REPRÉSENTATION GRAPHIQUE

Matplotlib est probablement l'un des packages Python les plus utilisés pour la représentation de graphiques en 2D. Il fournit aussi bien un moyen rapide de visualiser des données grâce au langage Python, que des illustrations de grande qualité dans divers formats. Le **module pyplot** de **matplotlib** est l'un de ses principaux **modules**. Il regroupe un grand nombre de fonctions qui **servent** à créer des graphiques et les personnaliser (travailler sur les axes, le type de graphique, sa forme et même rajouter du texte).

I. Rappels : listes d'entiers.

La fonction **range()** génère une liste d'entiers.
range(n) : n premiers entiers de 0 à n – 1 inclus.
range(n, m) : entiers de n inclus à m exclu par pas de 1.
range(n, m, p) : entiers de n inclus à m exclu par pas de p avec p entier.

II. Tableau à une dimension (1D).

On utilise la bibliothèque **numpy**.

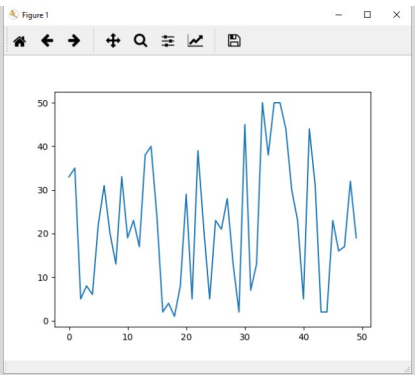
import numpy as np
np.array(liste) : convertit la liste en tableau .
np.linspace(a, b, n) : n valeurs régulièrement espacées de a à b inclus.
np.arrange(a, b, n) : de a inclus à b exclu par pas de p.

III. Création d'une courbe.

3.1 Utilisation de plot().

L'instruction plot() permet de tracer des courbes qui relient des points dont les abscisses et ordonnées sont fournies dans des tableaux.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias
from random import randint

plt.clf() #efface la figure
x = [i for i in range(50)]
y = [randint(1, 50) for i in range(50)]
plt.plot(x, y)

plt.show() # affiche la figure a l'écran
```



On peut construire x autrement :

```
x = [0, 1, 2, 3, …, 50] #fastidieux !
x = np.array([0, 1, 2, 3]) #fastidieux aussi !
x = range(50)
x = list(range(50))
x = [i for i in range(50)]
x = np.linspace(0, 50, 50)
x = np.arange(0, 50, 1)
```

IV. Fonctions utiles pyplot

plt.arrow(x, y, x + dx, y + dy, color = 'g', head_with = 1)	Trace une flèche verte (g) depuis le point de coordonnées (x, y) jusqu'au point de coordonnées (x + dx, y + dy)
plt.axis([xmin, xmax, ymin, ymax])	Fixe les dimensions des axes
plt.axis("equal")	Repère orthonormé
plt.clf()	Efface la figure
plt.grid()	Affichage de la grille
plt.legend()	Affichage de la légende
plt.plot(x, y)	Place et relie les points de coordonnées (x, y)
plt.plot(x, y, label = *str*)	Place et relie les points de coordonnées (x, y) Insère la légende de la courbe. Pour faire afficher le label, il ne faut pas oublier de faire appel à l'instruction plt.legend()
plt.plot(x, y, "--r", label = *str*, linewidth = *int*)	Place et relie les points de coordonnées (x, y) avec un trait pointillé rouge de largeur *int* Insère la légende de la courbe
plt.savefig("matplotlib.png")	Sauvegarde du graphique au format png par défaut.
plt.scatter(x, y, linewidth = *int*)	
plt.show()	Affiche la figure
plt.text(x, y, str)	Affichage du texte aux coordonnées (x, y)
plt.title(*str*)	Ajout d'un titre
plt.xlabel(*str*)	Affiche la légende pour l'axe des abscisses
plt.ylabel(*str*)	Affiche la légende pour l'axe des ordonnées
plt.xlim(xmin, xmax)	Fixe le domaine des abscisses
plt.ylim(xmin, xmax)	Fixe le domaine des ordonnées

Remarques :

- La liste de toutes les fonctions du modules pyplot est accessible de la façon suivante dans la console :

```
import matplotlib.pyplot as plt
dir(matplotlib)
```

- En ajoutant un r devant une chaîne de caractères, on peut afficher des formules mathématiques pour p eu que l'on connaisse un minimum la syntaxe LATEX. Exemple :
- Lorsqu'on exécute le script, l'affichage se réalise dans une fenêtre indépendante qu'il faut fermer après l'avoir visualisée.

Couleur.

Chaîne	Effet
b	Bleu (blue)
g	Vert (green)
r	Rouge (red)
c	Cyan (cyan)
m	Magenta (magenta)
y	Jaune (yellow)
k	Noir (black)
w	Blanc (white)

Style de ligne.

Chaîne	Effet
-	Trait plein
--	Pointillé long
:	Pointillé court
-.	Pointillé mixte

Si on ne veut pas faire apparaître de ligne, il suffit d'indiquer un symbole sans préciser un style de ligne.

Symbole ("marker")

Chaîne	Marqueur
.	Gros points
,	Pixels
o	Cercles
v	Triangle vers le bas
^	Triangle vers le haut
<	Triangle vers la gauche
>	Triangle vers la droite
1	Tripodes vers le bas
2	Tripodes vers le haut
3	Tripodes vers la gauche
4	Tripodes vers la droite

Chaîne	Marqueur
s	Carrés
p	Pentagones
*	Étoiles
h	Hexagone version 1
+	Signes +
x	Signes ×
X	Signes X
D	Diamants carrés
d	Losanges
| (AltGr + 6)	Traits verticaux
_ (AltGr + 8)	Traits horizontaux

Largeur de ligne : linewidth= <x>

Ne pas hésiter à se rendre sur https://matplotlib.org pour avoir la liste des fonctions et aussi de nombreux exemples de graphiques.

TRACÉ DE COURBES : TESTS

I. Listes des valeurs d'une fonction.

On se donne une fonction polynomiale : $P(x)=2x^2-3x-2$.

1°) Écrire une fonction liste_P qui prend en argument une liste de nombre réels x été qui retourne la liste des valeurs $P(x)$.

Exemple :

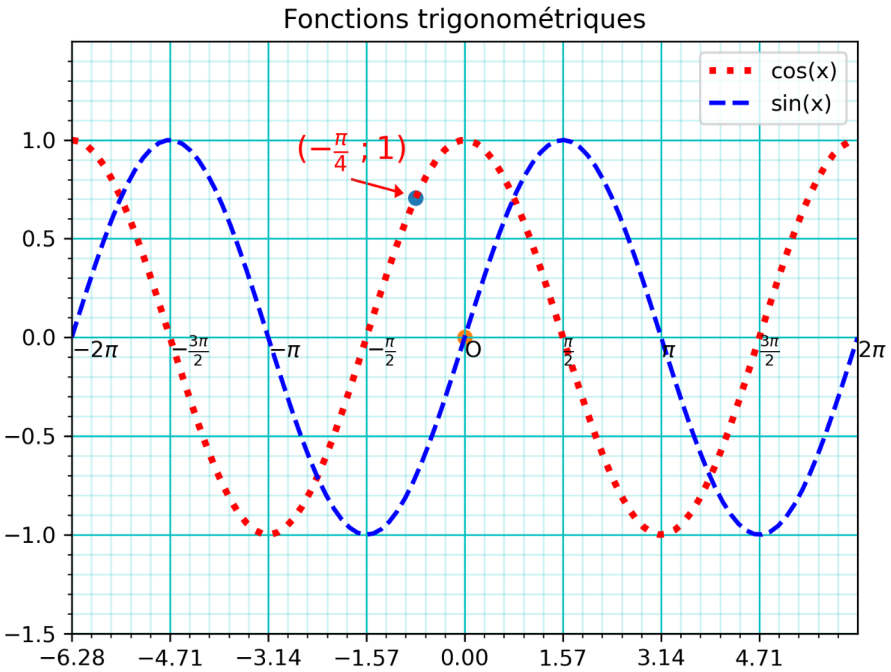
En entrée : liste_P([0, 1, 2, 3, 4]).

En sortie : [-2, -3, 0, 7, 18].

2°) Tracer la courbe représentant la fonction polynôme sur l'intervalle $[-10\ ;\ 10]$.

II. Fonctions trigonométriques.

À l'aide du module matplotlib, vous devez reproduire la figure ci-contre du mieux que vous le pouvez.



CORRECTION TRACÉ DE COURBES : TESTS

I. Listes des valeurs d'une fonction.

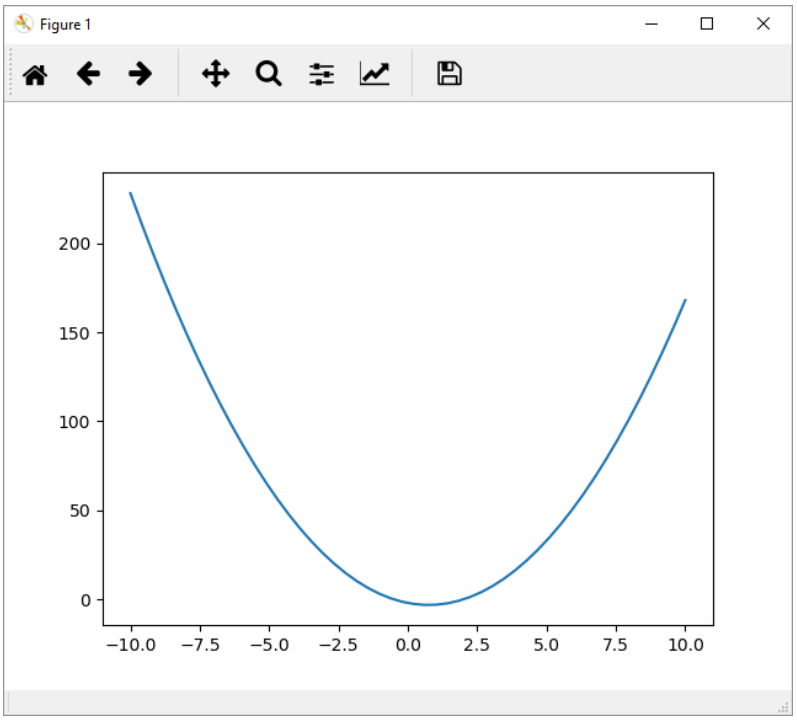
1°)

```
def P(x):
    return 2*x**2-3*x-2

def liste_P(liste_x):
    return [P(x) for x in liste_x]

print(liste_P([0, 1, 2, 3, 4]))
```

```
>>>
[-2, -3, 0, 7, 18]
>>>
```

2°)

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

def P(x):
    return 2*x**2-3*x-2

def liste_P(liste_x):
    return [P(x) for x in liste_x]

x = np.linspace(-10, 10) # valeurs de x entre -10 et 10
y = liste_P(x)  # et les y correspondants

plt.plot(x, y)
plt.show()
```



II. Fonctions trigonométriques.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias
from math import pi, sqrt

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1) #identique à fig.add_subplot(111)

x_min, x_max = -2*pi, 2*pi
y_min, y_max = -1.5, 1.5

plt.title("Fonctions trigonométriques")
plt.xlim(x_min, x_max) #domaine pour l'axe des abscisses
plt.ylim(y_min, y_max) #domaine pour l'axe des ordonnées

x = np.linspace(x_min, x_max, 100)
y1 = np.cos(x)
y2 = np.sin(x)
plt.plot(x, y1, ":r", label="cos(x)", linewidth=3)
plt.plot(x, y2, "--b", label="sin(x)", linewidth=2)
plt.legend() #affichage de la légende

y_pos = -0.1
plt.text(-2 * pi, y_pos, r'${-2\pi}$')
plt.text(-3 * pi/2, y_pos, r'$-\frac{3\pi}{2}$')
plt.text(-pi, y_pos, r'$-\pi$')
plt.text(-pi / 2, y_pos, r'$-\frac{\pi}{2}$')
plt.text(pi / 2, y_pos, r'$\frac{\pi}{2}$')
plt.text(pi, y_pos, r'$\pi$')
plt.text(3 * pi / 2, y_pos, r'$\frac{3\pi}{2}$')
plt.text(2 * pi, y_pos, r'$2\pi$')

#tracé du point A de coordonnées (-pi/4 ; rac(2)/2 )
plt.scatter(-pi/4,sqrt(2)/2) #tracé d'un point
#affichage du texte en indiquant les coordonnées
plt.text(-2.7, 0.9, r'$(-\frac{\pi}{4}\; ; 1)$', fontsize = 14, color='r')
#affichage de la flêche en indiquant les coordonnées
plt.arrow(-1.8, 0.8, 0.7, -0.06, head_width = 0.05, head_length = 0.1, color = 'r')
#tracé de l'origine du repère
plt.scatter(0,0) #tracé d'un point
plt.text(0, y_pos, r'O') #affichage du texte en indiquant les coordonnées

#Réglage des échelles pour les abscisses
grid_x_ticks_minor = np.arange(x_min, x_max, pi/10)
grid_x_ticks_major = np.arange(x_min, x_max, pi/2)
ax.set_xticks(grid_x_ticks_minor, minor=True)
ax.set_xticks(grid_x_ticks_major)

#Réglage des échelles pour les ordonnées
grid_y_ticks_minor = [i / 10 for i in range(-15, 15, 1)]
grid_y_ticks_major = [i / 10 for i in range(-15, 15, 5)]
ax.set_yticks(grid_y_ticks_minor, minor=True)
ax.set_yticks(grid_y_ticks_major)
```

```
ax.grid(which='both', linestyle='-', color='c')
ax.grid(which='minor', alpha=0.2)

#plt.grid() #affichage de la grille

#sauvegarde du graphique
plt.savefig("Fonctions_trigo.png", dpi = 300)
#affichage du graphique
plt.show() # affiche la figure à l'écran
```

Remarques : fig.add_subplot(1, 1, 1)

subplot(* args , ** kwargs)
subplot(nrows, ncols, index, **kwargs)

- Soit un entier à 3 chiffres, soit trois entiers séparés décrivant la position du sous-tracé. Si les trois entiers sont nrows , ncols et index dans l'ordre, le sous-tracé prendra la position d' index sur une grille avec nrows lignes et ncols colonnes. l'index commence à 1 dans le coin supérieur gauche et augmente vers la droite.

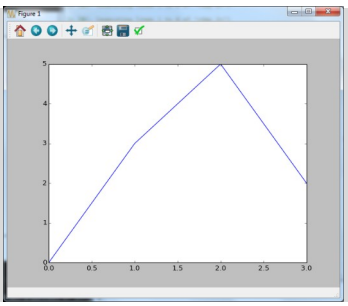
- pos est un entier à trois chiffres, où le premier chiffre est le nombre de lignes, le second le nombre de colonnes et le troisième l'indice du sous-tracé. c'est-à-dire fig.add_subplot (235) est le même que fig.add_subplot (2, 3, 5). Notez que tous les entiers doivent être inférieurs à 10 pour que ce formulaire fonctionne.

III. Exemples supplémentaires : compléments.

3.1 Utilisation de plot()

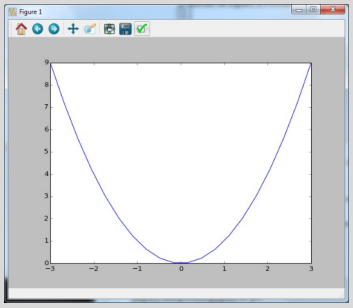
Pour tracer des courbes, Python n'est pas suffisant et nous avons besoin des bibliothèques numpy et matplotlib.
L'instruction plot() permet de tracer des courbes qui relient des points dont les abscisses et ordonnées sont fournies dans des tableaux.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

plt.clf() #efface la figure
x = np.array([0, 1, 2, 3])
y = np.array([0, 3, 5, 2])
plt.plot(x, y) #place et relie les points sur l'écran
plt.show() #affiche la figure à l'écran
```



Tracé de la fonction carré.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

plt.clf() #efface la figure
x = np.linspace(-3, 3, 20) #(x_min, x_max, nb_valeurs)
y = x**2
plt.axis([-3, 3, 0, 9]) #détermine les dimensions des axes
plt.plot(x, y) #place et relie les points sur l'écran
plt.show() #affiche la figure à l'écran
```

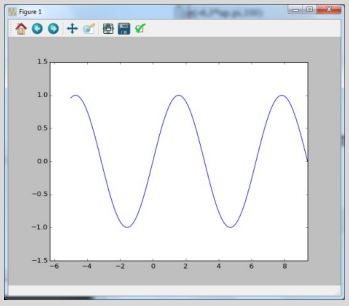


3.2. Définition du domaine des axes avec xlim() et ylim().

Il est possible fixer indépendamment les domaines des abscisses et des ordonnées en utilisant les fonctions xlim() et ylim().

Tracé de la fonction sinus.

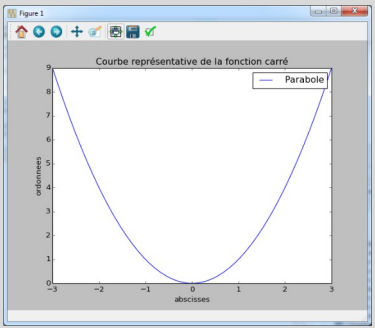
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

plt.clf() #efface la figure
x = np.linspace(-5, 3*np.pi, 100)
y = np.sin(x)
plt.plot(x, y) #place et relie les points sur l'écran
plt.xlim(-2*np.pi, 3*np.pi)
plt.ylim(-1.5, 1.5)
plt.show() # affiche la figure à l'écran
```



3.3 Ajout d'un titre, de légendes et de labels sur les axes.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

plt.clf() #efface la figure
```

```
plt.title("Courbe représentative de la fonction carré") #ajout d'un titre

x = np.linspace(-3,3,50)
y = x**2
plt.plot(x,y,label = "Parabole")
plt.legend() #affichage de la légende

plt.xlabel("abscisses") #label pour l'axe des abscisses
plt.ylabel("ordonnees") #"label pour l'axe des ordonnées
plt.show() #affiche la figure a l'écran
```

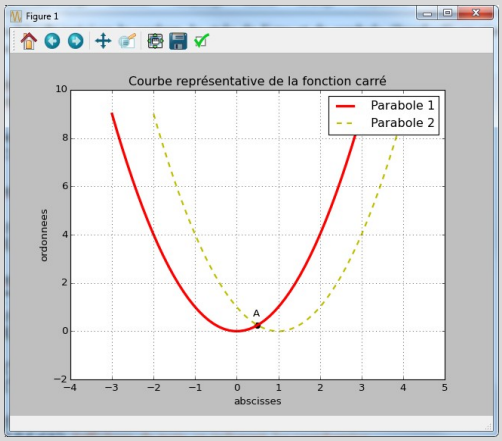


Pour faire afficher le label, il ne faut pas oublier de faire appel à l'instruction `legend()`

3.4 Formats de courbes.

Il est possible de préciser la couleur, le style de ligne et de symbole ("marker") en ajoutant une chaîne de caractères. Voir l'annexe pour les différentes possibilités.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

plt.clf() #efface la figure
plt.title("Courbe représentative de la fonction carré") #ajout d'un titre

x = np.linspace(-3, 3, 50)
y = x**2
plt.plot(x, y, "-r", label=" Parabole 1", linewidth=3)
plt.legend() #affichage de la légende

x = np.linspace(-2, 4, 50)
y = (x-1)**2
plt.plot(x, y, "--y", label = " Parabole 2", linewidth = 2)
plt.legend() #affichage de la légende

plt.xlabel("abscisses") #label pour l'axe des abscisses
plt.ylabel("ordonnees") #"label pour l'axe des ordonnées

#tracé d'un point
plt.text(0.4, 0.6, r'A') #affichage du texte en indiquant les coordonnées
plt.grid() #affichage de la grille
plt.show() #affiche la figure à l'écran
```

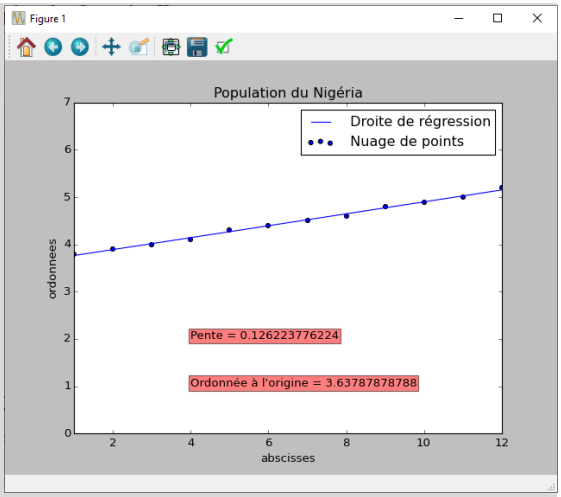


3.5 Touches.

G ou g : affiche ou cache la grille sur le graphique.
Ctrl – F : plein écran.

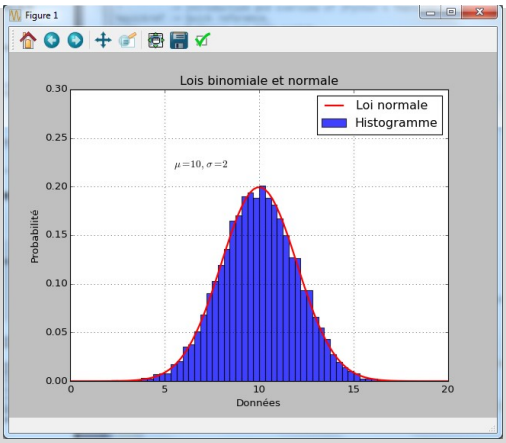
3.6 Nuage de points.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias
plt.clf() #efface la figure
plt.title("Population du Nigéria") #ajout d'un titre

plt.xlim(1, 12)
plt.ylim(0, 7)
x = [i for i in range(1, 13)]
y = [3.8, 3.9, 4, 4.1, 4.3, 4.4, 4.5, 4.6, 4.8, 4.9, 5, 5.2]
plt.scatter(x, y, label = " Nuage de points")
plt.legend() #affichage de la légende
plt.xlabel("abscisses") #label pour l'axe des abscisses
plt.ylabel("ordonnees") #"label pour l'axe des ordonnées

#Régression polynomiale de degré 1
a,b = np.polyfit(x, y, 1)
xd = [x[0], x[-1]]
yd = [a * x[0] + b, a * x[-1] + b]
plt.plot(xd, yd,label = " Droite de régression") #place et relie les points sur l'écran
plt.text(4, 2, "Pente = "+str(a), bbox=dict(facecolor='red', alpha=0.5))
plt.text(4, 1, "Ordonnée à l'origine = "+str(b), bbox=dict(facecolor='red', alpha=0.5))
plt.legend() #affichage de la légende
plt.show() #affiche la figure a l'écran
```



3.7. Histogramme.

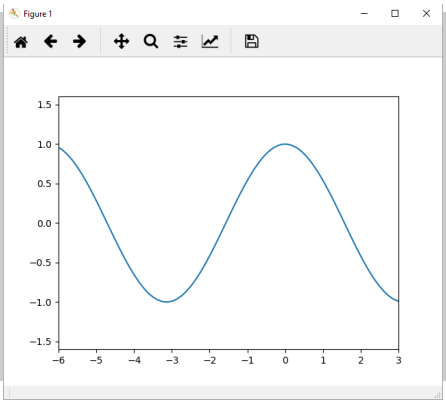
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias
from math import *
plt.clf() #efface la figure

#Tracé de la loi normale
mu, sigma = 10, 2
x = np.linspace(0, 20, 100) #100 valeurs de 0 à 20
y = np.exp(-0.5*((x-mu)/sigma)**2)/(sigma*sqrt(2*pi))
plt.plot(x, y, 'r',label="Loi normale", linewidth=2)
plt.legend() #affichage de la légende

#Tracé de l'histogramme
s = np.random.normal(mu, sigma, 10000)
n, bins, patches = plt.hist(s, bins = 100, density = True, facecolor = 'b', alpha = 0.75, label =
"Histogramme")
plt.legend() #affichage de la légende
plt.title('Lois binomiale et normale') #affichage du titre
plt.xlabel('Données') #affichage légende en abscisse
plt.ylabel('Probabilité') #affichage légende en ordonnée
plt.text(5.5, .22, r'$\mu=10,\ \sigma=2$') #affichage texte dans le graphique
plt.axis([0,20,0,0.3]) #réglage des échelles
plt.grid(True) #affichage de la grille

plt.show()
```



3.8 Définition du domaine des axes - xlim() et ylim()

Il est possible fixer indépendamment les domaines des abscisses et des ordonnées en utilisant les fonctions xlim() et ylim().

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y = np.cos(x)
plt.plot(x, y)
plt.xlim(-6, 3)                 # xlim(xmin, xmax)
plt.ylim(-1.6, 1.6)             "ylim(ymin, ymax)

plt.show() # affiche la figure à l'écran
```



3.9 Ajout d'un titre - title().

On peut ajouter un titre grâce à l'instruction title().

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

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y = np.cos(x)
plt.plot(x, y)
plt.title("Fonction cosinus")

plt.show() # affiche la figure à l'écran
```

3.10 Ajout d'une légende - legend().

On peut ajouter une légende grâce à l'instruction legend().

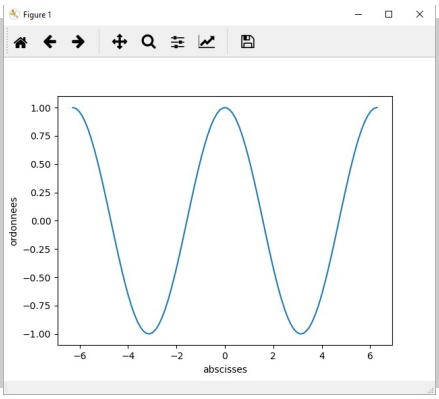
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y = np.cos(x)
plt.plot(x, y, label="cos(x)")
plt.legend() #affichage de la légende

plt.show() # affiche la figure à l'écran
```

Remarque : pour faire afficher le label, il ne faut pas oublier de faire appel à l'instruction legend().

3.11 Labels sur les axes - xlabel() et ylabel()

Des labels sur les axes peuvent être ajoutés avec les fonctions xlabel() et ylabel().

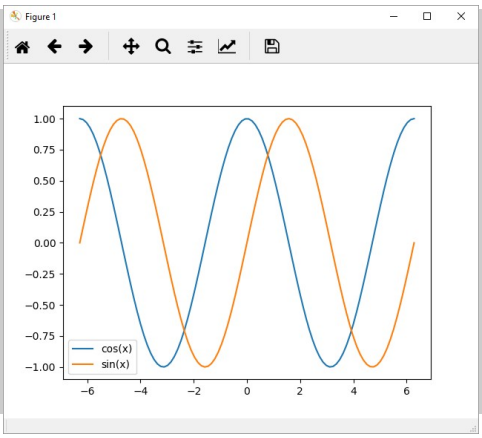
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y = np.cos(x)
plt.plot(x, y)
plt.xlabel("abscisses") #label pour l'axe des abscisses
plt.ylabel("ordonnees") #label pour l'axe des ordonnées

plt.show() # affiche la figure à l'écran
```



3.12 Affichage de plusieurs courbes

Pour afficher plusieurs courbes sur un même graphe, on peut procéder de la façon suivante :

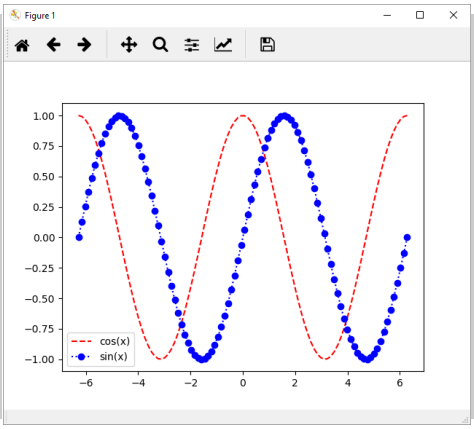
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y1 = np.cos(x)
y2 = np.sin(x)
plt.plot(x, y1, label="cos(x)")
plt.plot(x, y2, label="sin(x)")
plt.legend() #affichage de la légende

plt.show() # affiche la figure à l'écran
```



3.13 Formats de courbes

Il est possible de préciser la couleur, le style de ligne et de symbole (« marker ») en ajoutant une chaîne de caractères de la façon suivante :

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y1 = np.cos(x)
y2 = np.sin(x)
plt.plot(x, y1, "r--", label="cos(x)")
plt.plot(x, y2, "b:o", label="sin(x)")
plt.legend() #affichage de la légende

plt.show() # affiche la figure à l'écran
```

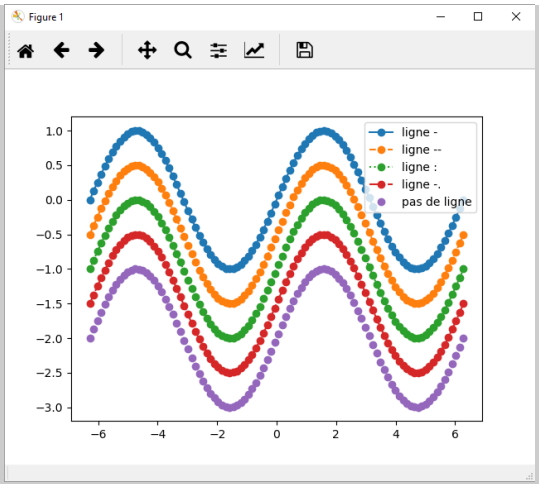


3.14 Style de ligne.

Les chaînes de caractères fournis en annexe permettent :

- de définir le style de ligne ;
- le symbole "marker" ;
- la couleur.

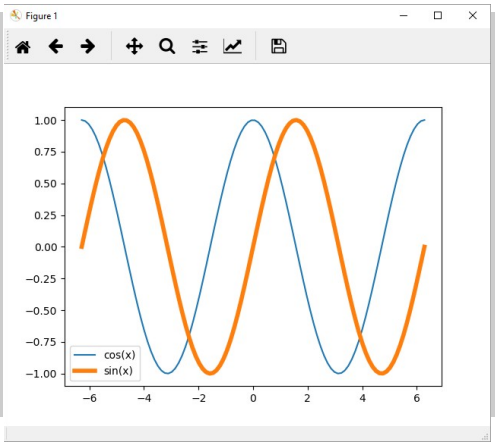
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y, "o-", label="ligne -")
plt.plot(x, y-0.5, "o--", label="ligne --")
plt.plot(x, y-1, "o:", label="ligne :")
plt.plot(x, y-1.5, "o-.", label="ligne -.")
plt.plot(x, y-2, "o", label="pas de ligne")
plt.legend() #affichage de la légende

plt.show() # affiche la figure à l'écran
```



3.15 Largeur de ligne.

Pour modifier la largeur des lignes, il est possible de changer la valeur de l'argument linewidth.

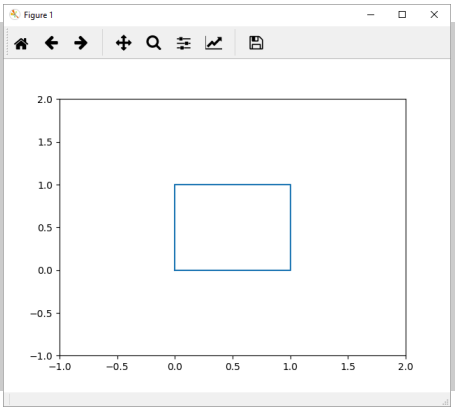
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y1 = np.cos(x)
y2 = np.sin(x)
plt.plot(x, y1, label="cos(x)")
plt.plot(x, y2, label="sin(x)", linewidth=4)
plt.legend() #affichage de la légende

plt.show() # affiche la figure à l'écran
```



3.16 Tracé de formes.

Comme la fonction plot() ne fait que relier des points, il est possible de lui fournir plusieurs points avec la même abscisse.

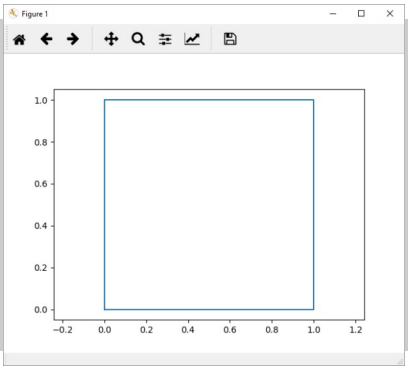
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.array([0, 1, 1, 0, 0])
y = np.array([0, 0, 1, 1, 0])
plt.plot(x, y)
plt.xlim(-1, 2)
plt.ylim(-1, 2)

plt.show() # affiche la figure à l'écran
```



3.17 L'instruction axis("equal")

L'instruction axis("equal") permet d'avoir la même échelle sur l'axe des abscisses et l'axe des ordonnées afin de préserver la forme lors de l'affichage. En particulier, grâce à cette commande un carré apparaît vraiment comme un carré, de même pour un cercle.

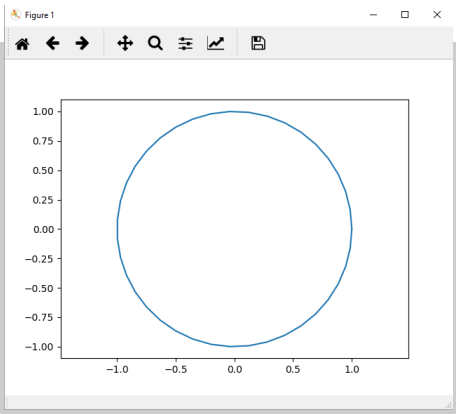
```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

x = np.array([0, 1, 1, 0, 0])
y = np.array([0, 0, 1, 1, 0])
plt.plot(x, y)
plt.axis("equal")

plt.show() # affiche la figure à l'écran
```



3.18 Tracé d'un cercle.

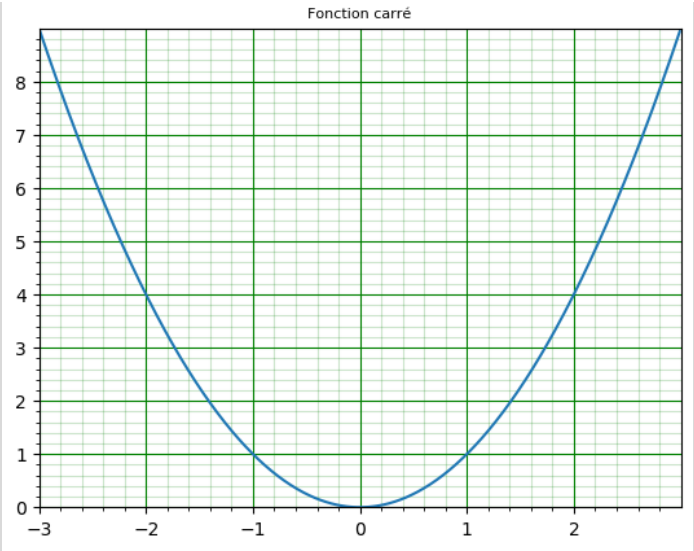
On peut tracer utiliser une courbe paramétrique pour tracer un cercle.
Sans axis("equal"), la courbe n'apparaît pas comme un cercle.

```
import numpy as np #création d'un alias
import matplotlib.pyplot as plt #création d'un alias

theta = np.linspace(0, 2*np.pi, 40)

x = np.cos(theta)
y = np.sin(theta)
plt.plot(x, y)
plt.axis("equal")

plt.show() # affiche la figure à l'écran
```



3.19 Fonction carré avec papier millimétré.

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

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

x_min, x_max = -3, 3
y_min, y_max = 0, 9
plt.xlim(x_min,x_max)
plt.ylim(y_min,y_max)

x = np.arange(x_min, x_max, .01)
y = x**2
plt.plot(x,y)

grid_x_ticks_minor = np.arange(x_min, x_max, 0.2)
grid_x_ticks_major = np.arange(x_min, x_max, 1)

ax.set_xticks(grid_x_ticks_minor, minor=True)
ax.set_xticks(grid_x_ticks_major)
```



```
grid_y_ticks_minor = np.arange(y_min, y_max, 0.2)
grid_y_ticks_major = np.arange(y_min, y_max, 1)
ax.set_yticks(grid_y_ticks_minor, minor=True)
ax.set_yticks(grid_y_ticks_major)

ax.grid(which='both', linestyle='-', color='g')
ax.grid(which='minor', alpha=0.2)

plt.title('Fonction carré', fontsize=8)

plt.savefig("Fonction carré.png", bbox_inches='tight')

plt.show()
```