Créer des graphiques avec Python : *le module matplotlib.pyplot*

IREM d'Aquitaine – Groupe Algorithmique

Planning de la journée

Matinée

- Présentation de matplotlib.pyplot (outils graphiques)
- Exercices graphiques, de base... ou pas

Après-midi

Menu à la carte :

- Retour sur les parcours de la 1re journée, ou de la matinée
- Listes : pour aller toujours plus loin...
- TP « Gestion de stock » (conception et mise en œuvre)
- Manipulation d'images

Matplotlib : qu'est-ce que c'est ?

Matplotlib est une bibliothèque Python (ensemble de fonctions prédéfinies) permettant de dessiner des graphiques en deux ou trois dimensions.

Il s'agit d'une bibliothèque très riche, qui permet d'enregistrer les graphiques générés sous différents formats (png, pdf...).

Dans le cadre du lycée, on se limitera aux graphiques en deux dimensions, et on utilisera le sous-module pyplot de matplotlib, que l'on importera de la façon suivante :

```
from matplotlib import pyplot as plt
```

ou encore :

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

plt = alias utilisé par convention...

Un premier exemple (simple) (1)

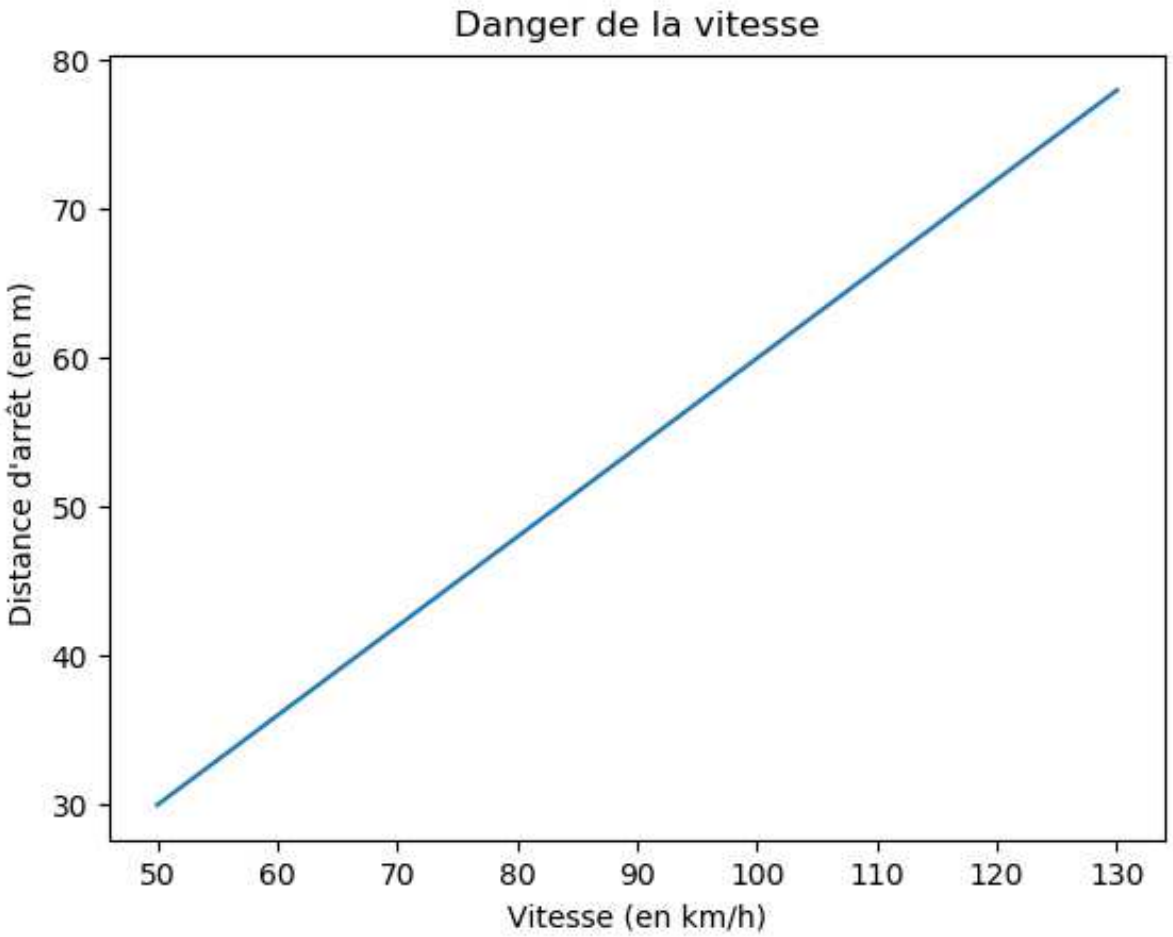
La fonction suivante dessine le graphique de la fonction estimant la distance d'arrêt d'un véhicule en fonction de sa vitesse.

```
import matplotlib.pyplot as plt

def graphique1 ():
    # Graphique de la fonction estimant la distance d'arrêt
    # d'un véhicule en fonction de sa vitesse.

    plt.title("Danger de la vitesse")          # titre du graphique
    plt.plot([50,130],[30,78])                 # points (liste des X, liste des Y)
    plt.xlabel('Vitesse (en km/h)')            # légende axe des abscisses
    plt.ylabel("Distance d'arrêt (en m) ")     # légende axe des ordonnées
    plt.show()                                 # dessin...
```

l'instruction show « suspend » l'exécution de la fonction... qui reprendra lorsque la fenêtre graphique sera refermée.

Un premier exemple (simple) (2)

Le graphique obtenu est le suivant :

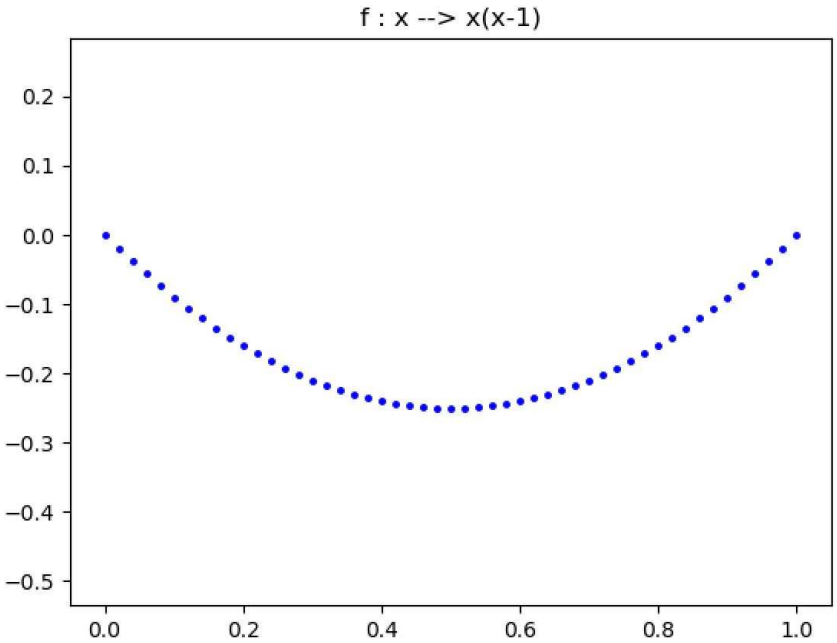


Tracé des points (1)

Les points composant le graphique peuvent être donnés un à un (tracé point à point), ou tous ensemble sous forme de deux listes (les points sont alors reliés) :

```
def f(x):
    return x * (x - 1)


def graphique2 (nbpoints):
    # Graphique de la fonction
    # f : x --> x(x-1) sur [0;1]
    # avec nbpoints + 1 points

    plt.axis("equal")
    plt.title("f : x --> x(x-1)")
    x = 0
    for i in range(nbpoints + 1):
        plt.plot(x, f(x), "b.", ms = 5)
        x = x + 1 / nbpoints
    plt.show()
```

repère normé



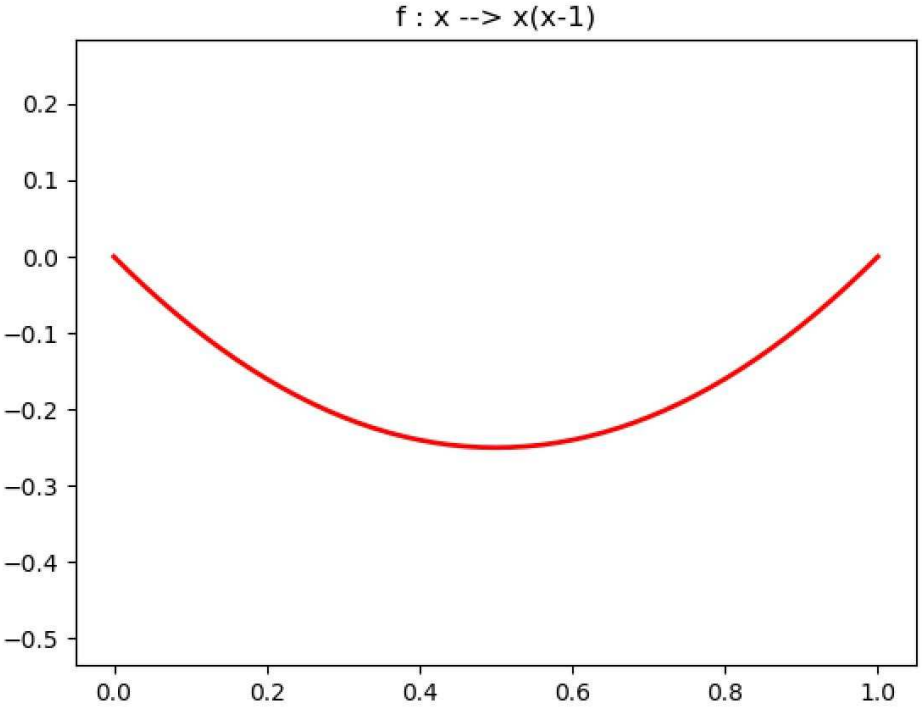
taille des points = 5 (marker size)

points (.) bleus (b)

Tracé des points (2)

Les points composant le graphique peuvent être donnés un à un (tracé point à point), ou tous ensemble sous forme de deux listes (les points sont alors reliés) :

```
def f(x):
    return x * (x - 1)


def graphique2bis (nbpoints):
    # Graphique de la fonction
    # f : x --> x(x-1) sur [0;1]
    # avec nbpoints + 1 points

    plt.axis("equal")
    plt.title("f : x --> x(x-1)")
    listeX = [x / nbpoints for x in range(nbpoints + 1)]
    listeY = [f(x) for x in listeX]
    plt.plot(listeX, listeY, "r-", lw = 2)
    plt.show()
```



épaisseur tracé = 2 (line width)

Modification des paramètres pris par défaut (1)

Il est naturellement possible de définir ses propres paramètres de dessin...

- **Limite des axes du graphique : axis ([xmin, xmax, ymin, ymax])**

```
plt.axis(xmin = 60, xmax = 120, ymin = 30, ymax = 120)
```

ou : `plt.axis([60, 120, 30, 120])`

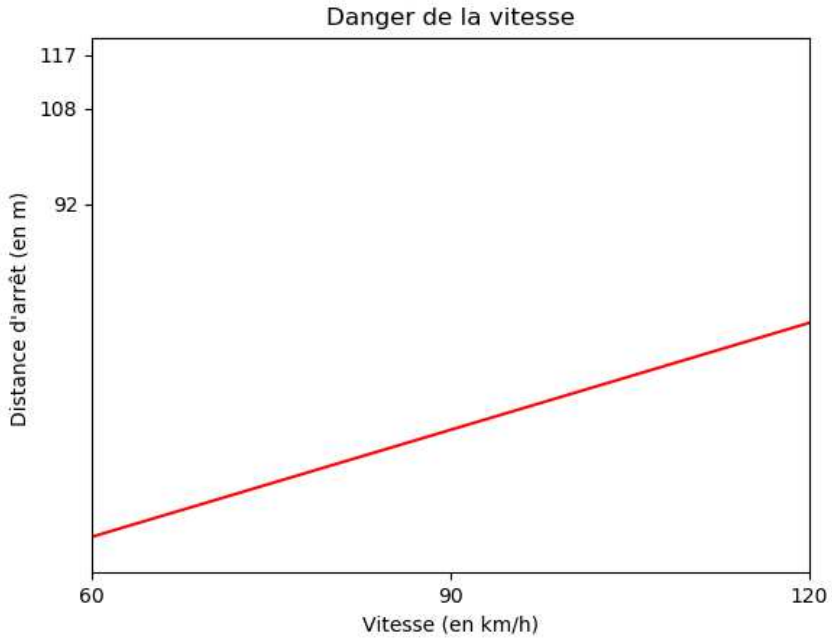
Seule la partie du graphique figurant dans cette zone sera affichée.

On peut aussi récupérer les limites du graphiques courant en écrivant :

```
rectangle = plt.axis()
```

- **Graduations : xticks (liste) et yticks (liste)**

```
plt.xticks([60, 90, 120])
plt.yticks([92, 108, 117])
```



Modification des paramètres pris par défaut (2)

- **Paramètres de la courbe : plot (...)**
  - **couleur du tracé (color) : b, g, r, c, m, y, k, w...**

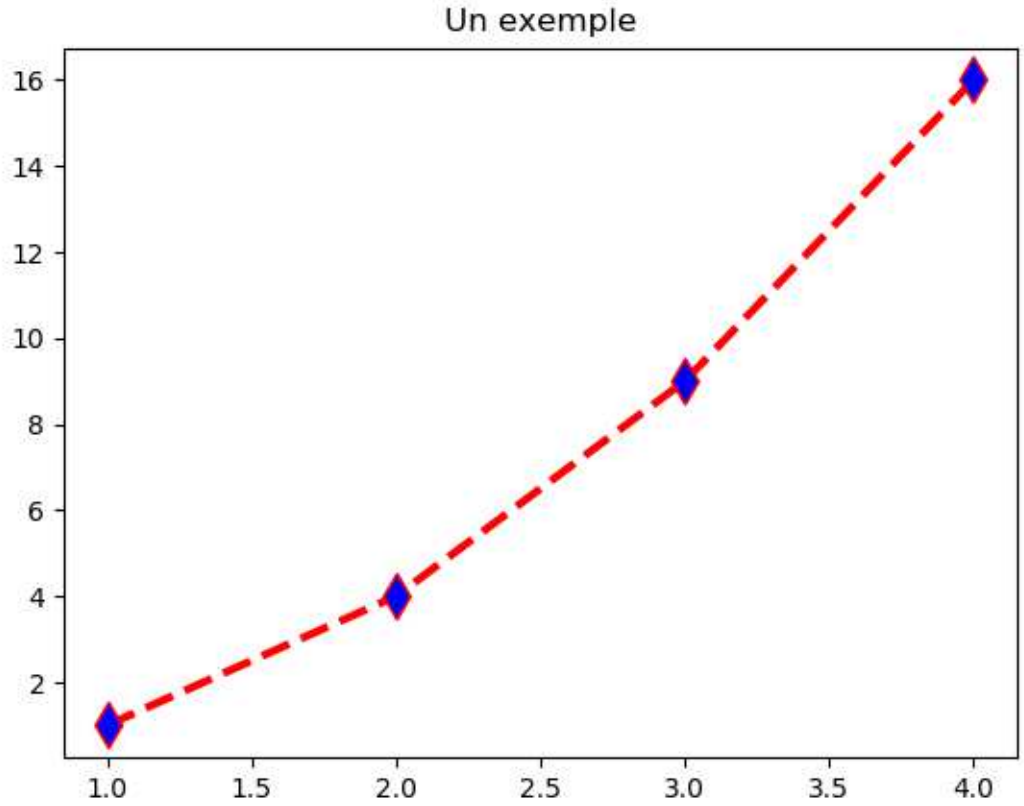
    b (bleu), g (vert), r (rouge), c (cyan), m (mauve), y (jaune), k (noir), w (blanc), ou triplet RGB (p. ex. (0.5, 0.1, 0.3))...
  - **style du tracé (linestyle) : -, --, -., :**
  - **épaisseur du tracé (linewidth ou lw) : nombre entier**
  - **forme des points (marker) : ., o, x, <, >, v, ^, *, s, p, h, d, +...**
  - **taille des points (markersize ou ms) : un entier**
  - **etc. (voir documentation sur matplotlib.org)**

Modification des paramètres pris par défaut (3)

```
plt.plot([1,2,3,4],[1,4,9,16],color='red',
         linestyle='dashed', linewidth=3,
         marker='d', markerfacecolor='blue',
         markersize=12)
```

Autre format d'écriture (moins lisible ?...) :

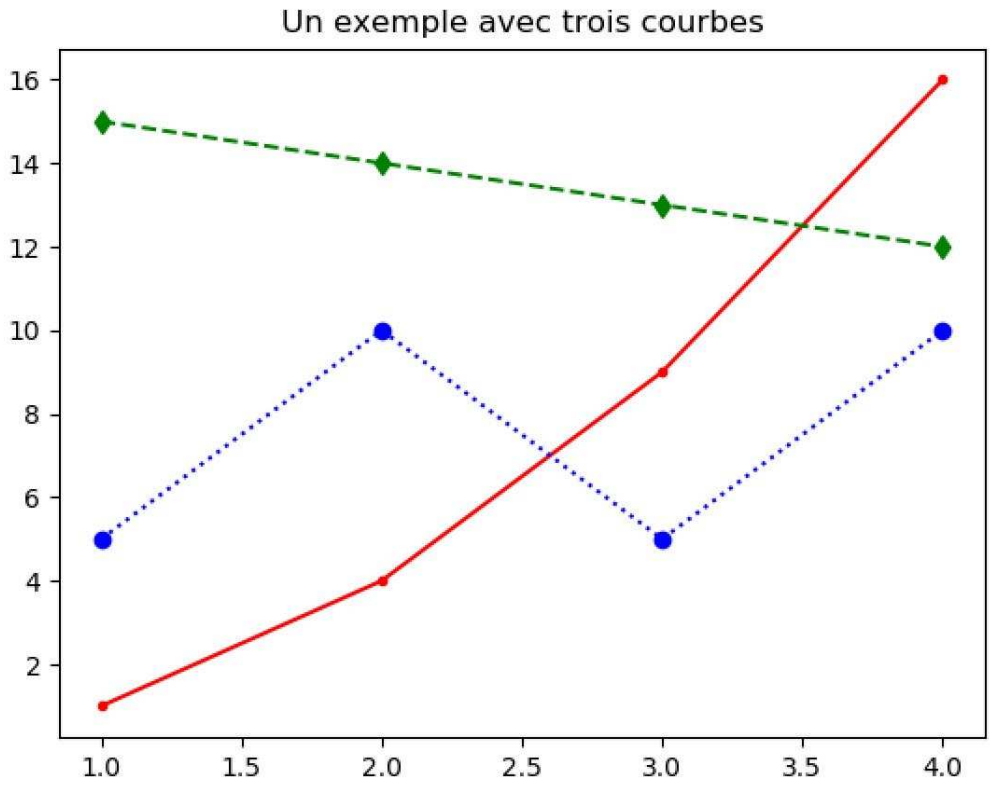
```
plt.plot([1,2,3,4],
         [1,4,9,16],
         "rd--",
         lw=3,
         mfc='b',
         ms=12)
```



Superposition de courbes

Pour cela, il suffit de placer plusieurs instructions plot avant l'instruction show...

```
plt.plot([1,2,3,4], [1,4,9,16], "r.-")
plt.plot([1,2,3,4], [5,10,5,10], "bo:")
plt.plot([1,2,3,4], [15,14,13,12], "gd--")
plt.show()
```

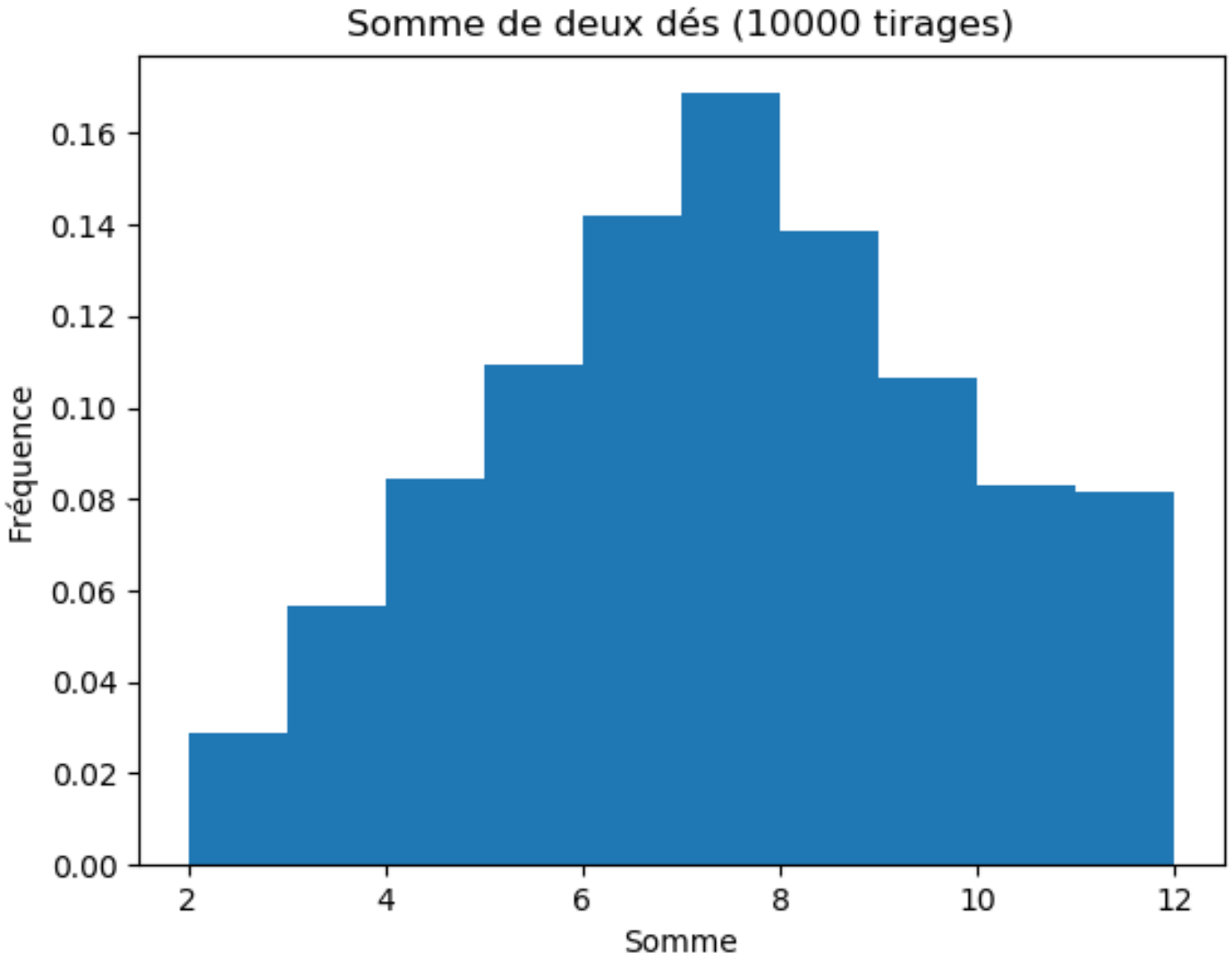


Histogrammes (1)

```
import matplotlib.pyplot as plt
import random

def tireSommesDeuxDés (n):
    # Cette fonction réalise n tirages de deux dés
    # et renvoie une liste donnant les sommes obtenues

    L = []
    for i in range(0,n):
        d1 = random.randint(1,6)
        d2 = random.randint(1,6)
        L.append(d1 + d2)
    return (L)

def histogrammeSommesDeuxDés (n):
    # Cette fonction réalise n tirages de deux dés
    # et affiche l'histogramme des sommes obtenues

    L = tireSommesDeuxDés (n)
    plt.title('Somme de deux dés (' + str(n) + ' tirages)')
    plt.xlabel('Somme')
    plt.ylabel('Fréquence')
    plt.hist(L, normed=1)
    plt.show()
```

l'histogramme est calculé... ici normalisé à 1 (nombreuses autres options disponibles, voir la documentation sur matplotlib.org)

Histogrammes (2)

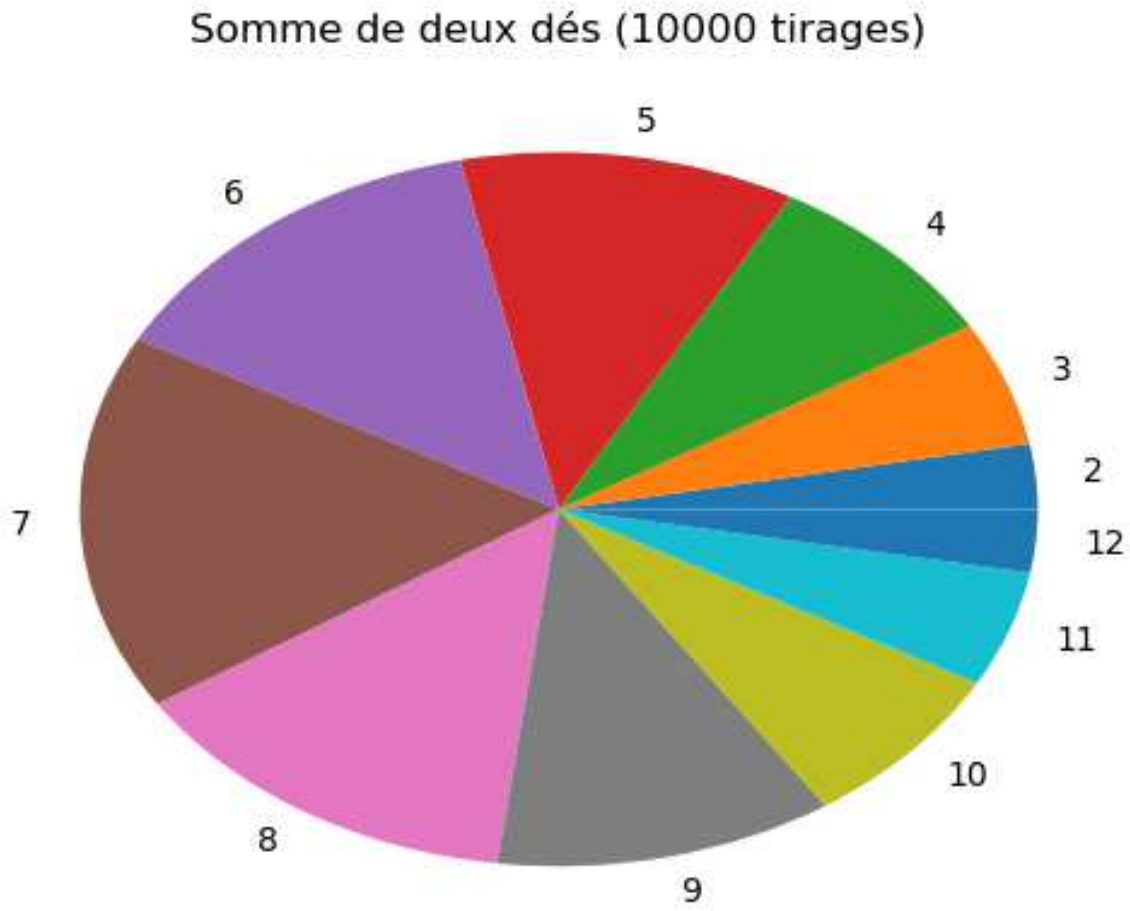


Camemberts (1)

```
import matplotlib.pyplot as plt
import random

def fréquenceSommesDeuxDés (n):
    # Cette fonction réalise n tirages de deux dés
    # et renvoie une liste donnant le nombre de fois où
    # chaque somme (de 2 à 12) est obtenue

    LF = 11*[0]
    for i in range(0,n):
        d1 = random.randint(1,6)
        d2 = random.randint(1,6)
        LF[d1 + d2 - 2] = LF[d1 + d2 - 2] + 1
    return (LF)

def camembertSommesDeuxDés (n):
    # Cette fonction réalise n tirages de deux dés
    # et affiche le camembert des fréquences des sommes
    # obtenues

    L = fréquenceSommesDeuxDés (n)
    sommes = [x for x in range(2,13)]
    plt.title('Somme de deux dés (' + str(n) + ' tirages)')
    plt.pie(L, labels=sommes)
    plt.show()
```

légende des « parts »

Camemberts (2)



Ne pas oublier...

Lien vers le questionnaire de satisfaction commun à tous les stages de mathématiques :

https://tinyurl.com/evastage