08.Recherche dans une liste

Nous utiliserons dans ce TP le langage de programmation *Python* au moyen de l'environnement de développement *IDLE*.

Exercice 1. Tester le programme suivant :

```
def positifs(l):
    '''extraction des valeurs positives d'une liste de nombres l'''
    L=[]
    for k in range(0,len(l)):
        if l[k]>=0:
            L.append(l[k])
    return(L)
```

Exercice 2. Créer puis tester une fonction *indicesdespositifs* de la variable l retournant la liste des indices des valeurs positives d'une liste de nombres l.

(par exemple $indicesdespositifs([1,0,-1,-5,8])$ vaut $[0,1,4]$)

Exercice 3. Créer puis tester une fonction *recherche* des variables v et l retournant la liste des indices des occurrences de la valeur v dans la liste de nombres l.

(par exemple $recherche(2,[1,2,-1,2,5,-7,2])$ vaut $[1,3,6]$)

Exercice 4. Créer puis tester une fonction *suppression* des variables v et l retournant la liste obtenue en supprimant les occurrences de la valeur v dans la liste de nombres l.

(par exemple $suppression(2,[1,2,-1,2,5,-7,2])$ vaut $[1,-1,5,-7]$)

Pour les exercices suivants, on représente une matrice comme la liste de ses lignes.

Exercice 5. Créer puis tester une fonction *diagonale* de la variable M retournant la diagonale de la matrice carrée M.

(par exemple $diagonale([[1,2,3],[4,5,6],[7,8,9]])$ vaut $[1,5,9]$)

Exercice 6. Créer puis tester une fonction *matricediagonale* de la variable d retournant une matrice carrée de diagonale d dont les coefficients non diagonaux sont nuls.

(par exemple $matricediagonale([1,2,3])$ vaut $[[1,0,0],[0,2,0],[0,0,3]]$)

Exercice 7. Créer puis tester une fonction *decomposition* de la variable M retournant une matrice de nombres positifs et une matrice de nombres négatifs dont la somme est égale à M.

(par exemple $decomposition([[1,-2,3],[-1,-5,7]])$ vaut $[[1,0,3],[0,0,7]],[[0,-2,0],[-1,-5,0]]$)

Réponses

2)
```
def indicesdespositifs(l):
    '''retourne les indices des valeurs positives d'une liste de nombres l'''
    L=[]
    for k in range(0,len(l)):
        if l[k]>=0:
            L.append(k)
    return(L)
```

3)
```
def recherche(v,l):
    '''retourne les indices des occurrences de la valeur v dans la liste de nombres l'''
    L=[]
    for k in range(0,len(l)):
        if l[k]==v:
            L.append(k)
    return(L)
```

4)
```
def suppression(v,l):
    '''retourne la liste obtenue en supprimant les occurrences de la valeur v dans la liste de nombres l'''
    L=[]
    for k in range(0,len(l)):
        if l[k]!=v:
            L.append(l[k])
    return(L)
```

5)
```
def diagonale(M):
    '''retourne la diagonale de la matrice carrée M'''
    d=[]
    for i in range(0,len(M)):
        d.append(M[i][i])
    return(d)
```

6)
```
def matricediagonale(d):
    '''retourne une matrice carrée de diagonale d dont les coefficients non diagonaux sont nuls'''
    M=[]
    for i in range(0,len(d)):
        M.append([])
        for j in range(0,len(d)):
            if i==j:
                M[i].append(d[i])
            else:
                M[i].append(0)
    return(M)
```

7)
```
def decomposition(M):
    '''retourne une matrice de nombres positifs et une matrice de nombres négatifs dont la somme vaut M'''
    Mp=[]
    Mn=[]
    for i in range(0,len(M)):
        Mp.append([])
        Mn.append([])
        for j in range(0,len(M[i])):
            if M[i][j]>=0:
                Mp[i].append(M[i][j])
                Mn[i].append(0)
            else:
                Mp[i].append(0)
                Mn[i].append(M[i][j])
    return(Mp,Mn)
```