Exercice 1 On donne la fonction `Python` suivante où le paramètre `L` est une liste :

```
def Tri(L) :
    n = len(L)
    for p in range(n-1) :
        pmin = p
        for j in range(p,n) :
            if L[j] < L[pmin]:
                pmin = j
        L[pmin],L[p] = L[p],L[pmin]
    return L
```

1. On saisit l'instruction `L = Tri([7,4,3,2,5])`. Remplir le tableau de variables suivant permettant d'illustrer le fonctionnement de ce tri :

p	0	1	2	3
pmin	3			
L	[2,4,3,7,5]			

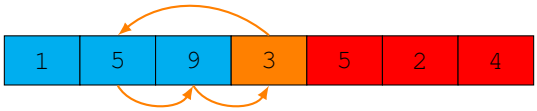
2. Décrire le fonctionnement de ce tri.
3. Comment peut-on appeler ce tri ? Quelle est sa différence avec celui vu en cours ?
4. Donner le nom d'un autre algorithme de tri.

Exercice 2 Réécrire la fonction `Tri` de l'exercice précédent pour trier dans l'ordre décroissant et non dans l'ordre croissant.

Exercice 3 On considère l'algorithme suivant où le paramètre `L` est un tableau :

```
def TriInsertion(L):
    n = ...........
    for i in range(1,n):
        temp = L[i]
        p = 0
        while ..... < temp:
            p = p + 1
        for j in range(i-1,p-1,-1):
            L[j+1] = ...........
        L[p] = ...........
    return L
```

1. Compléter cette fonction pour qu'elle effectue un tri par insertion dans l'ordre croissant de la liste `L`.
2. A quoi correspondent les variables `i` et `p` dans le cadre de la liste `L` ? Placer ces deux variables sur le schéma ci-dessous :



3. Combien de comparaisons sont effectuées dans le pire des cas ? Comparer ce résultat avec ce qui a été vu dans le cours.

Exercice 4 Réécrire la fonction `TriInsertion` de l'exercice précédent pour trier une liste dans l'ordre décroissant et non dans l'ordre croissant.

Exercice 5 (Tri par dénombrement) On considère une liste L de n nombres entiers, distincts ou non, et dont les valeurs sont comprises entre 0 et $K-1$, K étant un nombre entier supérieur ou égal à 1 fixé une fois pour toute et indépendant de n.
Ecrire un algorihme de complexité linéaire sur les comparaisons permettant de trier le tableau L par ordre croissant.