Exercice (tri insertion-Ordre croissant)

On rappelle l'algorithme du tri insertion sur un tableau A[1..n]

```
1   Pour j allant de 2 a n :
2      cle ← A[ j ]
3      // Insere A[j] dans la suite triee A[1,..,j-1]
4      i ←j-1
5      Tant que i > 0 et A[ i ] > cle Faire :
6            A[ i+1 ]← A[i]
7            i← i-1
8      Fin Tant Que
9      A[ i+1 ] ←cle
10  Fin Pour
```

1. Appliquer l'algorithme à la liste de nombres : 42 15 19 8 23
2. Indiquer le numéro de la ligne où se passe l'insertion éventuelle.
3. Pour chaque itération de la boucle **Pour** , comptez le nombre de comparaisons (A[i] > cle) de la ligne **5**, le nombre d'affectations de la ligne **6** (décalages vers la droite) .
   (Pour ce faire ,recopiez et compléter le tableau ci-dessous)

itération(s)	nombre de comparaisons(ligne 5)	nombre d'affectations (ligne 6)
1ere itération		

4. Reprendre la question précédente en utilisant la liste: 50 10 15 20 31 35
5. Dans le cas où l'on applique l'algorithme à une liste de n ($n \geqslant 2$) nombres ordonnés dans l'ordre décroissant , donner le total du nombre d'opérations indiquées dans la question 3 en fonction de n.
   Quelle est dans ce cas la complexité de l'algorithme de tri par insertion?

Exercice 2 (tri à bulles)

Dans le tri à bulles , on commence par comparer le premier élément A[0] avec le suivant A[1] et s'il est plus grand on échange A[0] et A[1].Puis on refait de même en comparant A[1] et A[2] et ainsi de suite .
De cette manière le plus grand élément du tableau est le dernier.

Puis on recommence l'étape précédente avec les n-1 premiers éléments du tableau.
Quand il ne reste plus qu'un élément , le premier en l'occurrence, l'algorithme s'arrête et le tableau est trié.

On traduit alors cet algorithme à l'aide du pseudo-code.

```
Tri à bulles sur un tableau A[1..n]
1  i=n-1
2  Tant Que i >=1 faire
3     Pour j allant de 1 à i
4        Si A[ j ] > A[ j+1 ]
5           Echanger A[ j ] et A[ j+1 ]
6        Fin Si
7     Fin Pour
8     i = i-1
9  Fin Tant Que
```

1. Appliquer l'algorithme à la liste de nombres : 71 15 50 8 12 32
2. Pour chaque itération de la boucle **Tant Que** de la ligne 2 , comptez le nombre de comparaisons de la ligne **4**, le nombre d'échanges de la ligne **5** .
   On pourra copier et remplir le tableau suivant:

itération(s)	nombre de comparaisons(ligne 4)	nombre d'échanges(ligne 5)
1ere itération(i=5)		
2eme itération (i=4)		

3. Refaire les questions 1 et 2 avec la liste de nombres 71 50 32 15 12 8
4. Refaire la même chose avec la liste : 8 12 15 32 50 71
5. Dans le cas où la liste de nombres à trier contient n nombres, donner en fonction de n la complexité dans le pire des cas puis dans le meilleur des cas.

Exercice 3 (une autre version du tri à bulles)

Cette fois-ci , on commence par comparer le dernier élément A[n] avec le précédent A[n-1] et s'il est plus petit on les échange .Puis on refait de même en comparant A[n-1] et A[n-2] et ainsi de suite .
De cette manière le premier élément du tableau est le plus petit.

Puis on recommence l'étape précédente avec les n-1 derniers éléments du tableau.
Quand il ne reste plus qu'un élément , le dernier en l'occurrence, l'algorithme s'arrête et le tableau est trié.

Ecrire cet algorithme en pseudo-code puis le programmer en Python.

Exercice 4

1. Dans l'algorithme de Tri à bulles ,si lors d'une étape de la boucle **Tant Que** principale, aucun échange n'est effectué dans la boucle **Pour** interne, c'est que le tableau est trié. Il est donc inutile de poursuivre la boucle externe.
   Décrivez un algorithme qui tient compte de cette remarque.

2. Combien de comparaisons d'éléments du tableau sont-elles effectuées lors du tri d'un tableau de longueur n avec cette variante du tri à bulles ? Décrivez le meilleur et le pire des cas.

3. Réalisez une fonction qui trie le tableau passé en paramètre selon cet algorithme.