Βασικές Ασκήσεις από τα Εργαστήρια της PYTHON (εβδομάδα 7)

► Συνάρτηση που δέχεται σαν παράμετρο λίστα αριθμών και υπολογίζει και επιστρέφει τον μέσο όρο τους. (prog01.py):

```
def listAverage(alist):
    suma = 0
    for i in alist:
        suma += i
    return suma/len(alist)

mylist=[1,2,3,4,5,6]
print('%s MO= %s' % (mylist, listAverage(mylist)))
print('%s MO= %s' % ([2, 42.2, 3.14], listAverage([2, 42.2, 3.14])))
```

► Για να "πιάσουμε" και την περίπτωση λάθους, όταν η λίστα που περνάμε σαν παράμετρο στην συνάρτηση να είναι κενή, η συνάρτηση ελέγχει το μήκος της λίστας για να αποφύγει τη διαίρεση με μηδέν. Εφόσον το μήκος είναι 0, η συνάρτηση επιστρέφει 0.0. Στην προηγούμενη εκδοχή της συνάρτησης η κενή λίστα θα δημιουργούσε τότε "**ZeroDivisionError: division by zero**".

Εδώ προσθέτουμε και την συνάρτηση **showAverage**, που δέχεται μία λίστα και τυπώνει την λίστα και τον μέσο όρο των αριθμών που περιέχει. Αυτή η συνάρτηση καλεί την **listAverage**. Η μία συνάρτηση καλεί την άλλη.

Δοκιμάστε να τροφοδοτήσετε την **showAverage** με tuple αντί για λίστα. Ποια είναι διαφορά μεταξύ tuple και λίστας; (prog02.py)

```
def listAverage(alist):
    size = len(alist)
    if(size==0):
      return 0.
    suma = 0
    for i in alist:
        suma += i
    return suma/size

def showAverage(prlist):
    print('%s MO= %s' % (prlist, listAverage(prlist)))

mylist=[1,2,3,4,5,6]
showAverage(mylist)

showAverage([2, 42.2, 3.14, 5])
showAverage([])
```

► Με την type() μπορούμε να ελέγξουμε τον τύπο μίας μεταβλητής στο shell, επειδή επιστρέφει το όνομα της κλάσης της οποίας αντικείμενο είναι. Προγραμματιστικά όμως, ρωτάμε αν η μεταβλητή είναι αντικείμενο ενός τύπου μέσω της **isinstance**() που επιστρέφει True ή False.

```
>>> i=1
>>> type(i)
<class 'int'>
>>> isinstance(i, int)
True
>>> isinstance(i, float)
False
```

► Το επόμενο προγραμματάκι χρησιμοποιεί την isinstance() για να αναγνωρίσει τον τύπο[1] της παραμέτρου που του περνάμε: (prog03.py)

```
def reportType(x):
    if (isinstance(x, int)):
        print("int!")
    elif (isinstance(x, str)):
        print("str!")
    elif (isinstance(x, float)):
        print("float!")
    elif (isinstance(x, tuple)):
        print("tuple!")
    else:
        print("i do not know...")

i=40
reportType(i)
reportType('hello')
reportType(3.14)
reportType((1,2,23))
reportType([1,2,3])
```

1 Επίτηδες δεν έχει μπει αναγνώριση για list.

► Σε αυτή τη μορφή, η συνάρτηση μας δεν τυπώνει τίποτα, αλλά επιστρέφει string με το όνομα του τύπου της παραμέτρου που του περνάμε. Τον μιγαδικό αριθμό δεν τον αναγνωρίζει (prog04.py):

```
def returnType(x):

    if (isinstance(x, int)):
        return 'int'
    elif (isinstance(x, str)):
        return 'str'
    elif (isinstance(x, float)):
        return 'float'
    elif (isinstance(x, tuple)):
        return 'tuple'
    elif (isinstance(x, list)):
        return 'list'
    else:
        return 'unknown'

i=40
print(returnType(i))
print(returnType('hello'))
print(returnType(3.14))
print(returnType((1,2,23)))
print(returnType([1,2,3]))
print(returnType(4+5j))
```

►Αυτό το πρόγραμμα καλεί μία συνάρτηση που δέχεται μία παράμετρο και ψάχνει και μετράει πόσα ακέραια στοιχεία περιέχει. Όταν το t είναι 'abc2' όμως, το 2 που βρίσκεται στην συμβολοσειρά το αναγνωρίζει ως '2', κατά συνέπεια δεν το μετράει επειδή δεν είναι int. (prog05.py)

```
def countIntegers(a):
    num = 0
    for i in a:
        if(isinstance(i, int)):
            num += 1
    return num

t=(1,2,'a','b',35)
print('%s has %d integers'%(t, countIntegers(t)))
t=[0,0,0,23,23.14,'abc']
print('%s has %d integers'%(t, countIntegers(t)))
t='abc2'
print('%s has %d integers'%(t, countIntegers(t)))
```

► Η συνάρτηση find ψάχνει να βρει ένα string μέσα σε ένα άλλο. Όταν το βρει, επιστρέφει τη θέση στην οποία το βρήκε. Αν δεν το βρει επιστρέφει -1. Η δεύτερη παράμετρος λέει από ποια θέση να αρχίσει η αναζήτηση. Η default τιμή της, αν δεν χρησιμοποιηθεί είναι 0. Η τρίτη παράμετρος ορίζει έως που να ψάξει. Εάν δεν χρησιμοποιηθεί ψάχνει μέχρι το τέλος του string.

```
>>> text='arnaki aspro aspro kai paxy'
>>> text.find('aspro')
7
>>> text.find('aspro', 7+1)
13
>>> text.find('aspro', 13+1)
-1
>>text.find('ro', 14, 19)
16
```

► Το επόμενο πρόγραμμα ψάχνει με την συνάρτηση find να βρει πόσες φορές υπάρχει το string word μέσα στο string text. Όταν βρει το πρώτο, κρατάει τη θέση στη μεταβλητή pos, και ψάχνει από αυτήν τη θέση και μετά, μεγαλώνοντας τον μετρητή cnt κατά ένα. Εάν δεν βρει, η find επιστρέφει -1 και η επανάληψη σταματάει. Τέλος η συνάρτηση επιστρέφει την τιμή της cnt. (prog06.py)

```
def countTimes(txt, wrd):
    cnt=0
    pos=txt.find(wrd)
    while(pos!=-1):
        cnt+=1
        pos=txt.find(wrd, pos+1)
    return cnt


text='arnaki aspro aspro kai paxy'

word=input('word or letter: ')
while(word != 'end'):
    print('%s appears %d times'%(word, countTimes(text, word)))
    word = input('word or letter: ')
```

► Μπορούμε να διατρέξουμε σύνθετες λίστες με χρήση της isinstance(). Κάθε στοιχείο της αρχικής μας λίστας είναι ή αριθμός ή λίστα, οπότε πριν προσθέσουμε ελέγχουμε. Αν δεν είναι λίστα τότε προσθέτουμε την τιμή. Εάν είναι διατρέχουμε τη λίστα και ελέγχουμε ή προσθέτουμε και ούτω καθεξής: (prog07.py)

```
def addAll(c):
    suma=0
    for i in c:
        if isinstance(i,list):
            for j in i:
                if isinstance(j,list):
                    for k in j:
                        if isinstance(k, list):
                            for l in k:
                                suma+=l
                        else:
                            suma+=k
                else:
                    suma+=j
        else:
            suma+=i
    return suma

t=[1,2,[3,4,[5,6,[7,8],9],10,11],12,13,14]
print(addAll(t))
```

Για να ελέγξουμε γρήγορα το αποτέλεσμα, χρησιμοποιούμε τιμές που μπορούμε να αναπαράξουμε εύκολα με ένα for στο shell:

```
>>> sum=0
>>> for i in range(0,15):
      sum+=i

>>> sum
105
```

Για να αποφύγουμε τις προσαρμογές του αριθμού των nested επαναλήψεων στον κώδικα, ανάλογα με το βάθος των εσωτερικών λιστών, βολεύει περισσότερο να χρησιμοποιήσουμε **αναδρομή**. Για να υπολογίσουμε το άθροισμα χρησιμοποιούμε την global μεταβλητή suma: (prog08.py)

```
suma=0

def add_lol(the_list):
    global suma
    for each_item in the_list:
        if isinstance(each_item, list):
            add_lol(each_item)
        else:
            suma+=each_item

t=[1,2,[3,4,[5,6,[7,8],9],10,11],12,13,14]
add_lol(t)
print('Sum', suma)
```

► Μπορούμε εύκολα με χρήση της αναδρομής όχι μόνο να μικρύνουμε τον κώδικα, αλλά να του δώσουμε τη δυνατότητα να είναι τελείως ανεξάρτητος από τη δομή της λίστας μας, σε αντίθεση με τη μη αναδρομική λύση, όπου πρέπει να έχουμε nested ζεύγη for και if, τόσα όσο είναι το βάθος της λίστας μας. Αυτό το πρόγραμμα τυπώνει τα περιεχόμενα της λίστας με αναδρομή : (prog09.py)

```
def print_lol(the_list):
    for each_item in the_list:
        if isinstance(each_item, list):
            print_lol(each_item)
        else:
            print(each_item)

t=[1,2,[3,4,[5,6,[7,8],9],10,11],12,13,14]
print_lol(t)
```