TP1

September 12, 2018

1 TP 1 : Rappel sur les librairies Numpy et matplotlib

1.1 1 - Introduction

L'objectif de ce TP est de faire des rappels sur les librairies numpy et matplotlib. Il sera à réaliser en python 3. Les librairies utilisées sont installées sur les machines de l'université, vous pouvez néanmoins les installer sur vos propres machines à l'aide de l'utilitaire pip présent par défaut avec python.

N hésite pas à regarder régulièrement la documentation de ces librairies, des exemples d'utilisation accompagnent généralement l'explication de chaque fonction.

- Python 3: https://docs.python.org/3/
- Numpy: https://docs.scipy.org/doc/numpy/reference/
- Scipy: https://docs.scipy.org/doc/scipy/reference/
- Matplotilb: https://matplotlib.org/contents.html

À part si cela est précisé, vous ne devez pas utiliser directement de boucle (`for`,`while`) ou de branchement conditionnel (`if`) durant ce TP..

```
In [1]: import numpy as np
        import scipy as sc
        import matplotlib.pyplot as plt
```

1.2 2 - Vecteurs

1.2.1 2.1 Création de vecteurs

Créez un vecteur de taille 5 contenant que des zéros. Affichez-le et affichez sa taille.

```
In [2]:
```

```
Affiche du vecteur:  [0. 0. 0. 0. 0.]
Dimension du vecteur: (5,) . Nombre d élément du vecteur: 5
```

Créez un vecteur contenant les nombres compris entre 3 et 12 (exclus) avec un pas de 0.5

```
In [3]:
```

```
Out[3]: array([ 3. ,  3.5,  4. ,  4.5,  5. ,  5.5,  6. ,  6.5,  7. ,  7.5,  8. ,
                8.5,  9. ,  9.5, 10. , 10.5, 11. , 11.5])
```

Créez un vecteur contenant les carrés des entiers compris entre -5 et 5 (exclus).

```
In [4]:
```

```
Out[4]: array([25, 16,  9,  4,  1,  0,  1,  4,  9, 16])
```

Créez un vecteur contenant toutes les puissances de deux d'entiers compris entre 1 et 65536 ($= 2^{16}$) inclus. Le résultat attendu ressemble à : <p style="text-align:center";> [1, 2, 4, 8, 16,. . . , 32768, 65536]

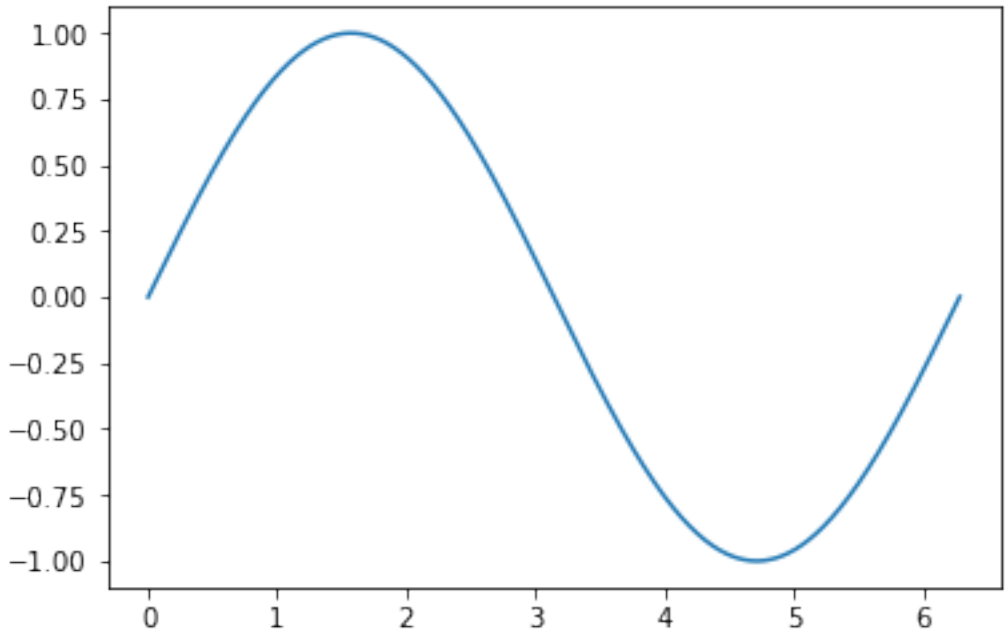
```
In [5]:
```

```
Out[5]: array([    1,     2,     4,     8,    16,    32,    64,   128,   256,
                 512,  1024,  2048,  4096,  8192, 16384, 32768, 65536])
```

1.2.2 2.2 Visualisation de fonctions

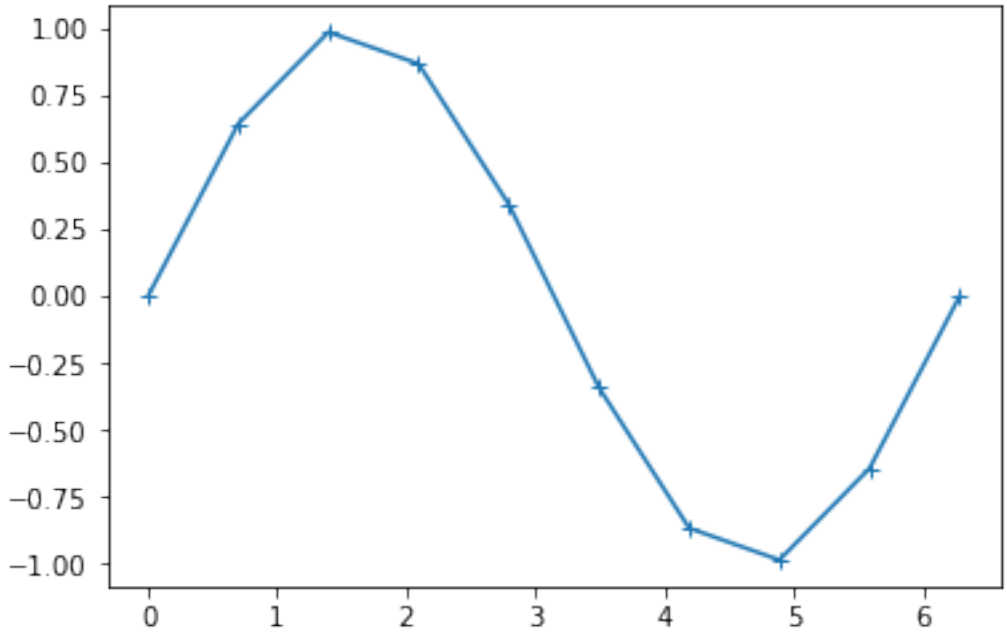
Tracez avec matplotlib la fonction sinus entre 0 et 2*np.pi en calculant un point tous les 1e-3. Vérifiez que les valeurs en abscisse et en ordonnée sont correctes.

```
In [6]:
```



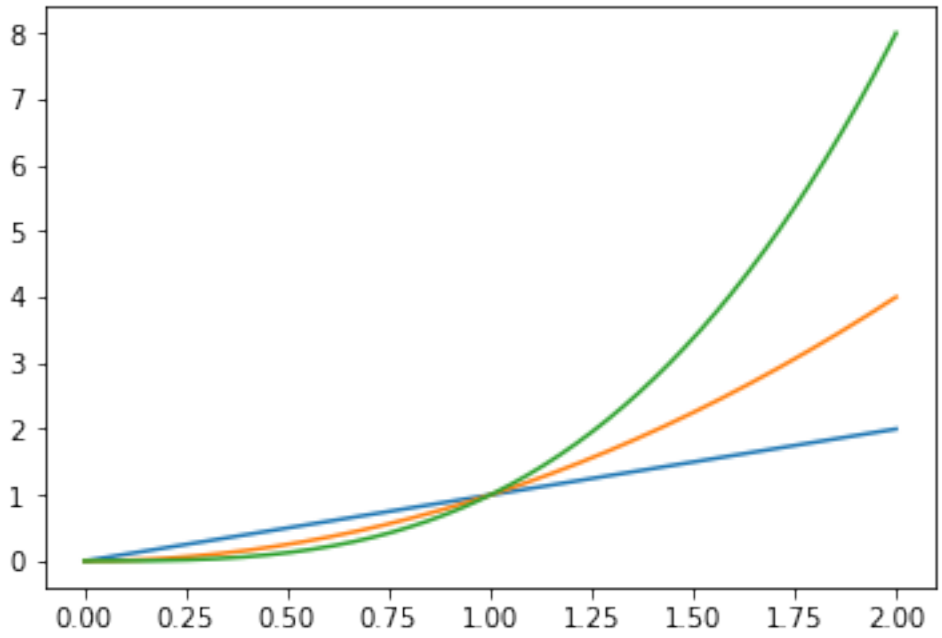
Faites l'affichage de la précédente question, mais avec uniquement 10 points. Vous pouvez visualiser ces points en ajoutant '+-' aux arguments de la fonction plot.

In [7]:



Tracez sur la même figure les fonctions x,x^2,x^3 entre 0 et 2. Choisissez un écart entre chaque point suffisamment petit pour que les courbes paraissent lisses.

In [8]:



1.3 3 - Comparaison python classique et fonction numpy

Le code suivant permet de mesurer le tempse d exéuction de plusieurs instructions.

```
In [9]: %%time
        1+1

CPU times: user 4 µs, sys: 1e+03 ns, total: 5 µs
Wall time: 9.3 µs


Out[9]: 2
```

Calculez la somme des entiers entre 0 et 100 millions avec une **boucle for** en python sans utiliser de librairie. Vous mesurez le temps d'exéuction de votre code.

```
In [10]:

5000000050000000
CPU times: user 19 s, sys: 71.7 ms, total: 19.1 s
Wall time: 19.1 s
```

Créez un vecteur contenant tous les entiers entre 0 et 100 millions puis calculez la somme des valeurs de ce vecteur en utilisant numpy. **Aucun for ou while ne devra être utilisé.**

Comparez les vitesses d'exécution des deux scripts.

```
In [11]:

5000000050000000
CPU times: user 316 ms, sys: 275 ms, total: 591 ms
Wall time: 602 ms
```

Question bonus: Calculez cette somme en utilisant vos connaissances sur la somme des suites arithmétiques. Retrouvez-vous le même résultat ?

```
In [12]:

5000000050000000
CPU times: user 166 µs, sys: 49 µs, total: 215 µs
Wall time: 181 µs
```

1.4 4 - Les matrices

1.4.1 4.1 - Création de matrices

Créez une matrice de taille (12,4) contenant que des zéros. Affichez la et affichez sa taille.

```
In [13]:
```

```
Matrice: [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Dimension de la matrice: (12, 4)
```

Créez une matrice identité de dimension (16,16). Affichez la et affichez sa taille.

In [14]:

```
Matrice: [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
Dimension de la matrice: (16, 16)
```

Créez et affichez la matrice $\begin{bmatrix} 1 & 31 \\ 51 & 12 \end{bmatrix}$

In [15]:

```
[[ 1 31]
 [51 12]]
```

Affichez uniquement la ligne 0 de la matrice précédente. Affichez la colonne 1.

In [16]:

```
Ligne 0: [ 1 31]
Colonne 1: [31 12]
```

1.4.2 4.2 Lecture et manipulation d'une image

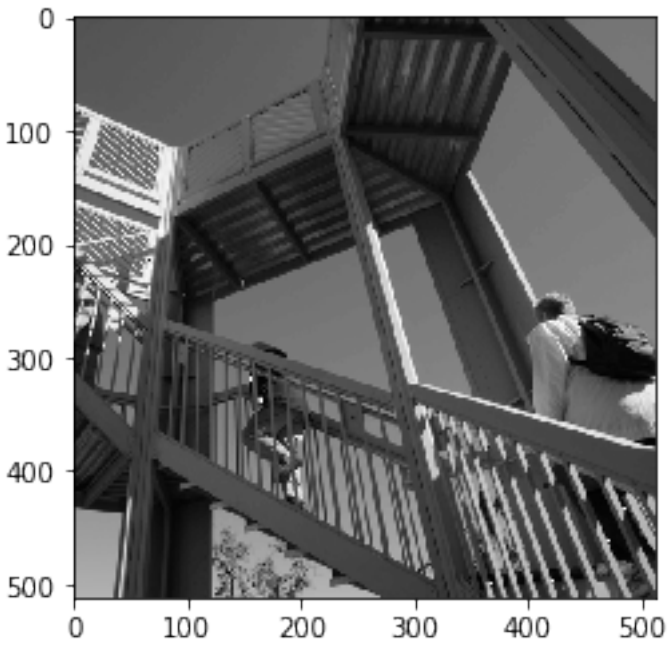
L'objectif de cet exercice et de lire une image, de lui ajouter un bord noir et de l'afficher. Nous allons procéder pour cela en différentes étapes, comme expliqué ci-dessous.

Il existe dans la librairie scipy deux images de test que vous pouvez utiliser. L'une d'elles peut être récupéré par le code suivant:

```
In [17]: import scipy as sc
         import scipy.misc
         im = sc.misc.ascent()
```

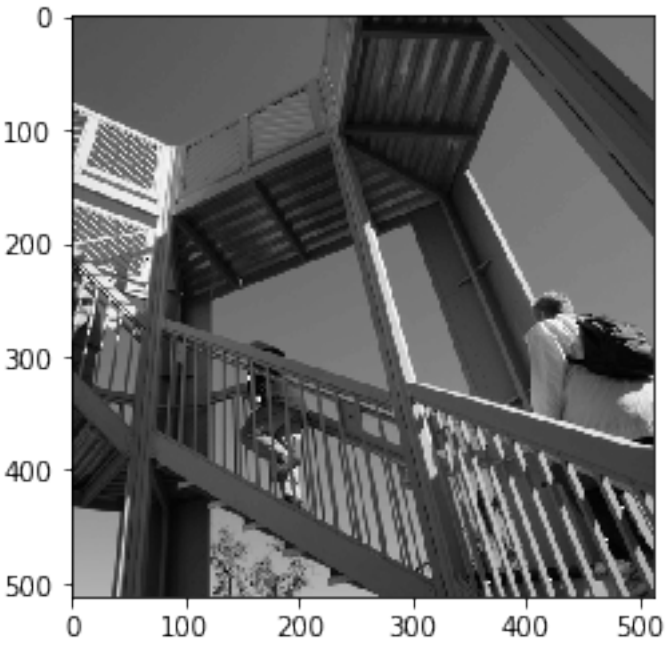
À l'aide de la fonction plt.imshow de matplotlib affichez l'image précédente. L'option `cmap='gray'` permet de préciser que l'image est en noir et blanc. Les options `vmin = 0` et `vmax = 255` permet de préciser que les pixels sont définis par des valeurs allant de 0 (noir) à 255 (blanc). N'oubliez pour la fonction plt.show() après plt.imshow .

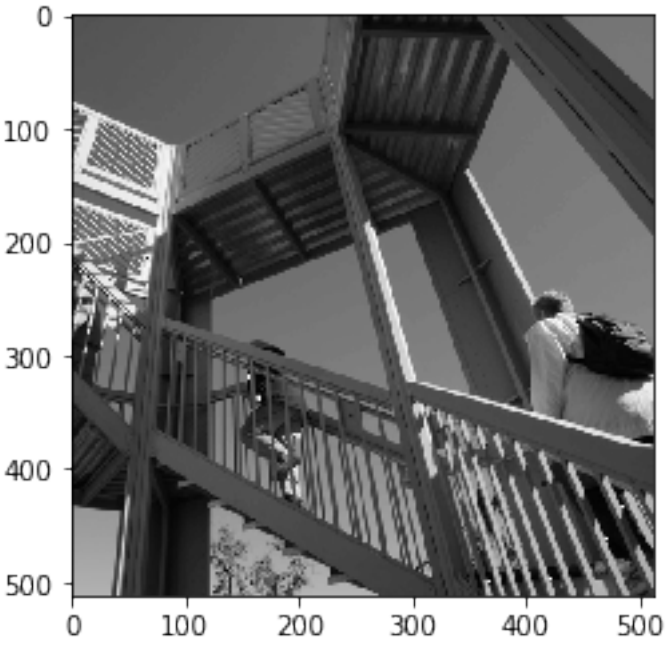
```
In [18]:
```



La fonction plt.show permet d'exécuter toutes les instructions d'affichage demandées. Il est parfois nécessaire d'afficher plusieurs fenêtres d'affichage. Pour cela vous pouvez appeler la fonction `plt.figure(n)` avec n le numéro de la fenêtre à gérer. Lorsque plt.show sera appelé, toutes les fenêtres seront affichées. Affichez dans deux fenêtres en parallèle l'image précédente. **Vous utiliserez qu'une seule fois la fonction plt.show().**

In [19]:





Nous allons ajouter un cadre noir à notre image. Pour cela, construisez un tableau numpy contenant uniquement des zéros dont les dimensions sont celles de l'image augmentées de 100 en

ligne et colonne. Recopiez ensuite l'image à partir de la ligne 50, colonne 50 (sans utiliser de for ou de while).

In [20]: