Devoir surveillé n°1 - Corrigé

Exercice 1 : 1. Somme de deux matrices parcimonieuses :

```
def somme(M,N):
    ''' ajoute deux matrices parcimonieuces codées sous forme de dictionnaire '''
    S={'dim':M['dim']}
    for clef in M:
        if clef!='dim':
            if clef in N:
                S[clef]=M[clef]+N[clef]
            else:
                S[clef]=M[clef]
    for clef in N:
        if clef!='dim':
            if clef not in M:
                S[clef]=N[clef]
    return S
```

2. Pour calculer la complexité, nous devons remarquer deux choses :
   - la première boucle "`for clef in M`" va tourner (c_1+1) fois (le nombre de clefs dans `M`), et la deuxième boucle va tourner (c_2+1) fois
   - le test "`if clef in N`" va se réaliser **en temps constant** (particularité des dictionnaires : la recherche d'une clef dans un dictionnaire se fait en temps constant), de même que le test "`if clef not in M`".

   La complexité dette fonction est :$C(c_1,c_2) = 1+(1+c_1)(1+1+2)+(1+c_2)(1+1+1) = 4c_1+3c_2+8$,

   donc la complexité est en $O(c_1+c_2)$.

Exercice 2 : 1. `id_titre` est un entier (type `int`) égal à 49987654 à l'issue des instructions

`zones` est une liste de deux entiers (type `list`) égale à `[1,3]`

`date_fin` est une liste de trois entiers (type `list`) égale à `[2025,8,31]`

2. Voici le code :

```
passages=[] #passage sera une liste de listes
for i in range(2,len(lignes)): #on parcourt la fin des lignes du fichier
    donnees_passage=lignes[i].rstrip('\n').split(',') #on retire le retour
                    #chariot et on sépare selon les virgules
    date=donnees_passage[0].split('-') #on sépare ensuite la date selon
                    #les tirets
    heure=donnees_passage[1].split(':') #on sépare l'heure selon les
                    #deux points
    passages.append([int(date[0]),int(date[1]),int(date[2]),int(heure[0]),
    int(heure[1]),int(heure[2]),int(donnees_passage[2])]) #on rajoute la
    #liste demandée en convertissant toutes les données de chaînes
    #de caractères à entiers.
```

3. `date1` est antérieur à `date2` si son année est strictement antérieure, ou si les années sont identiques et le mois est strictement antérieur, ou si les années et mois sont identiques et le jour est antérieur.

```
def estAvant(date1,date2):
    return (date1[0]<date2[0]) or (date1[0]==date2[0] and date1[1]<date2[1])
    or (date1[0]==date2[0] and date1[1]==date2[1] and date1[2]<=date2[2])
```

4. Voici le code :

```
def nbSecondesEntre(heure1,heure2):
    return 3600*(heure1[0]-heure2[0])+60*(heure1[1]-heure2[1])+heure1[2]-heure2[2]
```

5. On passe en arguments toutes les données définies dans l'énoncé, les données du lecteur avec des majuscules et les données du titre avec des minuscules.

```
def testPassage(Zone,Id_point, Liste_noire,Maintenant,id_titre,zones,date_fin,passages):
    for x in Liste_noire: #on vérifie si l'id du titre est dans la liste noire
        if x==id_titre:
            print('Titre refusé')
            return False
    if Zone<zones[0] or Zone>zones[1]: #on vérifie si la Zone dépasse les zones autorisées
        print("Non valide dans cette zone")
        return False
    if estAvant(date_fin, Maintenant[:3]): #on vérifie si la date
    #d'expiration est avant ou après la date actuelle
        print("Titre expiré")
        return False
    for x in passages: #on regarde tous les passages de la carte
        heure_passage=x[3:6]
        point=x[6]
        if (point==Id_point and
        nbSecondesEntre(Maintenant[3:],heure_passage)<450): #on vérifie
        #qu'il n'y a pas eu de passage trop récent au même point.
            print("Titre déjà validé")
            return False
    return True #on renvoie True si le titre a passé tous les tests
```

Exercice 3 : .

1. Obtention du numéro correspondant à un couple (x, y) :

```
def numero(x,y):
    if x==0 and y==0:
        return 0
    if y==0:
            #dans ce cas on est sur l'axe des ordonnées
        return numero(0,x-1)+1
    return numero(x+1,y-1)+1
```

2. Obtention du couple correspondant à un numéro n :

```
def couple(n):
    if n==0:
        return (0,0)
    x,y=couple(n-1)
    if x==0:
        #dans ce cas on est sur l'axe des abscisses
        return (y+1,x)
    else:
        return (x-1,y+1)
```