Chapitre 4 : Représentations graphiques

Axel Carpentier

Informatique

Python

Table des matières

La bibliothèque `matplotlib.pyplot`

La bibliothèque matplotlib.pyplot

Le module matplotlib.pyplot est une bibliothèque python qui permet de faire de nombreuses choses mathématiques, entre autre des graphiques.

En liaison avec le module numpy, le module matplotlib.pyplot contient un certain nombre de fonctions dédiées au tracé de graphes. La première chose à faire consiste donc à importer ces deux modules :

```
1 import numpy as np
2 import matplotlib.pyplot as plt
```

La bibliothèque `matplotlib.pyplot`

Les fonctions du module `numpy` sont désormais utilisables à condition d'être préfixées par `np` ; celles du module `matplotlib.pyplot` par le préfixe `plt`.

Il est naturellement possible de choisir un autre préfixe pour importer ces modules, ceux choisis ici sont les plus conventionnellement utilisés et universalisés.

Le principe général d'un tracé de graphe consiste, à partir d'une figure initialement vide, à lui ajouter progressivement des éléments graphiques (graphes, axes, légendes, ...) qui vont se superposer via les différentes fonctions intégrées dans le module `matplotlib.pyplot`.

Tracé simple

Tracé simple

La fonction `plt.plot` permet de tracer des points.

```
1  x=2
2  y=3
3
4  plt.plot(x,y)
5  plt.show()
```

On voit alors le tracé effectué apparaître dans un fenêtre annexe.

Tracé simple

Pour tracer plusieurs points, on peut préférer définir une liste de points, ce qui va permettre, entre autre, de relier les points entre eux, par exemple :

```
1  x=[2,4,5]
2  y=[3,5,8]
3
4  plt.plot(x,y)
5  plt.show()
```

Tracé simple

On a, de manière générale, pour $X = [x_0, x_1, x_2, \ldots, x_n]$ et $Y = [y_0, y_1, y_2, \ldots, y_n]$ deux listes numériques de même taille, la fonction `plt.plot(X, Y)` ajoute à la figure une ligne brisée qui relie les points de coordonnées (x_i, y_i) pour $0 \leq i \leq n$.

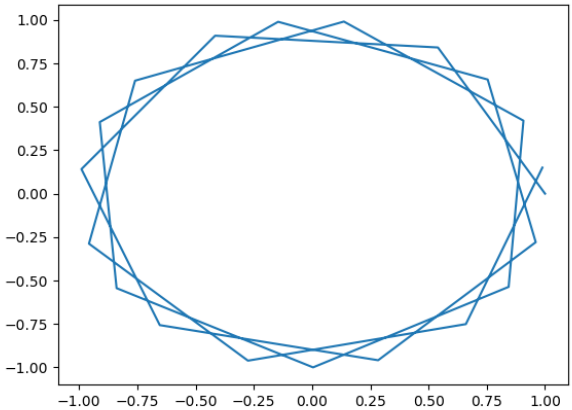
Exemple :

On considère le code suivant :

```
1  X = [np.cos(t) for t in range(20)]
2  Y = [np.sin(t) for t in range(20)]
3
4  plt.plot(X, Y)
5  plt.show()
```

Tracé simple

En exécutant le programme précédent, un fenêtre va s'ouvrir affichant le graphique suivant :



Tracé du graphe d'une fonction

Fonctionnement

Une fois une fonction définie, on peut tracer des points de sa courbe représentative, par exemple, pour la fonction carré et en utilisant une boucle `for` (et plus précisément une boucle avec `range`):

```
1 def f(x):
2     y=x**2
3     return y
4
5 for i in range(-10,10):
6     x=i
7     plt.plot(x,f(x))
8 plt.show()
```

Le programme précédent place un point de la courbe représentative à chaque abscisse entière, ce qui fait en tout peu de points.

Fonctionnement

On appelle "pas" l'écart entre deux abscisses consécutives, qui vaut donc 1 dans le programme précédent.

On peut bien sûr tracer autant de points et avec un pas aussi fin qu'on le souhaite.
On modifie par exemple le programme précédent pour tracer des points de la courbe avec un pas de un dixième:

```
1 def f(x):
2     y=x**2
3     return y
4
5 pas=0.1
6 for i in range(-10,10):
7     x=i*pas
8     plt.plot(x,f(x))
9 plt.show()
```

Fonctionnement

De manière générale, pour tracer le graphe d'une fonction d'équation $y = f(x)$ sur l'intervalle $[a, b]$, il suffit de subdiviser cet intervalle avec un pas suffisamment fin $x_0 = a < x_1 << x_{n-1} = b$, puis, pour chaque valeur x_i de cette subdivision, calculer la valeur $y_i = f(x_i)$.

La fonction `linspace` du module `numpy` réalise simplement cette subdivision.
En effet, `np.linspace(a,b,n)` subdivise $[a, b]$ en n points équirépartis.

Tracé du graphe d'une fonction

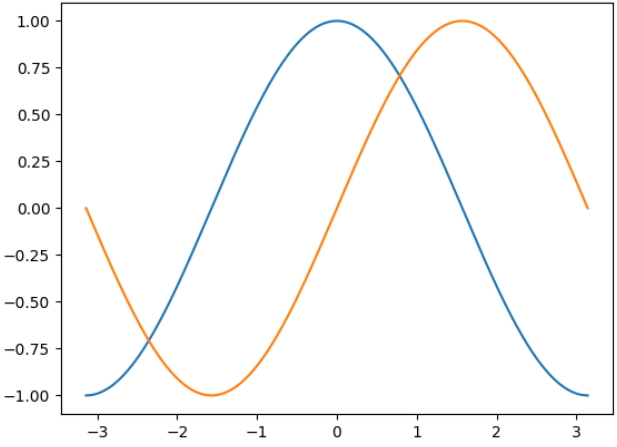
Personnalisation d'un graphique sur un exemple

On souhaite représenter graphiquement les fonctions $t \mapsto \sin(t)$ et $t \mapsto \cos(t)$ sur l'intervalle $[-\pi, \pi]$.

```
1 X = np.linspace(-np.pi, np.pi, 1000)
2 C = [np.cos(x) for x in X]
3 S = [np.sin(x) for x in X]
4
5 plt.plot(X, C)
6 plt.plot(X, S)
7 plt.show()
```

Personnalisation d'un graphique sur un exemple

En exécutant le programme précédent, une fenêtre s'ouvre affichant le graphique suivant :



Personnalisation d'un graphique sur un exemple

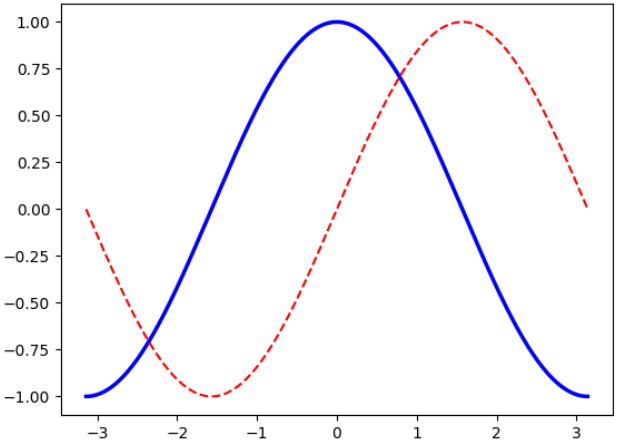
On remarque que les deux graphes ont été automatiquement tracés en deux couleurs différentes. Il est bien sur possible de choisir la couleur du graphe, ainsi que l'épaisseur du trait et le style du tracé.

On peut donc remplacer le tracé par :

```
1 plt.plot(X, C, color="blue", linestyle="-", linewidth=2.5)
2 plt.plot(X, S, color="red", linestyle="dashed")
```

Personnalisation d'un graphique sur un exemple

On obtient alors le graphique suivant :



Personnalisation d'un graphique sur un exemple

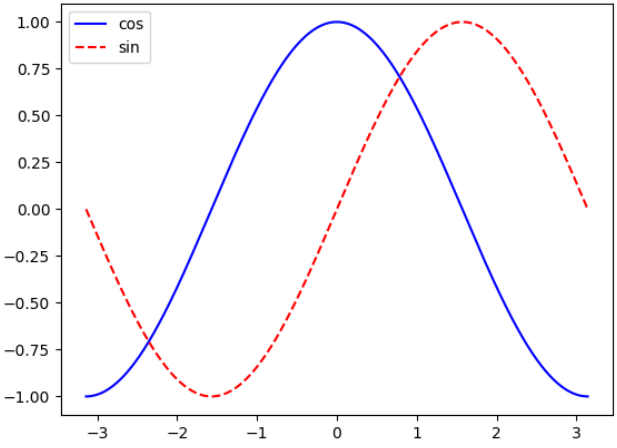
On peut aussi ajouter une légende associée à chacune des courbes tracées en utilisant le paramètre `label` de la fonction `plot`.

La fonction `legend` associe à la couleur du tracé le label que vous lui avez donné :

```
1 plt.plot(X, C, color="blue", linestyle="-", label='cos')
2 plt.plot(X, S, color="red", linestyle="dashed", label='sin')
3 plt.legend(loc='upper␣left')
```

Personnalisation d'un graphique sur un exemple

On obtient alors le graphique suivant :



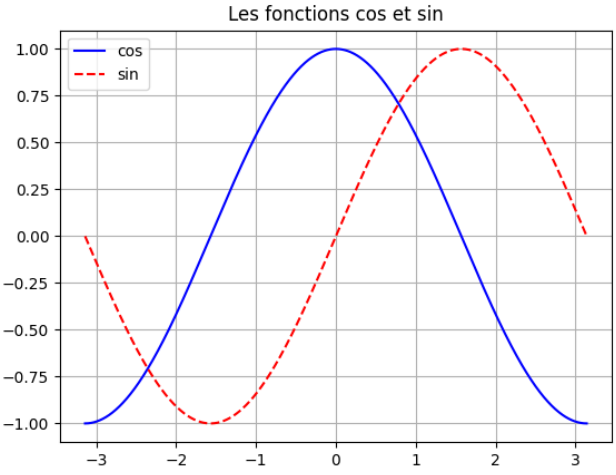
Personnalisation d'un graphique sur un exemple

La fonction `plt.grid` permet de faire apparaitre le quadrillage associé à la subdivision choisie. Enfin, la fonction `plt.title` ajoute un titre à la figure créée.
On obtient alors le programme complet suivant :

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  X = np.linspace(-np.pi, np.pi, 1000)
5  C = [np.cos(x) for x in X]
6  S = [np.sin(x) for x in X]
7  plt.plot(X, C, color="blue", linestyle="-", label='cos')
8  plt.plot(X, S, color="red", linestyle="dashed", label='sin')
9  plt.legend(loc='upper␣left')
10 plt.grid()
11 plt.title('Les␣fonctions␣cos␣et␣sin')
12 plt.show()
```

Personnalisation d'un graphique sur un exemple

En exécutant ce programme on obtient alors le graphique suivant :



Personnalisation d'un graphique sur un exemple

Les modifications et personnalisations effectuées ici sont naturellement arbitraires et ne constituent qu'un exemple.

Il est possible de représenter de nombreuses fonctions et de modifier et personnaliser le graphique à sa guise selon des critères esthétiques et pratiques (selon ce que l'on cherche à mettre en évidence sur une représentation graphique de fonction).