτητών, να φτιαχτεί ένα πινακάκι ,που θα έχει το όνομα του φοιτητή και δίπλα το βαθμό του.

```
names=['Μάκης','Εύα','Τάκης','Λίνα']
grades=[4,8,9,5]
for k,i in zip(names,grades):
    print('Όνομα Φοιτητή :',k,'Βαθμός :',i)
```

και τρέχοντάς το:

```
Όνομα Φοιτητή : Μάκης Βαθμός : 4
Όνομα Φοιτητή : Εύα Βαθμός : 8
Όνομα Φοιτητή : Τάκης Βαθμός : 9
Όνομα Φοιτητή : Λίνα Βαθμός : 5
```