

I – Généralités

La bibliothèque **matplotlib** n’est pas fournie par défaut avec Python.
La commande **pip** va permettre de télécharger et installer cette bibliothèque :

```
pip install matplotlib
```

Dans la suite, on considère que la ligne suivante est écrite au début du script :

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

II – Types de graphiques

Instruction	Description
<code>plt.plot()</code>	Permet de tracer des courbes
<code>plt.scatter()</code>	Permet d’afficher un nuage de points
<code>plt.bar()</code>	Permet de générer des diagrammes en bâtons
<code>plt.hist()</code>	Permet de générer des histogrammes
<code>plt.pie()</code>	Permet de générer des diagrammes circulaires (<i>camemberts</i>)

Par exemple, pour tracer une courbe représentative, l’instruction sera :

```
1 plt.plot(X, Y, style, linewidth = 1, label = 'y = f(x)')
```

- ⌚ **X** est une liste d’abscisses ;
- ⌚ **Y** est une liste d’ordonnées **y** telles que **y = f (x)**, avec **x** l’élément de la liste **X** de même indice que **y** dans **Y** ;
- ⌚ **style** est une chaîne de caractères décrivant l’aspect du tracé (*ci-à droite*) ;
- ⌚ **linewidth** est l’épaisseur (*en pixels*) du tracé ;
- ⌚ **label** est la légende associée au tracé.

III – Aspect du tracé

De manière générale, **style** est une chaîne comportant trois informations :

- ⌚ une **couleur**, celle de la courbe.
*On peut aussi la donner indépendamment avec le paramètre **color**='... '.*
- ⌚ un **marqueur**, la forme d’affichage des points.
*On peut aussi le donner indépendamment avec le paramètre **marker**='... '.*
- ⌚ un **style**, celui de la ligne tracée entre les points.
*On peut aussi le donner indépendamment avec le paramètre **linestyle**='... ', parfois abrégé en **ls**='... '.*

Si des caractères ne sont pas précisés, alors le style par défaut est appliqué.

Chaîne	Couleur
b	bleu
g	vert
r	rouge
c	cyan
m	magenta
y	jaune
k	noir
w	blanc

Chaîne	Ligne
-	continue
--	tirets
:	pointillés
- .	tirets points

Chaîne	Marqueur
.	point
,	pixel
o	cercle
v	triangle point en bas
^	triangle point en haut
<	triangle point à gauche
>	triangle point à droite
1	croix à trois branches vers le bas
2	croix à trois branches vers le haut
3	croix à trois branches vers la gauche
4	croix à trois branches vers la droite
s	carré
p	pentagone
*	étoile
h	hexagone
+	plus
x	croix
d	carreau
 	barre verticale
-	barre horizontale

IV – Paramètres du repère

▫ Définir les abscisses (ordonnées) minimales et maximales

```
plt.xlim(x_min, x_max)           # entiers ou flottants
plt.ylim(y_min, y_max)
```

L'instruction suivante permet de paramétrer les deux axes en une seule fois :

```
plt.axis( [x_min, x_max, y_min, y_max] )
```

▫ Ajouter une grille

```
plt.grid()
```

▫ Libeller les axes

```
plt.xlabel('axe des abscisses')
plt.ylabel('axe des ordonnées')
```

▫ Ajouter un titre au graphique

```
plt.title('titre du graphique')
```

Il est possible d'utiliser la syntaxe LATEX en faisant précéder la chaîne de caractère de la lettre « **r** » :

```
plt.title(r'y = $\frac{1}{2} x^2$')
```

▫ Ajouter une légende (utilise la valeur de label)

```
plt.legend()
```

▫ Afficher le graphique

```
plt.show()
```

V – Types de graphiques

En utilisant plusieurs fois la fonction **plot()** (ou **scatter()**, **bar()**, etc...), les éléments sont tracés dans le même graphique.

Il est possible de créer des **grilles de graphes**, solution très pratique pour empiler des graphes qui doivent être regardés ensemble mais qui n'ont pas les mêmes ordres de grandeurs en matière d'abscisses et d'ordonnées.

La grille de graphes est décrite par la fonction **subplot()**. Cette fonction définit aussi l'emplacement du graphe suivant dans la grille de graphes.

```
plt.subplot(nb_lignes, nb_colonnes, index, options)
```

▮ **nb_lignes** est le nombre de lignes de la grille de graphes ;

▮ **nb_colonnes** est le nombre de colonnes de la grille de graphes ;

▮ **index** est le numéro du graphe défini après l'appel **subplot()**.
la numérotation se fait de gauche à droite et de haut en bas.

▮ **options** est définie dans la documentation officielle :

https://matplotlib.org/stable/api/as_gen/matplotlib.pyplot.subplot.html

Par exemple :

```
1 # Graphes l'un à côté de l'autre (1 ligne, 2 colonnes)
2 plt.subplot(1, 2, 1)           # Graphe de gauche
3 plt.plot(X, Y, 'k-', linewidth=2)
4
5 plt.subplot(1, 2, 2, facecolor='c') # Graphe de droite
6 plt.plot(T, Y, 'r+:', linewidth=1)
```

VI – Éléments supplémentaires

▫ Ajouter du texte aux coordonnées (x, y) indiquées

```
plt.text(abscisse, ordonnee, 'texte affiché')
```

▫ Ajouter une annotation aux coordonnées **xytext** et tracer une flèche jusqu'au point de coordonnées **xy**

```
plt.annotate('annotation', xy=(x_fin, y_fin),
             xytext=(x_debut, y_debut),
             arrowprops=dict(facecolor='black', arrowstyle='->'))
```

Il est aussi possible d'ajouter des vecteurs, des figures, etc...

Vous trouverez la documentation complète officielle à cette adresse :

▮ <https://matplotlib.org/>

Pour approfondir, des exemples complémentaires d'utilisation sont disponibles (en français) à ces adresses :

▮ <http://www.python-simple.com/python-matplotlib/pyplot.php>

▮ <https://www.datacorner.fr/matplotlib/>