# -\*- coding: utf-8 -\*-

"""

@author: JMG

"""

import numpy as np

import pylab as pl

### Exercice 1

# Fonction auxiliaire

def insertion(L,i): # L[:i-1] est déjà triée !!!

 # Cette fonction insère clé=L[i] à sa bonne place dans L[:i-1]

 cle=L[i]

 p=i

 while p>0 and cle<L[p-1]:

 L[p]=L[p-1]

 p=p-1

 L[p]=cle

# Fonction tri par insertion

def triInsertion(L):

 i=1

 c=0

 for x in L[1:]:

 p=i

 while p>0 and x<L[p-1]:

 L[p]=L[p-1]

 p=p-1

 c=c+1

 L[p]=x

 i=i+1

 return L

# Tests

L=[4,3,7,9,8,5] ; print(triInsertion(L))

# Fonction résumée

def tri\_insertion(L):

 n=len(L)

 for i in range(1,n):

 cle=L[i]

 p=i

 while p>0 and cle<L[p-1]:

 L[p]=L[p-1]

 p=p-1

 L[p]=cle

 return L

# Tests

L=[4,3,7,9,8,5] ; print(tri\_insertion(L))

# La fonction de tri par insertion vue dans le TP

def tri\_insertion2(L):

 compteur=0 #pour la question avec compteur

 i=1

 for x in L[1:]:

 p=i

 while p>0 and x<L[p-1]:

 L[p]=L[p-1]

 p=p-1

 compteur+=1 #pour la question avec compteur

 L[p]=x

 i=i+1

 return (L,compteur) #pour la question avec compteur

 #return L

# Tests

L=[4,3,7,9,8,5] ; print("insertion TP : ",tri\_insertion2(L))

print("insertion TP pour range(1000): ",tri\_insertion2(list(range(1000))))

print("insertion TP pour range(1000,0,-1): ",tri\_insertion2(list(range(1000,0,-1))))

def mediane(L):

 tri\_insertion2(L)

 n=len(L)

 if n%2==1:

 return L[n//2]

 else:

 return 1/2\*(L[(n//2)-1]+L[n//2])

test=list(range(1000,0,-1))

print("test avec tri\_insertion :",tri\_insertion2(test))

print("La médiane de test est :",mediane(test))

""" D'autres fonctions d'insertion similaires, qui effectuent la même action :

def triInsertion1(t):

 n=len(t)

 #boucle externe

 for i in range(1,n):

 clef = t[i]

 j=i-1

 #boucle interne

 while j>=0 and t[j] > clef:

 t[j+1] = t[j]

 j=j-1

 #Insertion de la clef - Attention au + 1

 t[j+1] = clef

#L=[4,3,7,9,8,5] ; triInsertion1(L) ; print(L)

def triInsertion2(t):

 i=0

 #boucle externe

 for x in t:

 j=i-1

 #boucle interne

 while j>=0 and t[j][0] > x[0]:

 t[j+1] = t[j]

 j=j-1

 #Insertion de la clef - Attention au + 1

 t[j+1] = x

 i=i+1

"""

### Exercice 2

def insertionR(Lt,e):

 n=len(Lt)

 if Lt==[] or e>=Lt[n-1]:

 return Lt+[e]

 else:

 return insertionR(Lt[:n-1],e)+[Lt[n-1]]

Lt=[1,2,3,5]

print(insertionR(Lt,4))

print(Lt)

def tri2(L):

 n=len(L)

 Lt=[L[0]]

 for e in L[1:]:

 Lt=insertionR(Lt,e)

 return(Lt)

 # ou :

def tri2(L):

 n=len(L)

 Lt=[]

 for e in L:

 Lt=insertionR(Lt,e)

 return(Lt)

import sys

sys.setrecursionlimit(1020)

L2=range(1000,0,-1)

print("tri insertionR pour range(1000,0,-1): ", tri2(L2))

 ### Exercice 3

# Une proposition pas très esthétique. Si vous trouvez mieux... On programmera au prochain TP l’algorithme dichotomique récursif d’insertion.

def dichoto(L,e):

 n = len(L)

 if e > L[-1]:

 return n

 elif e<L[0]:

 return 0

 else:

 g, d = 0, n-1

 while g<d:

 m = (g+d)//2

 if e==L[m]:

 return m

 elif e< L[m]:

 if e>L[m-1]:

 return m

 else:

 d = m-1

 else:

 if e<L[m+1]:

 return m+1

 else:

 g = m+1

 return d

L=[2,3,4,5,6]

print(dichoto(L,1.5))

### Exercice 4

def tri\_selection(L):

 n = len(L)

 for i in range(0,n):

 min = i

 for j in range(i+1,n) :

 if L[j] < L[min] :

 min = j

 if min != i :

 L[i],L[min]=L[min],L[i]

L=[4,3,7,9,8,5] ; tri\_selection(L) ; print(L)

### Exercice 5

def tri\_rapide(L):

 if len(L) <= 1:

 return L

 L1, L2 = [], []

 cle = L.pop(0)

 for x in L:

 if x < cle:

 L1.append(x)

 else:

 L2.append(x)

 return tri\_rapide(L1)+[cle]+tri\_rapide(L2)

L=[4,3,7,9,8,5] ; print("Par tri rapide : ",tri\_rapide(L))

# Autre implémention possible

def tri\_rapide2(L):

 if len(L) <= 1:

 return L

 L1, L2 = [], []

 cle = L.pop()

 L1=[x for x in L if x<cle] # Listes en compréhension, pas mal !

 L2=[x for x in L if x>cle]

 return tri\_rapide2(L1)+[cle]+tri\_rapide2(L2)

#L=[4,3,7,9,8,5] ; print("Par tri rapide 2 : ",tri\_rapide2(L))