TP 5 : Les matrices

I] Matrices (tableaux) :

Pour créer des matrices, il faut faire appel à la bibliothèque numpy de Python.
Il y a 2 façons de faire appel à numpy :
from numpy import* ou **import numpy as np**

La deuxième façon est plus lourde dans la pratique car il faut indiquer dans les opérations que l'on va taper l'expression **np.** C'est ce que l'on appelle un alias.

Exemples :
Taper les scripts suivants :

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])     # On crée la matrice (1 2 3 / 4 5 6) ( array )
print(a)
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

```
import numpy as np
a=array([[1,2,3],[4,5,6]])     # La matrice (1 2 3 / 4 5 6) n'est pas créée.
print(a)                         Python ne comprend pas l'instruction.
                                 Il manque l'alias np.
```

```
from numpy import*
a=array([[1,2,3],[4,5,6]])     # On crée la matrice (1 2 3 / 4 5 6)
print(a)
```

J'utilise plutôt la 1ère instruction `from numpy import*` mais la deuxième est à connaitre, notamment pour les concours ! Habituez-vous à utiliser les deux notations.

1) Nombre d'éléments d'une matrice :

Pour savoir combien d'éléments contient une matrice on utilise l'instruction **size (ou np.size) .**

Exemple : Taper le script suivant et noter ce qui se passe.

```
from numpy import*
a=array([[1,2],[3,4]])
print(a)
print(size(a))
```

2) Dimensions d'une matrice :

Pour connaitre le nombre de lignes et le nombre de colonnes d'une matrice , on utilise l'instruction **shape (ou np.shape).**

Exemples : Taper les scripts suivants et noter ce qui se passe.

```
from numpy import*
a=array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
print(a)
print(shape(a))
```

```
from numpy import*
a=array([[1,2,3]])
print(a)
print(shape(a))
```

```
from numpy import*
a=array([[1],[2],[3]])
print(a)
print(shape(a))
```

3) Produit matriciel **:**

Rappel : Si on dispose d'une matrice A de taille (n, p) et d'une matrice B de taille (p, q) alors on peut multiplier ces deux matrices et obtenir une matrice AB de taille (n , q).

Pour effectuer ce produit, on utilise l'instruction **dot** (ou **np.dot**).

Exemples : Taper les scripts suivants et noter ce qui se passe.

```
from numpy import*
a=array([[1,2],[4,5]])
b=array([[8,9],[11,12]])
print(a)
print(b)
print(dot(a,b))
print(dot(b,a))
```

```
from numpy import*
a=array([[1,2,3],[4,5,6]])
b=array([[8,9],[11,12]])
print(a)
print(b)
print(dot(b,a))
print(dot(a,b))
```

Remarque : Selon les versions de Python, on peut aussi utiliser l'instruction @ pour effectuer le produit matriciel. Testez si votre version accepte cette écriture.

```
from numpy import*
a=array([[1,2],[4,5]])
b=array([[8,9],[11,12]])
print(a)
print(b)
print(a@b)
print(b@a)
```

4) Produit terme à terme **:**

Il existe une instruction sous Python qui permet de réaliser le produit de deux matrices A et B de taille (n, p). Ce produit n'a rien à voir avec le produit matriciel. Les termes de la matrice s'obtiennent en multipliant le terme a_{ij} et le terme b_{ij} .
On peut effectuer le produit A*B ou B*A qui seront égaux.

Pour effectuer ce produit , on utilise ***** (Cette instruction fonctionne sans l'alias np).

Exemple :

```
from numpy import*
a=array([[1,2,3],[4,5,6]])
b=array([[7,8,9],[10,11,12]])
print(a)
print(b)
print(a*b)
print(b*a)
```

5) Transposée **:**

Pour transposer une matrice on utilise l'instruction **.T** .

Exemple :

```
from numpy import*
a=array([[1,2,3],[4,5,6]])
b=array([[7,8,9],[10,11,12]])
print(a)
print(b)
print(a.T)
print(b.T)
```

6) Matrice identité **:**

La matrice identité de taille n×n s'obtient par l'instruction **eye(n)** **(ou np.eye(n)**).

Exemple :

```
from numpy import*
print(eye(3))
print(eye(5))
```

7) Matrice nulle **:**

La matrice de taille m×n ne contenant que des 0 s'obtient en tapant **zeros((m,n))** **(ou np.zeros((m,n))**).

Exemple :

```
from numpy import*
print(zeros((3,4)))
print(zeros((5)))
```

8) Matrice ne contenant que des 1 **:**

La matrice de taille m×n ne contenant que des 1 s'obtient en tapant **ones((m,n))** **(ou np.ones((m,n))**).

Exemple :

```
from numpy import*
print(ones((3,4)))
print(ones((5)))
```

9) Calcul du déterminant **:**

On peut calculer le déterminant d'une matrice carrée avec l'instruction **det( )** .
Il faut cependant chercher la bibliothèque numpy.linalg. (from numpy.linalg import*)

Exemple :

```
from numpy.linalg import*
a=array([[1,2],[4,5]])
b=array([[7,8,9],[10,11,12],[ 13,14,15]])
print(a)
print(b)
print(det(a))
print(det(b))
```

10) Calcul de l'inverse d'une matrice **:**

On peut calculer l'inverse d'une matrice carrée grâce à l'instruction **inv( )** .
Il faut cependant chercher la bibliothèque numpy.linalg. (from numpy.linalg import*)

Exemple :

```
from numpy.linalg import*
a=array([[1,2],[4,5]])
b=array([[7,8,9],[3,1,4],[ 1,7,8]])
print(a)
print(b)
print(inv(a))
print(inv(b))
```

11) Extraire une sous-matrice d'une matrice **:**

Exemples :

Taper les instructions suivantes et comprendre ce qui se passe.

```
>>> a = np.array([12, 25, 34, 56, 87])
>>> a[1:3]
array([25, 34])
```

```
>>> a[1:]
array([25, 34, 56, 87])
>>> a[:3]
array([12, 25, 34])
>>> a[:]
array([12, 25, 34, 56, 87])
```

```
>>> a = np.array([[1, 2, 3],
                  [4, 5, 6]])
>>> a[0,1]
2
>>> a[:,1:3]
array([[2, 3],
       [5, 6]])
>>> a[:,1]
array([2, 5])
>>> a[0,:]
array([1, 2, 3])
```

12) Résolution d'un système d'équations **:**

A vous de chercher sur le web !

II] Exercices :

Ex 1 :

Ecrire un script qui demande d'entrer une matrice de taille (n, p).

Ex 2 :

Ecrire une fonction **symetrique(A)** qui teste si la matrice est symétrique ou antisymétrique et renvoie un booléen.

Ex 3 :

a) Ecrire une fonction **mat1(n,p)** qui génère une matrice (n,p) dont les coefficients sont des réels aléatoires compris entre 0 et 1.

b) Ecrire une fonction **mat6(n,p)** qui génère une matrice (n,p) dont les coefficients sont des réels aléatoires compris entre 1 et 6 .

c) Ecrire une fonction **matpair(n,p)** qui génère une matrice (n,p) dont les coefficients sont des entiers aléatoires pairs.

Ex 4 :

Ecrire une fonction **diag(n)** qui retourne une matrice carrée de taille n avec sur la diagonale principale la valeur n, sur les diagonales au dessus et en dessous la valeur n – 1, et ainsi de suite.

Ex 5 :

Ecrire une fonction **diag2(n)** qui retourne une matrice carrée de taille n avec que des 1 sur les deux diagonales (pour former une croix) et que des 0 sinon.

Ex 6 :

On se donne $(n+1)$ couples de nombres $(x_i, y_i) \in \mathbb{R}^2$, avec $i \in [\![0, n]\!]$ et on cherche un polynôme $P \in \mathbb{R}_n[X]$ passant par tous les points (x_i, y_i). Autrement dit, on recherche $(n+1)$ coefficients $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ tels que le polynôme $P(X) = \sum_{k=0}^{n} a_k X^k$ vérifie pour tout $i \in [\![0, n]\!]$, $P(x_i) = y_i$.

1. Expliquer pourquoi cela revient à résoudre le système matricielle d'inconnues $(a_0, \ldots, a_n)$ suivant :

$$\begin{pmatrix} 1 & x_0^1 & \cdots & x_0^n \\ 1 & x_1^1 & \cdots & x_1^n \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_n^1 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

2. Plus particulièrement, on recherche un polynôme

$$P = a_0 + a_1 X + a_2 X^2 + a_3 X^3 \in \mathbb{R}_3[X]$$

passant par les points $E : (-2, -2)$, $F : (-1, 2)$, $G : (0, -4)$, $H : (1, 4)$. Poser le problème sous forme matricielle, *i.e.* déterminer une matrice X de taille 4×4 et un vecteur colonne Y de taille 4×1 tels que les coefficiens recherchés (a_0, a_1, a_2, a_3) forme un vecteur colonne A solution du système $XA = Y$.

3. Créer la matrice X et le vecteur colonne Y dans *Scilab* et résoudre le problème (on donnera l'écriture explicite du polynôme P passant par les points E, F, G et H).

III] Notes personnelles :