TP3 : Structure conditionnée et programmation du produit matriciel

1 Pour s'échauffer

Exercice 1
Construction d'une matrice avec 2 boucles for
En utilisant deux boucles for imbriquées, écrire le code qui permet de construire la matrice :

$$A = \begin{pmatrix} 2 & 4 & 6 & 8 \\ 4 & 8 & 12 & 16 \\ 8 & 16 & 24 & 32 \end{pmatrix}$$

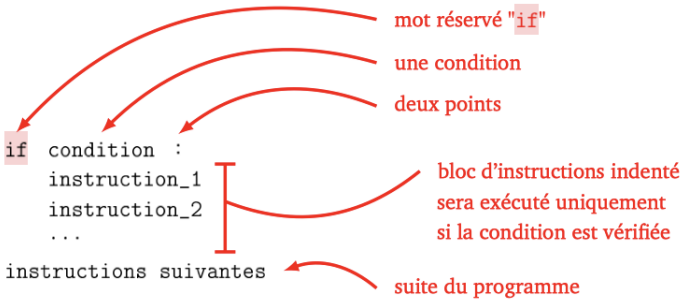
Remarque : d'un point de vue mathématiques $a_{i,j} = j * 2^i$

Exercice 2
Calcul de sommes

1. Écrire un script qui calcule la somme suivante : 2+4+6+8+10+.....+ 156.
   Remarque : On doit trouver que : 2+4+6+8+10+.....+ 156=6162.
2. Écrire une fonction nommée *sommeLigne* qui prend en entrée une matrice A et un numéro de ligne i et qui calcule la somme de tous les termes de la ligne i de la matrice A.
   Exemple : pour la matrice $A = \begin{pmatrix} 2 & 4 & 6 & 8 \\ 4 & 8 & 12 & 16 \\ 8 & 16 & 24 & 32 \end{pmatrix}$. La fonction *sommeLigne*(A,1) renvoie le résultat de 4+8+12+16=40.

2 L'instruction conditionnée IF

On a parfois besoin d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable. L'instruction conditionnée à la forme suivante :



condition est une expression dont le résultat peut être vrai ou faux, par exemple :

$\boxed{a == b}$ pour tester si a est égal b

$\boxed{a\ !\ = b}$ pour tester si a est différent de b

$\boxed{a >= b}$ pour tester si a est supérieur ou égal à b

$\boxed{a <= b}$ pour tester si a est inférieur ou égal à b

$\boxed{a > 1 \text{ and } a < 2}$ pour tester si a est supérieur à 1 **et** inférieur à 2

$\boxed{a > 1 \text{ or } a < -1}$ pour tester si a est supérieur à 1 **ou** inférieur à -1

On peut exécuter des instructions si la condition n'est pas remplie à l'aide du mot "else"

```
if condition :
    instruction
    instruction
    ...
else:
    instruction
    ...
instructions suivantes
```

bloc exécuté si la condition est vérifiée

bloc exécuté si la condition n'est pas vérifiée

Exemple : le programme suivant permet de vérifier si une matrice est de dimensions 2×2 (changer la matrice A pour tester pour d'autres matrices) :

```
A=array([[1,2,6],[8,3,4]])
n=shape(A)[0]
p=shape(A)[1]
if n==2 and p==3 :
    print('A est de taille 2x3')
else :
    print('A n'est pas de taille 2x3')
```

Exercice 3

Ecrire une fonction nommée *carre* qui prend pour variable d'entrée une matrice A et qui renvoie 1 si A est une matrice carrée et 0 sinon.

Exemple : Soit $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ alors carre(A) renvoie 1 et carre(B) renvoie 0.

On peut également imbriquer des séquences d'instructions conditionnées.

```
if   condition1 :
    instruction1
    ....
elif condition2 :
    instruction2
    ....
else :
    instruction3
    ....
Instructions suivantes
```

Bloc exécuté si la condition1 est vérifiée

Bloc exécuté si la condition2 est vérifiée

Bloc exécuté sinon

Exercice 4

1. Implémenter la fonction suivante : $h(x) = \begin{cases} 2x+1 & \text{si} \quad x < -1 \\ x^2 - 2 & \text{si} \quad x \in [-1;1] \\ -x & \text{si} \quad x > 1 \end{cases}$
2. Tester la fonction en calculant $h(-2)$, $h(0)$ et $h(4)$.

3 Le produit « terme à terme »

Nous avons vu dans le TP 1 que, si A et B sont des matrices de mêmes dimensions, alors $A * B$ renvoyait une matrice C (de mêmes dimensions que A et B) dont les coefficients étaient donnés par

$$c_{i,j} = a_{i,j} \times b_{i,j}$$

ATTENTION : Il ne faut pas confondre le produit terme à terme avec le produit matriciel classique (voir le section 4) !

Exercice 5 : Le produit "point"

1. À quelle(s) condition(s) sur A et B le produit $A * B$ est-il réalisable ?
2. Compléter le programme suivant pour obtenir une fonction *produitEtoile* qui prend en entrée deux matrices et qui renvoie le produit terme à terme.

```
def produitEtoile(A,B) :

    [NA, PA] = .........          # on récupère les dimensions de A
    [NB, PB] = .........          # on récupère les dimensions de B

    if ..............................   # on teste si le produit « terme à terme » est
                                       // réalisable
        return 'ERREUR de dimension des matrices, calcul impossible'
    else :
        C=zeros([NA,PA])
        for i in ..................          # on parcourt les lignes de C
            for j in ..................      # on parcourt les colonnes de C
                C[i,j] = ..................  # on calcule le coefficient C(i,j)
        return ......                        # retourne le résultat du calcul
```

3. Tester votre programme avec le produit $A * B$ pour les matrices suivantes :

$$A = \begin{pmatrix} 1 & 3 \\ 4 & 6 \\ 3 & 9 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 1 & 2 \\ 2 & 6 \\ 6 & 9 \end{pmatrix}$$

Attention : Test machine de 20 minutes à la fin du TP4

4 Le produit matriciel

Nous souhaitons écrire une fonction qui prend en entrée 2 matrices A et B et qui renvoie le produit matriciel AB. On rappelle que la formule générale qui donne le coefficient $C(i,j)$ de la matrice produit est :

$$C(i,j) = \sum_{k=1}^{p} A(i,k)B(k,j) \quad \text{pour} \quad i \in \{1,...,n\} \ \text{ et } \ j \in \{1,...,q\} \quad (\star)$$

avec $A(i,k)$ et $B(k,j)$ les coefficients des matrices A et B

Exercice 6 : programmation du produit matriciel

1. Tester le script suivant. Que calcule t-il ? (on pourra faire varier les valeurs de i et de j)

```
A=array([[1,2,3],[2,4,5]])
B=array([[-1,2],[-2,5],[3,0]])
i=1
j=1
n=3
coeff=0
for k in range(n) :
    coeff=coeff+A[i,k]*B[k,j]
print(coeff)
print(dot(A,B))
```

2. Recopier le code donné ci-dessous de la fonction *produitMatriciel* et compléter le en remplaçant les ..... pour qu'elle renvoie le produit matriciel de A par B :

```
def produitMatriciel(A,B) :
    [NA, PA] = .....                          # on récupère les dimensions de A
    [NB, PB] = .....                          # on récupère les dimensions de B
    if ..... :                                # on teste si le produit est réalisable
        print('Erreur de dimension : calcul produit matriciel impossible')
    else :
        C=zeros([NA,PB])
        for i in .....                        # on parcourt les lignes de C
            for j in .....                    # on parcourt les colonnes de C
                for k in .....                # on calcule la somme de la formule (⋆)
                    C[i,j] = .....
    return C
```

3. Tester votre programme avec les produits AB, BA pour les matrices suivantes (comparer votre résultat avec le produit matriciel préprogrammé en Python) :

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 1 & 2 \\ 2 & 6 \\ 6 & 9 \end{pmatrix}$$