Les Tris

Trier est une opération courante en informatique. Pour résoudre un problème on commence souvent par trier. Si L est une liste, la méthode pour trier une liste est

`L.sort()`.

NB : la liste est triée sur place et ne renvoie aucune valeur (ne pas écrire `L=L.sort()` mais seulement `L.Sort()`).

Exemple d'application : Médiane d'une liste L de nombres

Principe Il faut d'abord trier la liste L. La médiane correspond à la valeur qui sépare la série en 2.
n étant le nomber d'éléments de L, si n est impair, c'est L[(n+1)/2], si n est pair, c'est (L[n/2] +L[n/2+1])/2

```
def Mediane(L):
    '''L:list ->Med:élemnt de L'''
    L.sort()
    n=len(L)
    if n%2==0:return L[(n+1)//2]
    else : return (L[n//2] +L[n//2+1])/2
```

Tri élémentaire On recherche la plus grande valeur de L, on la retire et on la place dans une deuxième liste.

```
def TriMax(L):
      '''L: liste ->l: liste'''
      l=[]
      while len(L)!=0:
            M=max(L)
            l.append(M)
            L..remove(M)
      return l
```

Tri à bulles Le tri à bulles ou tri par propagation est un algorithme de l'algorithme parcourt le tableau, compare les couples d'éléments successifs. Lorsque les deux éléments ne sont pas dans l'ordre croissant, ils sont échangés. Après chaque parcours complet du tableau, l'algorithme recommence l'opération.
Deux options sont possibles :

- Lorsqu'aucun échange n'a lieu pendant un parcours, cela signifie que le tableau est trié. On arrête alors l'algorithme.
- A chaque étape, on sait que le maximum est repoussé en fin de liste, donc on recommence en allant à chaque étape une case moins loin. Dans ces conditions, le nombre de parcours est n-1 où n est le nombre d'éléments de la liste.

```
def TriBulle(V):
  """TriBulle version for-for,
  trie la liste V"""
  n=len(V)
  for i in range(n):
    for j in range(n-i-1):
      if V[j] < V[j+1]:
        V[j],V[j+1]=V[j+1],V[j]
  return V
```

```
def TriBulle(V):
  """TriBulle version while-for,
  trie la liste V"""
  n=len(V)
  ech=True
  while ech :
# Tant qu'il y a des échanges
# (ech=True)
    ech=False
    for j in range(n-1):
      if V[j] > V[j+1]:
        V[j],V[j+1]=V[j+1],V[j]
        ech=True
  return V
```

Tri par insertion Dans l'algorithme, on parcourt le tableau à trier du début à la fin. Au moment où on considère le i-ème élément, les éléments qui le précèdent sont déjà triés.
L'objectif d'une étape est alors d'insérer le i-ème élément à sa place parmi ceux qui précèdent.
Il faut pour cela comparer T[i] avec les valeurs précédentes de T. Si on trouve T[j] > T[i], on insère la valeur T[i] avant la valeur T[j] (pour cela, on detruit T[i] et on insere en j-eme position la valeur x=T[i])

```
def TriInser(T):
    n=len(T)
    for i in range(1,n):
        for j in range(i):
            if T[j]> T[i]:
                x=T[i] ; del(T[i]) ; T.insert(j,x)
                # autre possibilité  T[j,i+1]=T[i],T[j,i]
                break
        print(T)
    return(T)
```