Fiche de révision — Les tris

Vous pouvez revoir la feuille_ info_ 18, pour ceux qui veulent aller plus loin, il faudrait aller voir le tri fusion ou le tri rapide

1. Tri par sélection

Principe. À chaque position i, on cherche le plus petit élément parmi ceux qui restent et on l'amène en position i. Autrement dit : on extrait le minimum, encore et encore.

Version 1 *(recherche du minimum par indice, double boucle for).*

```
def tri_selection(L):
    n = len(L)
    for i in range(n):
        imin = i                        # indice du minimum de L[i:]
        for j in range(i+1, n):
            if L[j] < L[imin]:
                imin = j
        L[i], L[imin] = L[imin], L[i]   # on place ce minimum en position i
```

Attention c'est une fonction qui modifie `L` et qui renvoie `None`.

Exemple d'utilisation :

```
L = [3,2,1,5]
tri_selection(L)
print(L)
```

Version 2 — la plus simple à retenir (s'appuie sur `min` et `remove`).

```
def tri_selection(L):
    L = L[:]                # copie : sinon remove vide la liste d'origine
    R = []
    while L != []:
        m = min(L)
        L.remove(m)         # on retire ce minimum de L
        R.append(m)         # et on l'ajoute au resultat
    return R
```

2. Tri par insertion

Principe. On parcourt la liste de gauche à droite ; chaque nouvel élément est inséré à sa place dans la partie déjà triée à sa gauche — exactement comme on range des cartes dans sa main.

Version 1 *(Une boucle while dans une boucle for)*

```
def tri_insertion(L):
    n = len(L)
    for i in range(1, n):
        x = L[i]                # element a inserer dans L[0:i] (deja triee)
        j = i - 1
        while j >= 0 and L[j] > x:
            L[j+1] = L[j]       # on decale vers la droite
            j = j - 1
        L[j+1] = x              # on insere x a sa place
```

Attention c'est une fonction qui modifie `L` et qui renvoie `None`.

Version 2 - *(qui ne modifie pas la liste passée en argument)*

```
def tri_insertion(L):
    L = L[:]                    # copie : la liste d'origine n'est pas modifiee
    n = len(L)
    for i in range(1, n):
        x = L[i]                # element a inserer dans L[0:i] (deja triee)
        j = i - 1
        while j >= 0 and L[j] > x:
            L[j+1] = L[j]       # on decale vers la droite
            j = j - 1
        L[j+1] = x              # on insere x a sa place
    return L
```

3. Tri par comptage

Principe. Pour des entiers dont les valeurs restent dans un intervalle borné : on compte le nombre d'occurrences de chaque valeur, puis on reconstruit la liste triée à partir de ces effectifs. Aucune comparaison entre éléments.

```
def tri_comptage(L):
    if L == []:
        return []
    a, b = min(L), max(L)               # valeurs extremes
    C = [0] * (b - a + 1)               # tableau de comptage
    for x in L:
        C[x - a] += 1                   # on compte les occurrences de x
    R = []
    for v in range(b - a + 1):
        R += [v + a] * C[v]             # on reconstruit la liste triee
    return R
```