Python 7 : Les matrices

Dans ce cours tableaux et matrices sont synonymes.
Le mot vecteur est synonyme de matrice ligne ou colonne.
On peut définir et manipuler des matrices grâce à la bibliothèque numpy.
Dans toute la suite on suppose avoir importé la bibliothèque numpy avec la commande :

import numpy as np

1) Créer une matrice

On peut définir des matrices grâce à la bibliothèque numpy.

- A la main :
  np.array ([4,7,9]) : Renvoie un vecteur ligne à 3 colonne contenant (4,7,9)
  np.array ([[1,2], [4,5]]) : Renvoie la matrice $\begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$

- Matrices spécifiques :
  np.zeros(n) : Renvoie un vecteur ligne de taille n rempli de 0
  np.zeros ((n,p)) : Renvoie une matrice de taille n,p remplie de 0
  np.ones(n) : Renvoie un vecteur ligne de taille n rempli de 1
  np.ones((n,p)) : Renvoie une matrice de taille n,p remplie de 1 np.
  eye(n) : Renvoie la matrice identité de taille n,n

- Vecteur régulièrement espacé :
  np.linspace (a,b,n) : Renvoie un vecteur ligne de n valeurs régulièrement espacées entre a et b (donc le pas vaut (b-a)/(n-1))
  np.arange(6) : Renvoie la matrice (0,1,2,3,4,5)
  np.arange(2,8) : Renvoie la matrice (2,3,4,5,6,7)

  Exemple : np.linspace(0,1,11) renvoie : [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]

2) Repérer des coefficients

Comme pour les listes, on peut identifier des coefficients ou des lignes/colonnes :

Attention, comme pour les listes, l'indexation commence à 0.

A[i, j] Extrait le coefficient à la ligne i colonne j de A
A[: , j] Extrait la colonne j de A
A[i, :] Extrait la ligne i de A
A[i: j, :] Extrait les lignes i à j de A
A[[i, j], :] Extrait les lignes i et j de A

Par exemple, si A= $\begin{matrix} 1 & 9 \\ 2 & 3 \\ -1 & 0 \end{matrix}$, alors A[0,1] vaut 9, A[:,0] vaut $\begin{matrix} 1 \\ 2 \\ -1 \end{matrix}$, A[1,:] vaut (2 3)

3) Opérations sur les matrices

- On peut additionner deux matrices et les multiplier par un réel (comme les matrices du cours sur les matrices) :
  A+B renvoie la matrice A+B (et non la concaténation de A et B comme pour les listes)

np.sum(A) : Renvoie la somme des coefficients de la matrice A (donc renvoie un réel, type *float*)
np.mean(A) : Renvoie la moyenne des coefficients de la matrice A
np.cumsum(A) : Renvoie une matrice du même format que A remplie des sommes cumulées des coefficients de a
np.max(A) : Renvoie le plus grand des coefficients de la matrice A
np.min(A) : Renvoie le plus petit des coefficients de la matrice A
np.var(A) : Renvoie la variance des coefficients de A
np.std(A) : Renvoie l'écart type des coefficients de A
np.size(A) : Renvoie le nombre de coefficients de A (donc un entier, type *int*)
np.shape(A) : Renvoie un vecteur [n,p] avec le format de A [lignes, colonnes]

Exemple :

np.shape([0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]) renvoie [1,11]
np.cumsum([1,3,0,7] renvoie [1,4,4,11] (sommes cumulées de la première matrice)

- On peut également faire des produits matriciels ou des transposées :
  np.dot(A,B) renvoie la matrice AB (produit matriciel de A par B)
  np.transpose(A) renvoie la transposée de A

- Enfin on peut appliquer les fonctions usuelles (celles de la bibliothèque from math import*) aux matrices (cela veut dire qu'elles s'appliquent à chaque coefficient de la matrice).
  Par exemple si A=[1 4 e 10], log(A) vaut [0 ln(4) 1 ln(10)]

- Attention écrire A*B renvoie une nouvelle matrice dont les coefficients sont le produit de chaque coefficient de A par ceux de B (et non pas le produit matriciel).

4) Opérations de la bibliothèque linalg

Il existe une bibliothèque très utile accessible grâce à :
import numpy.linalg as al

Elle permet de faire des opérations :

al.inv(A) : Renvoie la matrice inverse de A (si elle existe)
al.matrix_rank(A) : Renvoie le rang de A (voir cours sur les applications linéaires)
al.matrix_power(A,n) : Renvoie la matrice A^n (si A est carrée bien sûr)
al.solve(A,Y) : Renvoie l'unique solution X du système AX=Y dans le cas où cette solution est unique, où A est une matrice et Y un vecteur colonne du bon format