TD Informatique – Tracés (TIPE)

MP
2024-2025

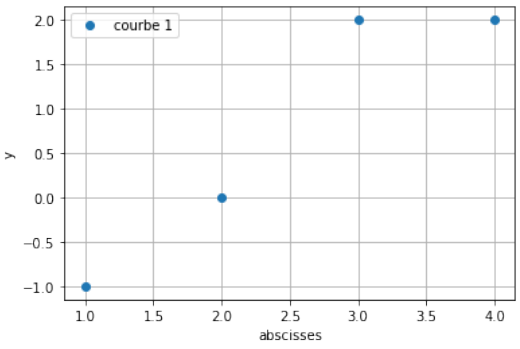
Principe de tracé

On importe un module de tracés. On prépare une liste d'abscisses x et une liste d'ordonnées y.
Si on a besoin de beaucoup de points, l'utilisation de listes définies *par compréhension* peut être efficace. Par exemple,

```
L = [k/10 for k in range(1,10)]
```

On utilise `plot` et ses multiples options.

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [-1, 0, 2, 2]
plt.plot(x, y, 'o', label='courbe 1')
plt.xlabel('abscisses')
plt.ylabel('y')
plt.legend()
# affiche le label
plt.grid() # quadrillage
plt.show()
```



(⋆) Exercice 1

Comment modifier le programme précédent pour que les points soient reliés ?

(⋆) Exercice 2

1. Tracer la fonction valeur absolue sur $[-2, 2]$.
2. On importe la fonction exponentielle (de la bibliothèque numpy par exemple). Tracer la fonction (gaussienne normalisée) suivante sur $[-5, 5]$.

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$$

3. Tracer la fonction suivante sur $[-3, 3]$.

$$g(x) = \begin{cases} 0 & \text{si } x < 0 \\ \sin x & \text{si } x \in [0, \frac{\pi}{2}] \\ 1 & \text{si } x > \frac{\pi}{2} \end{cases}$$

(⋆) Exercice 3 La méthode de Héron est une méthode efficace de calcul approché de racine carrée. Soit x un réel positif (dont on cherche la racine). On considère la suite u donnée par :

$$\begin{cases} u_0 & = \text{ le plus petit entier dépassant } x \\ u_{n+1} & = \frac{1}{2}(u_n + \frac{x}{u_n}) \text{ pour } n \in \mathbf{N} \end{cases}$$

1. Écrire une fonction `Heron(x, n)` qui renvoie u_n.
2. Représenter les 20 premiers termes de la suite lorsque $x = 2$. Essayer d'autres valeurs et commenter.

3. Tracer les valeurs de `Heron(x, 5)` pour 50 valeurs de $x \in [0, 10]$ et comparer avec la courbe représentative de $\sqrt{x}$. Commenter.

(⋆⋆) Exercice 4 FACULTATIF

Une figure fractale est une figure mathématique qui présente une structure similaire à toutes les échelles. En zoomant sur une partie de la figure, il est possible de retrouver toute la figure. Nous nous intéressons ici au flocon de Von Koch.
On définit une suite de courbes de la manière suivante. On part d'un segment de longueur nommée `cote`. On divise ce segment en 3 segments de même longueur et on remplace le segment du milieu par deux segments formant un triangle équilatéral avec le segment supprimé. On réitère le processus sur chacun des segments obtenus.

Le module `turtle` permet le déplacement d'une « tortue » qui, en se déplaçant, laisse un trait sur son passage.

`import turtle as t`	import du module
`t.forward(d)`	avance la tortue d'une longueur d
`t.left(a)`	fait tourner la tortue d'un angle a (en degrés)
`t.speed(0)`	permet d'accélérer la tortue
`t.color('blue')`	pour le choix d'une couleur (ne peut être choisi qu'une fois hors changement)
`t.exitonclick()`	garde la fenêtre ouverte

1. Écrire une fonction `basique` de paramètre `cote` qui trace la courbe obtenue à l'étape 1 (3 segments). Pour l'exécuter, on peut prendre `cote` égal à 200.

2. Écrire une fonction récursive `flocon` de paramètres `n` et `cote`, permettant d'obtenir le tracé à l'étape n.
3. Admirer :

```
t.color('blue')
flocon(3, 300)
t.left(-120)
flocon(3, 300)
t.left(-120)
flocon(3, 300)
t.exitonclick()
```