Quelques algorithmes importants sur les listes

Recherche de minimum et maximum :

Recherche d'un maximum ou minimum dans une liste :

Exercice 1.

```
List=[2,10,11,4,7,12]
max=List[0]
for k in range(len(List)):
     if List[k]>Max:
         Max=List[k]
print(Max)
```

Exercice 2. 1. Script de recherche de maximum

```
def creation_liste(n):
    List=[]
    for k in range(n):
            List=List+[int(input('entre l\'entier numéro suivant'))]
    return List
```

2. Script de création de liste

```
def MaximumListe(List):
    Max=List[0]
    for k in range(len(List)):
            if List[k]>Max:
                Max=List[k]
    return Max
```

3. Script de création de liste puis de recherche de maximum :

```
def max_liste(n):
    List=[]
    for k in range(n):
            List=List+[int(input('entre l\'entier numéro suivant'))]

    Max=List[0]
    for k in range(len(List)):
            if List[k]>Max:
                Max=List[k]
    return Max
```

ou bien en utilisant les deux fonctions

```
def max_liste(n):
    List=creation_liste(n)
    Max=Maximum_liste(List)
    return Max
```

4. Script de création de liste et de recherche de minimum :

```
def min_liste(n):
    List=[]
    for k in range(n):
            List=List+[int(input('entre l\'entier numéro suivant'))]

    Min=List[0]
    for k in range(len(List)):
            if List[k]<Min:
                Min=List[k]
    return Min
```

Recherche du deuxième maximum :

Exercice 3.

```
import numpy as np
def deuxieme_max(List):
    Max1=np.max(List)
    k=0
    List.remove(Max1)
    Max2=np.max(List)
    return Max2
```

1. L'algorithme suivant, à compléter, permet de parcourir une seule fois la liste et d'en donner les deux maxima.

```
def deux_maxima(List):
    save1=List[0]
    save2=List[1]
    if save1>save2:
        save1,save2=save2,save1
    for i in range(2,len(List)):
       if save2<=List[i]:
           save1,save2=save2,List[i]
       elif save1<List[i]<save2:
           save1=List[i]
    return(save1,save2)
```

Recherche de la distance minimale entre deux valeurs :

Exercice 4. On considère la liste suivante $L = [2, 3, 5, 11, 7, 5]$.

1. Compléter le script suivant pour qu'il permette de trouver la distance entre un nombre de la liste et le nombre 4

```
L=[2,3,6,11,7,5]
distance=4-L[0]
for k in range(len(L)):
        if (abs(4-L[k])<distance):
            distance=abs(4-L[k])
print(distance)
```

2.
```
L=[2,3,6,11,7,5]
distance=5-2
for k in range(len(L)):
    for i in range(len(L)):
            if (i!=k) & (abs(L[k]-L[i])<distance):
                distance=abs(L[k]-L[i])
print(distance)
```

3.
```
import numpy as np
def distance_liste(List):
    List_distances=[]
    for k in range(len(List)):
        for i in range(len(List)):
            if i!=k:
                List_distances=List_distances+[abs(List[i]-List[k])]
    return(np.min(List_distances))
```

4.
```
def distance_liste(List):
    distance=abs(List[0]-L[1])
    for k in range(len(List)):
```

```
            for i in range(len(List)):
                if (i!=k) & (abs(List[k]-List[i])<distance):
                    distance=abs(List[k]-List[i])
        return distance
```

Parcours de listes :

Boucle For avec les listes

Exercice 5. 1.

2.
```
def suppression_impairs(List1):
    List2=[x for x in List1 if x%2==0]
    return List2
```

Recherche dichotomique dans une liste triée.

Exercice 6.

```
def dichotomie(List, nombre):
    a = 0
    b = len(List) - 1
    while a <= b:
        m = (a + b) // 2
        if List[m] == nombre:
            return True
        elif List[m] < nombre:
            a = m + 1
        else:
            b = m - 1
    return False
```

Comptage dans une liste :

Exercice 7.

1.
```
def nb_mineurs(liste_personnes):
    compteur = 0
    for (nom,age) in liste_personnes:
        if age<18:
            compteur=compteur+1
    return compteur
```

2.
```
def nb_personnes(liste_personnes):
    c1,c2,c3 = 0,0,0
    for (nom,age) in liste_personnes:
        if age<=18:
            c1=c1+1
        if 18<age<=60:
            c2=c2+1
        if age>60:
            c3=c3+1
    return (c1,c2,c3)
```