



Représentations graphiques avec `matplotlib.pyplot`

◇ Objectifs : représenter graphiquement des suites, des fonctions (et des mesures expérimentales). Donner des noms aux courbes, aux axes, à la figure.

◇ Les différentes étapes pour effectuer le tracé avec PYTHON :

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

Lx = ...
Ly = ...

plt.plot(Lx, Ly, ...)

plt.legend(...)
plt.xlabel(...)
plt.ylabel(...)
plt.title(...)
plt.show()
```

Importer le module **Numpy**
Importer le module de tracé

Construction de la liste des abscisses
Construction de la listes des ordonnées
(**Lx** et **Ly** doivent être de même taille)
Représenter graphiquement les points
dont les abscisses sont dans *x* et les ordonnées dans *y* (*x* et *y* sont des listes, ou des tableaux, de même taille).

Répéter ces étapes pour chaque courbe à tracer

Donner les noms des différentes courbes
Donner les noms des axes

Donner le nom du dessin

Afficher la fenêtre graphique

Remarque. On peut ajouter à la commande **plot** des indications supplémentaires pour préciser le style du tracé sous forme d'une chaîne de caractères : "**ro**" pour un tracé avec des ronds rouges, "**b--**" pour un tracé avec une ligne en pointillés bleus, "**g-**" pour un tracé avec une ligne pleine verte, etc. On se reportera aux exemples vus en classe. □

◇ Pour construire la liste **Lx** des abscisses, on utilise souvent :

- Soit la commande **range** lorsque l'on désire des abscisses entières, par exemple :

```
Lx = list(range(-10, 11))
print(Lx)
```

```
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- Soit la commande `np.linspace(a,b,n)` pour obtenir n valeurs régulièrement espacées entre a et b :

```
Lx = np.linspace(-1,1,5)
print(Lx)
```

```
[-1. -0.5  0.  0.5  1. ]
```

◇ Pour construire la liste **Ly** des ordonnées, on utilise en général la construction :

```
Ly = [f(x) for x in Lx]
```

la fonction **f** ayant été définie auparavant.

Exemple. Représentation de la fonction $x \mapsto x^2 - 1$ sur l'intervalle $[-1, 2]$.

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

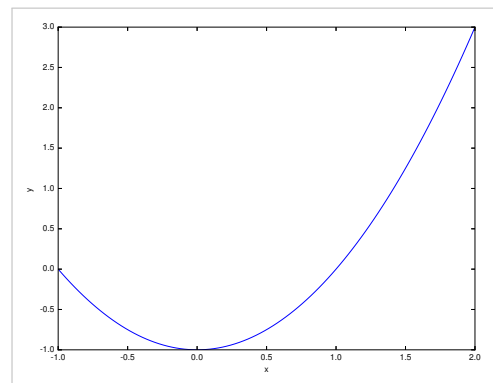
def g(x):
    return x**2-1

Lx = np.linspace(-1,2,100)
Ly = [g(x) for x in Lx]

plt.plot(Lx,Ly, 'b-')

plt.xlabel("x")
plt.ylabel("y")
```

```
plt.show()
```



Exemples de représentations graphiques

Fonctions

Exemple. Représenter les fonctions $f: x \mapsto e^{-x^2}$ et $g: x \mapsto \begin{cases} 0 & \text{si } x < 0 \\ 1.5 & \text{si } x \geq 0 \end{cases}$ sur $[-3, 3]$.

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

def f(x):
    return np.exp(-x**2)

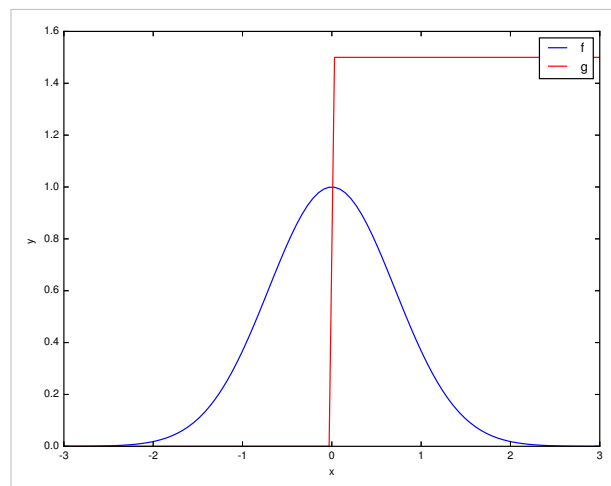
def g(x):
    if x < 0:
        return 0
    else:
        return 1.5

Lx = np.linspace(-3, 3, 100)
Lf = [f(x) for x in Lx]
Lg = [g(x) for x in Lx]

plt.plot(Lx, Lf, 'b-')
plt.plot(Lx, Lg, 'r-')

plt.legend(["f", "g"])
plt.xlabel("x")
plt.ylabel("y")
```

```
plt.show()
```



Suites

Exemple. On considère la suite $(u_n)_{n \geq 0}$ définie par :

$$u_0 = 1$$
$$\forall n \in \mathbb{N}, u_{n+1} = \sqrt{u_n + 1}$$

Représenter graphiquement les valeurs prises par la suite (u_n) pour $0 \leq n \leq 10$.

```
import matplotlib.pyplot as plt

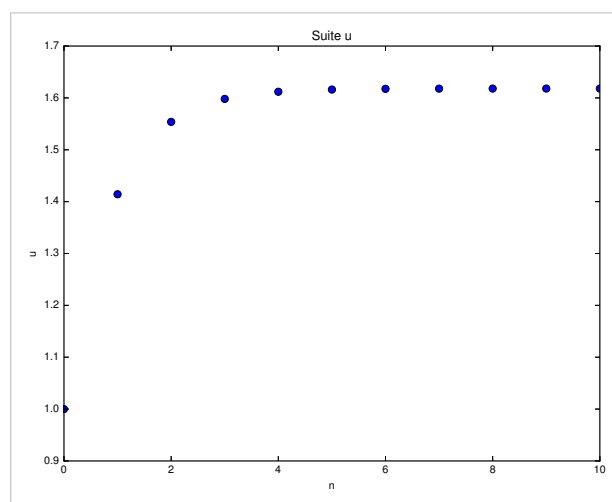
def u(n):
    u=1
    for k in range(n):
        u=(u+1)**0.5
    return u

Ln = range(0,11)
Lu = [u(n) for n in Ln]

plt.plot(Ln,Lu, 'bo')

plt.title("Suite u")
plt.xlabel("n")
plt.ylabel("u")
```

```
plt.show()
```



Un exemple en chimie

Exemple. On mesure la concentration $C(t)$ d'une espèce chimique à différents instants. On obtient le tableau ci-contre. Par ailleurs, une étude théorique montre que l'évolution de cette concentration peut être modélisée par la fonction :

$$C : t \mapsto \frac{1}{kt + \frac{1}{C_0}}$$

avec $k = 80.2 \text{ mol}^{-1} \cdot \text{L} \cdot \text{s}^{-1}$ et $C_0 = 5.0 \cdot 10^{-4} \text{ mol} \cdot \text{L}^{-1}$. Représenter sur le même graphique la courbe théorique et les valeurs expérimentales.

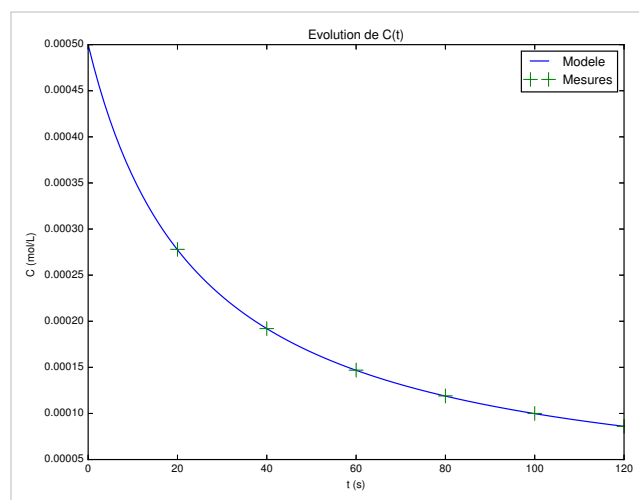
```
import matplotlib.pyplot as plt
import numpy as np

t_mes = [20, 40, 60, 80, 100, 120]
C_mes = [2.78e-4, 1.92e-4, 1.47e-4, 1.19e-4, 1.0e-4, 0.86e-4]
k = 80.2
C0 = 5.0e-4
t = np.linspace(0, 120, 100)

plt.plot(t, (k*t+1/C0)**(-1))
plt.plot(t_mes, C_mes, '+', markersize=12)

plt.legend(["Modele", "Mesures"])
plt.title("Evolution de C(t)")
plt.xlabel("t (s)")
plt.ylabel("C (mol/L)")
```

```
plt.show()
```



Un exemple en physique

Exemple. On considère l'équation différentielle du second ordre :

$$\frac{d^2 u}{dt^2} + \omega^2 + u = 0$$

avec $\omega = 10^4 \text{ rad} \cdot \text{s}^{-1}$ ainsi que la solution :

$$u(t) = A \cos(\omega t)$$

avec $A = 1 \text{ V}$. Représenter le portrait de phase sur l'intervalle $[0, 4T]$ avec $T = 2\pi/\omega$.

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

w = 1e4
A = 1
T = 2*np.pi/w

def u(t):
    return A*np.cos(w*t)

def v(t):
    return -A*w*np.sin(w*t)

Lt = np.linspace(0, 4*T, 100)

Lu = [u(t) for t in Lt]
Lv = [v(t) for t in Lt]
plt.plot(Lu, Lv, 'b-')

plt.xlabel("u (V)")
plt.ylabel("v (V/s)")
plt.title("Portrait de phase")
```

```
plt.show()
```

