TP6

Partie 1 — Visualisation avec matplotlib

Ex. 1 — "Scatter plot" (nuage de points)

Matplotlib est une excellente librairie graphique pour générer des figures scientifiques en 2D et 3D. Parmi les avanatges de cette librairie, on peut citer :

- prise en main facile,
- des figures de haute qualité et plusieurs formats PNG, PDF, SVG, EPS et PGF,

Comment utiliser la librairie matplotlib dans votre programme ?

Pour utiliser la librairie, il faut importer le module `matplotlib`. Ceci peut être réalisé avec la ligne de commande suivante :

```
import matplotlib
```

Les fonctions de visulaisation de la librairie que nous allons utiliser dans ce TP se trouvent dans le sous-module `pyplot` de `matplotlib`. Il est possible d'importer uniquement les fonctions du sous-module `pyplot`. La commande à utiliser dans ce cas est :

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

Nous nous intéressons ici à dessiner des nuages de points. Commençons par créer des points à afficher :

```
xs = range(1, 25)
ys = [1 / x for x in xs]
```

Le code ci-dessus permet de générer des points xs ayant comme valeurs de 1 à 25 et des points ys égales à l'inverse de ces valeurs.

La ligne de code `ys = [1 / x for x in xs]` permet de créer une liste `ys` dont les valeurs égales à `1/x` pour tout `x` dans `xs`.

- Pour dessiner un "scatter plot" (nuage de point), on utilise la méthode scatter comme suit :

  ```
  plt.scatter(xs, ys)
  ```

  `xs` contient les abscisses et `ys` les ordonnées des points à dessiner.

- On peut combiner ce graphique avec un autre ensemble de points. Créons d'autres points :

  ```
  zs = [1 / (25 - x) for x in xs]
  ```

- Pour afficher les deux courbes dans un même graphique :

```
plt.scatter(xs, ys)   plt.scatter(xs, zs)
```

- Pour afficher les courbes en ajoutant une légende, des axes et un titre à notre graphique :

```
plt.scatter(xs, ys, label="y=1/x")
plt.scatter(xs, zs, label="y=1/(25-x)$")
plt.xlabel("$x$")
plt.ylabel("Valeur")
plt.title("Mon premier \"scatter plot\" ")
plt.legend();
```

Question 1 :
Ecrire un programme qui permet de dessiner un "scatter plot" avec les points suivants :

```
xs = range(0,200,4)
ys = [(100 - x) ** 2 for x in xs]
```

Question 2 :
Ajouter une légende, des axes et un titre au graphique.

Correction

```
# Q1

import matplotlib.pyplot as plt
xs = range(0,200,4)
ys = [(100 - x) ** 2 for x in xs]
plt.scatter(xs, ys)


# Q2

plt.scatter(xs, ys, label="y=(100 - x)^2")
plt.xlabel("$x$")
plt.ylabel("Valeur")
plt.title(" Parabole\" ")
plt.legend();
plt.show()
```

Ex.2 — Plot

Nous venons de voir dans l'exercice 1 comment dessiner un nuage de point. On s'intéresse ici à dessiner des courbes.

La fonction de `matplotlib` permettant de représenter des courbes est `plot`. Comme la fonction `scatter`, elle prend comme paramètres les abscisses et les ordonnées des points de la courbe à dessiner.

Question 1 : Utiliser la fonction `range` pour générer une liste 'x contenant des valeurs entre 0 et 10.

Question 2 : Créer une liste `y` contenant les mêmes valeurs dans la liste `x` et une liste `z` contenant le carré des valeurs dans `x`.

Question 3 : Tracer les deux courbes `y` en fonction de `x` et `z` en fonction de `x`. De la même manière qu'avec la fonction `Scatter`, ajouter `lineaire en x` et `quadratique en x` comme légende pour chaque courbe.

Question 4 : Utilisant la fonction `scatter`, dessiner dans le même graphique les points permettant de représenter les courbes.

Correction

```
# Q1

x = range(0,10)


# Q2

y = x
z = [v*v for v in x]

# Q3

import matplotlib.pyplot as plt
plt.plot(x, y, label="lineaire en x")
plt.plot(x,z, label="quadratique en x")
plt.legend()



# Q4
```

```
plt.scatter(x, y)
plt.scatter(x, z)
plt.show()
```

Ex.3 — Hist

On sintéresse maintenant à la représentation graphique par histogramme. Un histogramme est un graphique permettant de représenter la répartition d'une variable continue en la représentant avec des colonnes verticales.

La fonction de `matplotlib` permettant de représenter un histogramme est `hist`. Comme pour les fonctions `scatter` et `plot`, la fonction `hist` se trouve dans le sous-module `pyplot` de la librairie `matplotlib`.

Pour représenter un histogramme, on utilise la commande suivante :

`hist(L,nombre)`

avec `L` une liste qui contient les données. C'est le seul argument obligatoire. Le paramètre 'nombre contient le nombre de barres de l'histogramme.

Question 1 : Les deux lignes de code suivants permettent de générer et afficher un nombre réel aléatoire dans l'intervalle [0,1).

```
from random import *
rd = random()
print(rd)
```

Utiliser la fonction `random` pour créer une liste de 100 éléments contenant des valeurs alétaoires.

Question 2 : Afficher l'histogramme de ces éléments. Utiliser plusieurs valeurs pour le nombre de barres de l'histogramme.

Correction

```
# Q1

from random import *
L = [random() for i in range(100)]


# Q2
```

```
import matplotlib.pyplot as plt
plt.hist(L,20)
plt.show()
plt.hist(L,5)
plt.show()
```

Partie 2 — Affichage et simulation d'un pendule

Ex.1 — Affichage d'un pendule

Le but de cette partie est d'afficher et simuler un pendule simple. En physique, le pendule est un système oscillant qui, écarté de sa position d'équilibre, y retourne en décrivant des oscillations, sous l'effet d'une force, par exemple le poids d'une masse.

Les librairies que nous allons utiliser sont : math (fonctions mathématiques), matplotlib (visualisation). Ci-dessous les lignes de code pour les importer :

```
import math
import matplotlib.pyplot as plt
```

Nous allons utiliser la fonction suivante pour afficher le pendule à une position donnée.

```
def penduleFromXY(x,y):
    plt.figure()
    plt.plot(x,y,'bo',markersize=20)
    plt.plot([0,x], [0,y])
    plt.xlim(-1.5,1.5)
    plt.ylim(-1.5,1.5)
    plt.show()
```

La fonction prend comme paramètre la position du pendule (x et y : coordonnées cartésiennes de la masse du pendule).

Question 1 :

Afficher le pendule avec la fonction `penduleFromXY` à la position `x=0` et `y=-1.`

Question 2 : On souhaite afficher le pendule suivant un angle d'inclinaison. La relation entre l'angle d'inclinaison `angle` et les coordonnées cartésiennes est donnée par les formules suivantes : $x = \sin(angle), \quad y = -cos(angle)$.

A titre d'exemple, les fonctions trigonométriques de python peuvent être appelées de la manière suiante : `math.sin(angle)` et `math.cos(angle)`.

Ecrire une fonction `pendule` qui prend en paramètre un angle et affiche le pendule à la position déterminée en fonction de cet angle. Pour cela utiliser la fonction `penduleFromXY`.

Question 3 :

Afficher le pendule à la position $\frac{\pi}{4}$.

Correction

```
import math
import matplotlib.pyplot as plt


def penduleFromXY(x,y):
    plt.figure()
    plt.plot(x,y,'bo',markersize=20)
    plt.plot([0,x], [0,y])
    plt.xlim(-1.5,1.5)
    plt.ylim(-1.5,1.5)
    plt.show()


pi = 3.14159

# Q1

penduleFromXY(0,-1)


# Q2

def pendule(angle):
    x = math.sin(angle)
    y = -math.cos(angle)
    penduleFromXY(x,y)



# Q3

pendule(pi/4)
```

Ex.2 — Simulation d'un pendule

Nous allons maintenant utiliser la fonction d'affichage du pendule de l'exercice 1 pour visualiser la trajectoire effectuée par le pendule pendant une simulation.

La trajectoire effectuée par le pendule dépend de la position (angle) et de la vitesse du pendule. Nous allons considérer une simulations suivant le modèle simple ci-dessous :

- $angle = angle + vitesse * intervalle\ de\ temps$
- $vitesse = vitesse + acc * intervalle\ de\ temps$

L'intervalle de temps est fixé à 0.1. Le paramètre `acc` est calculé à partir de la formule suivante :

- $acc = -1.5 * vitesse - 9.81 * sin(angle)$

La valeur de 9.81 dans l'équation ci-dessus représente l'accélération due à la pesanteur.

Question 1:

Ecrire une fonction `acc` qui prend comme paramètres un angle et une vitesse et qui retourne la valeur du pramètre `acc` calculée à partir de la l'équation ci-dessus.

Question 2:

Ecrire une fonction `nouvelAngle` qui prend comme paramètres une angle et une vitesse et qui retourne la nouvelle position (angle) du pendule suivant le modèle présenté ci-dessus.

Question 3:

Ecrire une fonction `nouvelleVitesse` qui prend comme paramètres un angle et une vitesse et qui retourne la nouvelle vitesse du pendule suivant le modèle présenté ci-dessus. La fonction fait appel à la fonction `acc`.

Question 4:

Ecrire une fonction `simulation` qui prend comme paramètres un angle initial, une vitesse initiale et un paramètre `nb`. La fonction retourne une liste de taille `nb` qui contient les angles successives après `nb` étapes.

Pour ceci utiliser les fonctions `nouvelAngle` et `nouvelleVitesse` dans une boucle afin de calculer à chaque fois la nouvelle position (angle) et la nouvelle vitesse du pendule.

Question 5:

Réaliser une simulation avec comme paramètres :

- angle initial = 0,

- vitesse initiale = 5,
- nb = 50.

Afficher à chaque étape de la simulation le pendule avec la fonction `pendule` de l'Exercice 1. La fonction `pendule` prend comme paramètre de manière successive les angles fournies par la simulation.

Correction

```
import math
import matplotlib.pyplot as plt


def penduleFromXY(x,y):
    plt.figure()
    plt.plot(x,y,'bo',markersize=20)
    plt.plot([0,x], [0,y])
    plt.xlim(-1.5,1.5)
    plt.ylim(-1.5,1.5)
    plt.show()


def pendule(angle):
    x = math.sin(angle)
    y = -math.cos(angle)
    penduleFromXY(x,y)




# Q1

def acc(angle,vitesse):
    return -1.5 * vitesse - 9.81 * math.sin(angle)



# Q2

def nouvelAngle(angle,vitesse):
    return angle + vitesse * 0.1
```

```
# Q3

def nouvelleVitesse(angle,vitesse):
    return vitesse + acc(angle, vitesse) * 0.1



# Q4

def simulation(angle,vitesse,nb):
    angles = [angle]
    for step in range(0,nb):
        angle = nouvelAngle(angle,vitesse)
        vitesse = nouvelleVitesse(angle, vitesse)
        angles.append(angle)
    return angles



# Q5

angles = simulation(0,5,50)
for angle in angles :
    pendule(angle)
```