C.R. T.P. 4 : début

1 Tableaux bidimensionnels vus comme listes doubles

La façon la plus simple pour fabriquer un tableau bidimensionnel en PYTHON est d'utiliser une *liste de listes*. Par exemple `L=[[1,2],[3,4]]` est ce qu'on appelle en informatique un *tableau bidimensionnel* qui code ce que les humains ont envie de voir comme le *tableau pour humain* : $\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$.

On dira qu'un tableau est carré si les lignes ont toutes la même longueur et le nombre de lignes est égal au nombre de colonnes. C'est le cas du tableau précédent. Ce n'est pas le cas de `M=[[1,2,3],[4,5,6]]` ni de `N=[[1,2,3],[4,5]]`.

a) **Question :** Fabriquer une fonction `affiche_tableau_pour_humain` qui est une fonction d'affichage : elle prend en argument un tableau informatique (liste de listes) et l'affiche ligne par ligne sans crochet ni virgules. Par exemple pour `L` elle donnera

```
1 2
3 4
```

Réponse :

```
def affiche_tableau_pour_humain(T):
    for i in range(len(T)):
        for j in range(len(T[i])):
            print(T[i][j],end=' ')
        print() # pour aller à la ligne à la fin de chaque boucle intérieure.
```

Cette fonction d'affichage doit vous permettre de bien visualiser les rôles respectifs de la boucle intérieure et de la boucle extérieure.

• La boucle extérieure est la boucle qui commence avec le `for i in range(len(T))`. L'indice `i` sera l'indice de la ligne qu'on va afficher.

• Pour chaque valeur de `i`, la boucle *intérieure* qui commence avec `for j in range(len(T[i]))` va permettre d'afficher toutes les entrées `T[i][j]` de la ligne `j`. L'indice `j` parcourt tous les numéros de colonnes de `0` au dernier indices de colonne de la ligne `T[i]`.

b) **Question :** Comment modifier `L` pour obtenir dans `L [[1,2],[3,5]]` ? **Réponse :** simplement avec `L[1][1]=5`. Il faut bien comprendre que pour la `L` qui est un liste de listes, `L[0]` est une liste, `L[1]` est aussi une liste et que chacune est modifiable.

c) Ecrire une fonction `Tzeros(m,n)` qui prend en arguments deux entiers `m` et `n` et retourne un tableau à `m` lignes et `n` colonnes rempli de zéros. **N.B.** Cette fois la fonction doit retourner un tableau, c'est-à-dire une liste de listes. On donne deux méthodes qui différent seulement dans le fait que dans la première on crée une liste `L` de liste qu'on recopie dans chaque ligne, alors que dans la seconde, à chaque tour de boucle, on recrée une liste de zéro.

• **Idée 1 (attention, mauvaise idée) :** on crée une liste `L` formée de n zéros, puis on la recopie m fois à l'intérieur d'une liste `T` :

```
def Tzerosv1(m,n):
    L=[0]*n
    T=[]
    for i in range(m):
        T=T+[L]
    return T
```

A priori tout va bien comme on peut le tester avec

```
In [8]: Tzerosv1(3,2)
Out[8]: [[0, 0], [0, 0], [0, 0]]
```

Attention : en fait cette idée 1 telle quelle n'est PAS une bonne idée ! Voir ce qui se passe à la question suivante !

• **Idée 2 (la bonne)** on recrée `L` à chaque tour de boucle.

```
def Tzerosv2(m,n):
    T=[]
    for i in range(m):
        L=[0]*n
        T=T+[L] # on rajoute cette ligne à T.
    return T
```

d) Soit `L=Tzeros(3,3)`. Modifier la première entrée de la première ligne de `L` pour y mettre un `1`. **C'est là qu'on va voir la différence entre les deux versions de Tzeros**

```
>>>T1=Tzerosv1(3,3)
>>>T2=Tzerosv2(3,3)
>>>T1==T2 # a priori pas de pb?
 True
>>> T1[0][0]=1
>>> T2[0][0]=1
>>> print(T1)
[[1, 0, 0], [1, 0, 0], [1, 0, 0]]
>>>print(T2)
[[1, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Pour `T1` la modification de la `T1[0][0]` a aussi modifié les entrés `0` des autres lignes ! ! Cela vient de la copie de la liste `L`, cf. cours !

e)
```
def testcarre(T):
    n=len(T)
    for i in range(n):
        if len(T[i])!=n:
            return False # permet de sortir de la fonction dès qu'on
           #détecte une mauvaise ligne.
    return True
```

Explications : Une erreur vue plusieurs fois sur la variable à passer en paramètre : ici une seule variable, le tableau `T`. La boucle `for` permet de tester pour chaque ligne `T[i]` de `T` si elle a la bonne longueur (qui est le nombre total de lignes `n`).
Le `return True` ne s'exécutera que si on a parcouru toute la boucle sans détecter une mauvaise ligne.

Questions f) et g) dans le C.R. suivant. Pour le g) on pourra se contenter de transformer le tableau en matrice rectangulaire :tous les lignes ont la même longueur, en complétant les lignes par des zéros. Pas besoin de le transformer en matrice carré, ce qui rajoute encore des zéros.

h)
```
def Transforme(L):
    t=len(L)
    m=(t//10)+1 # quotient de la div. eucl.
    T=Tzerosv2(m,10)# création d'un bon tableau de zéros.
    for i in range(t):
        T[i//10][i%10]=L[i]
    return T
```

Notez qu'on a utilisé par commodité la fonction `Tzerosv2` déjà créée, et on a utilisé à plein les opérations de divisions euclidienne.