

**DS n° 02 – DS02**

- Faire tous les exercices dans un même fichier NomPrenom.py à sauvegarder,
- mettre en commentaire l'exercice et la question traités (ex : # Exercice 1),
- ne pas oublier pas de commenter ce qui est fait dans votre code (ex : # Je crée une fonction pour calculer la racine d'un nombre),
- il est possible de demander un déblocage pour une question avec une (\*), mais celle-ci sera notée 0,
- il faut vérifier avant de partir que le code peut s'exécuter et qu'il affiche les résultats que vous attendez.

**Exercice 1 : import de fichier et recherche par dichotomie**

Le fichier `liste_nombres.csv` contient une liste de nombres réels, classés par ordre croissant. L'emplacement de ce fichier est le suivant : `/home/eleve/Ressources/PTSI/`

- 1 (\*) Importer le fichier. Récupérer le contenu du fichier sous forme d'une liste de **flottants**.

**Solution 1.**

```
fichier = open("liste_nombres.csv", "r")
```

```
contenu=fichier.read()
```

```
fichier.close()
```

```
lignes=contenu.split('\n')
```

```
L=[]
```

```
for i in lignes:  
    L.append(float(i))
```

- 2 Afficher les 30 premières valeurs de la liste.

**Solution 2.** `L[0:30]`

- 3 Ecrire une fonction `dicho` qui prend comme entrée une liste `L` et un nombre `a` et renvoie la position de `a` si `a` est dans le tableau et `False` sinon.

**Solution 3.**

```
def dicho(L, a):  
    debut = 0  
    fin = len(L) - 1  
    while debut <= fin:  
        m = (debut+fin) // 2  
        if L[m] == a:  
            return m  
        elif L[m] < a:  
            debut = m + 1  
        else:  
            fin = m - 1  
    return False
```

- 4 Afficher le résultat de la fonction `dicho` pour la liste précédemment importée pour le nombre `a = 1.56` puis pour le nombre `a = 2`.

**Solution 4.** `print(dicho(L,1.56)) print(dicho(L,2))`

- 5 Calculer et afficher la moyenne des nombres de la liste importée.

**Solution 5.**

```
somme=0  
for i in L:  
    somme=somme+i  
print(somme/len(L))
```

- 6 Afficher la médiane des nombres de la liste importée.

**Solution 6.** `print(L[len(L)//2])`

## Exercice 2 : recherche du maximum

Dans cet exercice, vous n'avez pas le droit d'utiliser les fonctions Python suivantes : `max` et `index`.

- 7 Ecrire une fonction `maxi` qui prend comme entrée une liste de nombres `L` et renvoie la valeur ainsi qu'une position du maximum.

**Solution 7.**

```
def maxi(L):
    maxi=L[0]
    position=0
    for i in range(1,len(L)):
        if L[i]>maxi:
            maxi=L[i]
            position=i
    return(maxi,position)
```

- 8 Ecrire une fonction `maxi_liste` qui prend comme entrée une liste `L` et renvoie la valeur ainsi que la liste de toutes les positions de cette valeur maximale.

**Solution 8.**

```
def maxi2(L):
    maxi=L[0]
    position=[0]
    for i in range(1,len(L)):
        if L[i]>maxi:
            maxi=L[i]
            position=[i]
        elif L[i]==maxi:
            position.append(i)
    return(maxi,position)
```

- 9 Afficher le résultat de la fonction suivante pour la liste :  $L = [9, 1, 3, 9, 2, 9]$ .

**Solution 9.**

```
print(maxi2([9,8,9,0,9]))
```

## Exercice 3 : algorithme Glouton

Vous avez un budget maximal  $B = 11$  euros et des bonbons à acheter. Pour chaque type de bonbon, vous avez deux informations : son prix et votre indice de satisfaction. L'objectif est de maximiser l'indice de satisfaction total.

Pour cela deux stratégies gloutonnes :

- stratégie 1 : choisir toujours le bonbon le plus satisfaisant compatible avec le budget restant ;
- stratégie 2 : choisir toujours le bonbon au plus fort rapport  $\frac{\text{satisfaction}}{\text{prix}}$ , compatible avec le budget restant.

La donnée est la suivante : une liste qui contient des couples `[prix,satisfaction]`.

```
bonbons=[[3,2.5],[1.5,1.5],[1,0.9]]
```

Par exemple, le premier bonbon coûte 3 euros et son indice de satisfaction est 2,5.

La liste est triée par ordre décroissant de satisfaction.

- 10 Recopier la ligne ci-dessus.

**Solution 10.** `bonbons=[[3,2.5],[1.5,1.5],[1,0.9]]`

- 11 Afficher `bonbons[2][0]` et `bonbons[2][1]`. A quoi correspondent ces deux nombres ? (réponse en commentaire).

**Solution 11.** Le premier nombre correspond au prix du 3ème bonbon et le second à son indice de satisfaction.

- 12 Ecrire un programme qui donne la liste des couples `[prix,satisfaction]` pour maximiser la satisfaction en suivant la stratégie 1.

**Solution 12.**

```

i=0
achat=[]
while i<len(bonbons) and B>0:
    if bonbons[i][0]<B:
        achat.append(bonbons[i])
        B=B-bonbons[i][0]
    else:
        i=i+1

```

- 13 Calculer (pas à la main) et afficher la satisfaction totale pour cette stratégie 1.

**Solution 13.**

```

Satisfaction_totale=0
for element in achat:
    Satisfaction_totale=Satisfaction_totale+element[1]
print(Satisfaction_totale)

```

- 14 (\*) On appelle ratio le quotient  $\frac{\text{satisfaction}}{\text{prix}}$ . Construire (pas à la main) une liste `bonbons2` constituée de sous-listes : `[ratio, prix, satisfaction]`.  
Ainsi, la liste `bonbons2` devra commencer de cette façon : `[[0.8333334, 3, 2.5], ...]`

**Solution 14.**

```

bonbons2=[]
for element in bonbons:
    bonbons2.append([element[1]/element[0],element[0],element[1]])

```

- 15 Recopier la commande suivante : `bonbons2.sort(reverse=True)`.  
La liste `bonbons2` est alors triée par valeur de ratio décroissante.
- 16 Ecrire un programme qui donne la liste des triplets `[ratio,prix,satisfaction]` pour maximiser la satisfaction en suivant la stratégie 2.

**Solution 15.**

```

B=12
i=0
achat2=[]
while i<len(bonbons2) and B>0:
    if bonbons2[i][1]<B:
        achat2.append(bonbons2[i])
        B=B-bonbons2[i][1]
    else:
        i=i+1

```

- 17 Afficher la satisfaction totale pour cette stratégie 2.

**Solution 16.**

```

Satisfaction_totale2=0
for element in achat2:
    Satisfaction_totale2=Satisfaction_totale2+element[2]
print(Satisfaction_totale2)

```

- 18 Conclure quant à la meilleure stratégie. (réponse en commentaire)

**Solution 17.** La satisfaction dans la stratégie 2 est plus grande. Donc, c'est la meilleure sur cet exemple.