Notes sur NumPy

Overview de NumPy

- calcul scientifique
- fonctionnalités et caractéristiques clefs :
  - manipulation de vecteurs, vectorisation, tableaux multidimensionnels,
  - algèbre linéaire,
  - mais aussi : stats, nombres pseudo-randomes, etc.

Objet *ndarray*

- comme un tableau classique, donc
  - taille fixe,
  - tous les éléments de même type :

```
>>> a = numpy.array([5, 1.0])
>>> a
array([ 5.,  1.])
>>> a.dtype
dtype('float64')
```

  - peut être multidimensionel.
- Accès aux éléments d'un tableau multidimensionnel :

```
## créer une matrice 2x3
>>> a = numpy.array([[1,2,3],[4,5,6]])
## accéder à un élément
>>> a[0,2]
3
## accéder à une colonne
>>> a[:,1]
array([2,5])
## accéder à une ligne
>>> a[0,] ## ou a[0]
array([1,2,3])
```

Vectorisation

- Opérations sur des vecteurs : élément par élément

```
## a*b = c
## t.q. c[i] = a[i]*b[i] pour tout i de 0 à n-1, n = len(a)
>>> numpy.array([1,2,3])*numpy.array([5,4,1])
array([5,8,3])
```

- ##
```
>>> x = np.array([[10, 4], [5, 8]])
>>> x > 5
array([[ True, False],
       [False,  True]])
```
- ## conversion de type pour tous les éléments
```
>>> a = numpy.array([True, False, True])
>>> a = a.astype(dtype = float)
>>> a
array([1., 0., 1.])
```

- « broadcasting » : si une opération est appliquée à deux ndarrays dont les dimensions ne sont pas compatibles, le ndarray à la dimensionnalité inférieure est répété («cloné ») autant de fois que ses dimensions deviennent égales à celles du deuxième
  - ## x + 5 : ajouter 5 à tous les éléments de x
```
>>> numpy.array([1,2,3]) + 5
[6,7,8]
```
  - >>> numpy.array([1,2,3]) == 2
```
array([False, True, False])
```

  NOTE : « broadcasting » marche comme « vector recycling » en R, mais avec les différences suivantes :

  - par lignes au lieu de colonnes ;
  - ne marche pas avec des ndarrays dont le nombre de dimensions sont pareils, mais les tailles des dimensions sont différentes
    - ## une seule dimension chacun => érreur
```
>>> numpy.array([1,2]) + numpy.array([1,2,3,4])
```
    - ## un vecteur + une matrice => pas d’erreur, broadcasting
```
>>> numpy.array([1,2]) + numpy.array([[1,2],[3,4]])
array([[2, 4], [4, 6]])
```

- Opérations sur les vecteurs en entier
  - >>> numpy.sum(numpy.array([1,2,3]))
```
6
>>> numpy.sum(numpy.array([[1,2],[3,4]]))
10
```
  - >>> a = numpy.array([[1, 4], [5, 8], [9,0]])
```
>>> a
array([[1, 4],
       [5, 8],
       [9, 0]])
## sommes de colonnes (à travers la dimension 0)
>>> a.sum(axis=0)
array([15,12])
## sommes de lignes (à travers la dimension 1)
>>> a.sum(axis=1)
array([5, 13, 9])
```
- Conversions implicites

```
>>> numpy.sum(numpy.array([False, True, True]))
2
```

- Indexation conditionnelle

```
>>> a = np.array([[10,  4],[ 5,  8]])
>>> a[a > 7]
array([10,  8])
## le résultat peut être utilisé en tant que l-value aussi
>>> a[a > 7] = 1
>>> a
array([[1, 4],
       [5, 1]])
```

Opérations sur des matrices

```
## produit de matrices
>>> a = numpy.array([1,2]) ; b = numpy.array([[3,4],[5,6]])
>>> numpy.matmul(a,b)
array([13, 16])
```

```
## transposition
b = numpy.transpose(a)
## ou
b = a.T
```