TP N°1 D'INFORMATIQUE

Graphiques sous Python

Anicet E. T. Ebou, ediman.ebou@inphb.ci

Introduction

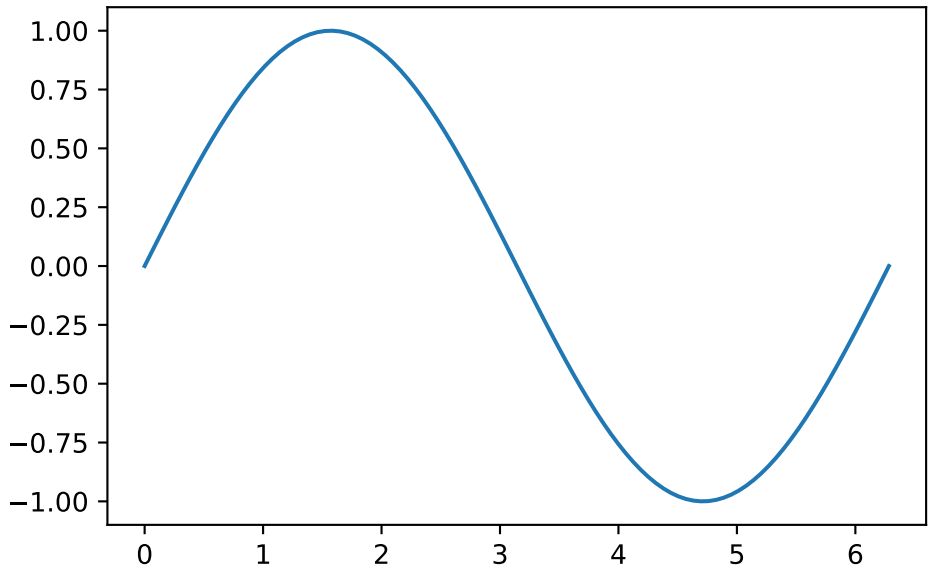
Le module pour réaliser des graphiques sous Python est `matplotlib`. Ce module permet de tracer:

- Des courbes simples;
- Des courbes multiples;
- Une série de points;
- Des histogrammes;
- Des camemberts;
- Une représentation par réseau de couleurs;
- Des courbes en 3D.

Courbes

Exemple minimal

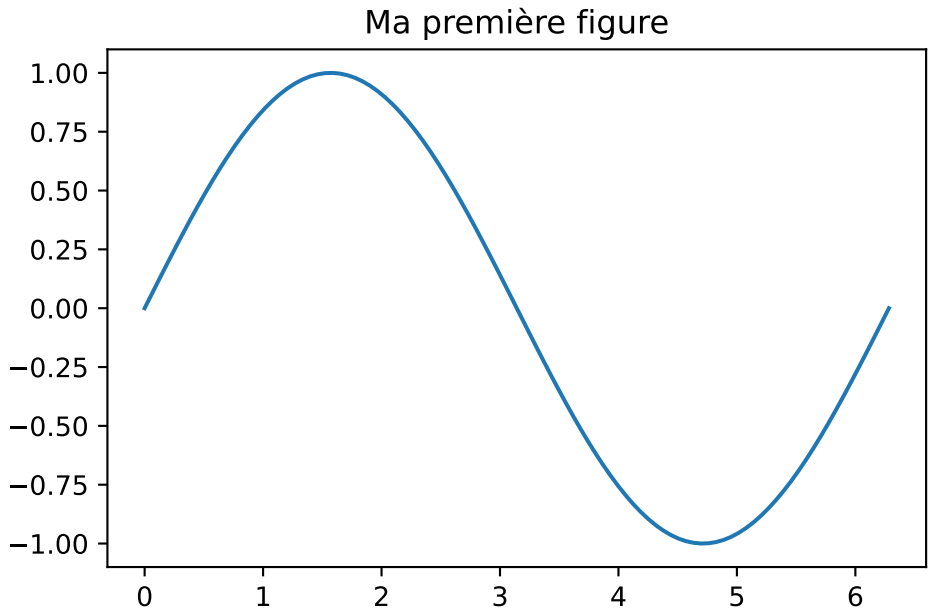
```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)  # Ajout d'une courbe
plt.show()  # Affichage d'une courbe
```



Ajout d'un titre au graphique

Pour ajouter un titre au graphique, on utilise la fonction `plt.title("...")`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)  # Ajout du d'une courbe

# Titre du graphique
plt.title("Ma première figure")

# Affichage d'une courbe
plt.show()
```



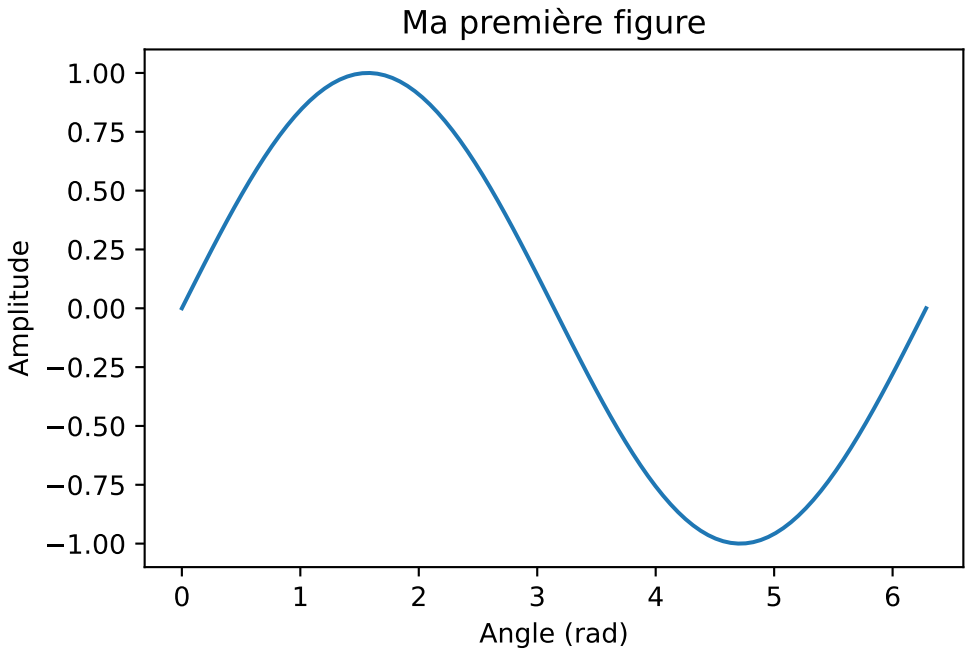
Exercice 2: Ajouter un titre au graphique précédent.

Ajout des titres des axes

Pour ajouter un titre sur les axes, on utilise les fonctions `plt.xlabel("...")` et `plt.ylabel("...")`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
```

```
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)  # Ajout du d'une courbe

# Titre du graphique
plt.title("Ma première figure")

plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x

plt.show()  # Affichage d'une courbe
```

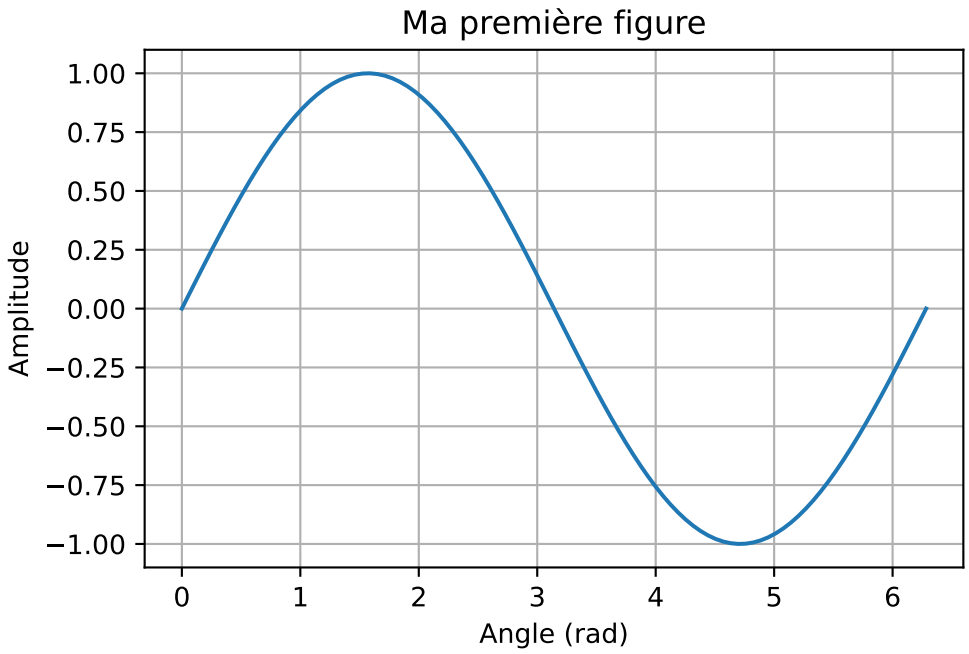


Affichage de la grille

Pour afficher la grille, on utilise la fonction `plt.grid(True)`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)  # Ajout du d'une courbe
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique

plt.grid(True)  # Affichage de la grille
```

```
plt.show()  # Affichage d'une courbe
```

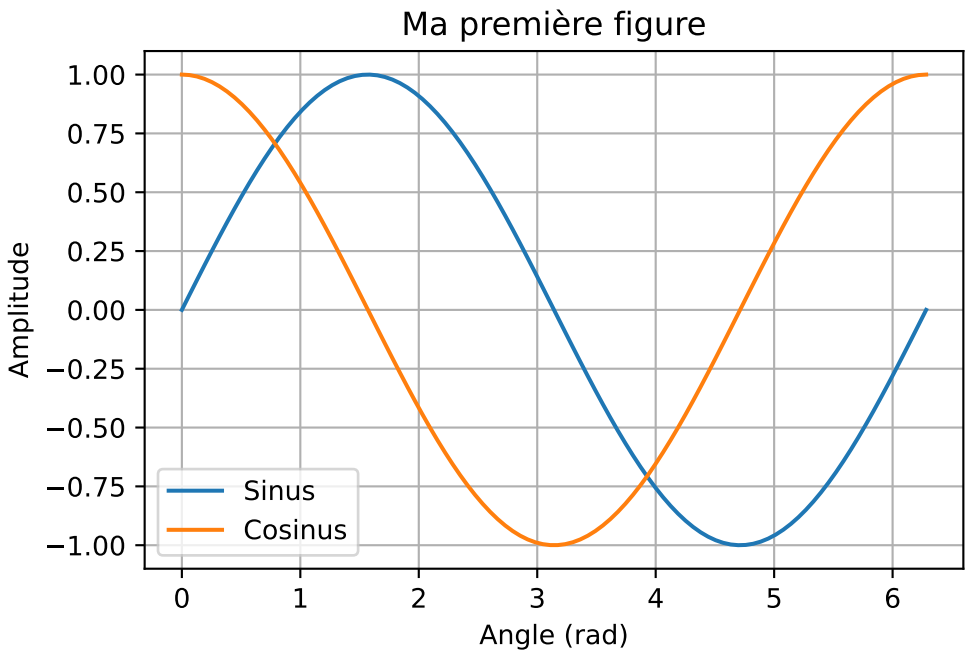


Ajout d'une deuxième courbe et d'une légende

Pour que la légende puisse s'afficher, il faut donner un label à chaque courbe avec le paramètre `label="..."` dans la fonction `plt.plot(...)`.

Par défaut, la légende se place automaqitue au mieux sur le graphique.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
y2 = np.cos(x)

# Ajout du d'une courbe avec label
plt.plot(x, y, label="Sinus")

# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.legend()  # Affichage de la légende

plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille
plt.show()  # Affichage d'une courbe
```

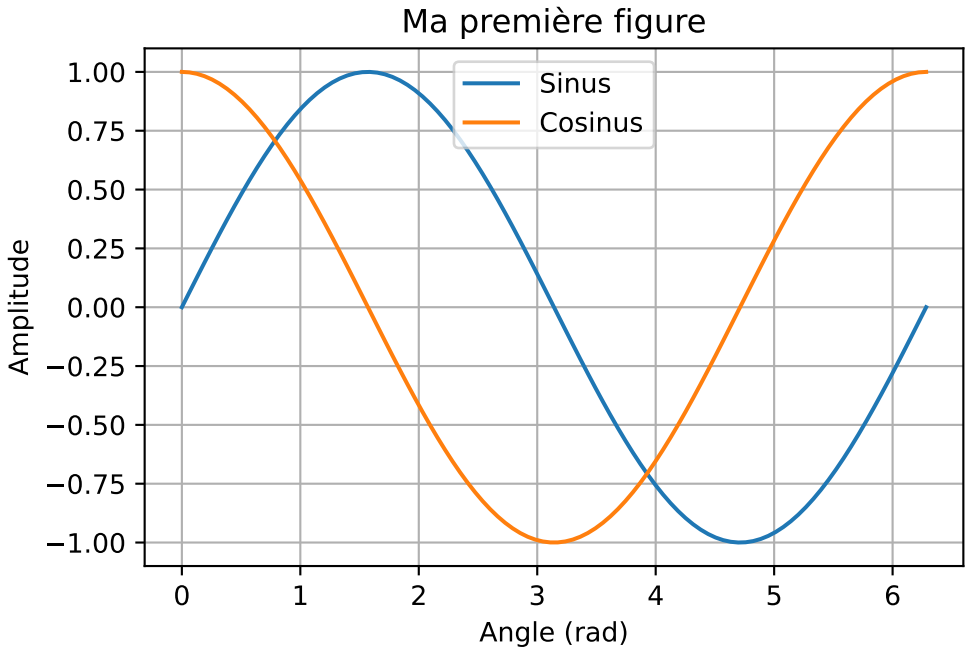


On peut modifier l'emplacement de la légende avec le paramètre `loc=x`, `x` étant la position voulue. Les différentes positions sont données dans le tableau ci-dessous.

Position	Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
y2 = np.cos(x)

# Ajout du d'une courbe avec label
plt.plot(x, y, label="Sinus")

# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.legend(loc=9)  # Affichage de la légende
```

```
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille
plt.show()  # Affichage d'une courbe
```

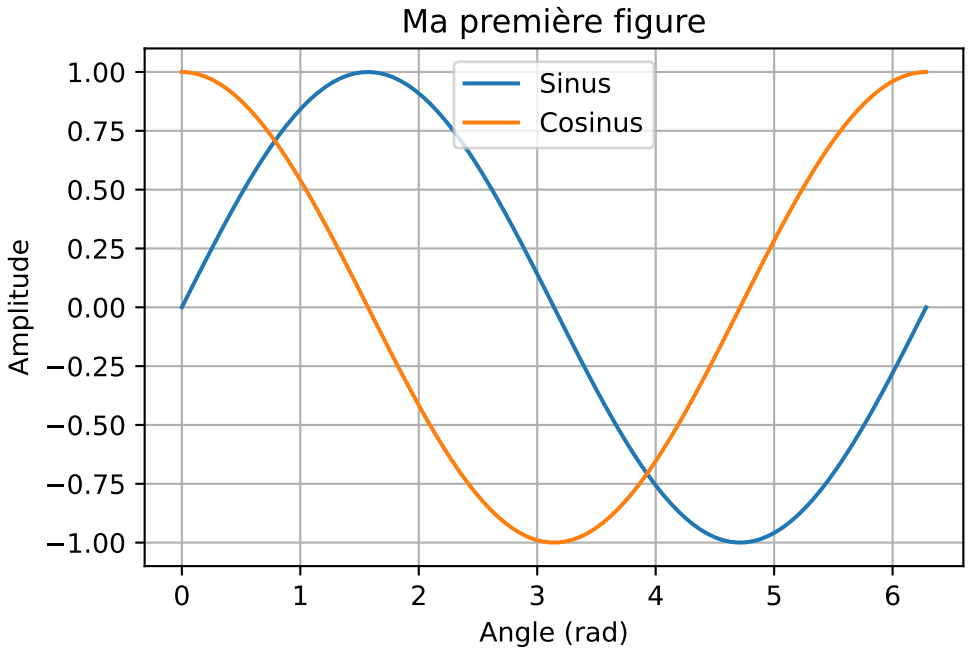


Sauvegarder le graphique

Il est possible de sauvegarder le graphique sous différents formats (pdf, png, jpeg ...) avec la commande `plt.sqvefig('Nom.pdf')`. L'extension du fichier détermine automatiquement son format.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
y2 = np.cos(x)

# Ajout du d'une courbe avec label
plt.plot(x, y, label="Sinus")

# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.legend(loc=9)  # Affichage de la légende
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
```

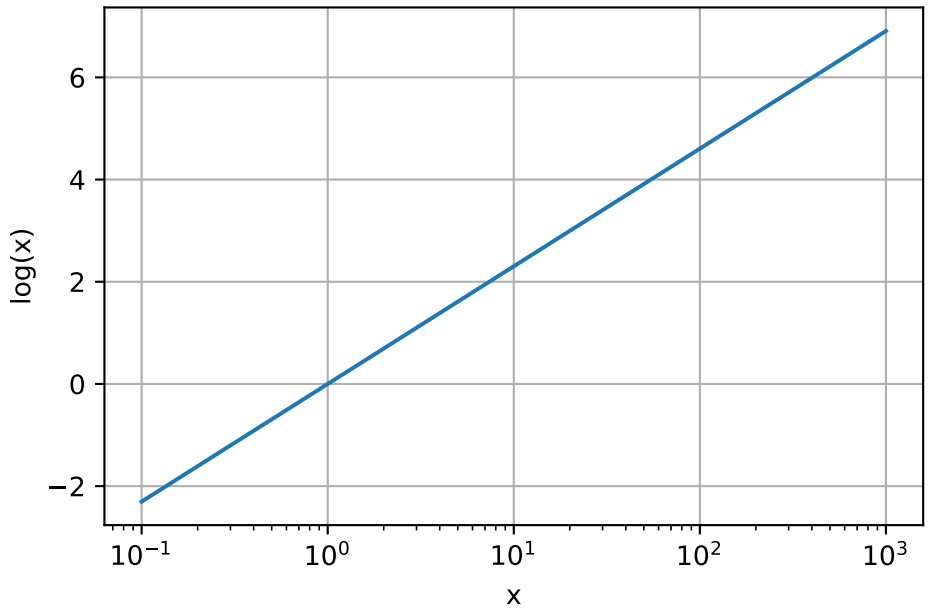
```
# Titre du graphique
plt.title("Ma première figure")
plt.grid(True)  # Affichage de la grille

# Sauvegarde en png avec un dpi de 300
plt.savefig("Graphique.png", dpi=300)

plt.show()  # Affichage d'une courbe
```



Echelle semi-logarithmique et logarithmique

On peut tracer une courbe avec une échelle semi-logarithmique en utilisant la fonction `plt.semilogx(x,y)`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0.1, 1000, 100)
y = np.log(x)

plt.semilogx(x, y)  # Ajout du d'une courbe
plt.ylabel('log(x)')  # Titre de l'axe y
plt.xlabel("x")  # Titre de l'axe x
plt.grid(True)  # Affichage de la grille
plt.show()  # Affichage d'une courbe
```

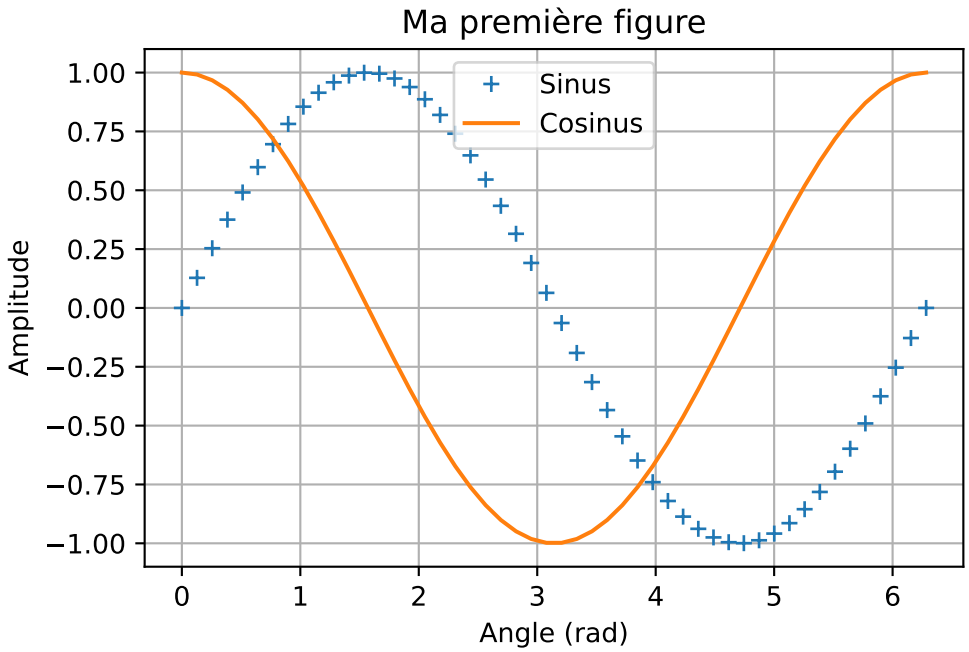


Affichage des points seuls

On peut afficher uniquement les points sans les relier ensemble en rajoutant le paramètre 'o' dans la fonction `plt.plot(...)`.

La liste des marqueurs possibles est disponible à l'adresse suivante: Marqueurs.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
y2 = np.cos(x)

# Ajout du des points uniquement
plt.plot(x, y, '+', label="Sinus")

# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.legend(loc=9)  # Affichage de la légende
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille
plt.show()  # Affichage d'une courbe
```



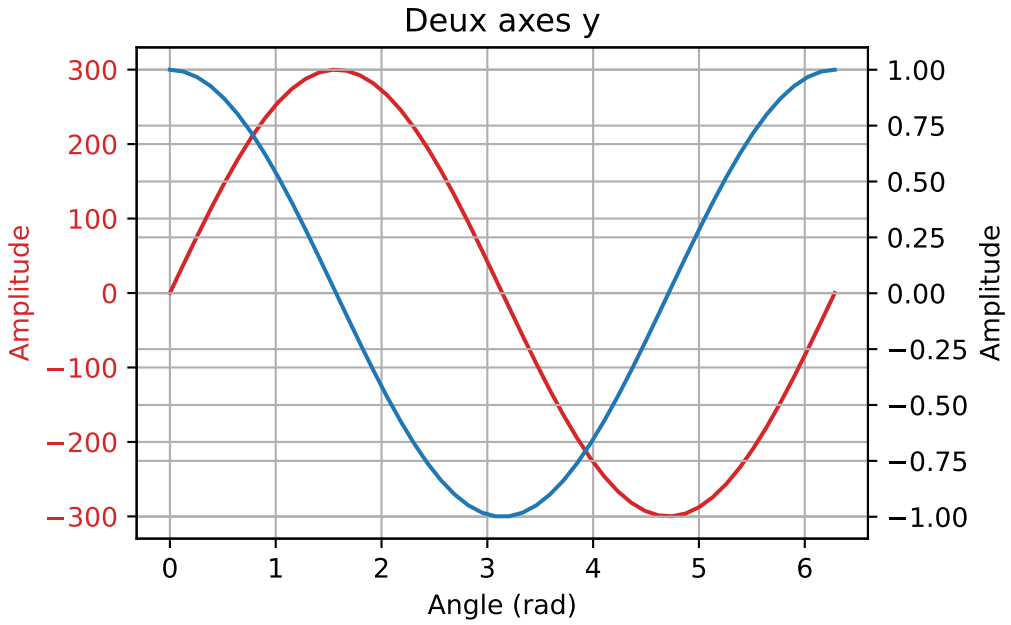
Axe secondaire

Si les données à afficher sont de dimensions différentes ou d'unités différentes, il est possible de construire un axe secondaire pour différencier ces données.

Création de deux axes y.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
y = 300*np.sin(x)
y2 = np.cos(x)

# Affichage des données de l'axe principal
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('Angle (rad)')
ax1.set_ylabel('Amplitude', color=color)
ax1.plot(x, y, color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax1.grid(True)

# Affichage de l'axe secondaire
# Création d'un deuxième axe qui partage le même axe x
ax2 = ax1.twinx()

color = 'tab:blue'
ax2.set_ylabel('Amplitude')#, color=color)  # Labelde l'axe x
```

```
ax2.plot(x, y2, color=color)
#ax2.tick_params(axis='y', labelcolor=color)
ax2.grid(True)


plt.title("Deux axes y")
fig.tight_layout()  # Amélioration de l'affichage
plt.show()
```

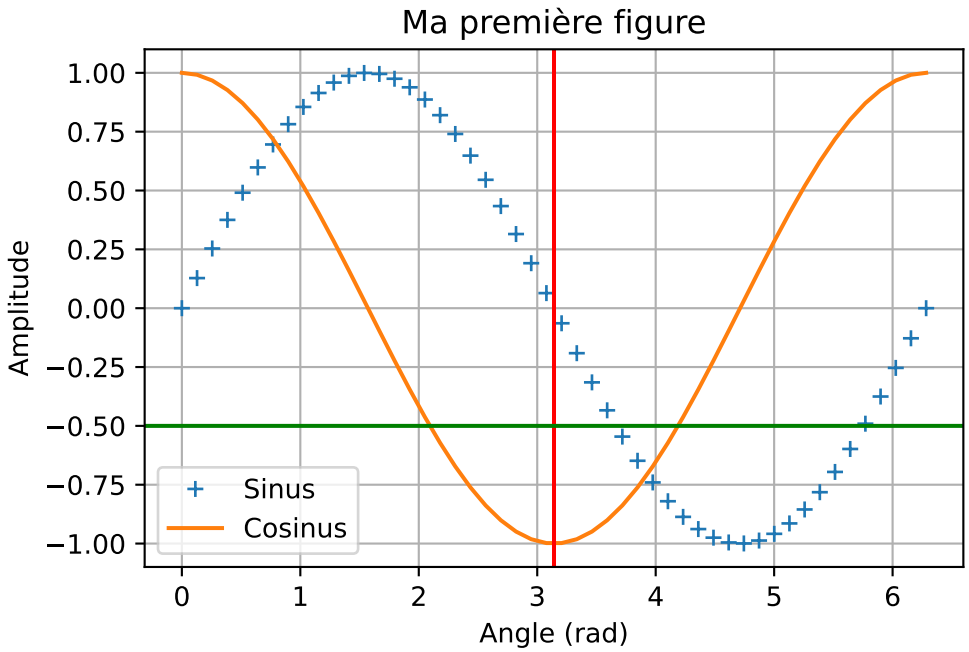


Barres verticales et horizontales

Il est parfois nécessaire d'afficher une barre verticale ou horizontale. Pour cela on utilise les fonctions `plt.axvline(x=0, ymin=0, ymax=1)` et `plt.axhline(y=0, xmin=0, xmax=1)`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
y2 = np.cos(x)

# Ajout du des points uniquement
plt.plot(x, y, '+', label="Sinus")

# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.legend()  # Affichage de la légende
plt.ylabel('Amplitude')  # Titre de l'axe y
```

```
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

# Barre verticale
plt.axvline(x=np.pi, ymin=0, ymax=1, color="red")

# Barre horizontale
plt.axhline(y=-0.5, xmin=0, xmax=1, color="green")

plt.show()  # Affichage d'une courbe
```



Sous figure ou afficher plusieurs graphiques sur la même figure

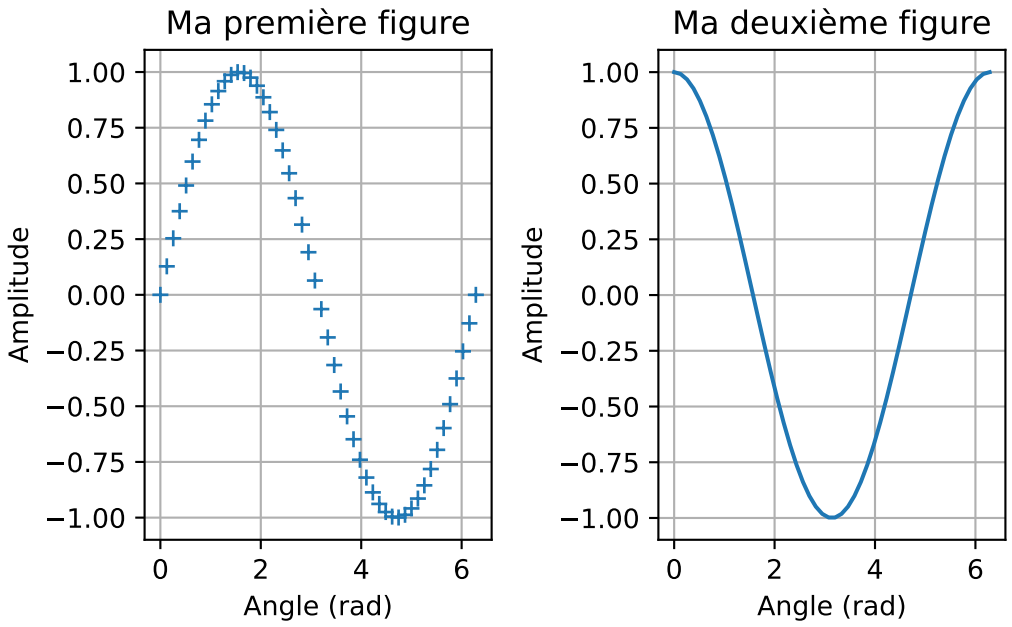
Il est possible d'afficher plusieurs graphiques côte à côte ou superposés sur la même figure grâce à la fonction `plt.subplot(lcp)`.

Avec comme paramètre 3 chiffres compris entre 1 et 9 pour définir une matrice de graphique avec `l` le nombre de lignes, `c` le nombre de colonnes et `p` l'emplacement où placer le graphique.

Pour éviter les chevauchements des graphiques, on utilise en plus la fonction `plt.tight_layout()`.

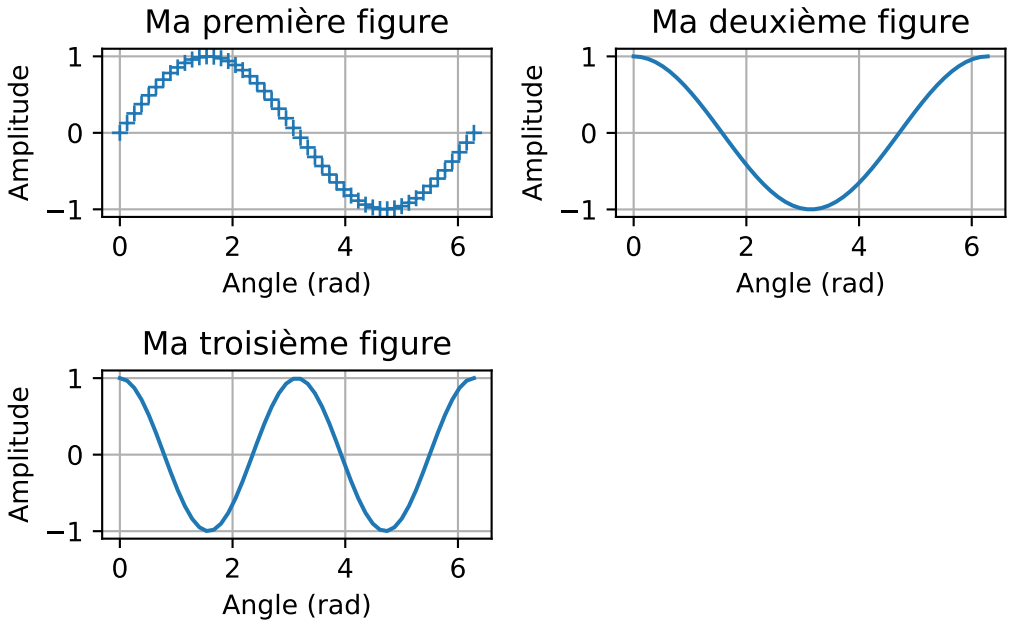
```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
y2 = np.cos(x)
```

```
plt.subplot(121) # Graphique côte à côte

 # Ajout du des points uniquement
plt.plot(x, y, '+', label="Sinus")

plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

plt.subplot(122) # Graphique superposé

# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma deuxième figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

plt.tight_layout() # Adaptation de l'affichage
plt.show()  # Affichage d'une courbe
```




```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
y2 = np.cos(x)
y3 = np.cos(2*x)
```

```
plt.subplot(221) # Graphique matrice 2x2

# Ajout du des points uniquement
plt.plot(x, y, '+', label="Sinus")
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

plt.subplot(222) # Graphique matrice 2x2
# Ajout d'une deuxième courbe avec label
plt.plot(x, y2, label='Cosinus')

plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma deuxième figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

plt.subplot(223) # Graphique matrice 2x2
# Ajout d'une troisème courbe avec label
plt.plot(x, y3, label='Cosinus')
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma troisième figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

plt.tight_layout() # Adaptation de l'affichage
plt.show()  # Affichage d'une courbe
```



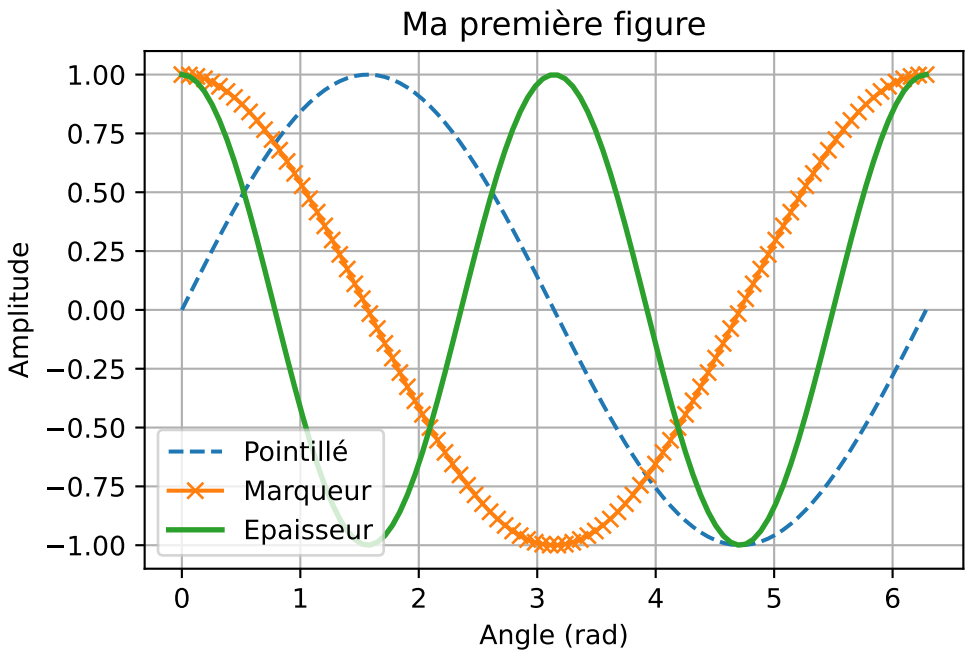
Personnalisation des courbes et autres options

Options d'affichage des lignes

Il est possible de modifier l'affichage des courbes en jouant sur les paramètres suivants (à inclure dans `plt.plot(...)`):

- `lw=1`: épaisseur de ligne (par défaut: 1);
- `ls='-'`: style de ligne -> ':', '-', '–' (par défaut: '-' ligne continue);
- `marker='.'`: type de marqueur pour les points -> '.', 'o', 'x', '+' ... (par défaut aucun), une liste des marqueurs possibles est disponible à l'adresse suivante: Marqueurs.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
y2 = np.cos(x)
y3 = np.cos(2*x)

# Ajout du d'une en pointillé
plt.plot(x, y, ls='--', label="Pointillé")

# Ajout d'une courbe avec un 'x' comme marqueur
plt.plot(x, y2, marker='x', label='Marqueur')
# Ajout d'une courbe d'épaisseur 2
plt.plot(x, y3, lw='2', label='Epaisseur')

plt.legend()  # Affichage de la légende
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

# Sauvegarde en png avec un dpi de 300
plt.savefig("Graphique.png", dpi=300)

plt.show()  # Affichage d'une courbe
```



Couleur des lignes

On peut changer la couleur des lignes à l'aide du paramètre `color='red'`.

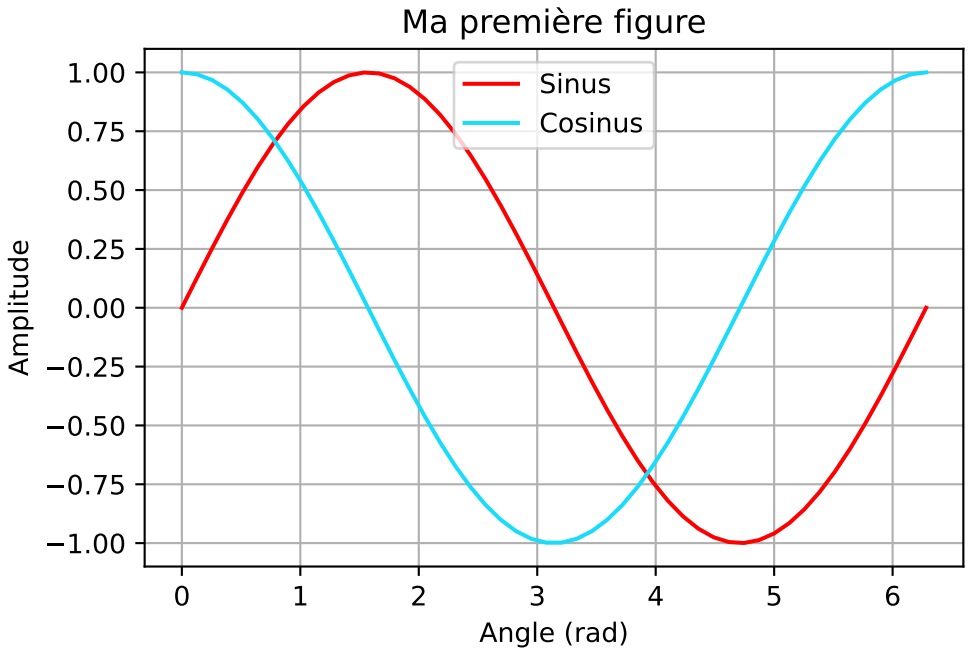
Liste de couleurs disponibles:

Alias	Couleur
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

Il est aussi possible de donnée le code hexadécimal de la couleur `color='#18DDFA'`. Pour trouver le code hexadécimal d'une couleur vous pouvez utiliser le site suivant: Couleurs hexa.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)
y2 = np.cos(x)

# Définition de la couleur par le nm
plt.plot(x, y, label="Sinus", color='red')
# Définition de la couleur par le code hexa
```

```
plt.plot(x, y2, label='Cosinus', color='#18DDFA')

plt.legend(loc=9)  # Affichage de la légende
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille
plt.show()  # Affichage d'une courbe
```

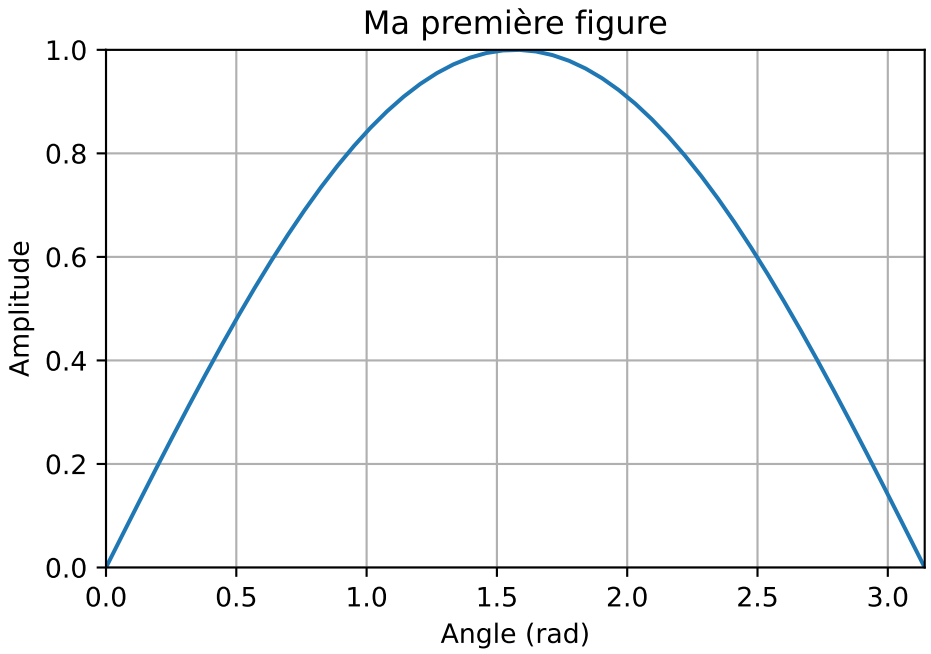


Bornes de l'affichage

Il est possible de réduire les bornes des axes abscisses et des ordonnées avec les fonctions `plt.xlim(min, max)` et `plt.ylim(min, max)`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)  # Ajout du d'une courbe
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

plt.ylim(0, 1)  # On limites l'axe des ordonnées
plt.xlim(0, np.pi)  # on limite l'axe des abscisses
```

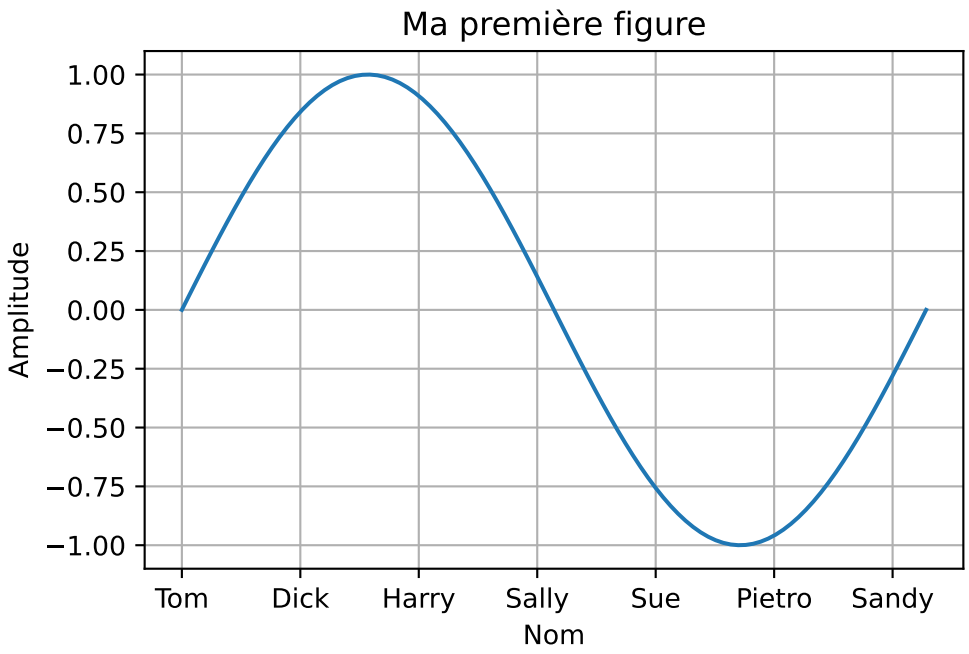
```
plt.show()  # Affichage d'une courbe
```



Personnalisation des étiquettes des axes

On peut personnaliser les étiquettes des axes en donnant une liste avec les fonctions `plt.xticks(...)` et `plt.yticks(...)`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)  # Ajout du d'une courbe
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Nom")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

# Changement des étiquettes de l'abscisse par des prénoms
plt.xticks(np.arange(2*np.pi), ('Tom', 'Dick',
                                'Harry', 'Sally', 'Sue', 'Pietro', 'Sandy'))

plt.show()  # Affichage d'une courbe
```

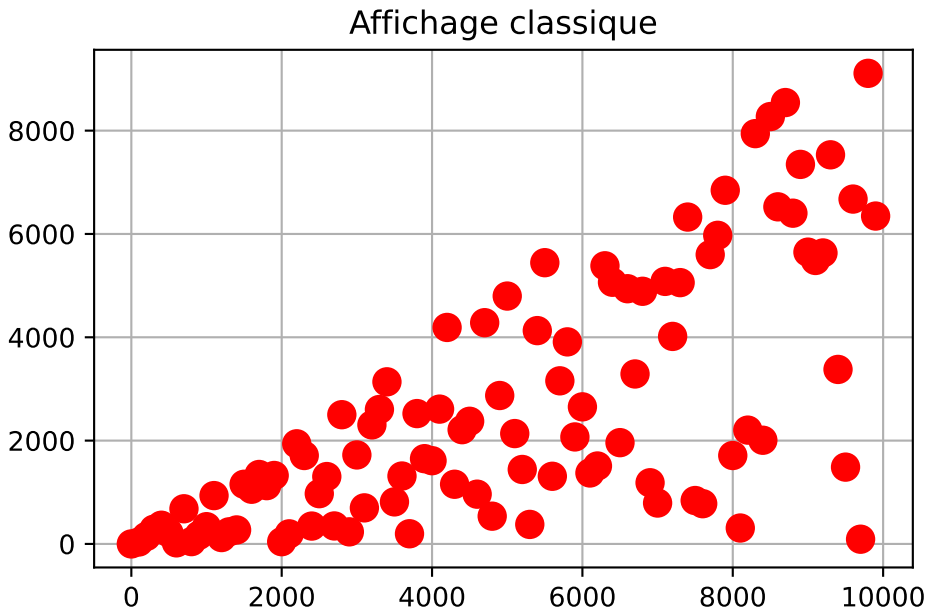


Affichage des étiquettes en écriture scientifique

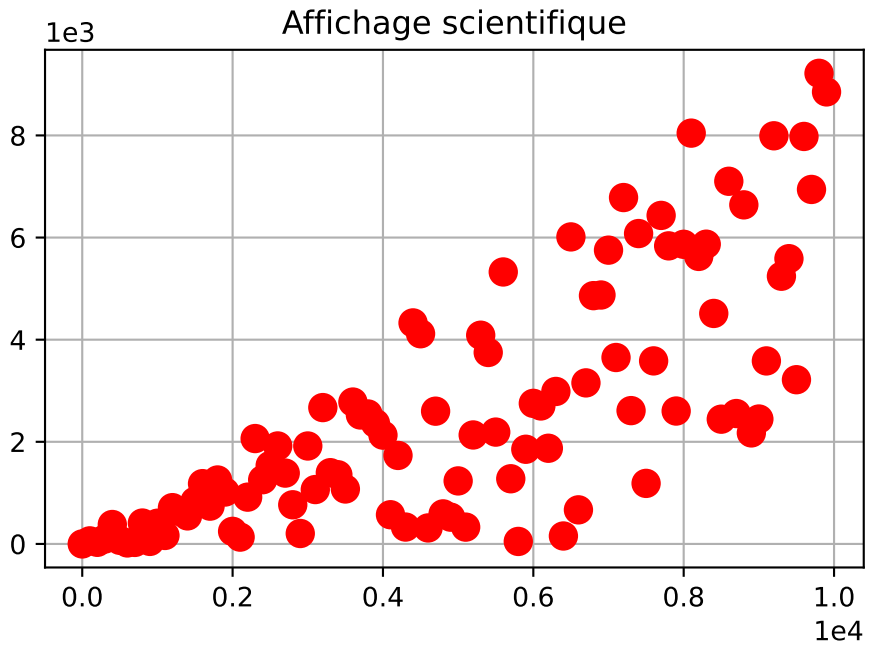
Pour de grands nombres, il est parfois nécessaire de changer le mode d'affichage des étiquettes et de le passer en écriture scientifique. Pour cela, on utilise la fonction `plt.ticklabel_format(axis='both', style='sci', scilimits=(0,0))`, avec les paramètres suivant:

- `axis='x'`: l'axe sur lequel appliquer le changement d'affichage -> 'x', 'y' ou 'both';
- `style='sci'`: pour appliquer l'écriture scientifique;
- `scilimiter=(m,n)`: limitation de l'écriture scientifique à l'extérieur de l'intervalle de 10^m à 10^n, on peut mettre `(0,0)` pour l'appliquer à toutes les valeurs.

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

# Création des données à afficher
x = np.arange(start=0, stop=10000, step=100)
y = np.random.rand(len(x))
y = x*y

# Affichage
plt.plot(x, y, 'ro', markersize=10)
plt.grid(True)
plt.title("Affichage classique")
plt.show()
```




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

# Création des données à afficher
x = np.arange(start=0, stop=10000, step=100)
y = np.random.rand(len(x))
y = x*y

# Affichage
plt.plot(x, y, 'ro', markersize=10)
plt.grid(True)
plt.title("Affichage scientifique")

plt.ticklabel_format(axis='both', style='sci', scilimits=(0, 0))

plt.show()
```



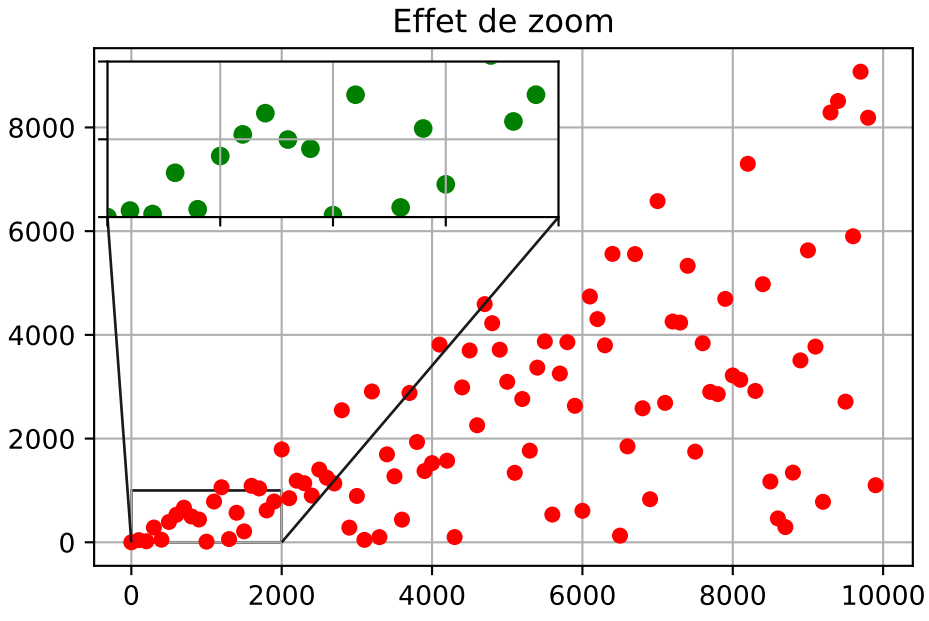
Zoom sur une partie du graphique

Il est possible d'inclure dans un graphique un effet de zoom sur une partie du graphique pour mettre en évidence un élément. L'emplacement du zoom est défini par un numéro, voir le tableau ci-dessous:

Position	Code
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

```
from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes, mark_inset
import matplotlib.pyplot as plt
import numpy as np

# Création des données à afficher
x = np.arange(start=0, stop=10000, step=100)
y = np.random.rand(len(x))
y = x*y

# Création des éléments d'axe et figure
fig, ax = plt.subplots()

# Affichage du graphique de base
```

```
plt.plot(x, y, 'ro', markersize=5)
plt.grid(True)
plt.title("Effet de zoom")

# Zoom
# Facteur de zoom: 3, location: en haut à gauche
axins = zoomed_inset_axes(ax, 3, loc=2)

# On dessine le graphique zoomé
axins.scatter(x, y, label='Zoom', color='green')
axins.set_xlim(0, 2000)  # Limite de l'axe x
axins.grid(True)
axins.set_ylim(0, 1000)  # Limite de l'axe y
plt.xticks(visible=False) # On cache les ticks sur x
plt.yticks(visible=False) # On cache les ticks sur y

# Effet de cadre sur le zoom
mark_inset(ax, axins, loc1=3, loc2=4, fc="None", ec="0.1")

plt.show()
```

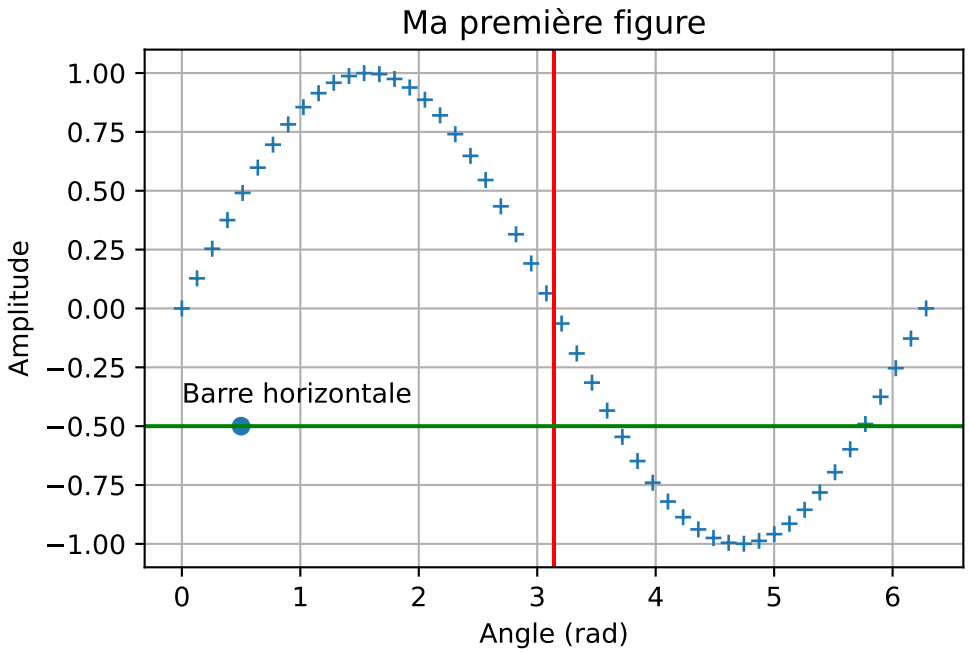


Annotations sur le graphique

Il est possible d'ajouter des annotations sur le graphique avec la fonction `plt.annotate(s, (x,y))`.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau pour le tracé
x = np.linspace(0, 2*np.pi, 50)
```

```
y = np.sin(x)

# Ajout du des points uniquement
plt.plot(x, y, '+', label="Sinus")
plt.ylabel('Amplitude')  # Titre de l'axe y
plt.xlabel("Angle (rad)")  # Titre de l'axe x
plt.title("Ma première figure")  # Titre du graphique
plt.grid(True)  # Affichage de la grille

# Barre verticale
plt.axvline(x=np.pi, ymin=0, ymax=1, color="red")
# Barre horizontale
plt.axhline(y=-0.5, xmin=0, xmax=1, color="green")

plt.scatter(0.5,-0.5) # Ajout d'un point
# Ajout d'une annotation
plt.annotate("Barre horizontale",(0,-0.4))


plt.show()  # Affichage d'une courbe
```



Exercice 1: Tracer la courbe de la fonction e^x et $ln(x)$ en reprenant chacune des étapes ci-dessus.

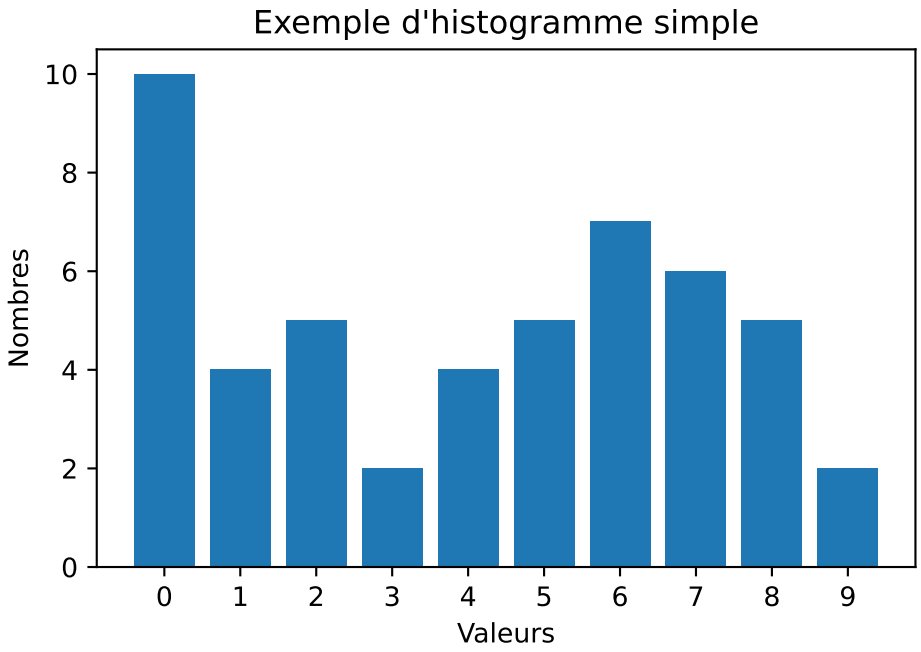
Histogramme

Les histogrammes se construisent à partir d'une liste de valeurs et des intervalles.

Histogramme simple

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau de 50 valeurs comprises entre 0 et 9
valeurs = np.random.randint(0, 10, 50)
print(valeurs)  # Affichage des valeurs
# Création d'un tableau avec les intervalles
# centrés sur la valeur entière
inter = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5]

# Création de l'histogramme
plt.hist(valeurs, bins=inter, rwidth=0.8)
plt.xlabel('Valeurs')
plt.xticks(np.arange(0, 10))
plt.ylabel('Nombres')
plt.title("Exemple d'histogramme simple")
plt.show()
```

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



Personnalisation des intervalles

Pour modifier les intervalles, il est possible d'utiliser plusieurs combinaisons des paramètres `range` et `bins`.

Paramètre	Détails
`bins`	Si c'est une valeur entière, défini le nombre de parts égal du paramètre `range` Si c'est une liste, un tableau ou une séquence, il défini les différents intervalles (valeur de gauche incluse et celle de droite exclue (sauf la dernière)). Les intervalles peuvent être de largeur inégale.
`range`	Valeurs min et max des intervalles

Exemples d'utilisation: - `plt.hist(valeurs, range=(0,9), bins=9)`: les valeurs sont comprises entre 0 et 9 avec 9 intervales de taille égale ([0,1[, [1,2[, ..., [8,9]); - `plt.hist(valeurs, range=(0,9))`: identique au précédent; - `plt.hist(valeurs, bins=range(10))`: même résultat que précédemment, mais en indiquant explicitement les intervalles; - `plt.hist(valeurs, bins=np.linspace(-0.5,9.5,11))`: création des intervalles de manière explicite centrés sur la valeur entière.

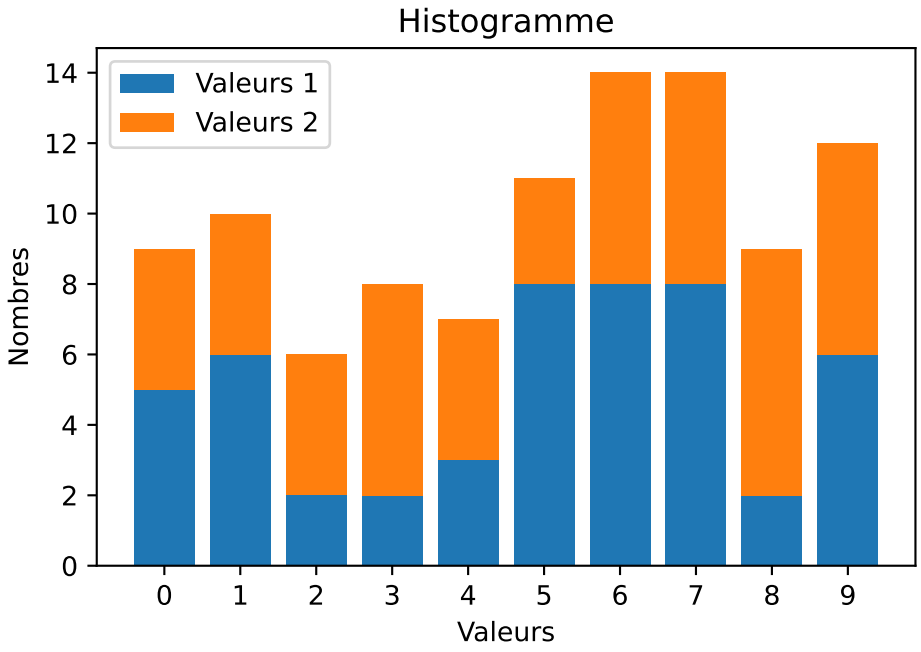
Personnalisation de l'histogramme

Les paramètres pour modifier l'apparence de l'histogramme sont disponibles ci-dessous:

Paramètre	Détails
`density=True`	Trace les fréquences plutôt que les nombres en ordonnée (somme vaut 1).
`weights=[1,2,1]`	Tableau de même dimension que les valeurs qui attribue un poids à chaque intervalle. S'applique uniquement si `density=True`.
`cumulative=True`	On construit l'histogramme du plus petit au plus grand est chaque barre prend ça valeur plus le cumul des valeurs plus petites (de gauche)
`histtype='bar'`	Défini le type d'histogramme à afficher : - 'bar' histogramme traditionnel avec des barres ; - 'barstacked' si on a plusieurs liste de valeurs, elles sont affichées empilées l'une sur l'autre et pas côte-à-côte ; - 'step' génère une courbe (contour) à partir de l'histogramme ; - 'stepfilled' génère une courbe (contour) et la zone entre le 0 et la valeur est coloriée.
`align='mid'`	Alignement de la barre par rapport au centre de l'intervalle : 'left', 'mid'ou 'right'.
`orientation='vertical'`	Orientation des barres : 'horizontal' ou 'vertical'.
`rwidth=0.8`	Largeur relative de la barre par rapport à l'intervalle (ici 80 %).
`log=True`	Utilise une échelle logarithmique
`color=red`	Couleur des barres
`edgecolor='black'`	Couleur du cadre des barres.
`label='Série 1`	Nom donné à la série. C'est le nom qui apparaît dans la légende.
`stacked=True`	Si il y a plusieurs séries de données, elles s'affichent empilées et non côte-à-côte.
`hatch='/'`	Hachuer les barres : '/', '.'

Exemple avec 2 séries de valeurs

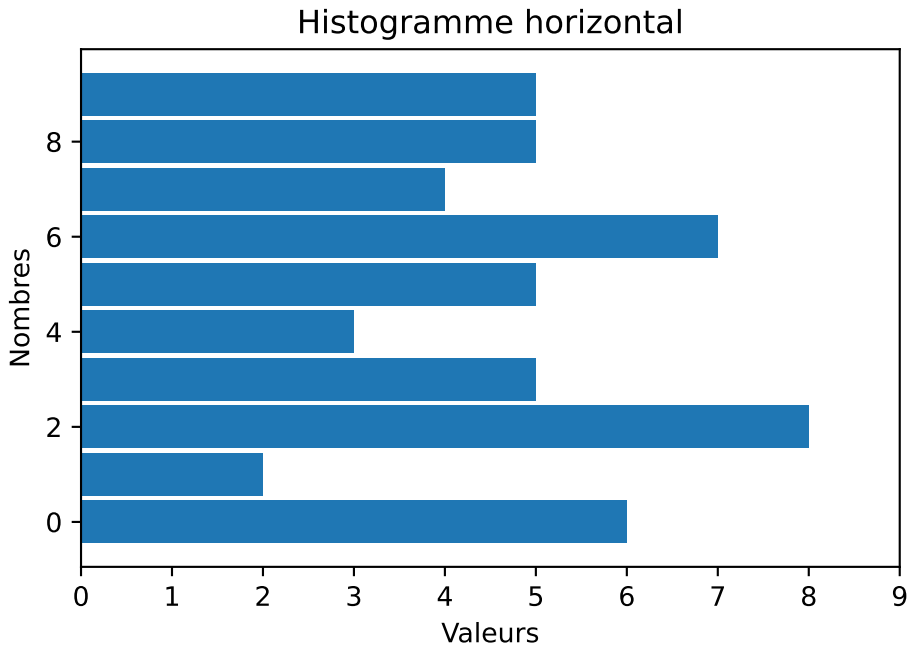
```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau de 50 valeurs comprises entre 0 et 9
valeurs = np.random.randint(0, 10, 50)
# Tableau de 50 valeurs comprises entre 0 et 9
valeurs2 = np.random.randint(0, 10, 50)
# Création d'un tableau avec les intervalles
# centrés sur la valeur entière
inter = np.linspace(-0.5, 9.5, 11)

# Création de l'histogramme
plt.hist([valeurs, valeurs2], bins=inter, rwidth=0.8, stacked=True,
         label=['Valeurs 1', 'Valeurs 2'])
plt.xlabel('Valeurs')
plt.xticks(np.arange(0, 10))
plt.ylabel('Nombres')
plt.title("Histogramme")
plt.legend()
plt.show()
```



Exemple barres horizontales

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np
```

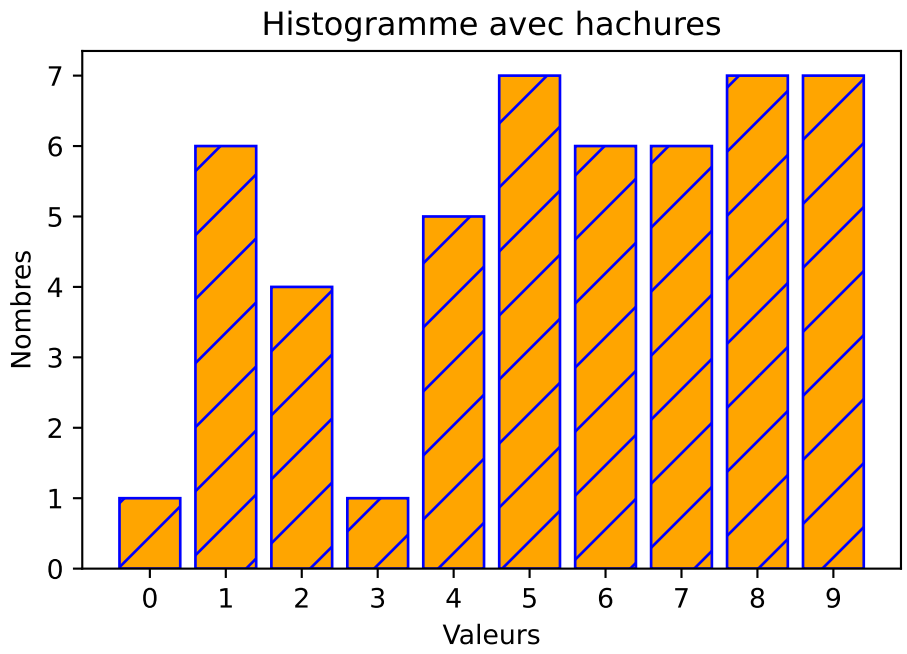
```
# Tableau de 50 valeurs comprises entre 0 et 9
valeurs = np.random.randint(0, 10, 50)
# Création d'un tableau avec les intervalles
# centrés sur la valeur entière
inter = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5]

# Création de l'histogramme
plt.hist(valeurs, bins=inter, orientation='horizontal',
         rwidth=0.9)
plt.xlabel('Valeurs')
plt.xticks(np.arange(0, 10))
plt.ylabel('Nombres')
plt.title("Histogramme horizontal")
plt.show()
```



Exemple décoration des barres

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Tableau de 50 valeurs comprises entre 0 et 9
valeurs = np.random.randint(0, 10, 50)
# Création d'un tableau avec les intervalles
# centrés sur la valeur entière
inter = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5]

# Création de l'histogramme
plt.hist(valeurs, bins=inter, rwidth=0.8, color='orange',
         edgecolor='blue', hatch='/')
```

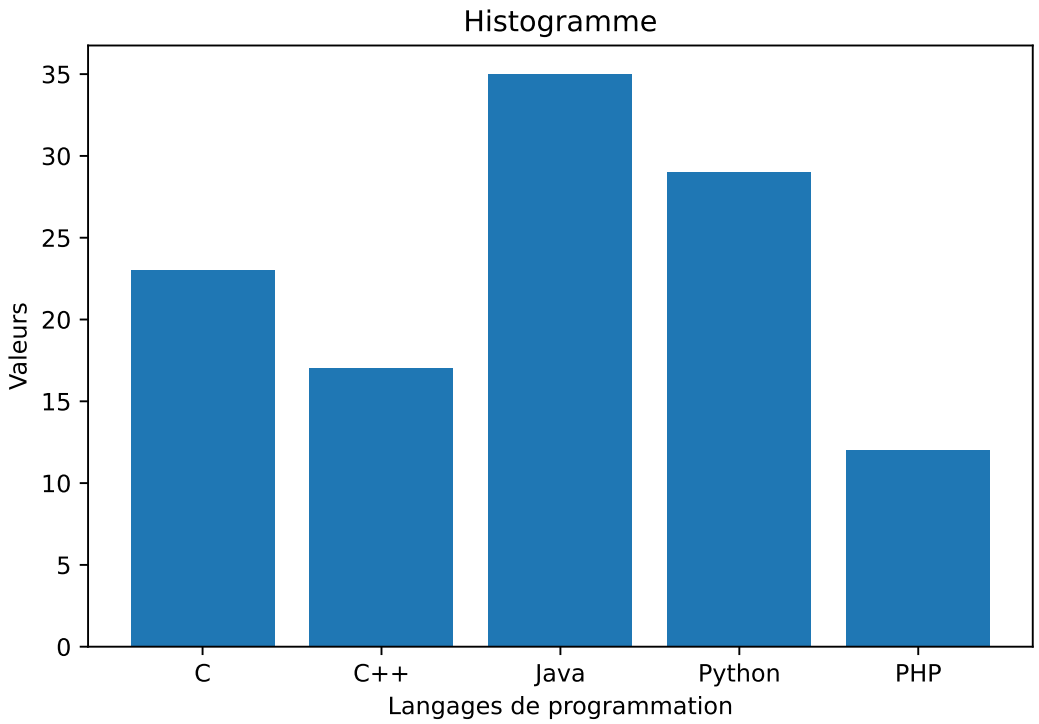
```
plt.xlabel('Valeurs')
plt.xticks(np.arange(0, 10))
plt.ylabel('Nombres')
plt.title("Histogramme avec hachures")
plt.show()
```



Histogramme avec étiquettes personnalisées

Pour afficher un histogramme associant une valeur avec une étiquette, il faut passer par la fonction `ax.bar(...)`.

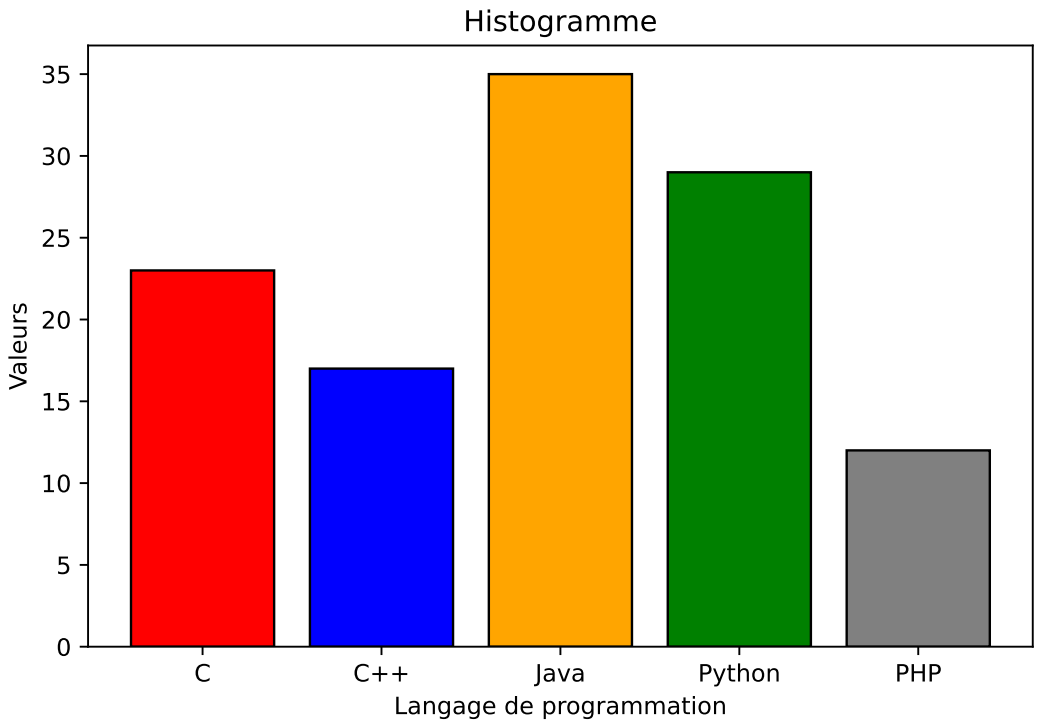
```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Préparation de la figure
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])

etiquettes = ['C', 'C++', 'Java', 'Python', 'PHP']
valeurs = [23, 17, 35, 29, 12]

# Affichage des données
ax.bar(etiquettes, valeurs)

plt.title("Histogramme")  # Titre du graphique
plt.ylabel('Valeurs')  # Titre de l'axe y
plt.xlabel('Langages de programmation')
plt.show()  # Affichage d'une courbe
```



Personnalisation de la couleur de chaque barre

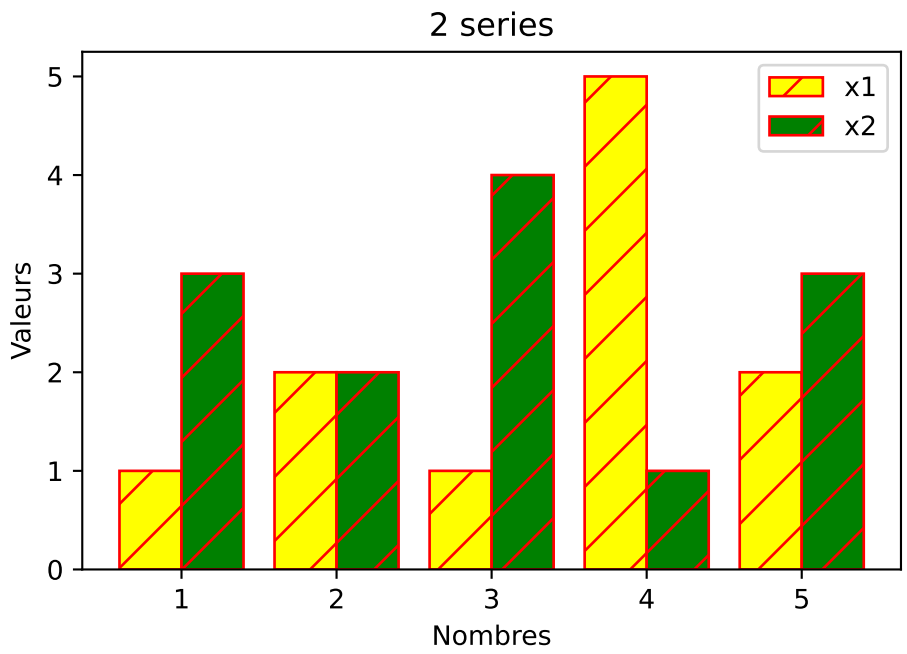
Pour personnaliser individuellement chaque couleur, on reprend le paramètre `color`, mais on lui passe un tableau de couleur qui correspond à chaque étiquette.

```
# Module pour tracer les graphiques
import matplotlib.pyplot as plt
import numpy as np

# Préparation de la figure
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])

etiquettes = ['C', 'C++', 'Java', 'Python', 'PHP']
valeurs = [23, 17, 35, 29, 12]

# Affichage des données
ax.bar(etiquettes, valeurs, color=[
       'red', 'blue', 'orange', 'green', 'gray'], edgecolor="black")

plt.title("Histogramme")  # Titre du graphique
plt.ylabel('Valeurs')  # Titre de l'axe y
plt.xlabel('Langage de programmation')
plt.show()  # Affichage d'une courbe
```

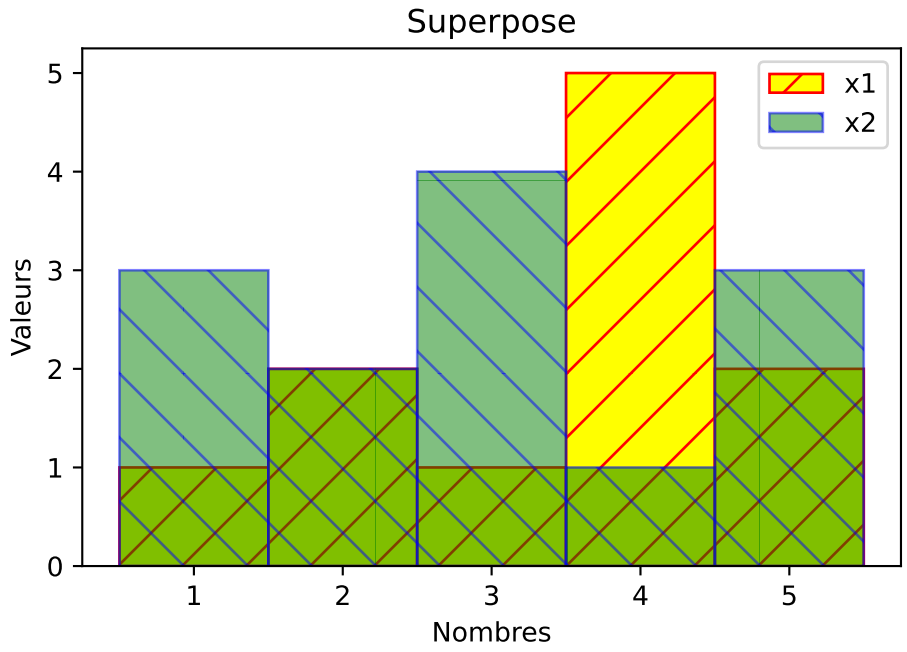


Autres exemples

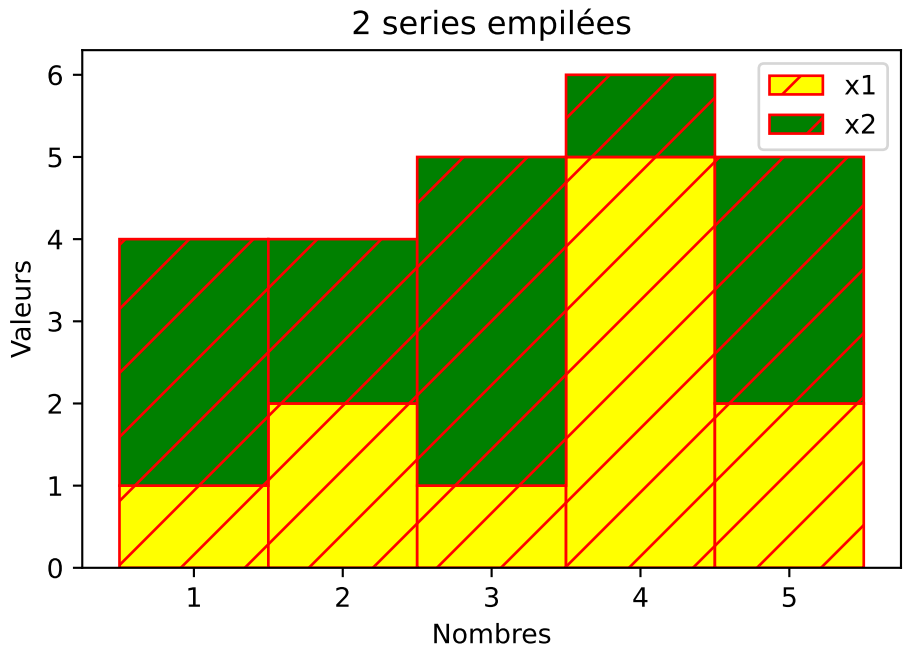
```
# Histogramme avec 2 séries
x1 = [1, 2, 2, 3, 4, 4, 4, 4, 4, 5, 5]
x2 = [1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 5, 5, 5]
bins = [x + 0.5 for x in range(0, 6)]

plt.hist([x1, x2], bins=bins, color=['yellow', 'green'],
         edgecolor='red', hatch='/', label=['x1', 'x2'],
         histtype='bar')  # bar est le defaut
plt.ylabel('Valeurs')
plt.xlabel('Nombres')
plt.title('2 series')
plt.legend()
plt.show()
```




```
# Histogrammes superposés avec un effet de transparence
x1 = [1, 2, 2, 3, 4, 4, 4, 4, 4, 5, 5]
x2 = [1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 5, 5, 5]
bins = [x + 0.5 for x in range(0, 6)]
plt.hist(x1, bins=bins, color='yellow',
         edgecolor='red', hatch='/', label='x1')
plt.hist(x2, bins=bins, color='green', alpha=0.5,
         edgecolor='blue', hatch='\\', label='x2')
plt.ylabel('Valeurs')
plt.xlabel('Nombres')
plt.title('Superpose')
plt.legend()
plt.show()
```



```
# Histogrammes empilés
x1 = [1, 2, 2, 3, 4, 4, 4, 4, 4, 5, 5]
x2 = [1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 5, 5, 5]
bins = [x + 0.5 for x in range(0, 6)]
plt.hist([x1, x2], bins=bins, color=['yellow', 'green'],
         edgecolor='red', hatch='/', label=['x1', 'x2'],
         histtype='barstacked')
plt.ylabel('Valeurs')
plt.xlabel('Nombres')
plt.title('2 series empilées')
plt.legend()
plt.show()
```



Exercice 2: Tracer l'histogramme de la liste generée par `[x for x np.random.normal(size = 100)]` en choisisant judicieusement la largeur des bandes.

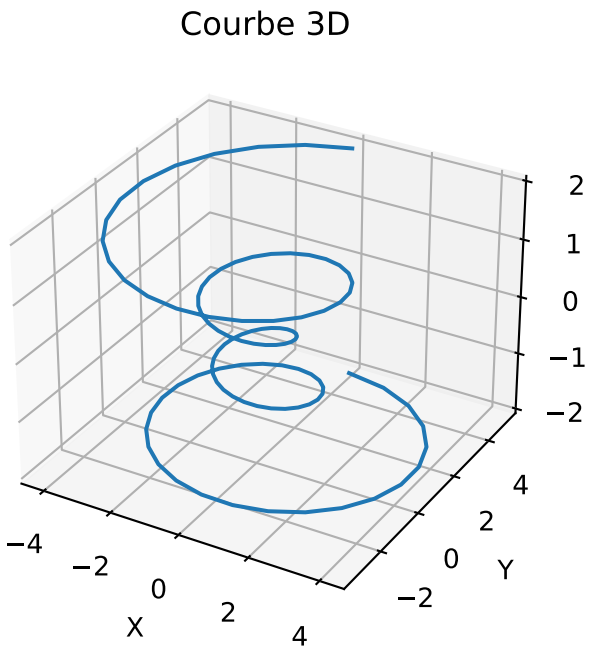
Graphiques 3D

Il est possible de réaliser des graphiques 3D sous Python pour visualiser des courbes, des surfaces, des points … dans un espace 3D.

Courbe 3D

```
import matplotlib.pyplot as plt
# Fonction pour la 3D
from mpl_toolkits.mplot3d import axes3d
import numpy as np

# Tableau pour les 3 axes
```

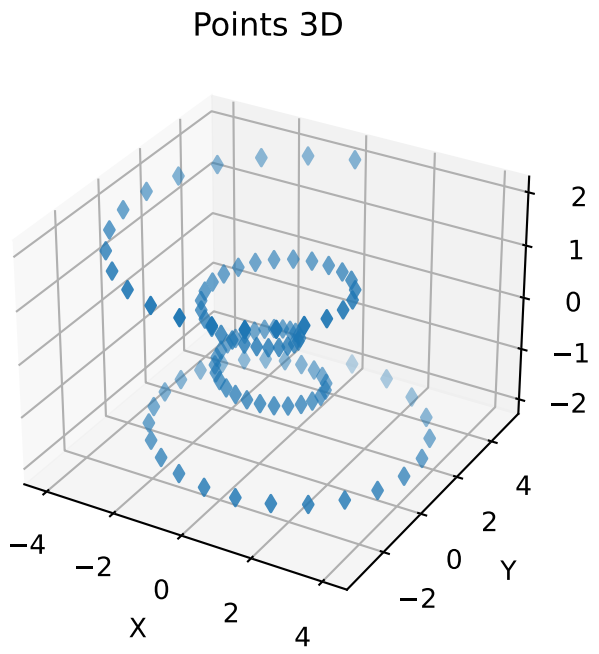
```
# Création d'un tableau de 100 points entre -4*pi et 4*pi
theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
# Création du tableau de l'axe z entre -2 et 2
z = np.linspace(-2, 2, 100)

r = z**2 + 1
x = r * np.sin(theta)  # Création du tableau de l'axe x
y = r * np.cos(theta)  # Création du tableau de l'axe y

# Tracé du résultat en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')  # Affichage en 3D
ax.plot(x, y, z, label='Courbe')  # Tracé de la courbe 3D
plt.title("Courbe 3D")
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.tight_layout()
plt.show()
```



Points 3D

```
import matplotlib.pyplot as plt
# Fonction pour la 3D
from mpl_toolkits.mplot3d import axes3d
import numpy as np

# Tableau pour les 3 axes
# Création d'un tableau de 100 points entre -4*pi et 4*pi
theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
# Création du tableau de l'axe z entre -2 et 2
```

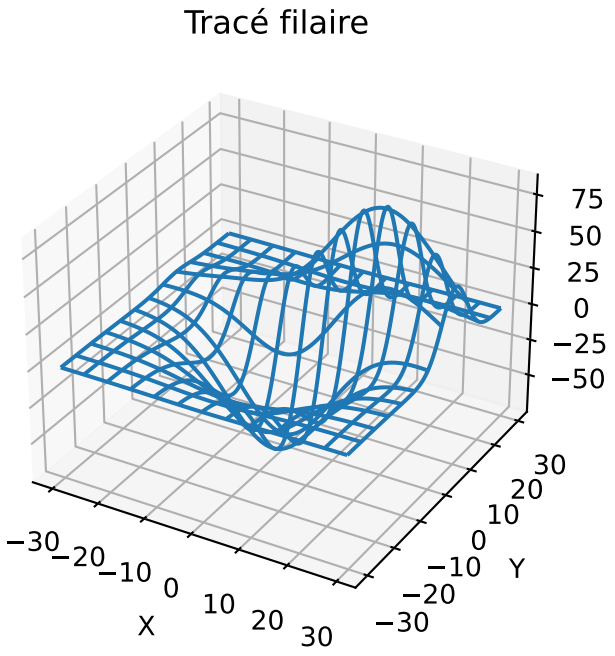
```
z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta)  # Création du tableau de l'axe x
y = r * np.cos(theta)  # Création du tableau de l'axe y

# Tracé du résultat en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')  # Affichage en 3D
# Tracé des points 3D
ax.scatter(x, y, z, label='Courbe', marker='d')
plt.title("Points 3D")
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.tight_layout()
plt.show()
```



Tracé filaire

```
import matplotlib.pyplot as plt
# Fonction pour la 3D
from mpl_toolkits.mplot3d import axes3d
import numpy as np

# Tableau pour les 3 axes
X, Y, Z = axes3d.get_test_data(0.05)


# Tracé du résultat en 3D
fig = plt.figure()
```

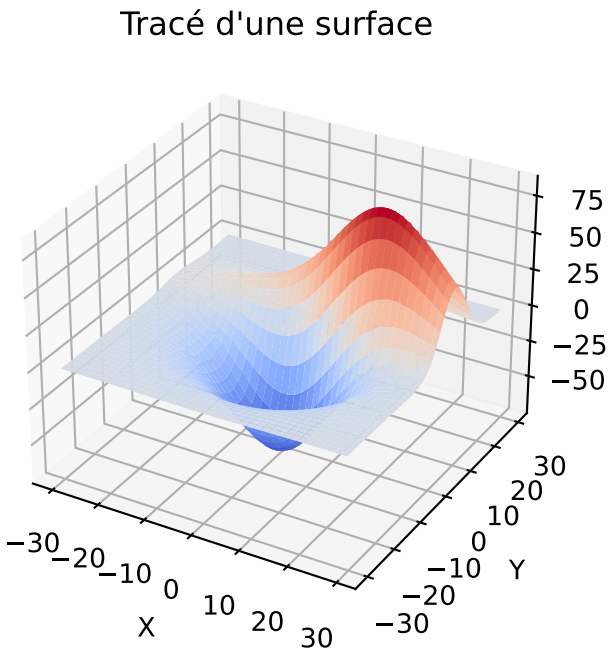
```
ax = fig.add_subplot(111, projection='3d')  # Affichage en 3D
# Tracé filaire
ax.plot_wireframe(X, Y, Z, rstride=10, cstride=10)
plt.title("Tracé filaire")
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.tight_layout()
plt.show()
```



Tracé d'une surface

```
import matplotlib.pyplot as plt
# Fonction pour la 3D
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np

# Tableau pour les 3 axes
X, Y, Z = axes3d.get_test_data(0.05)


# Tracé du résultat en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')  # Affichage en 3D
# Tracé d'une surface
ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, linewidth=0)
plt.title("Tracé d'une surface")
ax.set_xlabel('X')
```

```
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.tight_layout()
plt.show()
```

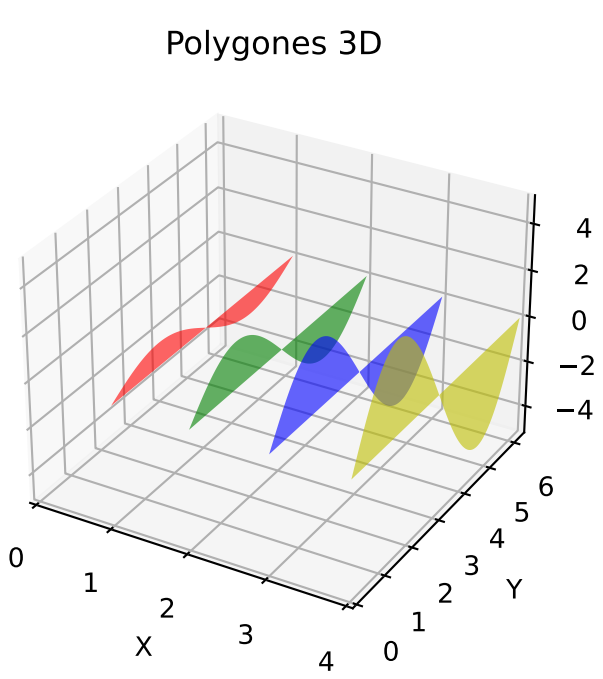


Tracé de polygones 3D ou courbes multiples

Permets de mettre côte à côte plusieurs courbes pour les comparer.

```
import matplotlib.pyplot as plt
# Fonction pour la 3D
from mpl_toolkits.mplot3d import axes3d
from matplotlib.collections import PolyCollection
from matplotlib import colors as mcolors
import numpy as np


def cc(arg):
    return mcolors.to_rgba(arg, alpha=0.6)


# Tableau pour les polygones
# Points pour chaque polygone
x = [1, 2, 3, 4]
# Création du tableau pour l'axe y
y = np.linspace(0, 2*np.pi, 100)

# Construction de chaque polygone pour les
# différents points de x
z = []
for xs in x:
```

```
    z.append(list(zip(y, xs*np.sin(y))))  # Axes (y,z)

# Création de la collection de polygones
poly = PolyCollection(z, facecolors=[cc('r'), cc('g'), cc('b'),
                                     cc('y')])
# Tracé du résultat en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')  # Affichage en 3D
# Tracé des différents polygones
ax.add_collection3d(poly, x, zdir='x')
plt.title("Polygones 3D")
ax.set_xlabel('X')
#ax.set_xticks(x,('Un','Deux','Trois', 'Quatre'))
ax.set_xlim3d(0, 4)  # Limites pour l'axe x
ax.set_ylabel('Y')
ax.set_ylim3d(0, 2*np.pi)  # Limites pour l'axe y
ax.set_zlabel('Z')
ax.set_zlim3d(-5, 5)  # Limites pour l'axe z
plt.tight_layout()
plt.show()
```



Histogramme 3D

Le tracé d'un histogramme 3D se construit barre par barre dans une ou plusieurs boucles `for`.

```
import matplotlib.pyplot as plt
# Fonction pour la 3D
from mpl_toolkits.mplot3d import axes3d
import numpy as np
```

```python
# Tracé du résultat en 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')  # Affichage en 3D

# Construction des histogrammes et affichage barre par barre
for c, z in zip(['r', 'g', 'b', 'y'], [30, 20, 10, 0]):
    x = np.arange(20)
    y = np.random.rand(20)

    # On peut définir une couleur différente pour chaque barre
    # Ici la première barre est en cyan.
    cs = [c] * len(x)
    cs[0] = 'c'
    # Ajout d'une barre
    ax.bar(x, y, z, zdir='y', color=cs, alpha=0.8)

plt.title("Histogramme 3D")
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.tight_layout()
plt.show()
```