Fiche : Tableaux en langage Python (le module `numpy`)

Un **tableau**, pour Python, c'est une variable de type `ndarray` (*array* = *tableau*). C'est un type spécifique à la librairie `numpy`, et pour cette raison on doit importer `numpy` pour l'utiliser. On commencera notre script par :

```
import numpy as np
```

Créer un tableau à partir d'une liste

`T = np.array(L)` ..... convertit la liste L en un tableau à une ligne (type array)

`T[i]` ..... valeur dans le tableau à la position i

`T[i:j]` ..... sous-tableau extrait de T (éléments en position i à j exclu)

`T[i]=10` ..... affecte la valeur 10 à la position i dans le tableau

Remarque : Comme pour les listes, la numérotation des indices commence à 0 et non pas à 1 !

Créer un tableau à deux entrées à partir d'une liste de lignes

`T = np.array([[1,2],[3,4],[5,6]])` ..... crée le tableau à 3 lignes (les 3 listes) et 2 colonnes

`np.shape(T)` ..... donne les dimensions d'un tableau

`np.size(T)` ..... donne le nombre total d'éléments du tableau

`T[i,j]` ..... valeur dans le tableau à la ligne i et colonne j

`T[i,:]` ..... sous-tableau extrait de T (toute la ligne i)

`T[i,j]=10` ..... affecte la valeur 10 à la position i,j dans le tableau

Remarque : Mathématiquement, un tableau à deux entrées est une matrice !

Créer des tableaux particuliers

`np.zeros((n,p))` ..... crée un tableau de taille $n \times p$ dont toutes les entrées sont nulles

`np.ones((n,p))` ..... crée un tableau de taille $n \times p$ dont toutes les entrées valent 1

`np.eye(n)` ..... créé la matrice identité de taille n

`np.random.random((n,p))` crée un tableau de taille $n \times p$ dont toutes les entrées sont des nombres flottants aléatoires entre 0 et 1

`np.random.randint(a,b,(n,p))` ... crée un tableau de taille $n \times p$ dont toutes les entrées sont des nombres entiers aléatoires entre a inclus et b exclu

`np.arange(a,b,p)` crée un tableau à une ligne comportant des valeurs entre a et b exclu choisies avec un pas de p

`np.linspace(a,b,n)` ..... crée un tableau à une ligne comportant n valeurs entre a et b

Opérations sur les tableaux de même taille

`A+B` ..... tableau des sommes des coefficients de A avec les coefficients correspondants de B

`A+3` ..... tous les coefficients de A plus 3

`A*B` ..... tableau des produits des coefficients de A avec les coefficients correspondants de B

`5*A` ..... tous les coefficients de A fois 5

`A**2` ..... tous les coefficients de A au carré

`A**B` ..... tableau des coefficients de A à la puissance les coefficients correspondants de B

`A/B` ..... tableau des coefficients de A divisés par les coefficients correspondants de B

`A==B` ..... tableau de booléens disant si chaque coefficient de A est égal au coefficient de B correspondant

Remarque : En clair, toutes les opérations que vous connaissez sur les nombres réels sont autorisées sur les tableaux. Elles sont, le plus simplement du monde, effectuées **coefficient par coefficient**. Notez la possibilité de faire aussi des opérations booléennes coefficient par coefficient.

Autres fonctions intéressantes

`np.dot(A,B)` ..... effectue le produit matriciel de deux matrices A et B de tailles compatibles

`np.sum(M)` ..... renvoie la somme de tous les coefficients du tableau M

`np.max(M),np.min(M)` ..... donne le plus grand (et le plus petit) coefficient dans le tableau M

`np.transpose(M)` ..... renvoie la matrice transposée de M

Il existe un « sous-module » du module `numpy` appelé `numpy.linalg` comportant des fonctions qui peuvent être utiles si les tableaux de nombres considérés correspondent à des matrices.

Fonctions du module numpy.linalg

`import numpy.linalg as la` ..... pour importer le module « algèbre linéaire »

`la.inv(M)` ..... renvoie l'inverse de la matrice M si M est inversible (erreur Singular matrix sinon)

`la.matrix_rank(M)` ..... renvoie le rang de la matrice M