ΜΕΜ104 Γλώσσα Προγραμματισμού Ι

4ο φυλλάδιο ασκήσεων

13 Νοεμβρίου 2020

1. Γράψτε ένα πρόγραμμα το οποίο κατασκευάζει μια λίστα με τους διαιρέτες ενός φυσικού αριθμού. Για παράδειγμα, αν δοθεί ο αριθμός 42 θα πρέπει να κατασκευαστεί η λίστα `[1, 2, 3, 6, 7, 14, 21, 42]`.

```
n = int(input('Enter n: '))
L = [1]
for i in range(2,n):
  if n % i == 0:
    L.append(i)
if n > 1:
  L.append(n)
```

2. Γράψτε ένα πρόγραμμα το οποίο ελέγχει αν δυο λίστες έχουν τουλάχιστον ένα κοινό στοιχείο.

```
L = [2, 13, 5, 4]
M = [1, 5, 3, 2, 8, 12]

found = 0
for e in L:
  if e in M:
    found = 1
    break

if found:
  print('Lists L and M have at least one common element')
else:
  print('Lists L and M do not have common elements')
```

3. Γράψτε ένα πρόγραμμα το οποίο αποθηκεύει σε μια λίστα τις λέξεις που εισάγει ο χρήστης και στη συνέχεια εμφανίζει μόνο τις λέξεις με περισσότερα από 3 γράμματα. Η είσοδος των λέξων θα πρέπει να τερματίζει όταν ο χρήστης δώσει τη λέξη 'end'.

```
L = []

while True:
  s = input('Enter word: ')
  if not s:
    continue
  if s != 'end':
    L.append(s)
  else:
    break

for e in L:
  if len(e) >= 3:
    print(e)
```

4. Γράψτε ένα πρόγραμμα το οποίο αποθηκεύει στη λίστα L τις τιμές της συνάρτησης f στα σημεία ενός ομοιόμορφου διαμερισμού του διαστήματος $[0, 1]$ με n εσωτερικά σημεία. Για παράδειγμα, ένας ομοιόμορφος διαμερισμός του $[0, 1]$ με τέσσερα εσωτερικά σημεία είναι ο $\{0, 0.2, 0.4, 0.6.0.8, 1\}$. Γενικότερα, τα σημεία ενός ομοιόμορφου διαμερισμού του $[0, 1]$ με n εσωστερικά σημεία είναι τα $x_i = i/(n + 1)$, για $i = 0, \ldots, n + 1$.

```
def f(x):
  return 1 / (1 + x)

n = 9
L = []
for i range(n+2):
  L.append( f(i/(n+1)) )
```

Θα μπορούσαμε ακόμα να φτιάξουμε την ομοιόμορφη διαμέριση του $[0, 1]$ και μετά να εφαρμόσουμε τη συνάρτηση f χρησιμοοιώντας την εντολή `map`.

```
def f(x):
  return 1 / (1 + x)

n = 9
x = []
for i in range(n+2):
  x.append( i/(n+1) )

L = list(map(f, x))
```

5. Γράψτε μια συνάρτηση η οποία με είσοδο δύο λίστες τα στοιχεία των οποίων είναι σε αύξουσα σειρά επιστρέφει μια λίστα με όλα τα στοιχεία και από τις δύο λίστες, πάλι σε αύξουσα σειρά. Για παράδειγμα, αν δοθούν οι λίστες `[2, 4, 5, 13]` και `[1, 2, 3, 5]` η συνάρτησή σας θα πρέπει να επιστρέψει τη λίστα `[1, 2, 2, 3, 4, 5, 5, 13]`.

```
L = [2, 4, 5, 13]
M = [1, 2, 3, 5, 13, 16]
K = []

i = 0;  j = 0

while i < len(L) and j < len(M):
  if L[i] <= M[j]:
    K.append( L[i] )
    i += 1
  else:
    K.append( M[j] )
    j += 1

if i < len(L):
  K.extend(L[i:])

if j < len(M):
  K.extend(M[j:])
```

Προφανώς, θα μπορούσε κανείς πρώτα να ενώσει τις δύο λίστες και στη συνέχεια να τις ταξινομήσει, δηλαδή

```
K = L + M
K.sort()
```

6. Γράψτε μια συνάρτηση η οποία υπολογίζει τη συχνότητα εμφάνισης όλων των στοιχείων μιας λίστας. Για παράδειγμα, αν δοθεί η λίστα `['m', 'a', 'r', 'i', 'a']`, τότε το πρόγραμμά σας θα πρέπει να επιστρέψει τη λίστα `[1, 2, 1, 1, 2]`.

```
L = list('Python programming is fun!')
freq = []
for e in L:
  freq.append(L.count(e))
```

7. Γράψτε ένα πρόγραμμα το οποίο αφαιρεί από μια λίστα τις πολλαπλές εμφανίσεις των στοιχείων της. Για παράδειγμα, η λίστα `[10, 12, 13, 13, 10, 9, 8, 7, 8, 10, 10, 8, 13]` θα πρέπει να γίνει `[10, 12, 13, 9, 8, 7]`.

```
L = [10, 12, 13, 13, 10, 9, 8, 7, 8, 10, 10, 8, 13]
M = []
for e in L:
  if e not in M:
    M.append(e)
```

Εναλλακτικά, θα μπορούσαμε να αφαιρέσουμε πολλαπλές εμφανίσεις ξεκινώντας από δεξιά προς τ' αριστερά:

```
for i in range(len(L)-1, -1, -1):
  if L.count(L[i]) > 1:
    L.pop(i)
```

8. Γράψτε ένα πρόγραμμα το οποίο αντιγράφει τα στοιχεία μιας λίστας L (υποθέτουμε ότι είναι όλα συμβολοσειρές) στη λίστα L1, αν περιέχουν τουλάχιστον τρία φωνήεντα, διαφορετικά στη λίστα L2.

```
def numv(s):
  k = 0
  for c in s:
    if c in 'aeiouAEIOU':
      k += 1
  return k

L = ['Maria', 'Michael', 'George', 'Nikos', 'Anna', 'Petros', 'Eleni']
L1 = []
L2 = []

for w in L:
  if numv(w) >= 3:
    L1.append(w)
  else:
    L2.append(w)
```

9. Μπορούμε να βρούμε το μικρότερο στοιχείο μιας λίστας χρησιμοποιώντας τη συνάρτηση `min`. Πως μπορούμε να βρούμε το *δεύτερο* μικρότερο στοιχείο μιας λίστας; Η άσκηση είναι δύσκολη – αν κάποιος ενδιαφέρεται για μια κομψή λύση.

```
L = [10, 7, 13, 5, 9, 8, 11, 14, 6, 12]

frst = sec = L[0]

for e in L:
  if e < frst:
    sec = frst
    frst = e
  elif e < sec:
    sec = e
```

Εναλλακτικά, θα μπορούσαμε να αφαιρέσουμε τον μικρότερο στοιχείο και να βρούμε εκ νέου το μικρότερο στοιχείο, δηλαδή,

```
if len(L) > 1:
  L.remove(min(L))
  sec = min(L)
```

ή, να ταξινομήσουμε τη λίστα και να διαλέξουμε το δεύτερο κατά σειρά στοιχείο:

```
if len(L) > 1:
  sec = sorted(L)[1]
```

10. Ο απλούστερος (και ο πιο αργός) τρόπος ταξινόμησης είναι ο λεγόμενος αλγόριθμος της φυσαλίδας (bubble sort), μια υλοποίηση του οποίου θα μπορούσε να είναι η ακόλουθη:

```
def bubbleSort(L):
    for n in range(len(L)-1,0,-1):
        for i in range(n):
            if L[i] > L[i+1]:
                tmp = L[i]; L[i] = L[i+1]; L[i+1] = tmp
```

Προσπαθήστε να καταλάβετε την ιδέα του αλγορίθμου τυπώνοντας τα περιεχόμενα της λίστας L μετά από κάθε επανάληψη. Για παράδειγμα, οι αλλαγές που πραγματοποιεί η παραπάνω συνάρτηση κατα την ταξινόμηση των στοιχείων της λίστας `[5, 1, 4, 2, 6, 8]` είναι:

```
5 1 4 2 6 3
1 4 2 5 3 6
1 2 4 3 5 6
1 2 3 4 5 6
```

Μετρήστε επίσης τον αριθμό των συγκρίσεων που κάνει η μέθοδος. Μπορείτε να ανακαλύψετε κάποια σχέση με το μήκος της λίστας;

Με τις συνεχείς συγκρίσεις των διπλανών στοιχείων και τις αντιμεταθέσεις τους αν είναι σε λάθος σειρά, τα μικρότερα στοιχεία ”ανεβαίνουν σα φυσαλίδες” στην κορυφή (αρχή) της λίστας. Στο πρώτο βήμα έχουμε n-1 συγκρίσεις, στο δέυτερο έχουμε n-2 συγκρίσεις, στο τελευταίο έχουμε 1 σύγκριση. Άρα, το άθροισμά τους είναι 1 + 2 + ... + (n-2) + (n-1) = n * (n-1) / 2 (οπου n το μέγεθος της λίστας).