L2 MIASHS – Informatique S3
TD3 : Tuples et listes

2024

Les exercices fondamentaux qui ne sont pas finis en TD doivent être faits chez vous dans la semaine. Les exercices complémentaires sont aussi conseillés. Ce sont de bons points d'appuis pour avancer et réviser. Le prochain contrôle portera sur le contenu du CM et aussi le contenu des exercices 1 à 4.

1 Rappels du cours

Exercice 1

1. Créez un tuple `t` contenant 5 éléments.
2. Essayez de modifier le troisième élément en faisant `t[2]=4`. Que se passe-t-il ?

2 Exercices fondamentaux

Exercice 2

1. Ecrivez une fonction prenant une liste comme paramètre et retournant un 2-uple (on appelle ça un couple) constitué du plus petit et du plus grand élément de la liste.
2. Testez votre fonction, et vérifiez que le résultat obtenu est bien un tuple en affichant le résultat de la commande `type(x)`. Cette commande retourne le type de `x`.

Exercice 3 – Représentation des matrices creuses

L'objectif ici est de stocker des matrices de très grandes dimensions dans peu de places en mémoire, sous certaines conditions.

En Python, une matrice peut être représentée sous forme de liste de listes. Chaque sous-liste représente une ligne de la matrice, et les éléments de cette sous-liste correspondent aux valeurs des colonnes de cette ligne. Par exemple, une matrice 3x3 contenant les valeurs suivantes :

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

peut être représentée en Python de la manière suivante :

```
matrice = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
```

ou de manière équivalente :

```
matrice = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
```

Ainsi, pour accéder à l'élément situé sur la deuxième ligne et la troisième colonne (qui est ici `6`), vous pouvez utiliser l'indexation comme suit :

```
element = matrice[1][2] # Cela renvoie 6
```

Pour parcourir tous les éléments de la matrice, on peut utiliser des boucles imbriquées comme suit :

```
for ligne in matrice:
    for element in ligne:
        print(element)
```

Ce code parcourt chaque élément de chaque ligne de la matrice et les affiche un par un. Cette structure permet de traiter chaque élément de manière individuelle dans une matrice de n'importe quelle dimension.

Matrices creuses : Une matrice creuse est une matrice dont l'immense majorité des valeurs est nulle (vaut 0). Dans ce cas, plutôt que de stocker la matrice dans une liste de listes, il peut être économe (pour la mémoire) de stocker un *triplet* constitué de :

- les dimensions de la matrice `n` et `m`,
- un dictionnaire `d`, dont les clés sont les couples des coordonnées non nulles.

Par exemple la matrice :

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

sera représenté par le triplet `(5,6,{(0,0):1,(0,3):3,(2,2):8,(3,0):4})`. Avec des très grandes matrices le gain est énorme.

1. Ecrire une fonction qui étant donné un triplet `(n,m,d)` représentant une matrice selon la forme décrite ci-dessus la transforme en une liste de listes selon la représentation classique des matrices.
2. Ecrire une fonction qui fait l'inverse.
3. Ecrire une fonction qui prend deux matrices de même taille représentées sous forme triplet et calcule leur somme (case à case). Le résultat doit être fourni sous forme de triplet.

4. **En utilisant des appels aux 3 fonctions précédentes**, écrire une fonction qui prend comme paramètre deux matrices représentées sous forme de liste de listes, et retourne une matrice qui est leur somme dans la même représentation. Cela nécessite donc de :
   — changer la représentation
   — sommer
   — changer dans l'autre sens
5. Pourquoi n'est-ce pas très efficace ?

Exercice 4 – Variables modifiables et fonctions

1. Que produit le code suivant ?

```
def f(x) :
    x[1] = 3

l = [1,2,3]
f(l)
print(l)
```

   Essayez de répondre sur le papier puis testez. Et celui-ci ?

```
def f(x) :
    x = 3

l = 4
f(l)
print(l)
```

   Retenir qu'il y a toujours un risque lorsqu'on utilise des objets modifiables, comme les listes : le risque qu'ils soient modifiés.
2. Ensembles et dictionnaires sont-ils modifiables ou immuables ?

3 Exercices complémentaires

Exercice 5 – Ensembles et modifiables

Peut-on utiliser des ensembles avec des types d'éléments modifiables ? Essayez avec des listes, puis avec des tuples.

Exercice 6

Cherchez à quoi correspond la méthode `get` des dictionnaires et comment indiquer une *valeur par défaut*.

Reprenez le code de l'exercice 3 en utilisant cette méthode.

Ajoutez une méthode qui prend une matrice représentée sous forme de listes de listes et l'affiche lisiblement.

Ajoutez une méthode qui calcule le produit entre deux matrices représentées sous forme creuse.