Κεφάλαιο 4. Δραστηριότητες

Δραστηριότητα 2

Να γραφεί πρόγραμμα σε γλώσσα Python που να διαβάζει τρεις ακεραίους αριθμούς και να υπολογίζει το μέσο όρο τους. Το αποτέλεσμα να εμφανίζεται στην οθόνη.

Απάντηση

Για να υπολογίσουμε το άθροισμα πολλών αριθμών τότε αρχικά δίνουμε την τιμή 0 στη μεταβλητή που θα υπολογίζει το άθροισμα σταδιακά

```
suma = 0.0
for i in range (3) :
   x = float (input ('Δώσε έναν αριθμό : '))
   suma = suma + x
average = suma / 3.0
print 'Ο μέσος όρος των 3 αριθμών είναι : ', average
```

Για τον υπολογισμό του μέσου όρου πρέπει ένας από τους αριθμούς της διαίρεσης να είναι αριθμός με δεκαδικό μέρος για να προκύψει σωστό αποτέλεσμα, καθώς ο τελεστής '/' επιστρέφει το ακέραιο πηλίκο μιας διαίρεσης ακεραίων.
Έτσι όταν οι αριθμοί είναι ακέραιοι τότε συνηθίζεται στην εύρεση του μέσου όρου να χρησιμοποιούμε έναν αριθμό με δεκαδικά ψηφία:
π.χ. για 3 αριθμούς mo = (a + b +c) / 3.0 ή mo = (a + b + c) / float(3)

Δραστηριότητα 3

Να γραφεί πρόγραμμα σε γλώσσα Python που να επιλύει την εξίσωση Ax+B = 0, για A και B πραγματικούς αριθμούς.

Απάντηση

```
a, b = input ('Δώσε τα α, β για την εξίσωση αx + β = 0 : ')
# εναλλακτικά
# a = input ('Για την εξίσωση αx + β = 0 δώσε το α : ')
# b = input ('και το β  : ')
if a != 0 :                          # α != 0
   x = (-1.0) * b / a
   print 'Η εξίσωση έχει λύση την x = ', x
elif b == 0 :                        # α = 0 και β = 0
   print 'Η εξίσωση είναι αόριστη'
else :                               # α = 0 και β <> 0
   print 'Η εξίσωση είναι αδύνατη'
```

Δραστηριότητα 4

Σε ένα ηλεκτρικό κύκλωμα υπάρχουν δύο αντιστάσεις R1 και R2. Θέλουμε να βρούμε την ολική αντίσταση R του κυκλώματος. Υπάρχουν δύο τρόποι σύνδεσης αυτών των αντιστάσεων : σε σειρά ή παράλληλα. Στην πρώτη περίπτωση, ο τύπος που δίνει την ολική αντίσταση είναι R = R1+R2. Στην παράλληλη σύνδεση, η ολική αντίσταση βρίσκεται από τον τύπο 1/R = (1/R1)+ (1/R2).
Να γραφεί πρόγραμμα σε γλώσσα Python που να :

• διαβάζει τις τιμές των αντιστάσεων R1 και R2
• διαβάζει τον τρόπο σύνδεσης, δίνοντας την επιλογή 1 για σύνδεση σε σειρά ή τη για παράλληλη σύνδεση, των αντιστάσεων R1 και R2
• υπολογίζει, εφαρμόζοντας τον κατάλληλο τύπο ανάλογα με τον τρόπο σύνδεσης, την τιμή της ολικής αντίστασης R
• εμφανίζει τις τιμές των αντιστάσεων R1, R2, καθώς και την τιμή της αντίστασης R.

Απάντηση

```
R1 = float (input ('Δώσε τις τιμές της αντίστασης R1 : '))
R2 = float (input ('Δώσε τις τιμές της αντίστασης R2 : '))
syndesh = input ("Δώσε τον τρόπο σύνδεσης των αντιστάσεων :  '1' για σύνδεση σε σειρά
                                        και '2' για σύνδεση παράλληλα  :  ")
if syndesh == 1 :
   R = R1 + R2
else :
   R = R1 * R2 / (R1 + R2)
print 'Η ολική αντίσταση των αντιστάσεων ', R1, ' και ', R2, ' είναι  :  ', R, ' Ω'
```

Δραστηριότητα 5

Να γραφεί πρόγραμμα σε γλώσσα Python που να βρίσκει σε ποιον όρο το άθροισμα 1+2+3+4+… γίνεται μεγαλύτερο του 2000.

Απάντηση

```
suma = 0
plithos = 0
while suma <= 2000 :
  plithos = plithos + 1
  suma = suma + plithos
print 'Το άθροισμα της αριθμητικής προόδου 1+2+3+... γίνεται 2000 στο όρο  :  ' , plithos
```

Δραστηριότητα 6

Να γραφεί πρόγραμμα σε γλώσσα Python που να δέχεται τριψήφιο θετικό ακέραιο και να ελέγχει αν είναι *παλίνδρομος*, δηλαδή αριθμός που παραμένει ο ίδιος, όταν αναστρέψουμε τη σειρά των ψηφίων του, όπως για παράδειγμα ο 131 ή ο 797.

Απάντηση

```
x = input ('Δώσε έναν τριψήφιο θετικό ακέραιο αριθμό  :  ')
y = x
x0 = y % 10    # απομόνωση των μονάδων
y = y // 10     # διώξιμο των μονάδων
x1 = y % 10    # απομόνωση των δεκάδων
x2 = y // 10    # απομόνωση των εκατοντάδων
if x0 == x2 :
   print 'Ο αριθμός ', x, ' είναι παλίνδρομος'
else :
   print 'Ο αριθμός ', x, ' δεν είναι παλίνδρομος'
```

Δραστηριότητα 7

Να γραφεί πρόγραμμα σε γλώσσα Python που υλοποιεί τον υπολογισμό της ακολουθίας Fibonacci.

Απάντηση

```
n = input ('Δώσε έναν φυσικό αριθμό  :  ')
if n <= 1 :
  Fib = n
f0 = 0
f1 = 1
for i in range (2, n+1) :
  Fib = f0 + f1
  f0 = f1
  f1 = Fib
print 'Η ακολουθία Fibonacci του ', n, ' είναι ο ', Fib
```

Δραστηριότητα 8

Να γραφεί πρόγραμμα σε γλώσσα Python που να υλοποιεί τον υπολογισμό της δύναμης χωρίς χρήση της συνάρτησης pow.

Απάντηση

```
a = input ('Δώσε τη βάση της δύναμης : ')
n = input ('Δώσε τον εκθέτη της δύναμης : ')
power = 1
if n != 0 :
  for i in range (n) :
    power = power * a
print 'Η δύναμη του ', a, ' στην  ', n, ' είναι : ', power
```

Δραστηριότητα 9

Να γραφεί πρόγραμμα σε γλώσσα Python που να εξετάζει και να εμφανίζει πόσοι από τους αριθμούς που υπάρχουν μεταξύ του 50 και του 500 είναι πολλαπλάσια του 3.

Απάντηση

```
plithos_multi = 0
for i in range (50, 501):
   if i % 3 == 0 :
      plithos_multi = plithos_multi + 1
print 'Το πλήθος των πολλαπλασίων του 3 είναι : ', plithos_multi
```

Δραστηριότητα 10

Να ορίσετε κατάλληλα μια συνάρτηση *παραγοντικό*, που να υπολογίζει και να τυπώνει το Ν! ενός θετικού αριθμού Ν.

Ν != 1, αν Ν = 0

Ν != 1*2*3*…. (Ν-1)*Ν, αν Ν > 0

Στη συνέχεια να γραφεί πρόγραμμα που να διαβάζει έναν ακέραιο θετικό αριθμό α, να γίνεται έλεγχος ώσπου να δοθεί αριθμός > = 0 και στη συνέχεια, καλώντας τη συνάρτηση *παραγοντικό* να υπολογίζει και να εμφανίζει τον παραγοντικό αριθμό του α.

Απάντηση

```
def parag(N):
  product = 1
  for i in range (2,n+1) :
    product = product * i
  return product

n = input ('Δώσε ένα φυσικό αριθμό διάφορο του 0 : ')
while n == 0 :
   n = input ('Δώσε ένα φυσικό αριθμό διάφορο του 0 : ')
paragontiko = parag (n)
print 'Το παραγοντικό του ', n, ' είναι: ', paragontiko
```

Κεφάλαιο 5. Δραστηριότητες

Δραστηριότητα 1. Βασικές επεξεργασίες λιστών
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο μεχρι το άθροισμά τους να ξεπεράσει το 1000, θα τους αποθηκεύει σε μια λίστα και στη συνέχεια θα υπολογίζει και θα εμφανίζει :
..
Απάντηση
Για να υπολογίσουμε των μέγιστο ή τον ελάχιστο πολλών αριθμών τότε αρχικά δίνουμε τον πρώτο αριθμό ως τιμή στην μεταβλητή που θα υπολογίζει το μέγιστο ή τον ελάχιστο αριθμό σταδιακά

```
suma = 0.0
n = 0
A = []
A.append (input ('Δώσε έναν αριθμό : '))
maxi = A[0]
while A[n] + suma <= 1000 :
  suma = suma + A[n]
  if A[n] > maxi :
    maxi = A[n]
    plithos_maxi = 1
  elif A[n] == maxi :
    plithos_maxi = plithos_maxi + 1
  A.append (input ('Δώσε τον επόμενο αριθμό : '))
  n = n + 1
if len (A) != 0 :
  average = suma / len (A)
print 'Ο μέγιστος των αριθμών είναι ο : ',  maxi,  'και το πλήθος των εμφανίσεων του είναι : ',
                                                                                    plithos_maxi
print 'Το άθροισμα των αριθμών είναι : ',  suma
print 'Ο μέσος όρος των αριθμών είναι : ',  average
```

Δραστηριότητα 2

Να δημιουργήσετε τη δομή δεδομένων στοίβα με τη χρήση λίστας, υλοποιώντας τις λειτουργίες push (ώθηση), pop (απώθηση) για λίστα.

Απάντηση

```
# ΠΡΟΓΡΑΜΜΑ ΩΘΗΣΗ ΚΑΙ ΑΠΩΘΗΣΗ
def push (stack, item) :          # ώθηση στοιχείου
   stack.append (item)
def pop (stack) :                 # απώθηση στοιχείου
   return stack.pop ()
def isEmpty (stack) :             # έλεγχος για ύπαρξη στοιχείων
   return len (stack) == 0
def createStack () :              # δημιουργία στοίβας με λίστα
   return [ ]

N =  input ('Δώσε το μήκος της στοίβας  : ')
A = createStack ()
top = 0
menu = ' '
while menu !=  '*' :
   print 'Δώσε ένα από τα παρακάτω σύμβολα για την αντίστοιχη λειτουργία της στοίβας : '
   print ' +  : Ώθηση'
   print ' -  : Απώθηση'
   print ' *  : Έξοδος'
   print
   menu = raw_input ('Δώσε τη επιλογή ( +, -, *) : ')
   while menu != '+' and menu != '-' and menu != '*' :
      print 'Λάθος επιλογή. Ξαναπροσπάθησε!!!'
      menu = raw_input ('Δώσε τη επιλογή ( +, -, *) : ')
   if menu == '+' :               # Ώθηση στοιχείου
      if top < N :
         item = input ('Δώσε το στοιχείο για ώθηση : stack [' + str  (top + 1) +  '] = ')
         push (A, item)
         top = top + 1
      else :
         print 'Η στοίβα είναι γεμάτη : ΥΠΕΡΧΕΙΛΙΣΗ ΣΤΟΙΒΑΣ !!!'
         print
   elif menu == '-' :             # Απώθηση στοιχείου
      if not isEmpty (A) :
         print 'Στοιχείο για απώθηση : stack [' + str (top + 1) +  '] = ', pop (A)
         top = top - 1
      else :
         print 'Η στοίβα είναι άδεια : ΥΠΟΧΕΙΛΙΣΗ ΣΤΟΙΒΑΣ !!!'
         print
   elif menu == '*' :
      print 'ΕΞΟΔΟΣ'
```

Δραστηριότητα 3

Να δημιουργήσετε τη δομή δεδομένων ουρά με τη χρήση λίστας, υλοποιώντας τις λειτουργίες enqueue (εισαγωγή), dequeue (εξαγωγή) για λίστα.

Απάντηση

```
# ΠΡΟΓΡΑΜΜΑ ΕΙΣΑΓΩΓΗ_ΚΑΙ_ΕΞΑΓΩΓΗ
def enqueue (queue, item) :          # εισαγωγή στοιχείου
   queue = queue.append (item)
def dequeue (queue) :                # εξαγωγή στοιχείου
   return queue.pop (0)
def isEmpty (queue) :                # έλεγχος για ύπαρξη στοιχείων
   return len (queue) == 0
def createQueue () :                 # δημιουργία ουράς  με λίστα
   return [ ]
N =  input ('Δώσε το μήκος της ουράς : ')
A = createQueue ()
front = 0
rear = 0
menu = ' '
while menu !=  '*' :
   print 'Δώσε ένα από τα παρακάτω σύμβολα για την αντίστοιχη λειτουργία της ουράς : '
   print ' +  : Εισαγωγή'
   print ' -  : Εξαγωγή'
   print ' *  : Έξοδος'
   print
   menu = raw_input ('Δώσε τη επιλογή ( +, -, *) : ')
   while menu != '+' and menu != '-' and menu != '*' :
      print 'Λάθος επιλογή. Ξαναπροσπάθησε!!!'
      menu = raw_input ('Δώσε τη επιλογή ( +, -, *) : ')
   if menu == '+' :                  # Εισαγωγή στοιχείου
      if rear < N :
         item = input ('Δώσε το στοιχείο για εισαγωγή : queue [' + str (rear + 1) +  '] = ')
         enqueue (A, item)
         rear = rear + 1
      else :
         print 'Η ουρά είναι γεμάτη : ΥΠΕΡΧΕΙΛΙΣΗ ΟΥΡΑΣ !!!'
         print
   elif menu == '-' :                # Εξαγωγή στοιχείου
      if not isEmpty (A) :
         print 'Στοιχείο για εξαγωγή: ', dequeue (A)
         front = front + 1
      else :
         print ' Η ουρά είναι άδεια : ΥΠΟΧΕΙΛΙΣΗ ΟΥΡΑΣ !!!'
         print
   elif menu == '*' :
      print 'ΕΞΟΔΟΣ'
```

Δραστηριότητα 4

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο, μέχρι να δοθεί ο αριθμός 0. Στη συνέχεια με τη χρήση λίστας θα εμφανίζει τους αριθμούς σε αντίστροφη σειρά από αυτή με την οποία τους διάβασε.

Απάντηση

```
A = []
x = input  ('Δώσε έναν αριθμό ή το 0 για τέλος : ')
while x != 0 :
  A.append (x)
  x = input ('Δώσε τον επόμενο αριθμό ή το 0 για τέλος : ')
print 'Οι αριθμοί που έδωσες με την αντίστροφη σειρά είναι :'
for i in range (len (A)-1, -1, -1) :
   print A[i],
```

Δραστηριότητα 5

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μια λίστα από 100 αριθμούς και θα διαχωρίζει τους αριθμούς σε δυο νέες λίστες, μια για τους θετικούς και μια για τους αρνητικούς. Οι αριθμοί πρέπει να παραμείνουν στη σειρά με την οποία δόθηκαν.

Απάντηση

```
A = []
POS = []
NEG = []
N = input ('Δώσε το πλήθος των αριθμών  :  ')

for i in range (N) :
  A.append (input ('Δώσε έναν αριθμό  :  '))
  if A[i] >= 0 :
    POS.append (A[i])
  else :
    NE.append (A[i])
print 'Η λίστα με τους αριθμούς που έδωσες είναι  : '
for i in range (N) :
  print A[i],
print
print 'Η λίστα με τους θετικούς αριθμούς που έδωσες είναι  : '
for i in range (len (POS)) :
   print POS[i],
print
print 'Η λίστα με τους αρνητικούς αριθμούς που έδωσες είναι  : '
for i in range (len (NEG)) :
   print NEG[i],
print
```

Δραστηριότητα 6

Τα βιβλία αναγνωρίζονται από ένα μοναδικό κωδικό ο οποίος είναι γνωστός ως ISBN. Ωστόσο τα τελευταία χρόνια χρησιμοποιείται και ένας άλλος κωδικός, ο ISSN. Να γράψετε πρόγραμμα το οποίο θα διαβάζει τους κωδικούς βιβλίων μέχρι να δοθεί ο κωδικός “0000” και στη συνέχεια θα διαχωρίζει σε δύο λίστες, τα βιβλία με κωδικό ISBN και εκείνα με ISSN.

Υπόδειξη : Να επεξεργαστείτε κάθε κωδικό σαν συμβολοσειρά (str) και να ελέγξετε το πρόθεμά του. Για παράδειγμα, σε ένα βιβλίο με ISBN, το ISBN θα καταλαμβάνει τις τέσσερις πρώτες θέσεις της συμβολοσειράς.

Απάντηση

```
ISSN = []
ISBN = []
b = raw_input ('Δώσε τον κωδικό ISSN ή ISBN βιβλίου ή το "0000" για τέλος : ')
while b != '0000' :
   if b[0:4] == 'ISSN' :
      ISSN.append (b)
   else :
      ISBN.append (b)
   b = raw_input ('Δώσε τον κωδικό ISSN ή ISBN βιβλίου ή το "0000" για τέλος : ')
 print 'Τα βιβλία με κωδικό ISSN είναι : '
print ISSN
print 'Τα βιβλία με κωδικό ISSN είναι : '
print ISBN
```

Δραστηριότητα 7. Αλγόριθμος Συγχώνευσης διατεταγμένων λιστών

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει δύο λίστες με αριθμούς. Η πρώτη λίστα περιέχει μόνο θετικούς και η δεύτερη μόνο αρνητικούς. Η εισαγωγή δεδομένων για κάθε λίστα τερματίζεται όταν δοθεί ο αριθμός 0, ο οποίος δεν πρέπει να αποθηκευθεί σε κάποια λίστα. Οι αριθμοί κάθε λίστας δίνονται σε αύξουσα σειρά, π.χ. 4, 6, 10, 145 και –100, –98, –10. Στη συνέχεια, το πρόγραμμα θα συγχωνεύει τις δύο λίστες σε μία, έτσι ώστε στην τελική λίστα οι αριθμοί να βρίσκονται σε αύξουσα σειρά κατά απόλυτη τιμή. Αν δύο ετερόσημοι αριθμοί έχουν την ίδια απόλυτη τιμή, υπερισχύουν οι θετικοί., π.χ. 4, 6, 10, –10, 98, –100, 145.

Απάντηση

```
POS = []
NEG = []
x = input ('Δώσε έναν αριθμό ή το 0 για τέλος : ')
while x != 0 :
  if x > 0 :
    POS.append (x)
  else :
    NEG.append (x)
  x = input  ('Δώσε τον επόμενο αριθμό ή το 0 για τέλος : ')
```

```
L = []
while POS != [ ] and NEG != [ ] :       # όσο οι δυο λίστες έχουν στοιχεία
  # Αν το 1ο στοιχείο της POS είναι μικρότερο από το 1ο στοιχείο της NEG κατά απόλυτες τιμές
  if abs (POS[0]) < abs (NEG[0]) :
      L.append (POS.pop (0))          # μεταφέρουμε το πρώτο στοιχείο της POS στο τέλος της L
  # Αν το 1ο στοιχείο της POS είναι ίσο με το 1ο στοιχείο της NEG κατά απόλυτες τιμές
  elif abs (POS[0]) == abs (NEG[0])
    # μεταφέρουμε το πρώτο στοιχείο της POS στο τέλος της L και μετά το πρώτο στοιχείο της NEG
      L.append (POS.pop (0))
      L.append (NEG.pop (0))
  else :
      L.append (NEG.pop (0))          # αλλιώς μεταφέρουμε το πρώτο στοιχείο της NEG στην L
# Αφού αφαιρούμε το στοιχείο[0] μιας λίστα το στοιχείο[1]
      # έρχεται αριστερά και γίνεται το νέο στοιχείο[0]

L = L + POS + NEG      # Στο τέλος προσθέτουμε τα στοιχεία που έχουν μείνει
# εναλλακτικά
# if POS == [ ] :
#    L = L + NEG
# else :
#    L = L + POS
print L
```

Δραστηριότητα 8. Αλγόριθμος κρυπτογράφησης του Καίσαρα

Με βάση τον αλγόριθμο κρυπτογράφησης του Καίσαρα, όπως φαίνεται στο παρακάτω σχήμα κρυπτογράφησης, δεχόμαστε ότι κάθε γράμμα θα αντιστοιχιστεί στο γράμμα που βρίσκεται τρεις θέσεις πίσω του στο αλφάβητο. Σε αυτή την περίπτωση μπορούμε να ορίσουμε ένα λεξικό.
cipher = { ‘A’ :’X’, ‘B’ :‘Y’, ‘C’ : ‘Z’, ‘D’ : ‘A’, …, ‘X’ :‘U’}
Πλέον η κρυπτογράφησή ενός κειμένου text είναι πολύ απλή, όπως φαίνεται και στο παρακάτω τμήμα κώδικα :

```
def encryptCeasar (text, alphabet) :
    cipherText = “”
    for letter in text :
        if letter in alphabet :
            cipherText = cipherText + cipher[letter]
        else :
            cipherText = cipherText + letter
    return cipherText
```

Να σχεδιάσετε μια συνάρτηση που να λαμβάνει τη μετατόπιση-κλειδί και να παράγει το λεξικό αντιστοίχισης των γραμμάτων cipher. Στη συνέχεια, να υλοποιήσετε τις αντίστοιχες συναρτήσεις για την αποκρυπτογράφηση.

Υπόδειξη : Να αναπτύξετε μια συνάρτηση η οποία να δέχεται το λεξικό κρυπτογράφησης και να παράγει το λεξικό αποκρυπτογράφησης, έτσι ώστε να μπορεί να χρησιμοποιηθεί η συνάρτηση encryptCeasar για αποκρυπτογράφηση.

Δραστηριότητα 9. Συχνότητα γραμμάτων

Να γράψετε πρόγραμμα στη γλώσσα Python το οποίο θα δέχεται ως είσοδο ένα κείμενο και θα ελέγχει αν εμφανίζονται όλα τα γράμματα του αλφαβήτου σε αυτό με την ίδια ακριβώς συχνότητα, όπως για παράδειγμα συμβαίνει στη συμβολοσειρά ‘abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz’, όπου όλα τα γράμματα εμφανίζονται δύο φορές.

Απάντηση

```
plithos = 0
coumt = 0
alphabet = 'abcdefghijklmnopqrstuvwxyz'
s = raw_input ('Δώσε ένα κείμενο : ')
N = len (s)

IsAllLetters = True
for item in alphabet :
   if item not in s :
      IsAllLetters = False

if IsAllLetters :
   print 'Εμφανίζονται όλα τα γράμματα της Αλφαβήτου'
   for i in range (1, N) :
      if s[0] == s[i] :
         plithos += 1

      IsSameFrequance = True
      for i in range (1, N) :
         count = 0
         for j in range (1, N) :
            if i != j :
               if s[i] == s[j] :
                  count += 1
         if count != plithos :
            IsSameFrequance = False
   if IsAllLetters :
      print 'με την ίδια συχνότητα'
   else :
      print 'αλλά ΟΧΙ με την ίδια συχνότητα'

else :
   print 'ΔΕΝ εμφανίζονται όλα τα γράμματα της Αλφαβήτου'
```

Δραστηριότητα 10

Να γραφεί πρόγραμμα που να διαβάζει μια λέξη και να ελέγχει αν η λέξη είναι *καρκινική*, εμφανίζοντας ανάλογο μήνυμα. Δηλαδή εξετάζει αν η λέξη είναι *συμμετρική*, δηλαδή αν μπορεί να διαβαστεί είτε από την αρχή είτε από το τέλος, όπως για παράδειγμα οι λέξεις ΑΝΝΑ ή RADAR.

Απάντηση

```
word = raw_input ('Δώσε μια λέξη : ')
palindrome = True
N = len (word)
i = 0
while i < N/2 and palindrome :
   if word[i] != word[N-i-1] :
      palindrome = False
   i += 1
if palindrome == True :
   print 'Η λέξη είναι παλίνδρομη'
else :
   print 'Η λέξη δεν είναι παλίνδρομη'
```

Δραστηριότητα 11

Βρείτε τι κάνει το παρακάτω πρόγραμμα. Επαληθεύστε την υπόθεση σας εκτελώντας το στο προγραμματιστικό περιβάλλον της γλώσσας.

```
sqlist = [ ]
for x in range (1, 11) :
    sqlist.append (x*x)
print sqlist
```

Απάντηση

Εμφανίζει τη λίστα με τα τετράγωνα των αριθμών από το 1 ως και το 100 :
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Δραστηριότητα 12

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα από λέξεις και θα επιστρέφει τη λέξη που σχηματίζεται από τα πρώτα γράμματα των λέξεων.

Απάντηση

```
def acronymio (L) :
    N = len (L)
    print 'Η λέξη που σχηματζεται από τα αρχικά των λέξεων είναιι : '
    for i in range (N) :
        print L[i][0],
A = []
s = raw_input ('Γράψε μία λέξη : ')
while s != 'end' :
    A.append (s)
    s = raw_input ('Γράψε ακόμα μία λέξη ή τη λέξη <end> για τέλος : ')

acronymio (A)
```

Δραστηριότητα 13

Να γράψετε ένα πρόγραμμα σε Python, το οποίο θα διαβάζει από το πληκτρολόγιο μια λίστα από λέξεις μέχρι να δοθεί η λέξη ‘end’.

Στη συνέχεια θα εμφανίζει τη λέξη ή τις λέξεις με τα περισσότερα γράμματα (υπόδειξη : Κατασκευάστε μια λίστα με το πλήθος των γραμμάτων κάθε λέξης).

Επίσης, θα εμφανίζει το πλήθος των λέξεων που τελειώνουν με το γράμμα ‘s

Απάντηση

```
A = [ ]
L = [ ]
count = 0
maxi = 0
s = raw_input ('Γράψε μία λέξη : ')
while s != 'end' :
    if s[-1] == 's' :
        count += 1
    A.append (s)
    L.append (len (s))
    s = raw_input ('Γράψε ακόμα μία λέξη ή τη λέξη <end> για τέλος : ')
N = len (L)
for i in range (N) :
    if L[i] > maxi  :
        maxi = L[i]
        pos = i
print 'Η μεγαλύτερη λέξη είναι η : ', A[pos]
print 'Το πλήθος των λέξεων που τελειώνουν σε s είναι : ', count
```

Κεφάλαιο 6. Δραστηριότητες

Δραστηριότητα 1. Το κόσκινο του Ερατοσθένη

Να αναζητήσετε στο Διαδίκτυο πληροφορίες για το κόσκινο του Ερατοσθένη και στη συνέχεια, αφού περιγράψετε τη λειτουργία του στην τάξη, να υλοποιήσετε τον αλγόριθμο αυτό σε Python.

Απάντηση

```
F = []
for i in range (1, 1001) :
    f = True
    for j in range (2, i) :
        if i % j == 0 :
            f = False
    if f == True :
        F.append (i)
print 'Πρώτοι αριθμοί είναι οι : ', F
```

Δραστηριότητα 2

Να σχεδιάσετε έναν άλλον αλγόριθμο για τον υπολογισμό του Μέγιστου Κοινού Διαιρέτη δύο αριθμών, εκτελώντας εξαντλητική αναζήτηση των πιθανών διαιρετών μέχρι να βρεθεί ο μέγιστος. Έχει σημασία από ποιον αριθμό θα ξεκινήσουμε;

Απάντηση

```
x = int (input ('Δώσε τον 1ο φυσικό αριθμό : '))
y = int (input ('Δώσε τον 2ο φυσικό αριθμό : '))
if y > x :
    x, y = y, x

i = y
found = False
while not found and i > 0  :
    if x % i == 0 and y % i == 0  :
        print 'Ο ΜΚΔ των αριθμών', x, 'και', y, 'είναι ο', i
        found = True
    i = i – 1
```

Δραστηριότητα 3

Για δύο αριθμούς α και β, τα διαδοχικά βήματα του αλγόριθμου του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη είναι :

ΜΚΔ (3600, _____) = **ΜΚΔ** (1100, _____) = **ΜΚΔ** (_____, _____)
= **ΜΚΔ** (_____, 100) = **ΜΚΔ** (_____, _____) = 100

Να συμπληρώσετε τα κενά στα παραπάνω βήματα του αλγόριθμου.

Απάντηση

ΜΚΔ (3600, 1100) = **ΜΚΔ** (1100, 300) = **ΜΚΔ** (300, 200)
= **ΜΚΔ** (200, 100) = **ΜΚΔ** (100, 0) = 100

Λύση

```
def mkd (a,b) :
    while b > 0 :
        a, b = b, a % b
        print a, b
    return a

x = int (input ('Δώσε τον 1ο φυσικό αριθμό : '))
y = int (input ('Δώσε τον 2ο φυσικό αριθμό : '))
if y > x  :
    x, y = y, x
z = mkd (x, y)
print 'Ο ΜΚΔ των αριθμών', x, 'και', y, 'είναι ο', z
```

Δραστηριότητα 4

Να σχεδιάσετε μια συνάρτηση, η οποία θα δέχεται δύο λίστες αριθμών και θα εμφανίζει πόσοι αριθμοί εμφανίζονται και στις δύο λίστες.

Απάντηση

```
def ItemInCommon (A, B) :
  count = 0
  for item in A :
    if item in B :
      count += 1
  return count
```

Δραστηριότητα 5

Να σχεδιάσετε μια συνάρτηση η οποία θα δέχεται δύο λίστες αριθμών και θα εμφανίζει πόσοι αριθμοί εμφανίζονται μόνο σε μία λίστα.

Απάντηση

Ένας τρόπος είναι να βρούμε τα κοινά στοιχεία ανάμεσα στις δύο λίστες χρησιμοποιώντας τη συνάρτηση της προηγούμενης δραστηριότητα και να αφαιρέσουμε από το άθροισμα όλων των στοιχείων των 2 λιστών το διπλάσιο των κοινών στοιχείων, αφού τα κοινά στοιχεία εμφανίζονται και στις δύο λίστες.

```
def ItemNotInCommon (A, B) :
  plithos = len (A) + len (B) - 2 * ItemInCommon (A, B)
  return plithos
```

Δραστηριότητα 6

Να τροποποιήσετε τον αλγόριθμο της σειριακής αναζήτησης, ώστε να υπολογίζει πόσες φορές εμφανίζεται το ζητούμενο στοιχείο στον πίνακα και να εμφανίζει στην οθόνη όλες αυτές τις θέσεις.

Απάντηση

```
def linearSearchFind (A, key) :
    N = len (A)
    count = 0
    i = 1
    while i < N :
        if A[ i ] == key :
            print 'Το στοιχείο', key, 'βρέθηκε στη θέση', i
            count = count +1
        i = i + 1
    return count
```

Δραστηριότητα 7

Ας υποθέσουμε ότι τα στοιχεία ενός πίνακα, στον οποίο μας ζητείται να εκτελέσουμε αναζήτηση, είναι ακέραιοι αριθμοί ταξινομημένοι σε αύξουσα σειρά. Αν, καθώς σαρώνουμε τον πίνακα, βρεθούμε σε στοιχείο το οποίο είναι μικρότερο από αυτό που ψάχνουμε, αυτό σημαίνει ότι το ζητούμενο θα το είχαμε ήδη συναντήσει, αν υπήρχε.

Να τροποποιήσετε τον αλγόριθμο σειριακής αναζήτησης, ώστε, όταν διαπιστώσει ότι το ζητούμενο στοιχείο δεν υπάρχει στον πίνακα, να τερματίζει.

Απάντηση

```
def linearSearch_inSorted (A, key) :
    N = len (A)
    pos = -1
    i = 1
    while pos < 0 and i < N :
        if A[ i ] == key :
            pos = i
            return pos
        i = i + 1
    return pos
```

Δραστηριότητα 8

Να τροποποιήσετε τον αλγόριθμο της ταξινόμησης με επιλογή, ώστε να ταξινομεί έναν πίνακα ακεραίων σε φθίνουσα σειρά.

Απάντηση

Αν αλλάξουμε μόνο τον τελεστή σύγκρισης στην minPos από "<" σε ">", ώστε να βρίσκει το μεγαλύτερο αντί το μικρότερο, τότε και η συνάρτηση selectionSort θα ταξινομεί σε φθίνουσα αντί σε αύξουσα. Έτσι η posMin θα γίνει :

```
def posMax (start, size, A) :
    pos = start
    for i in range (start, size) :
        if A[ i ] > A[ pos ] :
            pos = i
    return pos
```

Δραστηριότητα 9

Να τροποποιήσετε τον αλγόριθμο της ταξινόμησης με επιλογή, ώστε σε κάθε βήμα του να βρίσκει το μικρότερο και το μεγαλύτερο, να τοποθετεί το μικρότερο στην αρχή του πίνακα και το μεγαλύτερο στο τέλος.

Απάντηση

Ο αλγόριθμος αυτός θα χρησιμοποιεί ακόμη μία συνάρτηση για να υπολογίζει τη θέση του μέγιστου στοιχείου του πίνακα.

```
def posMax (start, size, A) :
    pos = start
    for i in range (size-1, start, -1) :
        if A[ i ] < A[ pos ] :
            pos = i
    return pos
```

και η συνάρτηση selectionSort θα τροποποιηθεί ως εξής :

```
def selectionSort (A) :
    N = len (A)
    for i in range (0, N) :
        pos = posMin (i, N-i, A)
        A[i], A[pos] = A[pos], A[i]
        posMax (i, N-i, A)
        A[N-1-i], A[pos] = A[pos], A[N-1-i]
```

Κεφάλαιο 7. Δραστηριότητες

Δραστηριότητα

Τι πιστεύετε ότι θα συμβεί, όταν θα εκτελεστούν οι παρακάτω κώδικες. Τεκμηριώστε την άποψή σας.

A)

```
L = [i**2 for i in range (1, 11)]
# Δημιουργεί μία λίστα τετραγώνων των αριθμών 1 - 10
f = open ('output.txt', 'w')
for item in L :
    f.write (str (item) + '\n')
f.close ()
```

B)

```
f = open ('input.txt', 'w')
f.closed              # θα τυπωθεί : False
f.close ()
f.closed              # θα τυπωθεί : True
```

C)

```
logfile = open ('test.log', 'w')
logfile.write ('test succeeded')
logfile.close ()
print file ('test.log').read ()      # θα τυπωθεί : test succeeded
logfile = open ('test.log', 'a')
logfile.write (' 2')
logfile.close ()
print file ('test.log').read ()          # θα τυπωθεί : test succeededline 2
```

D)

```
f = open ('new_file.txt', 'w')
f.write ('A new file begins')
f.write (' ... today!\n')
f.close ()
```

Γ τάξη Βιβλίο

Κεφάλαιο 5. Δραστηριότητες

Δραστηριότητα 1
Να αντιστοιχήσετε τους παρακάτω αλγόριθμους με τις κατάλληλες λειτουργίες :

Αλγόριθμος	**Στοιχειώδης Λειτουργία**
Ταξινόμηση με επιλογή	Α. Αντιμετάθεση ζευγών
Ταξινόμηση ευθείας ανταλλαγής	Β. Εισαγωγή στοιχείου σε ταξινομημένη λίστα
Ταξινόμηση με εισαγωγή	Γ. Εύρεση ελαχίστου

Δραστηριότητα 2
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 στην ενότητα 6.3 του βιβλίου της Β' τάξης;
Σε τι οφείλεται αυτό;

Απάντηση
Αν αλλάξουμε μόνο τον τελεστή σύγκρισης στην findMinPosition από ' < ' σε ' > ', ώστε να βρίσκει το μεγαλύτερο αντί το μικρότερο, τότε και η συνάρτηση selectionSortAscending θα ταξινομεί σε φθίνουσα αντί σε αύξουσα.

```
def findMaxPosition (start, end, List) :
    position = start
    for i in range (start, end) :
        if List[ i ] > List[ position ] :
            position = i
    return position

def selectionSortAscending (List) :
    position = None
    n = len (List)
    for i in range (0, n) :
        position = findMaxPosition (i, n, List)
        List[ i ], List[ position ] = List[ position ], List [ i ]
    return List
```

Καταφέραμε να αλλάξουμε τη λειτουργία της selectionSortAscending χωρίς να πειράξουμε τίποτα στον κώδικά της. Αυτό είναι ένα παράδειγμα της ανεξαρτησίας και της ευελιξίας του τμηματικού προγραμματισμού.

Δραστηριότητα 3
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. Υπόδειξη :
Χρησιμοποιήστε μια λογική μεταβλητή.

Απάντηση

Η λογική μεταβλητή ascending παραμένει True όσο δεν βρίσκουμε ένα ζευγάρι στοιχείων για το οποίο δεν ισχύει η σχέση που θέλουμε. Δηλαδή ψάχνουμε ένα ζευγάρι που να είναι σε φθίνουσα σειρά. Αν βρούμε ένα τουλάχιστον, δεν έχει πλέον νόημα να συνεχίσουμε.

```
# 1ος τρόπος με τη χρήση λογικής μεταβλητής
def isAscending (myList) :
    ascending = True
    i = 0
    N = len (myList)
    while ascending and i < N-1 :
        if (myList[ i ] > myList[ i+1 ]) :
            ascending = False
        i = i +1
    return ascending

# 2ος τρόπος χωρίς τη χρήση λογικής μεταβλητής
def isAscending2 (myList) :
    i = 0
    pred = myList[ 0 ]
    for item in myList :
        if item < pred :
            return False
        else :
            pred = item
    return True
```

Δραστηριότητα 4. (Βελτιωμένη φυσαλίδα)

Να αναπτύξετε 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 δε βρίσκονται στην επιθυμητή σειρά, καθώς η “φυσαλίδα ανεβαίνει στην επιφάνεια”.

Απάντηση

Χρησιμοποιούμε την ιδέα, όπου η λογική μεταβλητή isSorted μας δίνει την πληροφορία, αν η λίστα είναι ταξινομημένη στο τέλος κάθε περάσματος. Αν γίνει έστω και μία αντιμετάθεση, η isSorted θα γίνει False. Αν δεν γίνει αντιμετάθεση, σημαίνει ότι όλα τα στοιχεία είναι στη σωστή σειρά, άρα ταξινομημένα, οπότε ο αλγόριθμος δεν έχει λόγο να συνεχίσει.

1ος τρόπος : με λογική μεταβλητή

```
def optimizedBubbleSort (A) :
    N = len (A)
    isSorted = False
    i = 1
    while i < N and not isSorted :
        isSorted = True
        for j in range (N-1, i-1, -1) :
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
                isSorted = False
        i = i + 1
```

2ος τρόπος : με βίαιη διακοπή της επανάληψης for...in.

```
def optimizedBubbleSort2 (A) :
    N = len (A)
    for i in range (N) :
        isSorted = True
        for j in range (N-1, i, -1) :
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
                isSorted = False
            if isSorted :
                return
```

Δραστηριότητα 5

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.

Απάντηση

Ο πρώτος, πιο απλός και γρήγορος τρόπος, είναι να μετρήσουμε πόσα είναι τα **True**. Αν είναι έστω k, οπότε γνωρίζουμε το μέγεθος (**len**) της λίστας, μπορούμε να βρούμε και τα **False** (μέγεθος – k). Στη συνέχεια, θέτουμε **True** στα πρώτα k στοιχεία και **False** στα υπόλοιπα.

Α) Μετράμε πόσα είναι τα True :

1ος τρόπος :

```
def count_True_values (booleanList) :
    True_values = 0
    for item in booleanList :
        if item :                    # ή if item == True :
            True_values = True_values + 1
    return True_values
```

2ος τρόπος : (**True** = 1, **False** = 0). Μετράμε πόσα είναι τα True

```
def count_True_values2 (booleanList) :
    True_values = 0
    for item in booleanList :
        True_values = True_values + item
    return True_values
```

Β) Μετράμε πόσα είναι τα True και στη συνέχεια βάζουμε τόσα True στα πρώτα στοιχεία του πίνακα και στα υπόλοιπα False

1ος τρόπος :

```
def swapBooleanbyCounting (List) :
    N = len (List)
    True_values = count_True_values (List)
    for i in range (True_values) :
        List[ i ] = True
    for i in range (True_values, N) :
        List[ i ] = False
    return List
```

2ος τρόπος : Στο δεύτερο τρόπο ξεκινάμε με δύο δείκτες στα άκρα της λίστας και αντιμεταθέτουμε τα αντιδιαμετρικά στοιχεία (True, False) μέχρι να συναντήσουμε από αριστερά False ή από δεξιά True. Οι δείκτες κινούνται ταυτόχρονα και αντίθετα. Όταν συναντηθούν, έχουμε αντιμεταθέσει όλες τις τιμές που ήταν στη λάθος πλευρά.

```
def swapBooleanbyComparison (List) :
N = len (List)
left = 0
right = N-1
while left < right :
    if List[ right ]== True and List[ left ]== False :
        List[ left ], List[ right ] = List[ right ], List[ left ]
        left = left + 1
        right = right - 1
    elif List[ right ]== False :
        right = right - 1
    else :
        left = left + 1
return List
```

Δραστηριότητα 6

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, κ.λπ.

Απάντηση

Αρχικά σχεδιάζουμε μια συνάρτηση για να διαβάζουμε μια λίστα, η οποία θα μας χρειαστεί και σε επόμενες δραστηριότητες. Η συνάρτηση διαβάζει από το πληκτρολόγιο μια λίστα από αντικείμενα, ενώ η ανάγνωση της λίστας σταματάει, όταν δοθεί η τιμή None και ακολούθως επιστρέφει τη λίστα ως τιμή της συνάρτησης.

Αν θέλουμε να εισάγουμε αλφαριθμητικά, πρέπει να τα εισάγουμε ανάμεσα σε εισαγωγικά (quotes), για παράδειγμα ‘Γυμνάσιο’. Μπορούμε να διαπιστώσουμε ότι η συνάρτηση δουλεύει για κάθε τύπο, κάτι που αποτελεί ένα από τα ισχυρά πλεονεκτήματα της Python.

```
def readList () :
    print 'Δώσε τα στοιχεία της λίστας (Για το τέλος δώσε την τιμή None : )'
    index = 0
    L = [ ]
    value = input ('L[' + str (index) + '] = ')
    while value != None :
        L.append (value)
        index + = 1
        value = input ('L[' + str (index) + '] = ')
    return L
```

Μετράμε πόσα είναι τα True, διπλασιάζουμε το ποσό και το συγκρίνουμε με το πλήθος των στοιχείων της λίστας. Έτσι βρίσκουμε ποια τιμή εμφανίζεται τις λιγότερες φορές. Αυτό θα είναι και το πλήθος των ζευγών (True, False).

```
def Log_Counting () :
    List = readList ()
    N = len (List)
    # μετράει πόσα είναι τα True (ορίστηκε στην δραστηριότητα 5)
    True_values = count_True_values (List)
    # είναι τα True περισσότερα από τα False;
    if (2*True_values < N) :
        minValue = True
        couples = True_values
    else :
        minValue = False
        couples = N – True_values
    for i in range (0, 2*couples, 2) :
        List[ i ] = True
        List[ i+1 ] = False
    for i in range (2*couples, N) :
        List[ i ] = not minValue
    return List
```

Δραστηριότητα 7

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. Δεν επιτρέπεται να κάνετε καμία σύγκριση ούτε να χρησιμοποιήσετε τη δομή if.

Απάντηση

1ος τρόπος : Αφού οι τιμές True, False είναι μισές-μισές, τότε θέτουμε στις πρώτες μισές θέσεις True και στις υπόλοιπες False. Θέτουμε τα πρώτα N/2 True και τα επόμενα N/2 False.

```
def splitBoolean_byCounting (List) :
    mid = len (List) / 2
    for index in range (mid) :
        List[ index ] = True
        List[ index + mid ] = False
    return List
```

2ος τρόπος : Αρκούν N/2 περάσματα του αλγορίθμου ταξινόμησης.

```
def splitBoolean_bySorting (List) :
    N = len (List)
    mid = N / 2
    for i in range (mid) :
        for j in range (N-1, i, -1) :
            if List[ j ] > List[ j-1 ] :     # True > False
                List[ j-1 ], List[ j ] = List[ j ], List[ j-1 ]
    return List
```

Δραστηριότητα 8

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.

Απάντηση

Αρκεί να αλλάξουμε τη συνθήκη του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής, έτσι ώστε να ισχύει η διάταξη RWB. Έτσι σχηματίζουμε συνθήκη με όλες τις περιπτώσεις. Το σκεπτικό είναι ότι η αντιμετάθεση μεταξύ δυο στοιχείων θα γίνει, αν δεν ισχύει η σειρά R W B.

```
def dutchFlag (L) :
    N = len (L)
    for i in range (N) :
        for j in range (N-1, i, -1) :
            if (L[ j ]=='W' and L[ j-1 ]=='B') or (L[ j ]=='R' and L[ j-1 ]=='W') or (L[ j ]=='R' and L[j-1]=='B'):
                L[ j-1 ], L[ j ] = L[ j ], L[ j-1 ]
    return L
```

Σημείωση : Αν μια εντολή δε χωράει σε μια γραμμή και θέλουμε να συνεχιστεί στην επόμενη, χρησιμοποιούμε το σύμβολο \ .

Δραστηριότητα 9

Να γράψετε μια συνάρτηση σε Python η οποία διαβάζει αριθμούς από το χρήστη μέχρι να δοθεί η τιμή None, 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;

Απάντηση

Θέλουμε έναν αλγόριθμο ταξινόμησης ο οποίος να ταξινομεί σταδιακά ένα μέρος του πίνακα (incremental). Δε θέλουμε κάθε φορά που έρχεται ένα νέο στοιχείο, να εκτελούμε πάλι ταξινόμηση όλου του πίνακα, αλλά να τοποθετούμε το νέο στοιχείο στη σωστή θέση, έτσι ώστε ο πίνακας να παραμένει ταξινομημένος. Αυτή είναι η βασική ιδέα του αλγορίθμου ταξινόμησης με εισαγωγή. Κάθε φορά που διαβάζουμε ένα νέο αριθμό, εκτελούμε αναζήτηση για να βρούμε τη θέση που πρέπει να εισαχθεί και παράλληλα μετακινούμε μια θέση δεξιά, όσα στοιχεία πρέπει να βρίσκονται μετά από αυτόν.

```
def insertionSort () :
    print 'Δώσε την τιμή None για να σταματήσεις'
    number = input ('number = ')
    L = [ ]
    while (number is not None) :     # number != None
        L.append (number)
        value = L[ len (L) - 1 ]
        j = len (L) – 1
        while j > 0 and L[ j-1 ] < value :
            L[ j ] = L[ j-1 ]
            j = j-1
        L[ j ] = value # έξω από τη while
        number = input ('number = ')
    return L
```

Δραστηριότητα 10

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει από το χρήστη δύο 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 στην λίστα Α.

Υπόδειξη : Να θεωρήσετε ότι οι αριθμοί της λίστας Β είναι όλοι διαφορετικοί μεταξύ τους. Επίσης, να εκμεταλλευτείτε το γεγονός ότι τα στοιχεία της λίστας Α είναι ταξινομημένα σε αύξουσα σειρά.

Η εισαγωγή των αριθμών για κάθε λίστα σταματάει όταν δοθεί η τιμή None.

Απάντηση

Αρχικά, το πρόγραμμα διαβάζει δύο λίστες με τη συνάρτηση readList που έχουμε ορίσει σε προηγούμενη δραστηριότητα. Στη συνέχεια, εφαρμόζουμε έναν αλγόριθμο ταξινόμησης, ώστε να μπορεί να χρησιμοποιηθεί η δυαδική αναζήτηση μετά, για τον έλεγχο της ύπαρξης ενός αριθμού της λίστας Β στη λίστα Α. Για την καλύτερη οργάνωση του προγράμματος, αναπτύξαμε κάποιες συναρτήσεις.

```
# Ανάγνωση στοιχείων της λίστας
def readList () :
    print 'Δώσε τα στοιχεία της λίστας (Για το τέλος δώσε την τιμή None : )'
    index = 0
    L = [ ]
    value = input ('L[' + str (index) + '] = ')
    while value != None :
        L.append (value)
        index + = 1
        value = input ('L[' + str (index) + '] = ')
    return L
```

Αρχικά ορίζουμε την ταξινόμηση ευθείας ανταλλαγής :

```
# Ταξινομεί με τον αλγόριθμο της ευθείας ανταλλαγής τη λίστα A
def bubbleSort (A) :
    N = len (A)
    for i in range (N) :
        for j in range (N-1, i, -1) :
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
```

Ορίζουμε τη συνάρτηση που υλοποιεί τη δυαδική αναζήτηση :

```
# Ελέγχει την ύπαρξη ενός στοιχείου key στην ταξινομημένη λίστα L υλοποιώντας τον αλγόριθμο της δυαδικής αναζήτησης

def binarySearch (A, key) :
    last = len (A) - 1
    first = 0
    found = False
    while first <= last and not found :
        mid = (last + first) // 2
        if A[ mid ] == key :
            found = True
        elif A[ mid ] < key :
            first = mid + 1
        else :
            last = mid - 1
    return found
```

Στη συνέχεια παραθέτουμε το πρόγραμμα που χρησιμοποιεί τις παραπάνω συναρτήσεις. Παρατηρήστε ότι αυτό δε δουλεύει μόνο για αριθμούς. Δουλεύει επίσης για αλφαριθμητικά αλλά και για λογικές τιμές. Δοκιμάστε να εκτελέσετε το παραπάνω πρόγραμμα και με άλλους τύπους δεδομένων, όπως με αλφαριθμητικά. Αυτό για να εξοικειωθούν οι μαθητές με την ιδέα ότι στην Python έχουμε τη δυνατότητα να ορίζουμε τη λειτουργία του αλγόριθμου ανεξάρτητα από το είδος των δεδομένων. Δηλαδή δε χρειάζεται να ορίσουμε διαφορετικό αλγόριθμο για αλφαριθμητικά, διαφορετικό για αριθμούς κ.ο.κ.

```
A = readList ()
B = readList ()
bubbleSort (A)
count = 0
for item in B :
    if binarySearch (A, item) :
        count = count + 1
print 'Πλήθος κοινών στοιχείων = ', count
```

Κεφάλαιο 6. Δραστηριότητες

Δραστηριότητα 1

Ας πειραματιστούμε γράφοντας κώδικα σε Python, ο οποίος διαβάζει το αρχείο ‘input_file.txt’και το ξαναγράφει στο νέο αρχείο ‘output_file.txt’, όπου πριν από κάθε γραμμή θα προσθέτει τον αύξοντα αριθμό της. Τα αρχεία θεωρούμε ότι βρίσκονται στον τρέχοντα κατάλογο ‘TxtFiles’.

```
f_in = open ('TxtFiles\input_file.txt', 'r')
f_out = open ('TxtFiles\output_file.txt', 'w')
linecounter = 1
for line in f_in :
   f_out.write (str (linecounter) + '.' + line)
   linecounter += 1
f_in.close ()
f_out.close ()
```

Εναλλακτικά :

```
f_in = open ('TxtFiles\input_file.txt', 'r')
f_out = open ('TxtFiles\output_file.txt', 'w')
# Σημείωση : Η εντολή len (file) δεν δουλεύει στα αρχεία
L = []
for line in f_in :  # με αυτόν τον τρόπο βρίσκουμε τον αριθμό σειρών ενός αρχείου
   L.append (line)
linecounter = len (L)

# Αρίθμηση γραμμών
# Επειδή ήδη προσπελάσαμε μία φορά το αρχείο θα πρέπει να επαναφέρουμε
# το δείκτη της τρέχουσας θέσης στο 0
f_in.seek (0)
for i in range (linecounter):
   line = f_in.readline ()
   f_out.write ('line ' + str (i+1) + ': ' + line)
f_in.close ()
f_out.close ()

# Εμφάνιση του παραγόμενου αρχείου
f_out = open ('TxtFiles\output_file.txt', 'r')
# Σημείωση : Η εντολή print file δεν δουλεύει στα αρχεία
for i in range (linecounter) :
   print f_out.readline ()
f_out.close ()
```

Δραστηριότητα 2

Υποθέτουμε ότι έχουμε ένα πρόγραμμα σε Python που έχει αποθηκευθεί στο αρχείο με όνομα count.py και με τον ακόλουθο κώδικα :

```
# Περιεχόμενο αρχείο 'count.py'
def linecount (fname) :
    count = 0
    for line in open (fname) :      # εναλλακτικά :  f = open (fname)
                                    #                for line in f :
        count += 1
    return count

print 'Ο αριθμός των γραμμών αυτού του αρχείου είναι : ', lineCount ('count.py')
```

Εκτελέστε το παραπάνω πρόγραμμα και συζητήστε το αποτέλεσμα.

Υπόδειξη : Το αποτέλεσμα είναι ότι θα εμφανίσει το πλήθος των γραμμών του αρχείου.

Δραστηριότητα 3

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 με την αντίστροφη σειρά.

```
f = open ('input.txt', 'r')
L = [ ]
for line in f :
   L.append (line)                  # εναλλακτικά :  L.insert (0, line)
N = len (L)
for i in range (N-1, -1, -1) :      # εναλλακτικά :  for i in range (N) :
   print L[i],
f.close ()
```

Επέκταση (Εγγραφή σε αρχείο)

```
f_in = open ('input.txt', 'r')
f_out = open ('output.txt', 'w')
L = [ ]
for line in f_in :
   L.append (line)
N = len (L)
for i in range (N-1, -1, -1) :
   f_out.write (L[i])
f_in.close ()
f_out.close ()

f_out = open ('output.txt', 'r')
print f_out.read ()
f_out.close ()
```

Δραστηριότητα 4

Τι πιστεύετε ότι θα συμβεί, όταν θα εκτελεστούν τα παρακάτω σενάρια. Τεκμηριώστε την άποψή σας.

```
L = [i**2 for i in range (1, 11)]
f = open ('squares.txt', 'w')
for item in L :
    f.write (str (item) + '\n')   # η write δέχεται string ως όρισμα
f.close ()
f = open ('output.txt', 'r')
print f.read ()
f.close ()
```

Απάντηση

```
# Κατασκευή λίστας με τα τετράγωνα των αριθμών από 1 έως 10
L = [ ]
for i in range (1,11)
    L.append (i**2)
# ο παραπάνω κώδικας ισοδυναμεί με τον :  L = [i**2 for i in range (1,11)]

# Άνοιγμα αρχείου κειμένου για εγγραφή
f = open ('squares.txt', 'r')

# Εγγραφή των στοιχείων της λίστας στο αρχείο
for item in L :
   f.write (str (item) + '\n')     # η write δέχεται string ως όρισμα
f.close ()

# Άνοιγμα του αρχείου για ανάγνωση και εκτύπωση των περιεχομένων του
f = open ('squares.txt', 'r')
print f.read ()
# for line in f :                          # Εμφάνιση των τετραγώνων των αριθμών ως στήλη
#     print line
# ή εναλλακτικά :  print file ('squares.txt').read ()
f.close ()
```

Κεφάλαιο 7. Δραστηριότητες

Δραστηριότητα 1.
Τοπικές και καθολικές μεταβλητές
Το παράδειγμα αυτό αλλάζει το περιεχόμενο της global μεταβλητής name μέσα στη συνάρτηση func ( ).

```
name = 'black'
def func ( ):
   global name
   name = 'white'
   print 'Μέσα στη συνάρτηση : ', name

print 'Έξω από τη συνάρτηση : ', name
func ()
print 'Έξω από τη συνάρτηση : ', name

# θα τυπώσει :
# Έξω από τη συνάρτηση : black
# Μέσα στη συνάρτηση : white
# Έξω από τη συνάρτηση : white
```

Κεφάλαιο 8. Δραστηριότητες

Δραστηριότητα 1
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα εμφανίζει τα γράμματά της, ένα σε κάθε γραμμή.

Απάντηση
Διασχίζουμε τη λέξη, γράμμα – γράμμα, με την εντολή **for** … **in** …

```
# Εμφανίζει κάθε γράμμα της λέξης σε ξεχωριστή γραμμή
def splitLetters () :
    word = raw_input ('Δώσε μια λέξη : ')
    for letter in word :
        print letter
```

Δραστηριότητα 2
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα εμφανίζει πόσα κεφαλαία αγγλικά γράμματα περιέχει η λέξη.

Απάντηση
Ελέγχει αν ένα γράμμα είναι κεφαλαίο με χρήση του τελεστή in διασχίζοντας τη λέξη γράμμα – γράμμα, με την εντολή for … in …, αφού πρώτα κατασκευάσει ένα string/σύνολο με όλα τα κεφαλαία αγγλικά γράμματα.

```
def countCapitals (word) :
    enCapSet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    countCapitals = 0
    for letter in word :
        if letter in enCapSet :
            countCapitals += 1
    return countCapitals

lexi = raw_input ('Δώσε μια αγγλική λέξη : ')
print 'Η λέξη περιέχει', countCapitals (lexi), 'κεφαλαία γράμματα'
```

Δραστηριότητα 3
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μία λέξη και θα την εμφανίζει αντεστραμμένη, με τη χρήση μιας στοίβας.

```
# Αρχικά δίνουμε την υλοποίηση μιας στοίβας
def push (stack, item) :        # Ώθηση
    stack.append (item)
def pop (stack) :               # Απώθηση
    return stack.pop ()
def isEmpty (stack) :           # Έλεγχος άδειας στοίβας
    return len (stack)== 0
def createStack () :            # Δημιουργία νέας στοίβας
    return [ ]
```

 # Αντιστρέφει μια λέξη με τη χρήση στοίβας Το τελευταίο γράμμα της λέξης είναι το τελευταίο που θα εισέλθει στη στοίβα και θα βρίσκεται στην κορυφή της. Όταν ξεκινήσουμε να απωθούμε γράμματα από τη στοίβα θα βγει πρώτα το τελευταίο, μετά το προτελευταίο κ.ο.κ. Άρα, τα γράμματα θα εμφανιστούν σε αντίστροφη σειρά από αυτή με την οποία μπήκαν στη στοίβα.

```
def reverseWord (word) :
    stack = createStack ()
    for letter in word :
        push (stack, letter)
    while not isEmpty (stack) :
        reverse = reverse + pop (stack)
    print reverse
```

Δραστηριότητα 4

Να γράψετε μια συνάρτηση is*Substring (string, substring)* η οποία θα ελέγχει, αν η συμβολοσειρά substring περιέχεται στη συμβολοσειρά string και αν ναι, θα επιστρέφει True. Στη συνέχεια να υλοποιήσετε μια δεύτερη συνάρτηση, η οποία θα επιστρέφει πόσες φορές εμφανίζεται μια συμβολοσειρά μέσα σε μια άλλη.

```
# Αυτό μπορεί να γίνει με χρήση του τελεστή in πολύ απλά
def isSubstring (string, substring) :
    return substring in string

# Ελέγχει αν το substring εμφανίζεται στο string ξεκινώντας από τη θέση start
def isPrefix (string, substring, start) :
    i = start
    j = 0
    count = 0
    while j < len (substring) and i < len (string) :
        if string[ i ] == substring[ j ] :
            count += 1
        i += 1
        j += 1
    return count == len (substring)

# Εκτελούμε για κάθε θέση του string την προηγουμένη συνάρτηση και έτσι υπολογίζουμε
                                        πόσες φορές εμφανίζεται το substring στο String
def findSubstring (string, substring) :
    pos = 0
    count = 0
    while pos < len (string) :
        if isPrefix (string, substring, pos) == True :
            count += 1
        pos += 1
    return count
```

Δραστηριότητα 5

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, αν αυτό είναι δυνατόν και θα τους εμφανίζει στην οθόνη.

```
# γίνεται χρήση της στοίβας
def push (stack, item) :
    stack.append (item)
def pop (stack) :
    return stack.pop ()
def isEmpty (stack) :
    return len (stack)== 0
def createStack () :
    return [ ]

def stoivaOthApoth () :
   stack = createStack ()
   number = input ('Δώσε έναν αριθμό ή 0 για τέλος : ')
   while number != 0 :
      if number > 0 :
         push (stack, number)
      else :
         i = 1
         while i <= abs (number) and not isEmpty (stack) :
            pop (stack)
            i = i + 1
      print stack
      number = input ('Δώσε ακόμη έναν αριθμό ή 0 για τέλος : ')
   print stack

stoivaOthApoth ()
```

Δραστηριότητα 6

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.

1ος τρόπος : χωρίς λεξικό

```
# Επιστρέφει τη θέση ενός γράμματος letter στο αλφάβητο alphabet
def indexStr (letter) :
    capSet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    lowSet = 'abcdefghijklmnopqrstuvwxyz'
    for index in range (26) :
        if letter == capSet[index] or letter == lowSet[index] :
            return index
    return -1;
def countCapitals () :
    enCapitalSet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    text = raw_input ('Δώσε ένα κείμενο με αγγλικά γράμματα : ')
    # δημιουργία και μηδενισμός μιας λίστας μετρητών
    counter = [ ]
    for i in range (26) :
        counter.append (0)
    # αύξηση του μετρητή στην αντίστοιχη θέση του γράμματος
    for letter in text :
        index = indexStr (letter)
        if index >= 0 :
            counter[ index ] += 1
   print 'Letter Frequency'
   print '-----------------'
   for i in range (26) :
      print ' ', enCapitalSet[i], ' ', counter[i]
```

2ος τρόπος : με λεξικό (μόνο για τον εκπαιδευτικό)

```
def countCapitals2 () :
    messg = ('Δώσε ένα κείμενο με αγγλικά μικρά γράμματα : ')
    text = raw_input (messg)
    alphabet = dict ()
    for letter in text :
        if letter in alphabet :
            alphabet[ letter ] + = 1
        else :
            alphabet[ letter ] = 1
    print ' Letter Frequency'
    print '-------- -----------'
    for letter in alphabet :
        print ' ', letter, ' ', alphabet[letter]
    print '**************************************************'
```

Δραστηριότητα 7

Να γράψετε μια συνάρτηση σε Python η οποία θα δέχεται μια λίστα από λέξεις και θα επιστρέφει τη λέξη με το μεγαλύτερο μήκος.

```
def maxLength (wordList) :
    maxLen = 0
    maxWord = ''
    for word in wordList :
        if len (word) > maxLen :
            maxLen = len (word)
            maxWord = word
    return maxWord
```

Δραστηριότητα 8

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.

```
def num_of_Vowels (word) :
    vowels = 'AEIOUYaeiouy'
    count = 0
    for letter in word :
        if letter in vowels :
            count += 1
    return count

def maxVowels (wordList) :
    maxV = 0
    maxWord = ''
        for word in wordList :
            if num_of_Vowels (word) > maxV :
                maxV = num_of_Vowels (word)
                maxWord = word
    return maxWord
```

Κεφάλαιο 11. Δραστηριότητες

Δραστηριότητα εμπέδωσης

Δίνεται η παρακάτω κλάση:

```
class Car:
    def __init__ (self, make) :
        self.make = make
        self.speed = 60
    def speed_up (self, speed) :
        self.speed = speed
        print 'I am driving at a speed', self.speed, 'km/h'
    def turn (self) :
        print 'I am turning … '
```

1. Ποιος είναι ο κατασκευαστής (constructor) της κλάσης;
   Ο κατασκευαστής (constructor) της κλάσης είναι η μέθοδος **__init__** (self, make), ο οποίος δημιουργεί το αντικείμενο και αρχικοποιεί τις ιδιότητές του.

```
def __init__ (self, make) :
    self.make = make
    self.speed = 60
```

2. Να καταγράψετε τις ιδιότητες της κλάσης, τις μεθόδους της, καθώς και τις ενέργειες που πραγματοποιούν.
   Ιδιότητες της κλάσης Car : make, speed
   Μέθοδοι :
   α) speed_up (self, speed) : δίνει τιμή στην ιδιότητα speed του αντικειμένου και εμφανίζει στην οθόνη το μήνυμα 'I am driving at a speed' και στη συνέχεια, εμφανίζει την τιμή της ιδιότητας του αντικειμένου ακολουθούμενη από το μήνυμα 'km/h'
   β) turn (self) : εμφανίζει στην οθόνη το μήνυμα 'I am turning …'

   Ιδιότητες :
   self.make = make
   self.speed = 60
   Μέθοδοι :
   speed_up (**self**, speed)
   turn (**self**)
   Ενέργειες :
   self.speed = speed
   print 'I am driving at a speed', **self**.speed, 'km/h'

   print 'I am turning … '

3. Να προσθέσετε τις ιδιότητες color και year που αντιπροσωπεύουν το χρώμα και το έτος κυκλοφορίας του αυτοκινήτου αντίστοιχα και να αρχικοποιούνται στον κατασκευαστή.
   Ο κατασκευαστής της κλάσης αλλάζει ως εξής :

```
def __init__ (self, make) :
    self.make = make
    self.speed = 60
    self.color = color
    self.year = year
```

4. Να αλλάξετε τη μέθοδο turn, έτσι ώστε να δέχεται ως παράμετρο μια συμβολοσειρά που ορίζει αν το αυτοκίνητο θα στρίψει αριστερά ή δεξιά.

```
def turn (self, direction) :
    print 'I am turning … ', direction
```

5. Να δημιουργήσετε τα παρακάτω στιγμιότυπα της κλάσης:
   I. Αντικείμενο με όνομα convertible και μάρκα "bmw", χρώμα "μαύρο" και έτος κυκλοφορίας "2013".
   convertible = ***Car*** ('bmw', 60, 'μαύρο', 2031)
   II. Αντικείμενο με όνομα sedan και μάρκα "toyota", χρώμα "κόκκινο" και έτος κυκλοφορίας "2009".
   sedan = ***Car*** ('toyota', 60, 'κόκκινο', 2009)

6. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο convertible να στρίψει δεξιά.
   convertible.***turn*** ('right') :

7. Να καλέσετε την κατάλληλη μέθοδο, ώστε το αντικείμενο sedan να τρέχει με 90 χιλ/ώρα.
   sedan.***speed_up*** (90)

Δραστηριότητα 1

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.

Λύση

```
class Akeraios :
    def __init__ (self) :
        self.timi = 0
    def Anethese_timi (self, timi) :
        self.timi = timi
    def Emfanise_timi (self) :
        print self.timi

Artios = Akeraios ()
Artios.Anethese_timi (14)
Artios.Emfanise_timi ()
```

Δραστηριότητα 2

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.

Λύση

```
class Student :
    def __init__ (self, AM, name, grades) :
        self.AM = AM
        self.name = name
        self.grades = grades[ :8]
    def Emfanise_times (self) :
          print 'AM : ', self.AM
          print 'Όνομα : ', self.name
          print 'Βαθμοί : ', self.grades
    def Ypol_MO (self) :
        suma = 0
        for i in range (len (self.grades)) :
            suma = suma + self.grades[i]
        return suma / len (self.grades)

    chris = Student (209, 'Chris', [14, 16, 18, 19, 12, 20, 17, 20])
    chris.Emfanise_times  ()
    print 'Ο τελικός βαθμός του', chris.name, 'είναι :', chris.Ypol_MO ()
```