# Informatique

# 10

# Matplotlib

***Cours***

# Tracés de courbes

## 1.I. Introduction

Nous allons apprendre à tracer des courbes. Il ne me semble pas utile de détailler l'utilité de ce paragraphe…

Il est possible d'aller loin dans les possibilités qu'offrent les options de python. Nous ne développerons ici que les bases utiles le plus généralement. L'utilisateur qui souhaitera aller plus loin pourra trouver tout ce dont il a besoin sur internet en tapant simplement ce qu'il souhaite effectuer.

Il existe différentes méthodes pour tracer des courbes, je vous en propose une que j'ai adoptée et qui fonctionne bien.

Sachez que la création de courbes sous Python nécessite d'avoir deux listes, l'une pour les abscisses, l'autre pour les ordonnées. On n'écrit donc pas comme dans les calculatrices graphiques la fonction directement. C'est là une différence assez importante qu'il faut avoir comprise.

## 1.II. Import de la librairie

Pour tracer des courbes, il est nécessaire d'importer la librairie associée en écrivant :

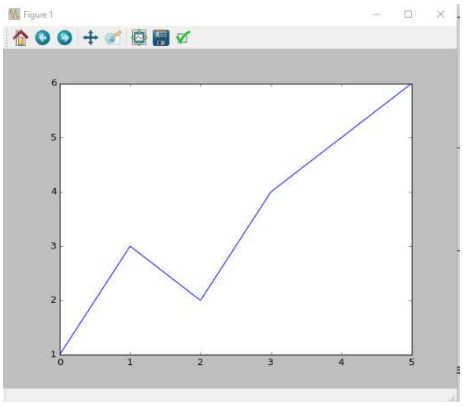
```
import matplotlib.pyplot as plt
```

Le fait de rajouter « `as plt` » permet par la suite de ne pas écrire pyplot mais juste plt. En fait, on change le nom de la fonction `pyplot` en `plt`.

## 1.III. Un exemple basique

```
X = [0,1,2,3,4,5]
Y = [1,3,2,4,5,6]
plt.plot(X,Y)
plt.show()
```

Le résultat est le suivant :



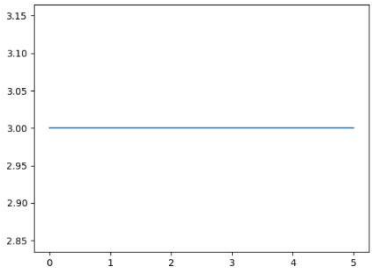
« `plt.plot(X,Y)` » met en mémoire un graphique affiché grace à la commande « `plt.show()` ».

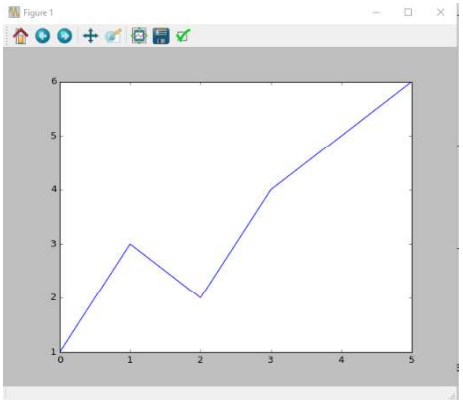
Sans fermer la figure qui vient de s'ouvrir, modifiez votre liste Y pour celle-ci et exécutez le code :

```
Y = [3,3,3,3,3,3]
```

Vous remarquerez que la figure ne s'est pas mise à jour ☹

Fermez la fenêtre, et réexécutez votre code :





Oui, la mise à jour de la figure nécessite de fermer les fenêtres après un changement du code et une nouvelle exécution. Mais il y a une solution ! Il suffit d'ajouter, juste après l'import du module matplotlib, la commande « `plt.close('all')` ». Elle fermera toutes les fenêtres au début de l'exécution, et permettra ainsi de voir les courbes souhaitées. Je le mets dans tous mes codes.

```
import matplotlib.pyplot as plt
plt.close('all')
```

## 1.IV. Les options

Il existe une multitude d'options permettant de changer les couleurs, styles de traits, épaisseurs, titres des graphiques, titres des axes etc… Voyons ici les plus importants.

Lors de la création du plot, on peut préciser :

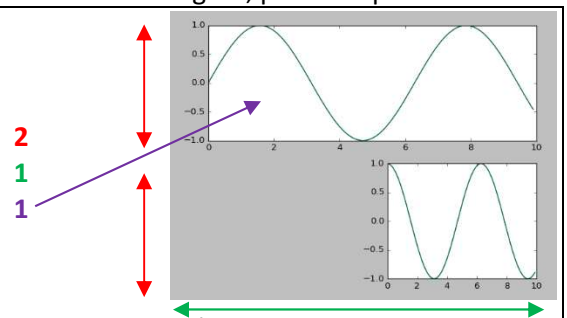
| | |
|---|---|
| `plt.plot(X, Y, linewidth=2.0)` | On règle ainsi l'épaisseur du trait |
| `plt.plot(X, Y, 'o')` | Crée un nuage de points |
| `plt.plot(X, Y, '--')` | Le trait est en pointillés |
| `plt.plot(X,Y,'r')` | On précise la couleur associée à la courbe<br>*b: blue - g: green - r: red - c: cyan - m: magenta - y: yellow - k: black - w: white* |
| `plt.plot(X,Y,label="Texte")`<br>`plt.legend()` | Ajout d'une légende à la courbe y(x)<br>Ligne nécessaire pour afficher la légende |
| `plt.plot(X, Y, '--r')` | Options cumulées : Pointillés rouges |

Après avoir créé un graphique via la commande « `plt.plot(X,Y)` » :

| | |
|---|---|
| `plt.xlim(-2,2)` | Définit l'intervalle des abscisses de la figure |
| `plt.ylim(-2,2)` | Définit l'intervalle des ordonnées de la figure |
| `plt.axis([0, 6, 0, 20])` | Définit l'intervalle des abscisses (0 à 6) puis des ordonnées (0 à 20) |
| `plt.axis('equal')`<br>`plt.axis('scaled')` | Repère orthonormé – A mettre avant xlim et ylim – Equal : redéfini les valeurs xlim ylim – scaled : redéfini la fenêtre |
| `plt.grid(True)` | Affiche la grille |
| `plt.title('Texte')` | Définit un titre au graphique |
| `plt.xlabel('Texte')` | Définit le nom des données en abscisses |
| `plt.ylabel('Texte')` | Définit le nom des données en ordonnée |
| `plt.show()` | Affiche le graphique |
| `plt.close()` | Ferme la dernière figure créée/appelée/ouverte |
| `plt.close('all')` | Ferme toutes les figures |
| `plt.clf()` | Vide la dernière figure créée/appelée/ouverte sans la fermer<br>(lors d'une simulation, il est beaucoup plus rapide de vider que de fermer/ouvrir) |
| `plt.pause(2)` | Permet d'attendre par exemple 2 secondes avant de continuer |
| `plt.savefig('Nom')` | Enregistrement dans le répertoire courant des figures préaffichées au format standard (png) avec le nom précisé – Pensez à enregistrer votre code dans un dossier avant de l'exécuter |

Il est possible de faire apparaître plusieurs graphiques dans la même figure, par exemple :

```
from math import cos,sin
n = 100
X = [i/10 for i in range(n)]
Y1 = [sin(X[i]) for i in range(n)]
Y2 = [cos(X[i]) for i in range(n)]
plt.subplot(211)
plt.plot(X,Y1)
plt.subplot(224)
plt.plot(X,Y2)
plt.show()
```




211 veut dire : séparation en 2 lignes et 1 colonnes, choix de la première cellule parmi 2
224 veut dire : séparation en 2 lignes et 2 colonnes, choix de la dernière cellule parmi 4

Google vous donnera le reste…

## 1.V. Gestion de plusieurs figures

Il n'existe pas une seule façon de faire. Personnellement, j'aime bien avoir la main sur les figures que je crée afin de pouvoir :

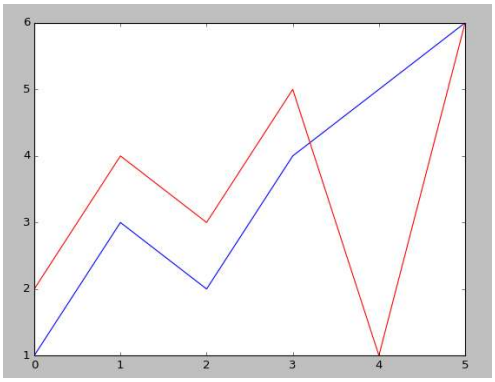
- En ouvrir autant que je le souhaite en parallèle
- Mettre à jour l'une d'elles
- Fermer l'une d'elles

Si on écrit :

```
X = [0,1,2,3,4,5]
Y = [1,3,2,4,5,6]
plt.plot(X,Y,'b')
plt.show()

X = [0,1,2,3,4,5]
Y = [2,4,3,5,1,6]
plt.plot(X,Y,'r')
plt.show()
```

On obtient :




Les deux courbes sont tracées sur le même graphique…

C'est pourquoi, à chaque fois que je fais une figure, j'utilise en premier lieu la commande :

```
plt.figure(i)
```

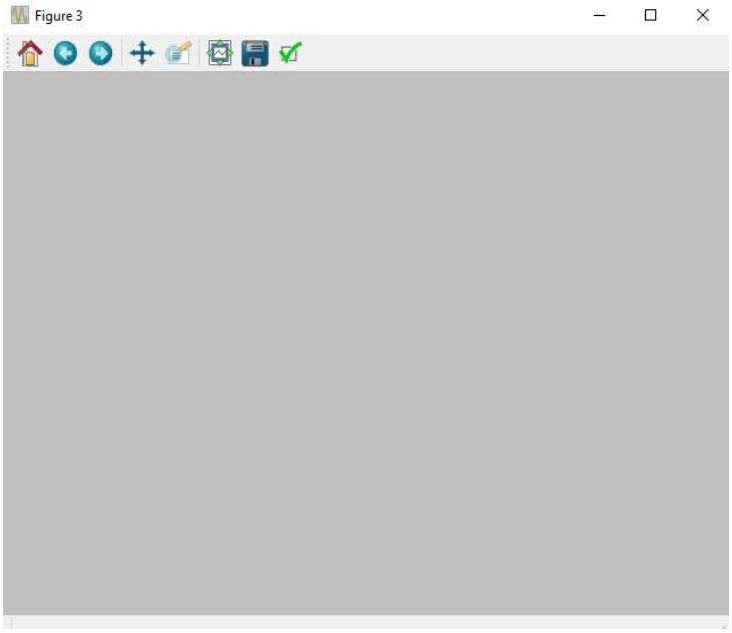
Où `i` est un nombre entier qui définit un numéro de figure.

Remarque : on peut mettre un nom (chaine de caractères) : `plt.figure("Nom")`

Ainsi, le code suivant ouvre la figure 3 :

```
plt.figure(3)
plt.show()
```

On obtient alors :




Il est alors très simple de fermer une figure parmi toutes les figures ouvertes à l'aide de la commande :

```
plt.figure(3)
plt.close()
```

De même, il est possible d'afficher une seconde courbe à la figure 3 en rappelant la figure en question :

```
plt.figure(3)
plt.plot(…)
plt.show()
```

Ou encore de vider la figure 2 en écrivant :

```
plt.figure(2)
plt.clf()
```

## 1.VI. Objet figure

Il est possible de créer un objet associé à une figure, pour cela il suffit d'écrire par exemple :

```
Fig_1 = plt.figure(1)
```

Il n'est plus alors obligatoire de rappeler la figure avec « `plt.figure(1)` » pour la fermer par exemple, il suffit d'écrire « `plt.close(Fig_1)` ».

Cela peut avoir d'autres intérêts que nous ne détaillerons pas ici.

## 1.VII. Le tout en une fonction

Idéalement, si vous souhaitez afficher une courbe, il est intéressant de créer une fonction qui prend en argument les deux listes de la courbe en question, et le numéro de la figure associée.

Ainsi, vous écrirez :

```
# Import librairie

from matplotlib import pyplot as plt

# Fermeture d'éventuelles fenêtres ouvertes

plt.close('all')

# Définition de la fonction

def f_courbe(X,Y,N_Fig,Legende):
    plt.figure(N_Fig)
    # Options à définir
    plt.plot(X,Y,label=Legende)
    plt.xlabel('Abscisses')
    plt.ylabel('Ordonnées')
    plt.legend()
    plt.show()

# Tracé

X = [1,2,3]
Y = [0,1,2]
f_courbe(X,Y,1,'Legende')
```

Remarque : plusieurs appels de `f_courbe` permettent de tracer plusieurs courbes sur la même figure. Cela fonctionne très bien depuis la zone de code avec F5, beaucoup moins bien depuis la console…