Les algorithmes classiques

I- Parcours séquentiel dans un tableau pour rechercher un élément.

```
1  def recherche_naive(tab,val):
2      for i in range(len(tab)):
3          if tab[i] == val:
4              return i
5
6      return -1
7
8  liste = [1,4,32,7,3,9]
9  valeur = 7
10 print (recherche_naive(liste, valeur))
```

Tester le programme avec valeur = 7 puis valeur= 10.

Dans le cas où la valeur est dans la liste, que renvoie le programme ? (à quoi cela correspond t-il ?)

Dans le cas où la valeur n'est pas dans la liste que renvoie le programme ?

Ce programme est-il efficace si la liste est triée ? Pourquoi ?

Rappel : len() renvoie la longueur du tableau

On parcours 1 fois tout le tableau qui contient n éléments.
On dit que la complexité est en O(n). Cela veut dire que le temps d'exécution va dépendre linéairement du nombre de variables n du tableau.

Faire une fonction maxi(liste) qui renvoie le maximum d'une liste.
(si besoin aide aide : cf cours algorithmique page 14)
Faire une fonction mini(liste) qui renvoie le minimum d'une liste.
Faire une fonction moyenne(liste) qui renvoie la valeur moyenne des nombres de la liste

II- Tri par sélection / Tri par insertion

→ Visualisez les 3 vidéos pour comprendre le principe des deux algorithmes (tri insertion et tri selection)

Introduction
https://www.youtube.com/watch?v=ra79TDfotno

Tri par Sélection :
https://www.youtube.com/watch?v=8u3Yq-5DTN8

```
1  def tri_selection(tab):
2
3      for i in range(len(tab)):
4
5          # trouver le min
6          min  = i
7          for j in range (i+1 , len(tab)):
8              if tab[min] > tab[j]:
9                  min = j
10
11         # permuter avec élément indice i
12         tmp = tab[i]
13         tab[i] = tab[min]
14         tab[min] = tmp
15
16
17     return tab
18
19
20 print (tri_selection([10,7,3,5,8]))
```

On recherche le plus petit élément dans toute la liste

On le positionne au début de la liste (en position indice i)

Tri par Insertion :
https://www.youtube.com/watch?v=bRPHvWgc6YM

```
1
2   def tri_insertion(tab):
3
4       # parcours de 1 à longueur du tableau
5       for i in range(1, len(tab)):
6           k = tab[i]
7           j = i-1
8
9           # Recherche emplacement à insérer
10          while j >= 0 and k< tab[j] :
11              tab[j+1] = tab[j]
12              j = j-1
13
14
15          # insertion
16          tab[j+1] = k
17
18
19  liste = [98, 22, 15, 32, 2, 74]
20  tri_insertion(liste)
21  print (liste)
```

On met dans k l'élément à positionner

On décale tous les éléments à droite

On place k à la bonne position

Travail à faire pour les deux programmes précédents (tri par sélection et insertion) :
Voir les TD associés.

III- Recherche Dichotomique dans un tableau (ou liste) Trié :
https://www.youtube.com/watch?v=RrZuuuJDLis

Principe :
https://professeurb.github.io/articles/dichoto/

```
def recherche_dichotomique(tab, val):
    gauche = 0
    droite = len(tab) - 1
    while gauche <= droite:
        milieu = (gauche + droite) // 2
        if tab[milieu] == val:
            # on a trouvé val dans le tableau,
            # à la position milieu
            return milieu
        elif tab[milieu] > val:
            # on cherche entre gauche et milieu - 1
            droite = milieu - 1
        else: # on a tab[milieu] < val
            # on cherche entre milieu + 1 et droite
            gauche = milieu + 1
    # on est sorti de la boucle sans trouver val
    return -1
```

Ecrire la fonction et exécuter la en lui passant comme paramètre une liste TRIEE et une valeur à rechercher.
Ex :
Tab= [1,3,7,8,12,15,25,37,42]
Et val = 14
Puis val = 12.

Que renvoie la fonction si val n'est pas dans la liste ?
Que renvoie la fonction si val est dans la liste ?

Complexité en $O(\log_2(n))$ (sera vu ultérieurement)