Module Numpy

Numpy

Ce document est une introduction au module numpy. Il n'est en aucun cas exhaustif :
Dans toute la suite on supposera qu'on a effectué : `import numpy as np`
Numpy fournit le type array qui est similaire à une liste de listes (list) mais correspond à des tableaux (une, deux, trois dimensions ou plus).
La contrainte est que tous les éléments d'un tableau sont de même type mais pour une matrice m, il est possible d'accéder directement à l'élément de ligne i et de colonne j avec la syntaxe m[i, j] et de faire du slicing sur les deux coordonnées.

Les commandes à connaître

np.array prend en argument une liste (L) et renvoie un tableau. `T=np.array(L)`.
Exemple : `L=[[1,2,3],[4,5,6]];T=np.array(L)`

slicing Comme pour les listes, le slicing extrait des tableaux. M étant une matrice à deux dimensions, on écrira `M[debut:fin:pas,debut:fin:pas]`.
Rappel : l'élément d'indice fin n'est pas inclus

.shape Si M est une matrice
- `M.shape` renvoie une liste [l,c] contenant les dimensions de M.
- `M.shape=[l,c]` reconstruit la matrice M avec les dimensions l×c.

np.zeros crée une matrice nulle. `np.zeros(l,c)` crée la matrice nulle de taille l×c. Attention, à priori, il s'agit de flottants, mais on peut écrire `np.zeros((l,c),dtype=np.int)`
Remarque : d'autres fonctions analogues existent : np.ones (avec des 1) et np.eye pour la matrice unité.

A retenir

Pour travailler sur une matrice M, on a souvent besoin de parcourir la matrice et d'avoir accès à chacune de ses cellules.

```
[L,C]=M.shape #  nombre de lignes (L), nombre de colonnes (C)
for i in range(L):
    for j in range(C):
        ... instruction pour M[i,j]
```

Comment créer un tableau ?

Voici différentes méthodes pour obtenir la matrice $T = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$.

- On rentre l'ensemble des valeurs, puis on redimensionne le tableau
  `T=np.array([1,2,3,4,5,6,7,8,9,10,11,12]);T=np.resize(T,[3,4])`
- On crée une matrice nulle de taille donnée puis on parcourt le tableau en donnant à chaque cellule la valeur souhaitée

```
T=np.zeros([3,4],dtype=np.int)
k=1
for i in range(3):
   for j in range(4):
       T[i,j]=k
       k=k+1
```

- On crée une liste de listes (possibilité de la créer en compréhension - délicat), puis on utilise array.
  `LL=[[i+j+1 for j in range(4)]for i in range(3)];T=np.array(LL,(4,3))`

Opérations sur les tableaux avec numpy

Les opérations usuelles (addition, soustraction, multiplication, division, puissance (A**n)) s'appliquent aux tableaux **en opérant coefficient par coefficient**.
Chaque fonction mathématique usuelle possède une fonction numpy qui lui est homonyme et qui calcule la même chose sur un tableau **en opérant coefficient par coefficient**.
Exemple `x=np.array([3,4])` puis `np.exp(x)` donne $[e^3, e^4]$
Néanmoins, on peut utiliser les tableaux comme des matrices et numpy fournit les fonctions nécessaires à l'aide du sous module linalg.
- Le produit matriciel A.B s'obtient avec `np.dot(A,B)`
- Le produit scalaire A.B (si A et B n'ont qu'une dimension) s'obtient avec `np.vdot(A,B)`
- L'inverse d'une matrice A s'obtient avec `np.linalg.inv (A)`
- La puissance n$^{\text{è}}$ de la matrice A avec `np.linalg.matrix_power(A,n)`
- `np.linalg.eig` permet d'obtenir les valeurs propres et les vecteurs propres d'une matrice (voir Maths 2$^{\text{è}}$ année).