3 Exercices

Exercice 1

1. Écrire une fonction `ind_max` renvoyant l'indice du maximum d'une liste de nombres.
2. Utiliser cette fonction pour écrire une fonction `tri_select_2` réalisant le tri d'une liste de nombres `L`, toujours dans l'ordre croissant, mais de la manière suivante :
   — sélectionner le plus grand élément de `L` et l'échanger avec le dernier élément de `L`,
   — sélectionner le deuxième plus grand élément de `L` et l'échanger avec l'avant-dernier élément de `L`,
   — etc.

   On commencera par décrire ce que doit contenir successivement la liste `L` lors de l'exécution de l'algorithme si initialement `L` vaut `[5,1,15,2,10,8,3]`. De plus, afin d'utiliser la fonction de la question 1, on pensera à utiliser des sous-listes.

Exercice 2

Écrire une fonction `tri_insert_2` réalisant le tri *dans l'ordre décroissant* d'une liste de nombres `L` en suivant une méthode similaire à celle du tri par insertion.

ex 1 : 1)
```
def indmax(L):
    n = len(L)
    ind = 0
    for k in range(1,n):
        if L[k] > L[ind]:
            ind = k
    return ind
```

2)
```
def triselect2(L):
    n = len(L)
    for i in range(n-1,-1,-1):   # ou range(n-1,0,-1)
        ind = indmax(L[0:i+1])
        L[ind], L[i] = L[i], L[ind]
    return L
```

L vaut [5,1,15,2,10,8,3]
[5,1,3,2,10,8,15]
[5,1,3,2,8,10,15]

[5,1,3,2,8,10,15]
[2,1,3,5,8,10,15]
[2,1,3,5,8,10,15]

[1,2,3,5,8,10,15]
[1,2,3,5,8,10,15]

nb : dernière étape inutile à cause de ⊛

ex 2 :
```
def tri_insert_2(L):
    n = len(L)
    for k in range(1,n):
        x = L[k]
        j = k
        while L[j-1] < x and j >= 1:
            L[j] = L[j-1]
            j = j-1
        L[j] = x
    return L.
```