Manip 1: Pratique NUMPY de base

- Création d'un tableau de base

In [1]:
```
import numpy as np
tab = np.array([1,2,3,4])
print(tab)
print(type(tab))
```

```
[1 2 3 4]
<class 'numpy.ndarray'>
```

- Creation d'un tableau avec arange

In [4]:
```
tab = np.arange(1,5)
print(tab)
```

```
[1 2 3 4]
```

Propriétés du tableau

- Dimension

In [5]:
```
print(tab.ndim)
```

```
1
```

- shape

In [9]:
```
print(tab.shape)
print(tab.shape[0])
```

```
(4,)
4
```

- type de données

In [10]:
```
print(tab.dtype)
```

```
int32
```

Matrice

In [12]:
```
mat = np.array([[1,2,3], [4,5,6]])
#dimension
print(mat.ndim)
```

```
2
```

In [13]:
```
#shape
print(mat.shape)
print(mat.shape[0])
print(mat.shape[1])
```

```
(2, 3)
2
3
```

Generation de matrices

In [15]:
```
mat_uns = np.ones((3,3))
print(mat_uns)
```

```
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
```

In [16]:
```
mat_zeros = np.zeros((4,4))
print(mat_zeros)
```

```
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
```

In [17]:
```
mat_diag = np.eye(5)
print(mat_diag)
```

```
[[ 1.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]]
```

In [19]:
```
mat_diag = np.eye(2,3)
print(mat_diag)
```

```
[[ 1.  0.  0.]
 [ 0.  1.  0.]]
```

- Matrice avec valeurs aleatoires

In [20]:
```
mat_ale = np.random.random((3,3))
print(mat_ale)
```

```
[[ 0.14995365  0.91051755  0.93605851]
 [ 0.04623208  0.2004003   0.57312611]
 [ 0.38750869  0.91801824  0.20195911]]
```

Indexation

In [23]:
```
tab = np.arange(10)
print(tab[2])
print(tab[9])
```

```
2
9
```

In [24]:
```
mat = np.array([[1,2,3], [4,5,6]])
print(mat[1,2])
```

```
6
```

- Slicing sur array-matrice

In [26]:
```
mat = np.array([[1,2,3,4], [5,6,7,8],[9,10,11,12]])
print(mat[:2, 1:3])
```

```
[[2 3]
 [6 7]]
```

Opérations mathématiques

In [32]:
```
A = np.array([[1,2], [3,4]])
B = np.array([[5,6], [7,8]])

# Addition
print(np.add(A,B))
#Soustraction
print("=" * 45)
print(np.subtract(A,B))
#Multiplication element x element
print("=" * 45)
print(np.multiply(A,B))
print("=" * 45)
print(np.matmul(A,B))
```

```
[[ 6  8]
 [10 12]]
=============================================
[[-4 -4]
 [-4 -4]]
=============================================
[[ 5 12]
 [21 32]]
=============================================
[[19 22]
 [43 50]]
```

Utilisation des axes 0 et 1

- Somme sur axes

In [33]:
```
print(np.sum(A))
```

```
10
```

In [34]:
```
print(np.sum(A, axis = 0))# par rapport aux colonnes
```

```
[4 6]
```

In [35]:
```
print(np.sum(A, axis = 1)) # par rapport aux lignes
```

```
[3 7]
```

In [37]:
```
print(np.min(A, axis = 0))
```

```
[1 2]
```

In [38]:
```
print(np.max(A, axis = 0))
```

```
[3 4]
```

Statistiques

In [45]:
```
x = np.array([23, 40, 16, 74, 55, 87])
print(np.mean(x))
```

```
49.1666666667
```

In [46]:
```
print(np.median(x))
```

```
47.5
```

In [47]:
```
print(np.std(x))
```

```
25.6607135953
```

In [ ]:
```

```