1. Directement présent en Python

1.1. Tri par la méthode sorted (to sort trier en anglais)

```
nombres = [1, 3, 4, 2]

print(sorted(nombres))                    [1, 2, 3, 4]

print(nombres)                            [1, 3, 4, 2]
```

La fonction sorted renvoie une nouvelle liste. La liste nombres n'est pas modifiée.

1.2. Tri par méthode dédiée : sort (trier en anglais)

```
nombres = [1, 3, 4, 2]
# Tri par valeurs croissantes
nombres.sort()
print(nombres)                                  [1, 2, 3, 4]
# Tri par valeurs decroissante
nombres.sort(reverse = True)
print(nombres)                                  [4, 3, 2, 1]
```

On remarque ici que le print(nombres) après l'application de la méthode sort renvoie bien la liste de départ. Le tri ici se fait en place (à la place de la liste originelle).

2. Tri par sélection

Il consiste à parcourir le tableau de la gauche à la droite en maintenant la partie déjà triée sur la gauche.

- A chaque étape on recherche dans la partie non triée le plus petit élément et on l'échange avec celui le plus à gauche.
- On recommence ainsi jusqu'à avoir parcouru tout le tableau.
- Pour effectuer l'échange on utilise une fonction « échange »

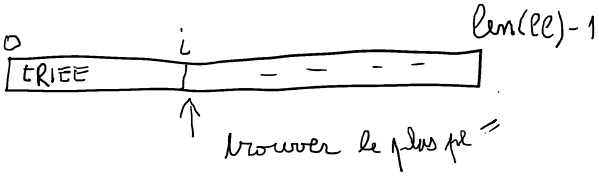
Trié	Non trié	Indice
	[10 ,2, -3 ,8 5,4]	0 : le plus petit est -3 on échange avec 10
[-3 ,	2, 10 ,8 ,5,4]	1 : le plus petit est 2 il est à la bonne place pas besoin de l'échanger
[-3 , 2	10 ,8 ,5,4]	2 : le plus petit est 4 on échange avec 10
[-3 , 2 , 4	,8 , 5 ,10]	3
[-3,2,4,5	,8,10]	4
[-3,2,4,5,8	,10]	5 : c'est fini le dernier terme est forcément le plus grand
[-3,2,4,5,8,10]		

On définie une fonction échange

```
def echange( t,i,j):
    temp = t[i]
    t[i]= t[j]
    t[j]=temp

ll =[1,2,3,4]
print(ll)
echange(ll,0,3)
print(ll)
```

Pour parcourir le tableau on utilise une fonction for :
for i in range(len(ll)):

La partie avant i est supposée déjà triée :



Pour parcourir la deuxième partie de la liste (de i à len(liste) -1)on utilise une boucle for et une variable m pour retenir le plus petit indice rencontré

```
m = i
for j in range(i+1,len(ll)):
    if t[j] <t[m] :
        m = j
```

Nous avons donc deux boucles for imbriquées

La première qui part de l'indice 0 pour balayer tous les indices du tableau
La deuxième qui ne s'intéresse qu'a la partie non triée de i+1 à lent(ll) -1
et qui échange les valeurs de l'indice i+1 avec un éventuel plus petit

```
def echange( t,i,j):
    temp = t[i]
    t[i]= t[j]
    t[j]=temp

ll =[-10,5,8,45,-96,12,897]
```

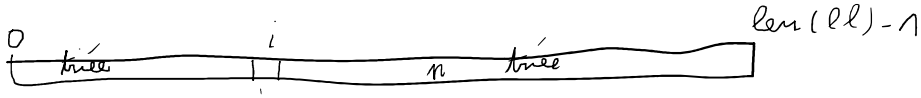
```
def tri_par_sélection(ll):
    for i in range(len(ll)):
        m = i
        for j in range(i+1,len(ll)):
            if ll[j] <ll[m] :
                m = j
        echange(ll,i,m)

tri_par_sélection(ll)
print(ll)
```

3. Tri par insertion

Autre algorithme souvent utilisé par les joueurs de cartes.

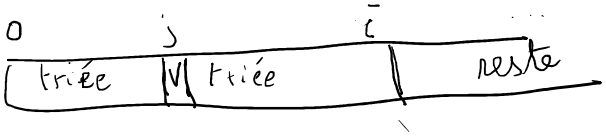
Même principe de départ parcours du tableau de la gauche vers la droite avec une partie gauche déjà triée.

On prend en compte le premier élément de la partie non triée



On cherche la place de cet élément dans la partie triée en recherchant sa position dans la partie triée

On insère cet élément à sa place.



On décale les éléments de la partie triée plus grands d'une position vers la droite

Trié	Non trié	Indice
[10 ,	2, -3 ,8 ,5,4]	
[10, 2,	-3 ,8, 5,4]	1 : on met 2 à la bonne place dans la partie triée ici on décale le 10 de un rang et on insère le 2 la bonne place
[2, 10,	-3 ,8, 5,4]	
[2, 10,-3	,8 5,4]	2 : on décale les valeurs jusqu'à mettre le -3 à la bonne place
[-3,2, 10,	,8 ,5,4]	
[-3,2, 10,8	,5,4]	3
[-3,2,8,10	,5,4]	

[-3,2,8,10,5	,4]	4
[-3,2,5,8,10	,4]	
[-3,2,5,8,10,4]		5
[-3,2,4,5,8,10]		fini

On définit une fonction insérer pour mette la valeur en fin de partie triée à la bonne place

```
ll =[-10,5,8,45,-96,12,897]

def inserer(t,i,v):
    j = i
    while j > 0 and t[j-1] > v :     # pas 0 car c'est déjà trié
                                     # on part de la droite donc de i

        t[j] =  t[j-1]   # décalage d'une position à droite des valeurs
        j = j-1  #décalage vers la droite pour le tour suivant
    t[j] = v # on insére v à sa place

def tri_par_insertion(t):
# on applique cette fonction de l'indice 1 au dernier.

    for i in range (1, len(t)):
        inserer(t,i,t[i])

tri_par_insertion(ll)
print(ll)
```