Comment créer une matrice en Python ?

I) Utilisation du module numpy.

Il existe un module additionnel à Python nommé numpy permettant de créer et d'effectuer des opérations sur les matrices. Ce module est librement téléchargeable sur internet à l'adresse suivante : http://numpy.scipy.org/

Les instructions décrites ci-dessous ne représentent qu'une partie des possibilités de ce module.
Pour utiliser ce module il faut en tête de votre programme saisir : from numpy import*

1. Créer des matrices.

a. Créer une matrice en saisissant une à une les valeurs de chacun des termes. L'instruction est "array".

array([(. . .,. . ., … ,. . .),(. . .,. . ., … ,. . .), … ,(. . .,. . ., … ,. . .)]

1ère ligne 2ème ligne dernière ligne

Le séparateur pour les différentes valeurs ou pour les différentes lignes est ",".

exemple : array([(1,2,3),(4,5,6)]) crée la matrice 2×3 : $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

résultat à l'affichage: [[1 2 3]
[4 5 6]]

b. Créer une matrice ne contenant que des 0, ou bien que des 1. Les instructions sont "zeros" et "ones".

Pour une matrice à une seule ligne :

zeros(*nombre de colonnes*) et ones(*nombre de colonnes*)

Pour les autres matrices :

zeros(*nombre de lignes*,*nombre de colonnes*) et ones(*nombre de lignes*,*nombre de colonnes*)

exemples. zeros(5) crée la matrice 1×5 : (0 0 0 0 0)

ones((2,3)) crée la matrice 2×3 : $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

c. Créer une matrice avec des valeurs séparées d'un pas régulier. L'instruction est "arange".

Pour les matrices avec les entiers consécutifs de 0 à *n*-1 :

les matrices à une seule ligne et contenant les entiers de 0 à *n*-1: arange(*n*)

les autres matrices *a*×*b* et contenant les entiers de 0 à *n*-1: arange(*n*).reshape(*a*,*b*)

Pour les matrices avec les réels débutants à *d*, finissant à *f* et séparés d'un pas *p*:

les matrices à une seule ligne : arange(*d*,*f*,*p*)

les autres matrices *a*×*b* : arange(*d*,*f*,*p*).reshape(*a*,*b*)

exemples : arange(4) crée la matrice 1×4 : (0 1 2 3)

arange(6).reshape(2×3) crée la matrice 2×3 : $\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix}$

arange(1,16,4.2) crée la matrice 1×4 : (1 5.2 9.4 13.6)

arange(1,16,4.2).reshape(2,2) crée la matrice 2×2 : $\begin{pmatrix} 1 & 5.2 \\ 9.4 & 13.6 \end{pmatrix}$

d. Créer une matrice *a*×*b* diagonale avec des 1. L'instruction est "eye" avec la syntaxe : eye(*a*,*b*).

exemples : eye(2,3) crée $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, eye(3,3) crée $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, eye(4,3) crée $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

2. Accéder aux termes d'une matrice.
Il faut retenir que les lignes et les colonnes sont numérotées à partir de 0.

a. Accéder à un terme d'une matrice A. Syntaxe : A[*numéro de ligne*, *numéro de colonne*]
exemple : Si A = $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ alors A[1,2] correspond au terme 6

b. Accéder à une ligne d'une matrice A. Syntaxe : A[*numéro de ligne*]
exemple : Si A = $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ alors A[1] correspond à la ligne n°1, résultat : [4 5 6]

c. Accéder à une colonne d'une matrice A. Syntaxe : A[:,*numéro de colonne*]
exemple : Si A = $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ alors A[:,1] correspond à la colonne n° 1, résultat : [2 5]

3. Opérations sur les matrices.

a. Effectuer une même opération sur tous les termes d'un même matrice. Avec A = $\begin{pmatrix} 1. & 2. & 3. \\ 4. & 5. & 6. \end{pmatrix}$.

Multiplier tous les termes :	2*A	donne $\begin{pmatrix} 2. & 4. & 6. \\ 8. & 10. & 12. \end{pmatrix}$
Diviser tous les termes :	A/2	donne $\begin{pmatrix} 0.5 & 1. & 1.5 \\ 2. & 2.5 & 3. \end{pmatrix}$
Additionner ou soustraire à tous les termes :	A-1	donne $\begin{pmatrix} 0. & 1. & 2. \\ 3. & 4. & 5. \end{pmatrix}$
Mettre tous les termes au carré :	A**2	donne $\begin{pmatrix} 1. & 4. & 9. \\ 16. & 25. & 36. \end{pmatrix}$
Calculer le sinus de tous les termes :	sin(A)	donne $\begin{pmatrix} 0.84147098 & 0.90929743 & 0.14112001 \\ -0.7568025 & -0.95892427 & -0.2794155 \end{pmatrix}$

b. Effectuer des opérations avec des matrices.
Avec A = $\begin{pmatrix} 1. & 2. & 3. \\ 4. & 5. & 6. \end{pmatrix}$, B = $\begin{pmatrix} 3. & 2. & 4. \\ 1. & 5. & 6. \end{pmatrix}$, C = $\begin{pmatrix} 1. & 3. \\ 2. & 4. \\ 2. & 5. \end{pmatrix}$ et D = $\begin{pmatrix} 3. & 2. \\ 1. & 2. \end{pmatrix}$.

Effectuer le produit, terme à terme, de deux matrices :	A*B	donne $\begin{pmatrix} 3. & 4. & 12. \\ 4. & 25. & 36. \end{pmatrix}$
Multiplier deux matrices :	dot(A,C)	donne $\begin{pmatrix} 11. & 26. \\ 26. & 62. \end{pmatrix}$
Additionner ou soustraire deux matrices :	A+B	donne $\begin{pmatrix} 4. & 4. & 7. \\ 5. & 10. & 12. \end{pmatrix}$
Inverser une matrice :	linalg.inv(D)	donne $\begin{pmatrix} 0.5 & -0.5 \\ -0.25 & 0.75 \end{pmatrix}$
Transposée d'une matrice :	D.transpose()	donne $\begin{pmatrix} 3. & 1. \\ 2. & 2. \end{pmatrix}$

Pour calculer une puissance d'une matrice, voici une fonction que l'on peut définir en tête de programme :

```
def puissance(mat,exp):
    m=mat
    for i in range(1,exp):
        mat=dot(mat,m)
    return mat
```

ainsi, puissance(D,2) donne $\begin{pmatrix} 11. & 10. \\ 5. & 6. \end{pmatrix}$

II) Créer une matrice avec les listes.

On peut créer une matrice avec deux boucles "for" imbriquées. On obtient alors des listes de listes.
Etant donné l'existence du module numpy, nous ne proposerons pas ici la programmation pour réaliser les différentes opérations sur ces matrices. Cela peut éventuellement faire l'objet d'exercices.

Exemple. On veut créer la matrice 3 lignes par 4 colonnes : $\begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

```
M = [[0 for j in range(0,4)] for i in range(0,3)]
M[0][0]=4
M[1][2]=8
print M
```

Résultat: [[4, 0, 0, 0], [0, 0, 8, 0], [0, 0, 0, 0]]

Commentaires :

- "for j in range(0,4)" crée les colonnes numérotées de 0 à 3, et " for i in range(0,3)" crée les lignes numérotées de 0 à 2.
- L'élément de la ligne n°1 et de la colonne n°3 est identifié par : M[1][3]

On peut alors aussi fabriquer une fonction pour créer des matrices d'une taille définie.
Exemple. Pour créer des matrices 2×3 :

```
def matrice2x3():
   return [[0 for j in range(0,3)] for i in range(0,2)]
A = matrice2x3()
print "A=", A
```

Résultat: A= [[0, 0, 0], [0, 0, 0]]

On peut alors aussi fabriquer une fonction pour créer des matrices d'une taille non définie.
Exemple. Pour créer des matrices i×j :

```
def matrice(i,j):
   return [[0 for q in range(0,j)] for p in range(0,i)]
A = matrice(3,4)
print "A=", A
```

Résultat: A= [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]