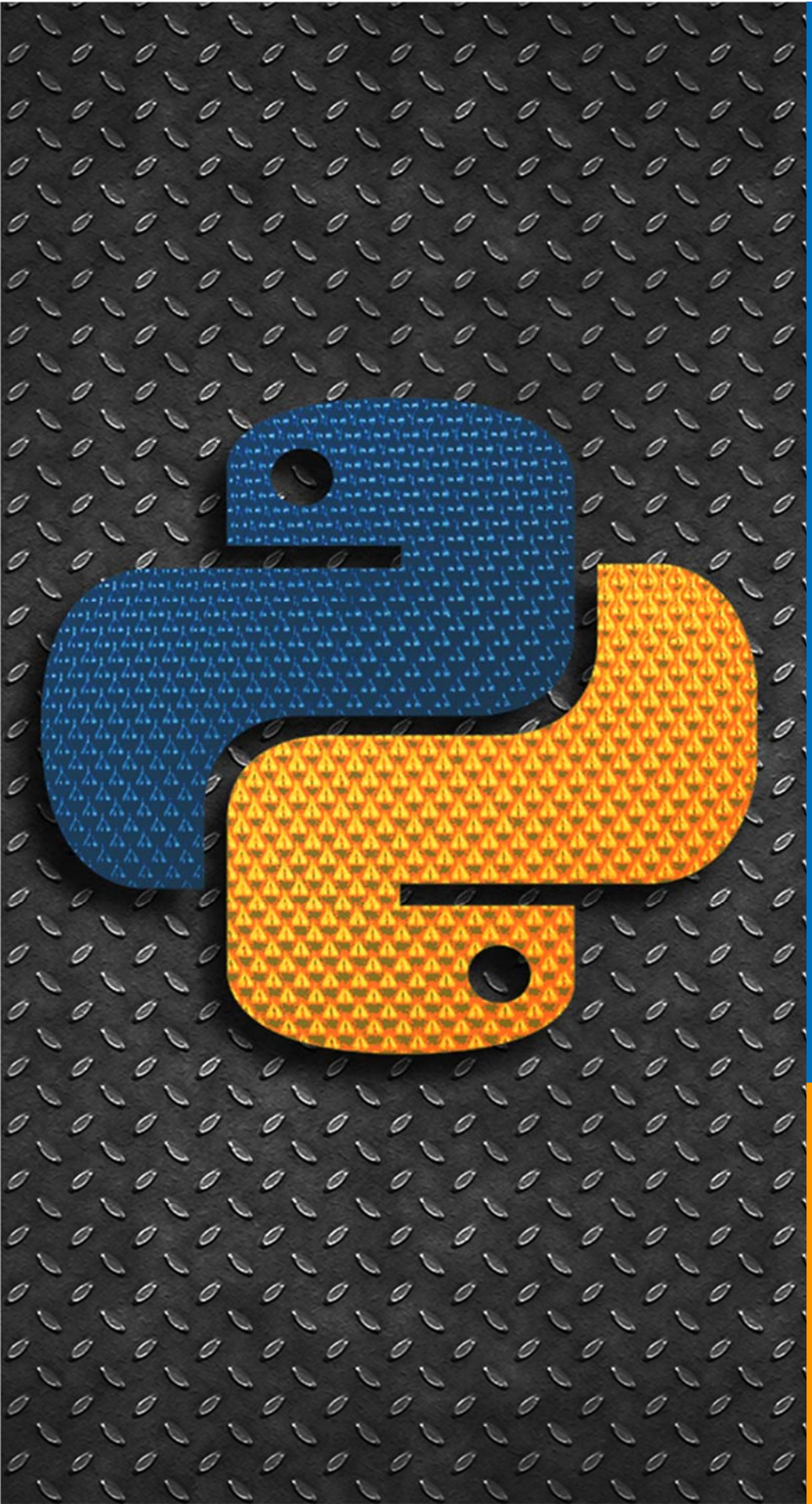

Επίλυση Δραστηριοτήτων του Βιβλίου Προγραμματισμός Υπολογιστών

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Γ' Τάξη ΕΠΑ.Λ.

Αναστάσιος Χατζηπαπαδόπουλος

Καθηγητής Πληροφορικής

VS 0.2 – 12/2016

Επίλυση Δραστηριοτήτων

του

Βιβλίου Μαθητή

του μαθήματος

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ

Γ’ Τάξη ΕΠΑΛ

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ISBN 978-960-93-8739-2

Αναστάσιος Χατζηπαπαδόπουλος
Καθηγητής Πληροφορικής ΠΕ19
chatzipapwork@gmail.com

Παρατηρήσεις

Αγαπητοί συνάδελφοι και μαθητές,

Οι υλοποιήσεις των αλγορίθμων των δραστηριοτήτων είναι ενδεικτικές.

Έγινε προσπάθεια να χρησιμοποιηθεί όπου ήταν εφικτό η διδαχθείσα ύλη, συνεπώς οι λύσεις μπορεί να μην είναι οι βέλτιστες. Έχουν δοθεί με τη σχετική τεκμηρίωση για ευκολία κατανόησης από τους μαθητές και για λόγους λειτουργικότητας των σεναρίων, με ένα σχετικό βαθμό ελευθερίας ως προς την αρχική εκφώνηση των προβλημάτων.

Έχει συμπεριληφθεί χωρίς να ζητείται πολλές φορές από την εκφώνηση για ευκολία επίδειξης και τμήμα κλήσης των συναρτήσεων των αλγορίθμων ώστε να παρέχονται άμεσα αποτελέσματα στην οθόνη.

Οι λύσεις συνοδεύονται από τα αντίστοιχα αρχεία κώδικα *(file name in doc header)* στον παρακάτω σύνδεσμο:

Scripts link: https://github.com/chatzipap/python_c_class

Οι προτάσεις σας για βελτίωση και διόρθωση των περιεχομένων είναι ευπρόσδεκτες.

☺ Happy Learning

Για τη συγγραφή και εκτύπωση των σεναρίων χρησιμοποιήθηκε το λογισμικό Notepad ++ έκδοση 7.2.2 (https://notepad-plus-plus.org/). Ενώ για την εκτέλεσή τους το περιβάλλον του IDLE (Python έκδοση 2.7.10).

Οι λύσεις των δραστηριοτήτων αφορούν στο βιβλίο του μαθήματος Προγραμματισμός Υπολογιστών των: Αράπογλου Α., Βραχνός Ε., Κανίδης Ε., Λέκκα Δ., Μακρυγιάννης Π., Μπελεσιώτης Β., Παπαδάκης Σπ., Τζήμας Δ. του τομέα Πληροφορικής της Γ’ ΕΠΑ.Λ. ΙΤΥΕ «ΔΙΌΦΑΝΤΟΣ».

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

Δραστηριότητα 2 / Σελίδα 87
Να τροποποιήσετε τον αλγόριθμο της ταξινόμησης με επιλογή, ώστε
να ταξινομεί μια λίστα ακεραίων σε φθίνουσα σειρά. Υπάρχει τρόπος
να το πετύχετε, χωρίς να κάνετε καμία απολύτως αλλαγή στον κύριο
αλγόριθμο που δίνεται σε αυτή την ενότητα; Σε τι οφείλεται αυτό;

'''

## Αρχικός αλγόριθμος ταξινόμησης με επιλογή

# Εύρεση της θέσης του μικρότερου στοιχείου μιας λίστας
# μεταξύ των θέσεων start & end
def find_min_posistion(start, end, alist):
    position = start
    for i in range(start, end):
        if alist[i] < alist[position] :
            position = i
    return position

# Αλγόριθμος ταξινόμησης selection sort
def selection_sort_ascending(alist):
    position = None
    n = len(alist)
    for i in range(0,n):
        position = find_min_posistion(i, n, alist)
        alist[i], alist[position] = alist[position], alist [i]
    return alist

# Η λύση στην άσκηση μπορεί να δοθεί με τους εξής τρόπους:
# 1ος Pythonic Way με μικροαλλαγή στον αρχικό κώδικα της ταξινόμησης
def selection_sort_descending(alist):
    position = None
    n = len(alist)
    for i in range(0,n):
        position = find_min_posistion(i, n, alist)
```

```
        alist[i], alist[position] = alist[position], alist [i]
    alist = alist.reverse()
    return alist

# 2ος χωρίς αλλαγή στον αρχικό κώδικα της ταξινόμησης
def find_min_max_position(start, end, alist, AscDesc):
    # Τροποποίηση της εύρεσης θέσης του min ώστε να βρίσκει και τη
    θέση του max
    # ανάλογα αν επιθυμούμε αύξουσα (Ascending=a) ή φθίνουσα
    (Descending=d) ταξινόμηση
    position = start
    for i in range(start, end):
        if AscDesc.lower()=='a':
            if alist[i] < alist[position] :
                position = i
        elif AscDesc.lower()=='d' :
            if alist[i] > alist[position] :
                position = i
    return position

def selection_sort2(alist,AscDesc):
    # Προσθήκη ακόμας μίας παραμέτρου στη συνάρτηση
    # AscDesc ('a' για αύξουσα, 'd' για φθίνουσα ταξινόμηση)
    position = None
    n = len(alist)
    for i in range(0,n):
        position = find_min_max_position(i, n, alist,AscDesc)
        alist[i], alist[position] = alist[position], alist [i]
    return alist

#-------------------------------------------------------------
# Δοκιμαστικές κλήσεις και αποτελέσματα
import random
start = 0
end = 21
alist = []

for i in range(start, end):
    alist.append(random.randint(1,100))

print 'Original List =', alist
print 'Min in alist is', alist[find_min_posistion(start,end,alist)],
'his posistion is ', find_min_posistion(start,end,alist)
blist = alist[:]
```

```
print 'List with selection sort ascending is ',
selection_sort_ascending(alist)
print 'Original list Min is ', blist[find_min_max_position(start,end,
blist, 'a')], 'in position ', find_min_max_position(start,end, blist,
'a')
print 'Original list Max is ', blist[find_min_max_position(start,end,
blist, 'd')], 'in position ', find_min_max_position(start,end, blist,
'd')
print 'Desc - Sorted list with selection_sort2 is',
selection_sort2(blist,'d')
print 'Asc - Sorted list with selection_sort2 is',
selection_sort2(blist,'a')
#------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 3 / Σελίδα 87
Να γράψετε μια συνάρτηση σε Python, η οποία θα δέχεται μια λίστα,
θα ελέγχει αν τα στοιχεία της είναι σε αύξουσα σειρά και θα επιστρέφει
αντίστοιχα true ή false.
Προτείνεται να χρησιμοποιήσετε μια λογική μεταβλητή

'''

# Αλγοριθμική λύση
def ascending(alist):
    ascend = True
    for i in range(len(alist) - 1):
        if alist[i+1] < alist[i] :
            ascend = False
    return ascend

# Pythonic Ways
def ascending2(alist):
    return all(alist[i+1] > alist[i] for i in range(len(alist) - 1))

def ascending3(alist):
    return sorted(alist) == alist

#-------------------------------------------------------------
# Δοκιμαστικές κλήσεις και αποτελέσματα
import random
start = 0
end = 21
alist = []
for i in range(start, end):
    alist.append(random.randint(1,100))
print 'Original List =', alist
print 'Is list asc sorted ??', ascending(alist)
print 'Sorted list =', sorted(alist)
print 'Is list asc sorted ??', ascending(sorted(alist))
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 4 / Σελίδα 87 (βελτιωμένη φυσαλίδα)
Να δώσετε τη βελτιωμένη έκδοση του αλγορίθμου ταξινόμησης
ευθείας ανταλλαγής η οποία τερματίζει, όταν διαπιστώσει ότι η
λίστα είναι ταξινομημένη, ώστε να αποφεύγονται περιττές συγκρίσεις.
Υπόδειξη: Χρησιμοποιήστε μια λογική μεταβλητή η οποία θα
αλλάζει τιμή, αν υπάρχουν τουλάχιστον δύο στοιχεία τα οποία
δεν βρίσκονται στην επιθυμητή σειρά,
καθώς η “φυσαλίδα ανεβαίνει στην επιφάνεια”.

'''


# Κλασική Φυσαλίδα
def bubblesort(alist):
    for i in range(len(alist)):
        for j in range(len(alist) - 1, i-1 , -1):
            if alist[j] < alist[j-1] :
                alist[j], alist[j-1] = alist[j-1], alist[j]


# Βελτιωμένη Φυσαλίδα
def short_bubblesort(alist):
        done = False
        i = 0
        # Σταματά η επανάληψη όταν ταξινομηθεί η λίστα
        while i < len(alist) - 1  and done == False :
            done = True
            for j in range(len(alist) - 1, i , -1):
                if alist[j] < alist[j-1] :
                    alist[j], alist[j-1] = alist[j-1], alist[j]
                    done = False

#-------------------------------------------------------------

# Δοκιμαστικές κλήσεις και αποτελέσματα
import random
start = 0
```

```
end = 21
alist = []
for i in range(start, end):
    alist.append(random.randint(1,100))
print 'Original List =', alist
short_bubblesort(alist)
print 'Sorted list=', alist
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 5 / σελίδα 89
Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα
με λογικές τιμές True/False και θα διαχωρίζει τις τιμές αυτές,
τοποθετώντας τα True πριν από τα False.

'''


# Διάταξη λογικών τιμών
def order_booleans(alist):
        i, j, n = 0, 0, len(alist) - 1
        while j <= n:
            if alist[j] > False:
                alist[i], alist[j] = alist[j], alist[i]
                i += 1
                j += 1
            else:
                j += 1

#------------------------------------------------------------
# Δοκιμαστικές κλήσεις και αποτελέσματα
import random
start = 0
end = 21
alist = []
for i in range(start, end):
    alist.append(random.choice([True, False]))
print 'Original List =', alist
order_booleans(alist)
print ''
print 'True Before False LIST=', alist
#------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz
'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 6 / σελίδα 89
Να γράψετε ένα πρόγραμμα σε Python το οποίο θα δέχεται μια λίστα
με λογικές τιμές True/False και στη συνέχεια θα καλεί την συνάρτηση
του προηγούμενου ερωτήματος, ώστε να τοποθετηθούν τα True
πριν από τα False. Στη συνέχεια θα τοποθετεί τις τιμές αυτές εναλλάξ,
δηλαδή True, False, True, False, κλπ,εκτελώντας τιςλιγότερες
δυνατές συγκρίσεις.

'''

# Διάταξη λογικών τιμών βλ Δρ.5. σελ.89
def order_booleans(alist):
        i, j, n = 0, 0, len(alist) - 1
        while j <= n:
            if alist[j] > False:
                alist[i], alist[j] = alist[j], alist[i]
                i += 1
                j += 1
            else:
                j += 1

# -----------Uncomment for manual input---------------------
### Εισαγωγή λογικών τιμών σε λίστα
##alist = []
##item = input('Enter Values 1=True, 0=False =')
##while item != None :
##    alist.append(item)
##    item = input('Enter Values 1=True, 0=False, None=Stop =')
##print 'Original List=', alist
##-----------------------------------------------------------

# ------------------AuTo values-------------------------------
# Δοκιμαστικές κλήσεις και αποτελέσματα
import random
start = 0
end = 21
alist = []
```

```
for i in range(start, end):
    alist.append(random.choice([1, 0]))
print 'Original List =', alist
##-------------------------------------------------------------

order_booleans(alist)
print 'Ones(1) before Zeros(0) List=', alist

# Βρες τη θέση εμφάνισης του 1ου False
pos = 0
for item in alist:
    if item == 1 :
        pos += 1
    else :
        break
print 'Change 1-->0 position=', pos, 'Values length=',len(alist)

# Βάλε ενναλάξ τα στοιχεία 0 & 1
j = 1
if alist.count(1) == alist.count(0):
    print 'Equal number of True(s) & False(s)'
    # Όταν ο αριθμός των True & False είναι ίδιος μπορούμε
    # να χρησιμοποιήσουμε την παρακάτω δομή με for
    for i in range(pos, len(alist)):
        alist[j], alist[i] = alist[i], alist[j]
        j += 2
else:
    print 'Different number of True(s) & False(s)'
    # Για να λειτουργεί ο αλγόριθμος και με διαφορετικό αριθμό True &
    False
    # πρέπει να χρησιμοποιήσουμε την while
    j = 1
    i=pos
    while i<len(alist) and j<len(alist):
        alist[i], alist[j] = alist[j], alist[i]
        j += 2
        i=i+1

#-------------------------------------------------------------
# Εμφάνιση αποτελεσμάτων
print '1 & 0 alternately=', alist
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 7 / σελίδα 89
Ας υποθέσουμε ότι σας δίνεται μια λίστα στην Python η οποία
περιέχει λογικές τιμές True/False εναλλάξ. Επίσης, το πλήθος των
True είναι ίσο με το πλήθος των False. Να γράψετε αλγόριθμο,
σε Python, ο οποίος δεδομένης της παραπάνω δομής της λίστας,
θα τοποθετεί τα True πριν από τα False εκτελώντας,
τις λιγότερες δυνατές μετακινήσεις. Δεν επιτρέπεται να κάνετε
καμία σύγκριση ούτε να χρησιμοποιήσετε τη δομή if.
Θεωρήστε ότι το πρώτο στοιχείο της λίστας έχει την τιμή True.

'''

# Δεδομένα δοκιμών
alist = [True, False, True, False, True, False, True, False,
         True, False, True, False, True, False, True, False, True,
         False,
         True, False, True, False, True, False, True, False, True,
         False]

# Διάταξε τις λογικές τιμές με αμοιβαίες αντιμεταθέσεις
def order_boolean(alist):
    n = len(alist)
    mid = n // 2
    for i in range(1, mid, 2) :
        alist[i], alist[n-i-1] = alist[n-i-1], alist[i]
    return alist

#-----------------------------------------------------------
# Κλήση συνάρτησης
x = order_boolean(alist)
print x
#-----------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 8 / σελίδα 89
Το πρόβλημα της ολλανδικής σημαίας αναφέρεται στην αναδιάταξη
μιας λίστας γραμμάτων, η οποία περιέχει μόνο τους
χαρακτήρες R, W, B. (Red, White, Blue), έτσι ώστε όλα τα R
να βρίσκονται πριν από τα W και όλα τα W να βρίσκονται πριν
από B. Να τροποποιήσετε έναν από τους αλγορίθμους ταξινόμησης που
παρουσιάστηκαν σε αυτήν την ενότητα, ώστε να επιλύει αυτό το
πρόβλημα.

'''

# Συνάρτηση Υλοποίησης αλγορίθμου Dutch Flag Problem
def order_flag_colors(alist):
        i, j, n = 0, 0, len(alist) - 1
        while j <= n:
            if alist[j] < 1:
                alist[i], alist[j] = alist[j], alist[i]
                i += 1
                j += 1
            elif alist[j] > 1:
                alist[j], alist[n] = alist[n], alist[j]
                n -= 1
            else:
                j += 1


#-------------------------------------------------------------
# Δοκιμαστικές κλήσεις και αποτελέσματα
# Red <--> 0, White <--> 1, Blue <--> 2
dic = {0:'R', 1:'W', 2:'B'} # Δημιουργία λεξικού για αντιστοίχιση
χρώματος/αριθ.
import random
start = 0
end = 21
alist = []
for i in range(start, end):
```

```
    alist.append(random.choice([0, 1, 2]))
print 'Original List =', alist

order_flag_colors(alist)
print 'Ordered numbered List =', alist
print 'Ordered Flag Color List=', [dic[i] for i  in alist]
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz


'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 9 / σελίδα 89
Να γράψετε μια συνάρτηση σε Python η οποία διαβάζει αριθμούς
από το χρήστη τους οποίους τοποθετεί σε μια λίστα σε φθίνουσα
σειρά, την οποία και επιστρέφει.
Κάθε φορά που διαβάζει έναν νέο αριθμό τον τοποθετεί στη σωστή
θέση στην ήδη ταξινομημένη λίστα, ώστε να διατηρείται η φθίνουσα
διάταξη των στοιχείων της λίστας. Ποιον αλγόριθμο ταξινόμησης
σας θυμίζει η παραπάνω λειτουργία;  Σε τι διαφέρει η συνάρτηση
που θα αναπτύξετε από τον αλγόριθμο αυτόν;

'''

# Αλγόριθμος ταξινόμησης με εισαγωγή
def insertion_sort( alist ) :
    for i in range(1, len( alist ) ) :
        value = alist[i]
        j = i
        while j > 0 and alist[j-1] < value :
            alist[j] = alist[j-1]
            j = j-1
            alist[j] = value

# Εισαγωγή στοιχείων, κλήση συνάρτησης ταξινόμησης
alist = []
item = input('Δώσε ένα νέο ακέραιο=')
while item != None :
    alist.append(item)
    insertion_sort(alist)
    print alist
    item = input('Δώσε ένα νέο ακέραιο (None to stop)=')
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 5ο ΚΛΑΣΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ
Δραστηριότητα 10 / σελίδα 89
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει από το χρήστη
δυο λίστες αριθμών Α και Β και θα ταξινομεί σε αύξουσα σειρά
τους αριθμούς της λίστας Α. Στη συνέχεια θα εμφανίζει πόσοι από
τους αριθμούς της λίστας Β εμφανίζονται στηλίστα Α.
Υπόδειξη: Να θεωρήσετε ότι οι αριθμοί της λίστας Β είναι όλοι
διαφορετικοί μεταξύ τους. Επίσης, να εκμεταλλευτείτε το γεγονός
ότι η λίστα Α είναι ταξινομημένη σε αύξουσα σειρά.

'''

# Συνάρτηση Εισαγωγής στοιχείων σε λίστα
def fill_a_list():
    alist = []
    item = input('Δώσε έναν ακέραιο=')
    while item != None :
        alist.append(item)
        item = input('Δώσε έναν ακέραιο (None to stop) =')
    return alist

# Κλήση συνάρτησης και εισαγωγή στοιχείων στη λίστα Α
print 'Input LIST A'
A = fill_a_list()

# Κλήση συνάρτησης και εισαγωγή στοιχείων στη λίστα Β
print 'Input LIST B'
B = fill_a_list()

# Εμφάνιση λιστών Α και Β
print 'A=',A
print 'B=',B

# Συνάρτηση αλγορίθμου Κλασικής Φυσαλίδας
def bubblesort(alist):
    for i in range(len(alist)):
        for j in range(len(alist) - 1, i , -1):
```

```
            if alist[j] < alist[j-1] :
                alist[j], alist[j-1] = alist[j-1], alist[j]

# Ταξινόμηση Λίστας A
bubblesort(A)

# Συνάρτηση Δυαδικής Αναζήτησης
def binary_search( array, key ) :
    first = 0
    last = len(array)-1
    found = False
    while first <= last and not found :
        mid = ( first + last ) // 2
        if array[ mid ] == key :
            found = True
        elif array[ mid ] < key :
            first = mid + 1
        else :
            last = mid-1
    return found

# Πόσοι από τους αριθμούς της λίστας B εμφανίζονται στην λίστα A
count = 0
common_itemsAB = []
for item in B :
    if binary_search(A,item) :
        count += 1
        common_itemsAB.append(item)
print 'Βρήκα %d στοιχεία της λίστας B να περιέχονται στη λίστα A ' %
count
print common_itemsAB
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @ Tasos Chatzipapadopoulos

'''

ΚΕΦΑΛΑΙΟ 6ο ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ
Δραστηριότητα 3 / σελίδα 101
Να γράψετε πρόγραμμα στη γλώσσα Python, το οποίο θα δέχεται ως
είσοδο το όνομα ενός αρχείου, θα εμφανίζει τα περιεχόμενά του
κατά γραμμή και στη συνέχεια θα γράφει σε ένα άλλο αρχείο,
τις γραμμές του αρχείου στην αντίστροφη σειρά.

'''

# Δημιουργία αναφοράς σε αρχείο
f = open ('testFile.txt', 'r')

# pythonic way
for line in reversed(open("testFile.txt").readlines()):
    print line

# Αλγοριθμικη προσέγγιση
lines = []
for line in f :
    lines.append(line)

# Χωρίς μέθοδο αναστροφής
for i in range(len(lines)-1,0,-1):
    print lines[i]

# Με χρήση μεθόδου αναστροφής
lines.reverse()
for line in lines :
    print line

f.close()
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 6ο ΔΙΑΧΕΙΡΙΣΗ ΑΡΧΕΙΩΝ
Δραστηριότητα 4 / σελίδα 101
Τι πιστεύετε ότι θα συμβεί, όταν θα εκτελεστούν τα παρακάτω σενάρια.
Τεκμηριώστε την άποψή σας.

'''

# List Comprehension

# Δημιουργία λίστας τετραγώνων αριθμών από 1.. 10
my_list = [i**2 for i in range(1,11)]

# Άνοιγμα αρχείου για εγγραφή
f = open('output.txt', 'w')

# Γράψε τα στοιχεία της λίστας στο αρχείο
for item in my_list:
    f.write(str(item) + '\n') # δέχεται string όρισμα η write
f.close()

# Άνοιξε το αρχείο και εκτύπωσε τα περιεχόμενά του
f = open('output.txt', 'r')
print f.readline()
print f.read()
f.close()
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 1 /σελίδα 151
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και
θα εμφανίζει τα γράμματά της, ένα σε κάθε γραμμή.

'''


# Εισαγωγή λέξης
word = raw_input('Δώσε μία λέξη=')

# Με διάσχιση
for letter in word:
    print letter

# Με τεμαχισμό (slicing)
for i in range(len(word)):
    print word[i]
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 2 /σελίδα 151
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα
εμφανίζει πόσα κεφαλαία αγγλικά γράμματα περιέχει η λέξη.

'''

# Κλήση βιβλιοθήκης string για πρόσβαση σε συμβολοσειρές αλφάβητων
import string
english_caps = string.ascii_uppercase

# Αντί διάβασμα λέξης χρησιμοποιούμε το παρακάτω κείμενο για εύρεση
κεφαλαίων λατινικών χαρακτήρων
my_text = '''
Python is a widely used high-level, general-purpose, Interpreted,
dynamic programming Language.[24][25] Its design philosophy
emphasizes code readability, and its SYNTAX allows programmers
to Express concepts in Fewer LINES of code than POSSIBLE in
Languages such as C++ or Java.[26][27] The language provides
constructs
INTENDED to enable WRITING clear programs on both a small and
large SCALE.[28] Η Python είναι μια υψηλού επιπέδου γλώσσα
προγραμματισμού[1][2] η οποία δημιουργήθηκε από τον Ολλανδό
Γκβίντο βαν Ρόσσουμ (Guido van Rossum) το 1990.
Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της
και η ευκολία ΧΡΗΣΗΣ της και το συντακτικό της επιτρέπει στους
Προγραμματιστές να εκφράσουν έννοιες σε λιγότερες
γραμμές κώδικα απ'ότι θα ήταν δυνατόν σε Γλώσσες όπως η C++ ή η Java.
[3][4] Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που
διευκολύνουν ΙΔΙΑΙΤΕΡΑ αρκετές συνηθισμένες εργασίες και
για την ταχύτητα εκμάθησης της.
'''

# Εύρεση κεφαλαίων Με τη χρήση του τελεστή in & με διάσχιση
count_caps = 0
caps = []
for letter in my_text:
```

```
    if letter in english_caps:
        caps.append(letter)
        count_caps += 1

print 'Found %d English Capital letters ' % count_caps
for item in caps:
    print item,
print ""

# Εύρεση κεφαλαίων Χωρίς τον τελεστή in & με slicing
count_caps = 0
caps = []
for i in range(len(my_text)):
    for j in range(len(english_caps)):
        if my_text[i] == english_caps[j]:
            caps.append(my_text[i])
            count_caps += 1

print 'Found %d English Capital letters ' % count_caps
for item in caps:
    print item,
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 3 /σελίδα 151
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και
θα την εμφανίζει αντεστραμμένη, με τη χρήση μιας στοίβας.

'''

word = raw_input('Δώσε μία λέξη να έρθουν τα πάνω κάτω =')

stack = []

# Βασικές λειτουργίες στοίβας
def push(stack, item) :
    stack.append(item)
    return stack

def pop(stack) :
    return stack.pop()

def is_empty(stack) :
    return len(stack) == 0

# Fill Stack with Letters
for letter in word:
    push(stack, letter)

# Prepare reverse word
wordR =''
while is_empty(stack) == False :
    letter = pop(stack)
    wordR += letter
print wordR

# Pythonic Way
print word[::-1]
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 4 /σελίδα 151
Να γράψετε μια συνάρτηση is_substring(string, substring) η οποία
θα ελέγχει, αν η συμβολοσειρά substring περιέχεται στην συμβολοσειρά
string και αν ναι, θα επιστρέφει True. Στη συνέχεια να
χρησιμοποιήσετε αυτή τη συνάρτηση, για να βρείτε
πόσες φορές εμφανίζεται μια συμβολοσειρά μέσα σε μια άλλη.

'''

# Η λύση εδώ παρουσιάζεται με μία αλλαγή καθότι η 1η συνάρτηση δεν
έχει ιδιαίτερο
# νόημα εφόσον ο τελεστης in (substring in string) δίνει τη λύση
# έτσι υλοποιούμε μία  συνάρτηση η οποία βρίσκει όχι μόνο πόσες φορές
# αλλά και σε ποιες θέσεις εμφανίζεται η substring μέσα στην string
# χρησιμοποιώντας την τεχνική slicing

# Δεδομένα δοκιμής
my_text =
'hellotherethisisascriptinpythonlanghellotherethisisascriptinpythonlang
'


# Αλγοριθμικός τρόπος
def is_substring(substring, string):
    n = len(substring)
    N = len(string)
    found = False
    start = 0
    end = n
    position = []   # λίστα θέσεων εμφάνισης της substring
    while end <= N :
        part = string[start:end] # τεμαχιζουμε την string σε φέτες
        μεγέθους n
        if substring == part :
            position.append(start)
        start, end = start + 1, end + 1
    print
```

```
    print 'Η υπο-συμβολοσειρά "%s" βρέθηκε στη συμβολοσειρά "%s"
    ακριβώς %d φορές στις \
θέσεις %s' % (substring, string, len(position), position)
    # return position

# Pythonic Way μόνο για το πλήθος όχι για τις θέσεις
def is_substring2(substring, string):
    return string.count(substring)

#-------------------------------------------------------------
 # Δοκιμαστική κλήση συνάρτησης
is_substring('hello', my_text)
is_substring2('hello', my_text)
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 5 /σελίδα 151
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς από το
πληκτρολόγιο, μέχρι να δοθεί ο αριθμός 0. Κάθε φορά που θα
διαβάζει έναν θετικό αριθμό, θα τον προσθέτει σε μια στοίβα.
Όταν διαβάζει έναν αρνητικό αριθμό θα αφαιρεί τόσους αριθμούς
από τη στοίβα, όσο είναι η τιμή του αριθμού.
Ο αλγόριθμος θα τερματίζει όταν αδειάσει η στοίβα.

'''
# ή όταν δοθεί ο αριθμός 0 μηδέν

stack = []

# Βασικές λειτουργίες στοίβας
def push(stack, item):
    stack.append(item)
    return stack

def pop(stack):
    return stack.pop()

def isEmpty(stack):
    return len(stack) == 0

# Εισαγωγή 1ου θετικού ακέραιου με έλεγχο δεδομένων
num = int(input('Δώσε έναν θετικό ακέραιο αριθμό='))
while num <=0:
    num = int(input('Δώσε έναν θετικό ακέραιο αριθμό='))
push(stack,num)


# Εισαγωγή υπολοίπων ακεραίων
while num != 0 and len(stack) != 0:
    num = int(input('Δώσε έναν ακέραιο αριθμό=')    )
    # εάν είναι θετικός τότε push
    if num > 0:
```

```
        push(stack,num)
    # εάν είναι αρνητικός
    elif num < 0:
        # κάνε έλεγχο εάν έχει στοιχεία η στοίβα
        if len(stack) >= abs(num):
            # και κάνε pop
            for i in range(abs(num)):
                item = pop(stack)
                print item,
        else :
            # Δεν επαρκούν τα στοιχεία που ζητήθηκαν
            print 'Not enough items in stack poping the items left'
            # Κάνε pop μόνο αυτά που έμειναν
            for i in range(len(stack)):
                item = pop(stack)
                print item,
        print
    print stack
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 6 /σελίδα 151
Να γράψετε πρόγραμμα στη γλώσσα Python το οποίο θα δέχεται ως
είσοδο ένα κείμενο και θα εμφανίζει πόσες φορές εμφανίζεται κάθε
γράμμα του αγγλικού αλφαβήτου σε αυτό.
Να χρησιμοποιήσετε ένα λεξικό.

'''

# Δεδομένα Δοκιμών ... κείμενο εισόδου
my_text ='''
 Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thought Alice `without pictures or conversation?'
  So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble
of getting up and picking the daisies, when suddenly a White
Rabbit with pink eyes ran close by her.
  There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to
itself, `Oh dear!  Oh dear!  I shall be late!'  (when she thought
it over afterwards, it occurred to her that she ought to have
wondered at this, but at the time it all seemed quite natural);
but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
POCKET, and looked at it, and then hurried on, Alice started to
her feet, for it flashed across her mind that she had never
before seen a rabbit with either a waistcoat-pocket, or a watch to
take out of it, and burning with curiosity, she ran across the
field after it, and fortunately was just in time to see it pop
down a large rabbit-hole under the hedge.
'''

# Εισαγωγή βιβλιοθήκης για αλφάβητο λατινικών χαρακτήρων
import string
```

```
letters = string.ascii_lowercase

# Επίλυση Με χρήση λεξικού
dic = {}
for letter in letters:
    dic[letter] = 0

for char in my_text:
    if dic.get(char.lower()) != None :
        dic[char.lower()] += 1

# Εκτύπωση λεξικού συχνότητας εμφάνισης χαρακτήρων
print dic

# Επίλυση και εκτύπωση συχνοτήτων χωρίς χρήση λεξικού
for letter in letters :
    f = 0  # Συχνότητα εμφάνισης γράμματος
    for char in my_text :
        if letter == char.lower():
            f += 1
    print letter, f
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 7 /σελίδα 151
Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα
από λέξεις και θα επιστρέφει τη λέξη με το μεγαλύτερο μήκος.

'''


# Δεδομένα Δοκιμών ... κείμενο εισόδου
my_text = ''' Either the well was very deep or she fell very
slowly for she
had plenty of time as she went down to look about her and to
wonder what was going to happen next  First she tried to look
down and make out what she was coming to but it was too dark to
see anything then she looked at the sides of the well and
noticed that they were filled with cupboards and bookshelves
here and there she saw maps and pictures hung upon pegs  She
took down a jar from one of the shelves as she passed it was
labelled ORANGE MARMALADE but to her great disappointment it
was empty she did not like to drop the jar for fear of killing
somebody so managed to put it into one of the cupboards as she
fell past it'''

# Δημιουργία λίστας λέξεων από κείμενο με τη μέθοδο split
# για να αποφύγουμε κοπιαστικό data entry
words = my_text.split()

# Εύρεση λέξης μεγαλύτερου μήκους
xmax = 0
xword = ''
for word in words :
    if len(word) > xmax :
        xmax = len(word)
        xword = word

print 'Η λέξη "%s" είναι η μεγαλύτερη με %d γράμματα' % (xword, xmax)
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 8ο ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΙΙ
Δραστηριότητα 8 /σελίδα 152
Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λέξη
και θα επιστρέφει το πλήθος των φωνηέντων που έχει. Στη συνέχεια,
να γράψετε μια δεύτερη συνάρτηση, η οποία θα δέχεται μια λίστα
από λέξεις και θα επιστρέφει τη λέξη με τα περισσότερα φωνήεντα.

'''

# Δεδομένα Δοκιμών ... κείμενο εισόδου
my_text = ''' Either the well was very deep or she fell very
slowly for she
had plenty of time as she went down to look about her and to
wonder what was going to happen next  First she tried to look
down and make out what she was coming to but it was too dark to
see anything then she looked at the sides of the well and
noticed that they were filled with cupboards and bookshelves
here and there she saw maps and pictures hung upon pegs  She
took down a jar from one of the shelves as she passed it was
labelled ORANGE MARMALADE but to her great disappointment it
was empty she did not like to drop the jar for fear of killing
somebody so managed to put it into one of the cupboards as she
fell past it'''

# Δημιουργία λίστας λέξεων από κείμενο με τη μέθοδο split
# για να αποφύγουμε κοπιαστικό data entry
words = my_text.split()
vowels = 'aoieyu'    # Λατινικά φωνήεντα

# Συνάρτηση καταμέτρησης φωνηέντων σε μία λέξη
def count_vowels(word):
    count = 0
    for letter in word:
        if letter.lower() in vowels :
            count += 1
    return count
```

```
# Συνάρτηση που δέχεται μια λίστα από λέξεις
# και  επιστρέφει τη λέξη με τα περισσότερα φωνήεντα
def max_vowels_word(words):
    xmax = 0
    xword = ''
    for word in words:
        vowels = count_vowels(word)
        if vowels > xmax:
            xmax = vowels
            xword = word
    return xword, xmax

#-------------------------------------------------------------
# Δοκιμαστική κλήση συναρτήσεων
xword, xmax = max_vowels_word(words)
print 'Η λέξη με τα περισσότερα φωνήεντα είναι η %s και είναι %d '
%(xword, xmax)
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 11ο ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Δραστηριότητα 1 /σελίδα 221
Να ορίσετε κλάση με όνομα Akeraios, η οποία θα έχει την ιδιότητα
timi και τις παρακάτω μεθόδους:
I. Anathese_timi(timi), η οποία θα αναθέτει τιμή στην ιδιότητα του
αντικειμένου.
II. emfanise_timi(), η οποία θα εμφανίζει την τιμή της ιδιότητας
του αντικειμένου. Στη συνέχεια να δημιουργήσετε στιγμιότυπο της
κλάσης Akeraios με όνομα Artios και να χρησιμοποιήσετε τις
παραπάνω μεθόδους για να δώσετε την τιμή 14 στην
ιδιότητα του αντικειμένου και να την εμφανίσετε.

'''

# Δημιουργία Κλάσης ακέραιος
class Akeraios:
    def __init__(self,timi):
        self.timi = timi
    def anathese_timi(self, timi):
        self.timi = timi
    def emfanise_timi(self):
        print self.timi
    # Επέκταση της κλάσης ώστε να ελέγχει εάν το αντικείμενο είναι
    περιττός...
    def is_odd(self):
        if self.timi % 2 == 1:
            return True
        else :
            return False
    # ... ή Άρτιος ακέραιος
    def is_even(self):
        if self.timi % 2 == 0:
            return True
        else :
            return False

#------------------------------------------------------------
```

```
# Δοκιμαστικές κλήσεις της κλάσης
artios = Akeraios(0)
artios.emfanise_timi()
artios.anathese_timi(14)
artios.emfanise_timi()
x,y = artios.is_even(), artios.is_odd()
print x,y
#-------------------------------------------------------------
```

```
# Επίλυση Δραστηριοτήτων του μαθήματος Προγραμματισμός Υπολογιστών
# Γ' ΕΠΑΛ
# ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
# worked with Python 2.7.10
# @TasosChatz

'''

ΚΕΦΑΛΑΙΟ 11ο ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Δραστηριότητα 2 /σελίδα 222
Να ορίσετε κλάση με όνομα Student η οποία θα έχει τρεις ιδιότητες
για τον αριθμό μητρώου, το ονοματεπώνυμο και τους βαθμούς σε
8 μαθήματα (πίνακας). Επίσης να ορίσετε τις παρακάτω μεθόδους
της κλάσης:
I. Μια μέθοδο για την ανάθεση τιμών στις ιδιότητες ενός
αντικειμένου.
II. Μια μέθοδο για την εμφάνιση των τιμών των ιδιοτήτων
ενός αντικειμένου.
III. Μια μέθοδο για τον υπολογισμό και την επιστροφή του
μέσου όρου βαθμολογίας στα 8 μαθήματα.
Στη συνέχεια να ορίσετε ένα στιγμιότυπο της κλάσης Student
με όνομα Chris και να χρησιμοποιήσετε τις παραπάνω μεθόδους
για να δώσετε τιμή στις ιδιότητες του αντικειμένου,
να τις εμφανίσετε και να υπολογίσετε το μέσο όρο βαθμολογίας του.

'''

# Δημιουργία Κλάσης φοιτητή
class Student:
    def __init__(self, am, name, grades):
        self.am = am
        self.name = name
        self.grades = grades
    def update_student(self, am, name, grades):
        self.am = am
        self.name = name
        self.grades = grades
    def print_student(self):
        print self.am, self.name, self.grades
    def calc_average(self):
        xsum=0
        for grade in self.grades:
            xsum += grade
        return xsum / float(len(self.grades))
```

```
#-------------------------------------------------------------
# Δοκιμαστικές κλήσεις της κλάσης
chris = Student('1234','christopher robert',[12,11,14,15,10,16,17,18])
chris.print_student()
chris.update_student('4321','Christofer
Robert\'s',[12,11,14,15,20,16,17,18])
chris.print_student()
avg = chris.calc_average()
print 'Μέσος όρος βαθμολογίας για %s είναι %.2f ' % (chris.name, avg)
#-------------------------------------------------------------
```

Αποποίηση ευθύνης

Παρά το γεγονός ότι έγινε έλεγχος των αλγορίθμων των λύσεων, καμιά εγγύηση εξασφάλισης ως προς την πλήρη ορθότητά τους δεν μπορεί να δοθεί. Οι προτάσεις επίλυσης των προβλημάτων δεν έγιναν με στόχο την αλγοριθμική βελτιστοποίησή τους, είναι υποκειμενικές, μπορεί να περιέχουν λάθη και δεν εκφράζουν το σύνολο της επιστημονικής κοινότητας. Διατηρείται το δικαίωμα το περιεχόμενο να αλλάξει, να διαγραφεί προσωρινά ή οριστικά, εξ ολοκλήρου ή εν μέρει ανά πάσα στιγμή και χωρίς προειδοποίηση. Οι αξιώσεις ευθύνης λόγω κάθε είδους ζημιών άυλων ή υλικών που προκύπτουν από τη χρήση του παρόντος αποκλείονται. Δεν φέρεται καμία ευθύνη για αναδημοσιεύσεις του περιεχομένου σε ιστοσελίδες τρίτων μέσω συνδέσμων ή μεταφορτώσεων ή και διανομής με τη μορφή κάθε είδους έντυπου υλικού. Το περιεχόμενο του παρόντος χρησιμοποιείται αποκλειστικά με δική σας ευθύνη.

ISBN 978-960-93-8739-2

Αναστάσιος Χατζηπαπαδόπουλος

ISBN 978-960-93-8739-2