Βασικές Ασκήσεις από τα Εργαστήρια της Python

Εργαστήριο 4

► Αντί να χρησιμοποιούμε πολλές ομοειδείς μεταβλητές του ίδιου τύπου, όπως παραδείγματος χάριν, οι 7 μέσες θερμοκρασίες μίας εβδομάδας, μέσω των οποίων θα υπολογιστεί ο μέσος όρος της εβδομάδας, είναι πολύ πιο βολικό να χρησιμοποιούμε λίστες. Το παρακάτω παράδειγμα υπολογίζει τον μέσο όρο των θερμοκρασιών επτά ημερών χωρίς λίστα. Είναι προφανές ότι πρόκειται για κακή προσέγγιση, λόγω του μεγάλου αριθμού μεταβλητών που αναγκαζόμαστε να χρησιμοποιήσουμε:

```
# program13.py

t1=int(input('Vale temp 1: '))
t2=int(input('Vale temp 2: '))
t3=int(input('Vale temp 3: '))
t4=int(input('Vale temp 4: '))
t5=int(input('Vale temp 5: '))
t6=int(input('Vale temp 6: '))
t7=int(input('Vale temp 7: '))

mo=(t1+t2+t3+t4+t5+t6+t7)/7.0
print('O MO einai %.2f'%mo)
```

► Το προηγούμενο πρόγραμμα, βελτιώνεται με τη χρήση της λίστας t. Είναι προφανές ότι αν θέλουμε να βρούμε τον μέσο όρο μηνός αρκεί να αλλάξουμε την “σταθερά” SIZE, πχ. από 7 σε 31:

```
# program14.py

SIZE=7

t=[]
for i in range(0,SIZE):
    t.append(int(input('Vale temp ' + str(i+1) +': ')))

print(t)

athroisma=0
for i in t:
    athroisma+=i

print(athroisma)

print('MO einai %.2f'%(athroisma/SIZE))
```

► Στην γλώσσα C, δηλώνουμε το μέγεθος ενός ακέραιου πίνακα με δέκα θέσεις γράφοντας int a[10]; Στην Python οι λίστες είναι **δυναμικές**. Λέγοντας a=[] δηλώνουμε μία κενή λίστα χωρίς στοιχεία. Με την μέθοδο append, προσθέτουμε στοιχεία στη λίστα αυτή. Με τον τελεστή + μπορούμε να προσθέσουμε λίστα σε λίστα.

```
>>> a=[]
>>> a
[]
>>> a.append(3)
>>> a
[3]
>>> a.append(5)
>>> a
[3, 5]
>>> a+2
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    a+2
TypeError: can only concatenate list (not "int") to list
>>> a+[3,4]
[3, 5, 3, 4]
>>> a
[3, 5]
>>> a=a+[3,4]
>>> a
[3, 5, 3, 4]
>>>
```

► Για να μπορέσουμε να χρησιμοποιήσουμε τις λίστες στην Python με τρόπο παρεμφερή με αυτόν που χρησιμοποιείται για τα arrays στην C, μπορούμε να δεσμεύσουμε δέκα θέσεις εκ των προτέρων, γράφοντας a=[0]*10. Δηλαδή δημιουργούμε μία λίστα με 10 στοιχεία με δείκτες από 0 έως 9, που το κάθε ένα έχει την τιμή 0.

```
>>> a=[]
>>> a[4]=1
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    a[4]=1
IndexError: list assignment index out of range
>>> a=[0]*10
>>> a[4]=1
>>> a
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
>>> a=[None]*10
>>> a
[None, None, None, None, None, None, None, None, None, None]
>>> a[4]=1
>>> a
[None, None, None, None, 1, None, None, None, None, None]
>>>
```

► Το επόμενο πρόγραμμα διανέμει ένα ποσό 50.000 € σε 5 σχολές, ανάλογα με το πλήθος των φοιτητών τους. Διαβάζει το πλήθος των φοιτητών για κάθε σχολή και υπολογίζει το ποσό που πρέπει να δοθεί σε κάθε σχολή.

```
#program15.py

s=[]
for i in range(0,5):
    s.append(int(input('sxoli '+str(i+1)+': ')))

print(s)

sum=0
for i in s:
    sum+=i

print(sum)
for i in range(0,5):
    print('%d %d' % (i+1, s[i]))

# posa ana sxoli

for i in range(0,5):
    print('Sxoli %d pairnei %.2f' % (i+1, 50000/sum*s[i]))
```

► Το παρακάτω πρόγραμμα βρίσκει την μέγιστη και την ελάχιστη από 6 θερμοκρασίες:

```
#program16.py

t=[]
for i in range(0,6):
    t.append(int(input('temp '+str(i+1)+': ')))

max=min=t[0]
for i in t:
    if(max<i):
        max=i
    if(min>i):
        min=i


print('max is %d, min is %d' % (max, min))
```

► Μία βασική διαφορά μεταξύ array και list, είναι ότι τα arrays περιέχουν μόνο στοιχεία ίδιου τύπου. Πχ. Ακέραιους, χαρακτήρες κλπ. Η Python επιτρέπει τη δημιουργία array με χρήση του αντικειμένου array, η υλοποίηση του οποίου βασίζεται σε λίστα, περιορίζοντας τους τύπους.

class `array.array`(*typecode*[, *initializer*])

Επιτρεπτοί τύποι:

Type code	C Type	Python Type	Minimum size in bytes
`'b'`	signed char	int	1
`'B'`	unsigned char	int	1
`'u'`	Py_UNICODE	Unicode character	2
`'h'`	signed short	int	2
`'H'`	unsigned short	int	2
`'i'`	signed int	int	2
`'I'`	unsigned int	int	2
`'l'`	signed long	int	4
`'L'`	unsigned long	int	4
`'q'`	signed long long	int	8
`'Q'`	unsigned long long	int	8
`'f'`	float	float	4
`'d'`	double	float	8

```
#program17.py

import array

i=array.array('I', [1,3,4,6,7])
i[4] = 123
print(i)
i.append(41)
print(i)
```

Σχετικά με τις λίστες στην Python:

Η Python γνωρίζει έναν αριθμό σύνθετων τύπων δεδομένων, που χρησιμοποιούνται για την ομαδοποίηση άλλων τιμών. Ο πιο ευέλικτος είναι η λίστα, η οποία να γραφτεί σαν λίστα τιμών (αντικειμένων) που χωρίζονται με κόμματα και βρίσκεται μέσα σε αγκύλες.

Οι λίστες μπορούν να περιέχουν αντικείμενα διαφορετικού τύπου, αλλά συνήθως τα στοιχεία έχουν όλα τον ίδιο τύπο.

Μέθοδοι των αντικειμένων τύπου list:

`list.append`(*x*)

Προσθέτει ένα αντικείμενο στο τέλος της λίστας. Ισοδύναμο με `a[len(a):] = [x]`.

`list.extend`(*iterable*)

Επεκτείνει την λίστα προσθέτοντας όλα τα στοιχεία ενός iterable. Equivalent to `a[len(a):] = iterable`.

```
>>> a=[1,2,3]
>>> type(a)
<class 'list'>
>>> a.extend([4,5])
>>> a
[1, 2, 3, 4, 5]
>>> a.extend('a')
>>> a
[1, 2, 3, 4, 5, 'a']
>>> b=[9,10]
>>> a.append(b)
>>> a
[1, 2, 3, 4, 5, 'a', [9, 10]]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 'a', [9, 10], 9, 10]
>>>
```

`list.insert`(*i*, *x*)

Προσθέτει ένα στοιχείο στην δεδομένη θέση. Η πρώτη παράμετρος είναι ο δείκτης του στοιχείου πριν το οποίο θα τοποθετηθεί το νέο, έτσι a.insert(0, x) εισαγάγει μπροστά από τη λίστα το στοιχείο x. Το insert(len(a), x) είναι ισοδύναμε με το a.append(x).

`list.remove(`*x*`)`

Διαγράφει το πρώτο στοιχείο της λίστας του οποίου η τιμή ισούται με x. Εάν δεν υπάρχει τέτοιο στοιχείο δημιουργείται σφάλμα.

`list.pop(`[*i*]`)`

Διαγράφει το στοιχείο που βρίσκεται στην θέση 'i' και το επιστρέφει. Εάν δεν δοθεί παράμετρος η pop() διαγράφει και επιστρέφει το τελευταίο στοιχείο της λίστας.

`list.clear()`

Διαγράφει όλα τα στοιχεία της λίστας. Ισοδύναμο με `del a[:]`.

`list.index(`*x*[, *start*[, *end*]]`)`

Επιστρέφει τον δείκτη της θέσης στην οποία βρίσκεται το πρώτο στοιχείο του οποίου η τιμή ισούται με x. Δημιουργεί `ValueError` εάν δεν υπάρχει το στοιχείο.

Οι προαιρετικές παράμετροι *start* και *end* επιτρέπουν την αναζήτηση σε συγκεκριμένο τμήμα της λίστας.

`list.count(`*x*`)`

Επιστρέφει τον αριθμό των φορών που το *x* εμφανίζεται στη λίστα.

`list.sort(`*key=None*, *reverse=False*`)`

Ταξινομεί τη λίστα. Δες την `sorted()`

`list.reverse()`

Αντιστρέφει τα στοιχεία της λίστας

`list.copy()`

Επιστρέφει αντίγραφο της λίστας. Ισοδύναμε με `a[:]`.

Παράδειγμα χρήσης των μεθόδων:

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4)  # Find next banana starting a position 4
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()
'pear'
```

Ίσως παρατηρήσατε ότι μέθοδοι όπως `insert`, `remove` ή `sort` οι οποίες μόνο μετατρέπουν την λίστα δεν επιστρέφουν τιμή για να τυπωθεί – επιστρέφουν το default `None`. Αυτή η σχεδιαστική αρχή ισχύει για όλες τις mutable δομές δεδομένων της Python.