Fiche mémo `python/numpy/matplotlib` – MT09
Fiche autorisée au médian et au final MT09

Système

```
%pwd # print working directory
%mkdir toto # make directory
%cd toto # change directory
%ls # list
# etc ...
```

Arrays 1d

```
import numpy as np
x = np.array([1,2,3], dtype=np.float64)
y = np.array([3,2,1])
z = x*y # produit elmt par elmt
x.size # taille
x.shape # (3,)
x.ndim # = 1
np.dot(x,y) # produit scalaire x^T y (float)
np.outer(x, y) # = x y^T (3x3)
I = np.arange(1,11) # [1,2,...,10]
J = np.arange(10,0,-1) # [10,9,...,1]
xd = np.linspace(0, 10, 6) # [0 2 4 6 8 10]
# Différences d'ensembles
I = np.arange(0,5) # [0,1,2,3,4]
J = np.array([2,3])
K = np.setdiff1d(I,J) # [0,1,4]
```

Boucles et conditions

```
I = np.arange(1,11)
for i in I:
    print("i = ", i, "\n")
print("Done.\n")
k = 0
while (k<10):
    k +=1
    print(k)
print("Done.\n")
```

Matrices

```
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
A.size # taille, = 9
A.shape # (3,3)
A.T # A transposée
A.ndim # = 2
B = np.ones( (3,3) )
C = np.zeros( (3,3) )
D = np.eye(3) # = I_3
BC = B@C # produit matrice-matrice

x = np.array([[1,2,3]]).T # matrice colonne
x.shape # = (3,1)
y = np.array([[1,2,3]]) # matrice ligne
y.shape # = (1,3)
y@x # matrice 1x1 (array2d)

A = np.array([[1,2,3],[4,5,6],[7,8,9]])
A[0,:] # 1ere ligne de A (array1d)
A[0:1,:] # 1e ligne de A (mat ligne, array2d)
A[:,0] # 1e colonne de A (array1d)
A[:,0:1] # 1e colonne de A (mat colonne, array2d)
```

Produits matrice-vecteur array2d-array1d

```
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
x = np.array([1,2,3])
A@x # = Ax (array1d)
x@A # x^T A (array1d)
x@A@x # x^T A x (float)
```

Module algèbre linéaire

```
import numpy.linalg as la
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = np.array([1,2,3])
x = la.solve(A,b) # résout Ax=b
print("Ax-b = ", A@x - b)
la.det(A) # déterminant
la.cond(A,1) # condt en norme 1
la.cond(A,2) # condt en norme 2
la.cond(A,np.inf) # en norme infinie
D, V = la.eig(A) # valeurs et vecteurs propres
print(D, '\n\n', V)
```

Fonctions

```
def maFonction(x,y):
    """Documentation ..."""
    z = x*np.sin(np.pi * y) # produit elmt par elmt
    return z
```

Exceptions

```
cond = True
if (cond):
    raise SystemExit("Arret : condition = ", cond)
```

Tracé de courbes

```
import matplotlib.pyplot as plt
a, b = 0, 5
x = np.linspace(a, b, 200)
plt.plot(x, x**2, '.-b', linewidth=2, markersize=2)
plt.plot(x, x**3, 'x-r', linewidth=2, markersize=2)
plt.legend(['x^2', 'x^3'])
plt.grid()
# voir aussi semilogy, loglog pour echelle log
plt.savefig('mafigure.png')
```