Matrices

Exercice 0 : manipulations élémentaires

Pour rentrer sous Python une matrice dont on connait les coefficients, il suffit de rentrer le tableau des tableaux de ses lignes.

1. Pour commencer, définir la matrice $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 7 & 8 & 0 \end{pmatrix}$, en tapant sous Python :

```
A = [[1, 3, 5], [2, 4, 6], [7, 8, 0]]
```

Si i et j sont deux entiers et que $1 \leq i \leq$ (nombre de lignes de A) et $1 \leq j \leq$ (nombre de colonnes de A), on peut alors accéder au coefficient $A_{i,j}$ en tapant sous Python `A[i-1][j-1]`

Il y a ce décalage car Python compte les indices de 0 à $n-1$ et non pas de 1 à n.

Exemple : dans cette matrice pour accéder au coefficient $A_{2,3}$ (qui vaut 6), il faut demander à Python `A[1][2]`.

2. Changer maintenant le 8 de la matrice en 18.
3. Pour afficher la matrice A, il suffit de taper `print(A)` ou juste `A`. Le problème est que Python affiche toutes les lignes à la file, et il est ainsi difficile de se repérer. Pour avoir un meilleur rendu, taper :

```
for line in A:
    print(line)
```

Bien sûr cet affichage n'est toujours pas convaincant dès qu'on a des coefficients qui n'ont pas tous le même nombre de chiffres...

Ces manipulations sont très intuitives et permettent de faire la plupart des travaux demandés. Par contre, quand on veut manipuler de grandes matrices, cela devient pénible (par ex. si l'on veut créer I la matrice identité de taille 11...). Cela pose également un souci quand les coefficients de la matrice sont donnés par une formule : on ne pas s'amuser à effectuer tous les calculs puis à rentrer à la main la matrice, mais on va plutôt utiliser l'ordinateur pour automatiser le tout.

Afin d'automatiser les calculs, il faut d'abord que l'on crée une matrice remplie de zéros. Si n et p sont deux nombres entiers strictement positifs, pour créer une matrice A de taille $n \times p$ remplie de zéros, il faut taper sous Python :

```
A = [[0 for x in range(p)] for x in range(n)]
```

Exemple : pour créer la matrice $B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ il suffit de taper sous Python :

```
B = [[0 for x in range(3)] for x in range(2)]
```

Remarque 1 : pour de très grosses matrices on utilisera la fonction *xrange* plutôt que la fonction *range*

Remarque 2 : on peut aussi taper `A = [p*[0] for x in range(n)]` mais pas `A = n*[p*[0]]`, dans le dernier cas chaque ligne de la matrice A pointe vers le même tableau, et si on modifie la case A[1][3] par ex. on modifie également A[0][3], A[2][3], A[3][3] ...

Exercice 1 : un problème d'identité

Créer une matrice I de taille 11×11 remplie de zéros, puis programmer une suite d'instructions permettant de transformer cette matrice en la matrice identité de taille 11 (avec des 0 partout sauf des 1 sur la diagonale)

Exercice 2 : le triangle de Pascal

Un triangle de Pascal de taille n peut être placé dans une matrice carrée P de taille n : on ne s'intéressera qu'à la partie triangulaire inférieure (les cases de la diagonale et en-dessous), le reste des cases sont des zéros. Pour cela, il faut remplir P de la manière suivante :

- D'abord, remplir la première colonne de 1, c'est-à-dire avec la formule :
$$P_{i,1} = 1 \quad \text{pour tout } 1 \leq i \leq n$$
- A ce moment, la ligne 1 est remplie, puisqu'on ne s'intéresse qu'aux coefficients sur la diagonale et en-dessous. Ensuite, pour tout $2 \leq i \leq n$, si la ligne $i-1$ est remplie, alors on remplit la ligne i avec la formule :
$$P_{i,j} = P_{i-1,j-1} + P_{i-1,j} \quad \text{pour tout } 2 \leq j \leq i$$
C'est-à-dire que chaque case est la somme de la case juste au-dessus, et de la case juste au-dessus, juste à gauche. Ainsi, les premières lignes de P sont :

```
1  0  0  0  0  0...
1  1  0  0  0  0...
1  2  1  0  0  0...
1  3  3  1  0  0...
1  4  6  4  1  0...
...
```

1. Ecrire une procédure initialisant un triangle de Pascal P à 13 lignes puis affichant P.
Indication : en Python, pour mettre en oeuvre la formule « $P_{i,1} = 1$ pour tout $1 \leq i \leq n$ », on pourra créer une boucle pour i de 0 à $n-1$ et y mettre l'affectation `P[i][0] = 1`
2. Ecrire une procédure demandant de saisir une valeur n puis initialisant un triangle de Pascal P à n lignes.

Exercice 3 : addition, multiplication de matrices

1. Programmer en Python une fonction qui a les spécifications suivantes :
   - entrées : deux matrices A et B
   - sortie : la matrice qui résulte de l'addition $A + B$ si c'est possible, message d'erreur sinon
2. Programmer en Python une fonction qui a les spécifications suivantes :
   - entrées : deux matrices A et B
   - sortie : la matrice qui résulte de la multiplication $A \times B$ si c'est possible, message d'erreur sinon