Atelier d'informatique

Épisode V : Tracés avec `matplotlib`

2 février 2017

1 Introduction

2 Principe

3 Définir une figure

4 Effectuer un tracé

5 Tracer le graphe d'une fonction compliquée

6 Des options multiples

Introduction

La librairie `matplotlib` permet de créer des graphes scientifiques en Python, tels que des courbes représentatives de fonctions, des histogrammes, des diagrammes en boîte, ou encore des nuages de points.

Introduction

La librairie `matplotlib` permet de créer des graphes scientifiques en Python, tels que des courbes représentatives de fonctions, des histogrammes, des diagrammes en boîte, ou encore des nuages de points.

On ne s'intéressera ici qu'au tracé de courbes représentatives de fonctions.

Introduction

La librairie `matplotlib` permet de créer des graphes scientifiques en Python, tels que des courbes représentatives de fonctions, des histogrammes, des diagrammes en boîte, ou encore des nuages de points.

On ne s'intéressera ici qu'au tracé de courbes représentatives de fonctions.

Pour toute la suite du cours, veillez à ce que votre script contienne les lignes suivantes

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

et qu'il ait été exécuté au moins une fois.

Principe

La librairie `matplotlib` crée des objets qui permettent de manipuler des figures, de type `Figure`, qui contiennent des zones de tracé, de type `Axes`, dans lesquels se situent des tracés, de type `Line2D` le plus souvent.

Principe

La librairie `matplotlib` crée des objets qui permettent de manipuler des figures, de type `Figure`, qui contiennent des zones de tracé, de type `Axes`, dans lesquels se situent des tracés, de type `Line2D` le plus souvent.

Utiliser `matplotlib` pour effectuer un tracé revient à définir une figure, une zone de tracé, puis à faire le tracé dans la zone.

Principe

La librairie `matplotlib` crée des objets qui permettent de manipuler des figures, de type `Figure`, qui contiennent des zones de tracé, de type `Axes`, dans lesquels se situent des tracés, de type `Line2D` le plus souvent.

Utiliser `matplotlib` pour effectuer un tracé revient à définir une figure, une zone de tracé, puis à faire le tracé dans la zone.

On peut le faire de manière interactive, en tapant des instructions directement dans la console, sans stocker chaque objet dans une variable : `matplotlib` comprend automatiquement qu'il doit se référer au dernier objet créé, dans la dernière figure créée.

Principe

La librairie `matplotlib` crée des objets qui permettent de manipuler des figures, de type `Figure`, qui contiennent des zones de tracé, de type `Axes`, dans lesquels se situent des tracés, de type `Line2D` le plus souvent.

Utiliser `matplotlib` pour effectuer un tracé revient à définir une figure, une zone de tracé, puis à faire le tracé dans la zone.

On peut le faire de manière interactive, en tapant des instructions directement dans la console, sans stocker chaque objet dans une variable : `matplotlib` comprend automatiquement qu'il doit se référer au dernier objet créé, dans la dernière figure créée.

Quand on utilise un script, il est conseillé de manipuler directement les objets et utiliser les méthodes qui sont définies dessus. Par souci de simplicité, on fera sans.

Définir une figure

Pour définir un objet `figure`, on utilise le constructeur associé, `plt.figure`. Il fonctionne comme une fonction, donc il faut mettre des parenthèses :

Définir une figure

Pour définir un objet `figure`, on utilise le constructeur associé, `plt.figure`. Il fonctionne comme une fonction, donc il faut mettre des parenthèses :

```
plt.figure()
```

Définir une figure

Pour définir un objet `figure`, on utilise le constructeur associé, `plt.figure`. Il fonctionne comme une fonction, donc il faut mettre des parenthèses :

```
plt.figure()
```

et on peut éventuellement préciser le numéro de la figure à créer en mettant un nombre entier entre les parenthèses.

Définir une figure

Pour définir un objet `figure`, on utilise le constructeur associé, `plt.figure`. Il fonctionne comme une fonction, donc il faut mettre des parenthèses :

```
plt.figure()
```

et on peut éventuellement préciser le numéro de la figure à créer en mettant un nombre entier entre les parenthèses. En mode interactif, la zone de tracé est définie **automatiquement** lorsque l'on effectue un tracé. `matplotlib` le définit alors dans la figure la plus récente.

Comment utiliser matplotlib

Effectuer un tracé

Étant donné une liste `X` des *abscisses* et une liste `Y` des *ordonnées* de la fonction que l'on veut tracer, il suffit d'utiliser l'instruction

```
plt.plot(X, Y)
```

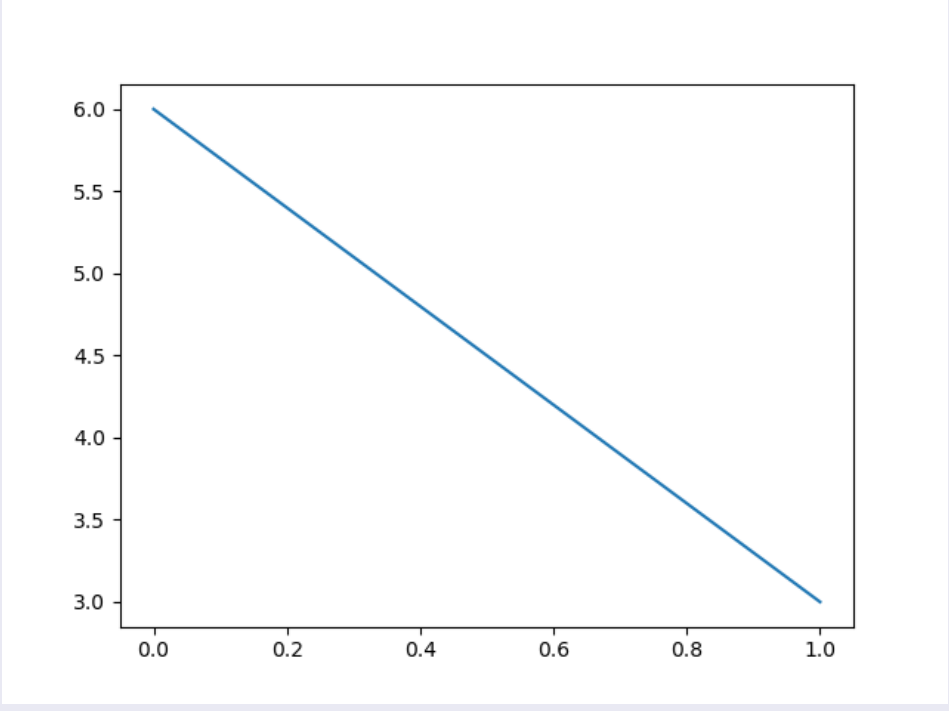
pour effectuer le tracé.

Pour afficher, on utilise l'instruction `plt.show()`.

Exemple 1

```
X = [0,1]
Y = [6,3]
plt.figure()
plt.plot(X,Y)
```

produit la figure suivante :



Étant donnée une fonction numérique (qui renvoie des valeurs) `f` et un intervalle *I* sur lequel on veut tracer son graphe, il nous faut d'abord représenter par une liste de points l'intervalle.

Étant donnée une fonction numérique (qui renvoie des valeurs) `f` et un intervalle *I* sur lequel on veut tracer son graphe, il nous faut d'abord représenter par une liste de points l'intervalle.

Pour cela, la librairie `numpy` nous fournit une fonction nommée `linspace`. On l'utilise de la façon suivante :

```
X = np.linspace(a, b, N)
```

où `a` et `b` sont les extrémités de l'intervalle *I* (par exemple -3 et 2 si $I = [-3, 2]$), et `N` le nombre de points avec lequel on veut représenter *I*. Le plus il y en a, le plus fidèle sera la courbe tracée.

Étant donnée une fonction numérique (qui renvoie des valeurs) `f` et un intervalle I sur lequel on veut tracer son graphe, il nous faut d'abord représenter par une liste de points l'intervalle.

Pour cela, la librairie `numpy` nous fournit une fonction nommée `linspace`. On l'utilise de la façon suivante :

```
X = np.linspace(a, b, N)
```

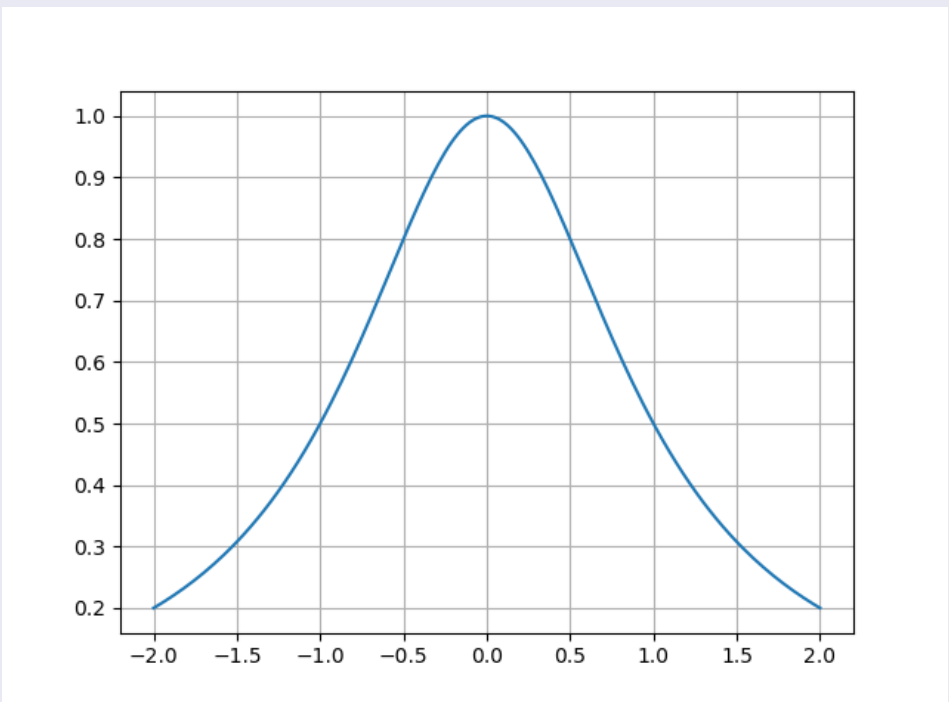
où `a` et `b` sont les extrémités de l'intervalle I (par exemple -3 et 2 si $I = [-3, 2]$), et `N` le nombre de points avec lequel on veut représenter I. Le plus il y en a, le plus fidèle sera la courbe tracée.

Enfin, on effectue l'instruction `Y = f(X)` pour calculer les ordonnées, et on fait comme décrit avant.

Exemple 2

```
def f(x):
        return 1/(1+x**2)
X = np.linspace(-2,2,200)
Y = f(X)
plt.figure()
plt.plot(X,Y)
plt.grid()
```

produit le graphe suivant :



Pour ajouter un quadrillage au repère, on peut utiliser `plt.grid`. C'est une méthode relative à dernière zone de tracé en mémoire, donc on lui fait appel via `plt.grid()`. Elle fonctionne comme un interrupteur, ajoute un quadrillage s'il n'y en a pas, l'enlève s'il y en a déjà un. On peut forcer l'ajout en précisant `plt.grid(True)`.

Pour ajouter un quadrillage au repère, on peut utiliser `plt.grid`. C'est une méthode relative à dernière zone de tracé en mémoire, donc on lui fait appel via `plt.grid()`. Elle fonctionne comme un interrupteur, ajoute un quadrillage s'il n'y en a pas, l'enlève s'il y en a déjà un. On peut forcer l'ajout en précisant `plt.grid(True)`.

On peut donner un titre au graphe via `plt.title`, qui est aussi une méthode, à laquelle on fait appel en passant pour argument la chaîne de caractère que l'on veut utiliser comme titre :

```
plt.title("La␣meilleure␣courbe␣du␣monde.␣Vraiment.")
```

Pour ajouter un quadrillage au repère, on peut utiliser `plt.grid`. C'est une méthode relative à dernière zone de tracé en mémoire, donc on lui fait appel via `plt.grid()`. Elle fonctionne comme un interrupteur, ajoute un quadrillage s'il n'y en a pas, l'enlève s'il y en a déjà un. On peut forcer l'ajout en précisant `plt.grid(True)`.

On peut donner un titre au graphe via `plt.title`, qui est aussi une méthode, à laquelle on fait appel en passant pour argument la chaîne de caractère que l'on veut utiliser comme titre :

```
plt.title("La␣meilleure␣courbe␣du␣monde.␣Vraiment.")
```

On peut également forcer à occuper le plus d'espace possible en utilisant la méthode `tight_layout`: `plt.tight_layout()` essaie de minimiser les espaces vides sur la figure.

Pour ajouter un quadrillage au repère, on peut utiliser `plt.grid`. C'est une méthode relative à dernière zone de tracé en mémoire, donc on lui fait appel via `plt.grid()`. Elle fonctionne comme un interrupteur, ajoute un quadrillage s'il n'y en a pas, l'enlève s'il y en a déjà un. On peut forcer l'ajout en précisant `plt.grid(True)`.

On peut donner un titre au graphe via `plt.title`, qui est aussi une méthode, à laquelle on fait appel en passant pour argument la chaîne de caractère que l'on veut utiliser comme titre :

```
plt.title("La␣meilleure␣courbe␣du␣monde.␣Vraiment.")
```

On peut également forcer à occuper le plus d'espace possible en utilisant la méthode `tight_layout`: `plt.tight_layout()` essaie de minimiser les espaces vides sur la figure.

Il existe encore plein d'options, que l'on peut trouver dans la documentation de `matplotlib` (attention, c'est en anglais...).