Σημειωματάριο Δευτέρας 13 Νοε. 2017

Bubble sort

Εδώ δείχνουμε το πώς λειτουργεί μια πολύ απλή μέθοδος ταξινόμησης, η bubble sort. Έχουμε να ταξινομήσουμε μια λίστα L αριθμών σε **αύξουσα σειρά**. Ο τρόπος που το κάνουμε είναι ο εξής. Κάνουμε πολλαπλές συγκρίσεις ανάμεσα σε ζεύγη θέσεων της λίστας. Σε κάθε τέτοια σύγκριση αν τα περιεχόμενα των δύο θέσεων δεν είναι στη σωστή σειρά τότε εναλλάσουμε τα περιεχόμενά τους. Έχει βέβαια μεγάλη σημασία το με ποια σειρά κάνουμε τους ελέγχους αυτούς. Αν n είναι το μήκος της λίστας (θέσεις από 0 έως n-1) τότε κάνουμε τους παρακάτω ελέγχους ζευγών:

```
(0, 1), (1, 2), (2, 3), ... , (n-4, n-3), (n-3, n-2), (n-2, n-1),
(0, 1), (1, 2), (2, 3), ... , (n-4, n-3), (n-3, n-2),
(0, 1), (1, 2), (2, 3), ... , (n-4, n-3),
...
(0, 1), (1, 2), (2, 3),
(0, 1), (1, 2),
(0, 1)
```

Η βασική παρατήρηση τώρα είναι ότι στο τέλος της πρώτης σειράς ελέγχων το μεγαλύτερο στοιχείο της L (ή ένα από τα μεγαλύτερα, αν έχει παραπάνω από ένα) έχει πάει στη θέση του, δηλ. στη θέση n-1. Μετά το τέλος της δεύτερης σειράς ελέγχων έχει πάει στη θέση του το δεύτερο μεγαλύτερο στοιχείο της L, δηλ. στη θέση n-2, κλπ. Έτσι μετά τον τελευταίο έλεγχο που γίνεται η λίστα είναι πλήρως ταξινομημένη σε αύξουσα σειρά.

In [ ]:

```
```

In [1]:

```
def bubblesort(L):
    # Η συνάρτηση αυτή αναδιατάσσει τα στοιχεία της λίστας L ώστε να είναι σε αύ
ξουσα σειρά
    n = len(L)
    if n <= 1: # Αν κενή ή έχει μήκος 1 δεν έχουμε τίποτε να κάνουμε
        return
    for i in range(1, n): # Για σταθερό i υλοποιούμε τους ελέγχους/εναλλαγές μια
ς γραμμής από τις παραπάνω (ένα πέρασμα)
        for j in range(i, n):
            if L[i-1] > L[j]: # Αν τα i-1 και j στοιχεία της L δεν είναι στη σωσ
τή σειρά τότε τα εναλλάσουμε
                L[i-1], L[j] = L[j], L[i-1]

X=[0, 1, 4, 3, -1, -5, 0, 3.4, 9, -10]
bubblesort(X) # ταξινομούμε τη X σε αύξουσα σειρά
print(X)
```

```
[-10, -5, -1, 0, 0, 1, 3, 3.4, 4, 9]
```

Γεννιέται όμως η ανάγκη να ταξινομούμε και με άλλους τρόπους, π.χ. σε φθίνουσα σειρά, ή να ταξινομούμε αντικείμενα που ο τρόπος που είναι διατεταγμένα είτε δεν είναι προφανής είτε θέλουμε να τον αλλάξουμε.

Γι' αυτό ξαναγράφουμε την `bubblesort` ώστε να παίρνει κι ένα δεύτερο όρισμα, μια συνάρτηση `f`. Η συνάρτηση αυτή `f` πρέπει να είναι έτσι φτιαγμένη ώστε να παίρνει ως όρισμα μια λίστα L και δύο θέσεις της λίστας `i` και `j` και να μας απαντάει αν οι τιμές της λίστας στις θέσεις αυτές είναι στη σωστή σειρά (True) ή όχι (False).

Τώρα η `bubblesort` καλεί αυτή τη συνάρτηση για να αποφασίσει αν θα εναλλάξει τα περιεχόμενα δύο θέσεων της λίστας που ταξινομεί. Κι εμείς δίνουμε στην `bubblesort` την κατάλληλη τέτοια συνάρτηση που να συμβαδίζει με την έννοια διάταξης που θέλουμε να εφαρμόσουμε.

In [2]:

```
def bubblesort(L, f):
    n = len(L)
    if n <= 1:
        return
    for i in range(1, n):
        for j in range(i, n):
            if not f(L, i-1, j): # Εδώ ελέγχουμε, καλώντας την f, αν χρειάζεται
 εναλλαγή.
                L[i-1], L[j] = L[j], L[i-1]
```

Αν θέλουμε να ταξινομήσουμε μια λίστα αριθμών σε φθίνουσα σειρά τότε φτιάχνουμε την παρακάτω συνάρτηση ελέγχου και την περνάμε στην `bubblesort`.

In [4]:

```
def compare(L, i, j):
    # Ελέγχει αν το L[i] και L[j] στοιχείο της L
    # είναι σε φθίνουσα σειρά
    if i <= j:
        return L[i] >= L[j]
    return L[j] >= L[i]

def bubblesort(L, f):
    n = len(L)
    if n <= 1:
        return
    for i in range(1, n):
        for j in range(i, n):
            if not f(L, i-1, j): # Εδώ ελέγχουμε, καλώντας την f, αν χρειάζεται
 εναλλαγή.
                L[i-1], L[j] = L[j], L[i-1]

X=[0, 1, 4, 3, -1, -5, 0, 3.4, 9, -10]
bubblesort(X, compare) # ταξινομούμε τη X σε φθίνουσα σειρά
print(X)
```

```
[9, 4, 3.4, 3, 1, 0, 0, -1, -5, -10]
```

Ενώ αν θέλουμε να ταξινομήσουμε με φθίνουσα σειρά της **απόλυτης τιμής** τότε χρησιμοποιούμε την παρακάτω συνάρτηση ελέγχου.

In [5]:

```
def comp1(L, i, j):
    if i <= j:
        return abs(L[i]) >= abs(L[j])
    return abs(L[j]) >= abs(L[i])

def bubblesort(L, f):
    n = len(L)
    if n <= 1:
        return
    for i in range(1, n):
        for j in range(i, n):
            if not f(L, i-1, j): # Εδώ ελέγχουμε, καλώντας την f, αν χρειάζεται
 εναλλαγή.
                L[i-1], L[j] = L[j], L[i-1]

X=[0, 1, 4, 3, -1, -5, 0, 3.4, 9, -10]
bubblesort(X, comp1) # ταξινομούμε τη X σε φθίνουσα σειρά της απολύτου τιμής
print(X)
```

```
[-10, 9, -5, 4, 3.4, 3, 1, -1, 0, 0]
```

Αν θέλουμε να ταξινομήσουμε μια λίστα από strings αγνοώντας το αν τα γράμματα είναι κεφαλαία ή μικρά τότε το κάνουμε με την παρακάτω συνάρτηση.

In [6]:

```
def compares(L, i, j):
    if i <= j:
        return L[i].lower() >= L[j].lower()
    return L[j].lower() >= L[i].lower()

def bubblesort(L, f):
    n = len(L)
    if n <= 1:
        return
    for i in range(1, n):
        for j in range(i, n):
            if not f(L, i-1, j): # Εδώ ελέγχουμε, καλώντας την f, αν χρειάζεται
 εναλλαγή.
                L[i-1], L[j] = L[j], L[i-1]

X = ["abc", "A", "ABCD", "xyz", "Xyz", ""]
bubblesort(X, compares) # ταξινομούμε την X σε φθίνουσα (λεξικογραφική) σειρά χω
ρίς να λογαριάζουμε κεφαλαία ή μικρά
print(X)
```

```
['xyz', 'Xyz', 'ABCD', 'abc', 'A', '']
```

Υπάρχουν και δύο έτοιμες συναρτήσεις/μέθοδοι της python για ταξινόμηση λίστας.

Η συνάρτηση `sorted(L)` που μας επιστρέφει τη λίστα L ταξινομημένη (χωρίς να πειράζει την L) και η μέθοδος `L.sort()` η οποία ταξινομεί εσωτερικά τη λίστα L.

In [11]:

```
X=[1, 2, 3, 2, 1, -1]

print(sorted(X))
print(X)
```

```
[3, 2, 2, 1, 1, -1]
[1, 2, 3, 2, 1, -1]
```

In [ ]:

```
X=[1, 2, 3, 2, 1, -1]

print(sorted(X, reverse=True)) # δίνοντας αυτή την παράμετρο η sorted ταξινομεί
 κατά φθίνουσα σειρά.
print(X)
```

In [7]:

```
X=[1, 2, 3, 2, 1, -1]
print(X)
X.sort()
print(X)
```

```
[1, 2, 3, 2, 1, -1]
[-1, 1, 1, 2, 2, 3]
```

In [10]:

```
X=[1, 2, 3, 2, 1, -1]
print(X)
X.sort(reverse=True) # δίνοντας αυτή την παράμετρο η sort ταξινομεί κατά φθίνουσ
α σειρά.
print(X)
```

```
[1, 2, 3, 2, 1, -1]
[3, 2, 2, 1, 1, -1]
```

Αν πρόκειται να ταξινομήσουμε αντικείμενα που δεν έχουν προφανή τρόπο διάταξης (φανταστείτε π.χ. ότι τα αντικείμενά μας είναι τα άτομα μέσα σε μια αίθουσα, που για το καθένα από αυτά έχουμε την εξής πληροφορία: το όνομά τους, την ηλικία τους, το ύψος τους και το βάρος τους. Τότε υπάρχουν πάρα πολλοί διαφορετικοί τρόποι με τους οποίους θα μπορούσαμε να ζητήσουμε να ταξινομηθούν.

Για να πούμε λοιπόν στην `sort` και την `sorted` πώς θέλουμε να ταξινομήσουμε τα αντικείμενα που είναι μέσα στη λίστα αυτό που κάνουμε είναι ότι φτιάχνουμε μια συνάρτηση (ίσως και να είναι ήδη έτοιμη στην python) η οποία σε κάθε αντικείμενο της λίστας μας αντιστοιχίζει έναν αριθμό (π.χ., στο προηγούμενο παράδειγμα, θα μπορούσαμε σε κάθε άτομο να αντιστοιχίσουμε το ύψος του) και με βάση αυτό τον αριθμό γίνεται η ταξινόμηση.

Ας πούμε για παράδειγμα ότι τα αντικείμενα που θέλουμε να ταξινομήσουμε είναι λίστες (διαφόρων περιεχομένων) και ότι θέλουμε να ταξινομήσουμε αυτές τις λίστες σε αύξουσα σειρά **μήκους**. Τότε περνάμε στην `sort` ως τιμή της παραμέτρου `key` τη συνάρτηση `len` της python η οποία σε κάθε λίστα αντιστοιχίζει το μήκος της.

In [12]:

```
X = [ 20*[0], [1, 2], [1, 2 , 3], [], ['a', 'b', []], 10*[1] ]

X.sort(key=len)
print(X)
```

```
[[], [1, 2], [1, 2, 3], ['a', 'b', []], [1, 1, 1, 1, 1, 1, 1, 1, 1,
1], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

Αν τα στοιχεία μας είναι λίστες αριθμών και θέλουμε να τα ταξινομήσουμε κατά φθίνουσα σειρά του **αθροίσματος των στοιχείων τους** τότε περνάμε στην παράμετρο `key` τη συνάρτηση `sum` της python η οποία για μια λίστα αριθμών επιστρέφει το άθροισμα των στοιχείων της.

In [13]:

```
X = [ 20*[0], [1, 2], [1, 2 , 3], 10*[1] ]

X.sort(key=sum, reverse=True)
print(X)
```

```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 2, 3], [1, 2], [0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

Ας πούμε τώρα ότι τα στοιχεία μας είναι strings και το μόνο που μας ενδιαφέρει γι' αυτά είναι το πόσα a έχουν μέσα, και θέλουμε να τα ταξινομήσουμε σύμφωνα με αυτόν τον αριθμό. Δεν έχουμε παρά να φτιάξουμε την αντίστοιχη συνάρτηση για την παράμετρο `key` η οποία να μετράει τα a σε ένα string.

In [14]:

```
def f(s): # επιστρέφει πόσα `a` έχει μέσα του το string s
    count=0
    for k in s:
        if k == 'a':
            count += 1
    return count

X = ["abc", "aa", "a123", "123", "aaa"]
print(sorted(X, key=f))
```

```
['123', 'abc', 'a123', 'aa', 'aaa']
```