Travaux Dirigés : Pratique de Matplotlib

Objectifs

Ce TD a pour objectif de renforcer votre maîtrise de la bibliothèque `matplotlib` en Python à travers une série d'exercices. Vous apprendrez à créer différents types de graphiques, à personnaliser les courbes et à ajouter des annotations, tout en utilisant des techniques de visualisation avancées.

Exercices

Exercice 1 : Tracer des fonctions trigonométriques multiples

Objectif : Tracer les fonctions $y = \sin(x)$, $y = \cos(x)$ et $y = \sin(2x)$ sur un même graphique.

1. Créez un tableau de valeurs `x` entre 0 et 2π.
2. Tracez chacune des fonctions avec des couleurs et des styles de ligne distincts.
3. Ajoutez des légendes pour chaque courbe et personnalisez les axes.

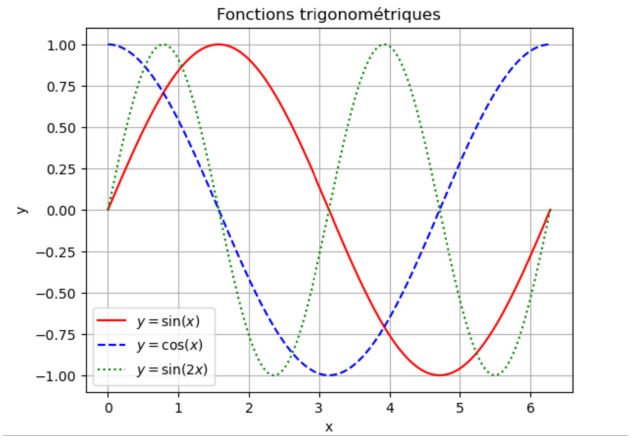

Figure 1: Fonctions trigonométriques

Exercice 2 : Courbe de Bézier

Objectif : Tracer une courbe de Bézier cubique donnée par les points de contrôle (P_0, P_1, P_2, P_3).

1. Définissez les points de contrôle : $P_0 = (0, 0)$, $P_1 = (1, 2)$, $P_2 = (2, 2)$, et $P_3 = (3, 0)$.
2. Utilisez la formule de la courbe de Bézier cubique pour calculer les points de la courbe :

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$$

3. Tracez la courbe pour $t \in [0, 1]$.

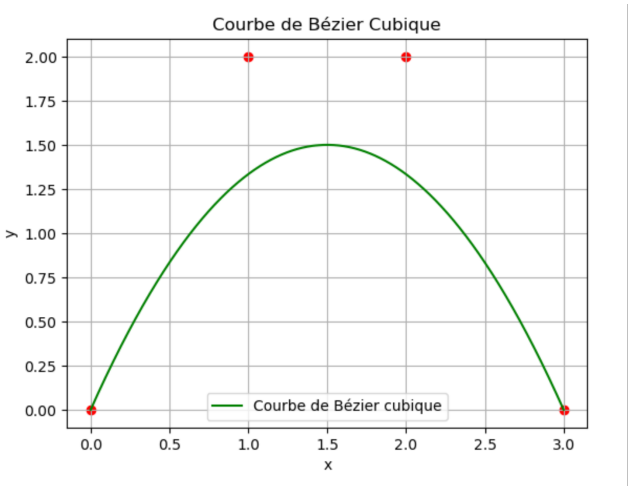

Figure 2: Courbe de Bezier

Exercice 3 : Graphe de la fonction complexe

Objectif : Tracer le module et l'argument d'une fonction complexe $f(z) = e^{iz}$.

1. Créez un tableau de valeurs `x` entre -2π et 2π.
2. Calculez la fonction complexe $f(z) = e^{iz}$, où $z = x + iy$ et les valeurs imaginaires sont données par `np.imag()`, les valeurs réelles sont données par `np.real()`.
3. Tracez à la fois la partie réelle et la partie imaginaire de la fonction.

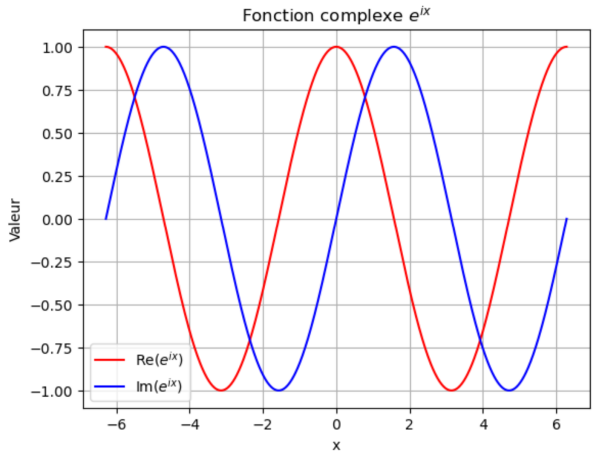

Figure 3: Graphe de fonction complexe

Exercice 4 : Ajouter des annotations

Objectif : Ajouter des annotations pour donner plus de détails sur un graphique.

1. Utilisez `plt.annotate()` pour ajouter des annotations à des points spécifiques sur le graphique.

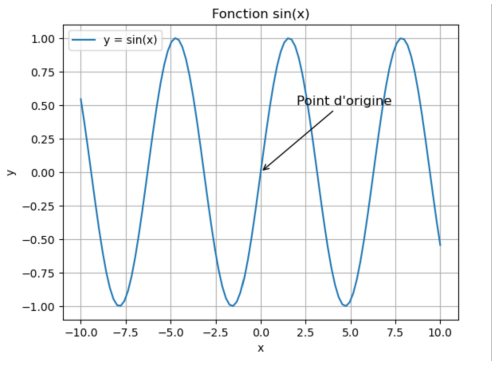

Figure 4: Annotation

Exercice 5 : Histogramme de distribution

Objectif : Créer un histogramme à partir de données aléatoires.

1. Utilisez `np.random.randn(1000)` pour générer 1000 valeurs suivant une distribution normale.
2. Utilisez `plt.hist()` pour afficher l'histogramme et ajoutez une grille avec `plt.grid()`.
3. Personnalisez les labels avec `plt.xlabel()` et `plt.ylabel()`.

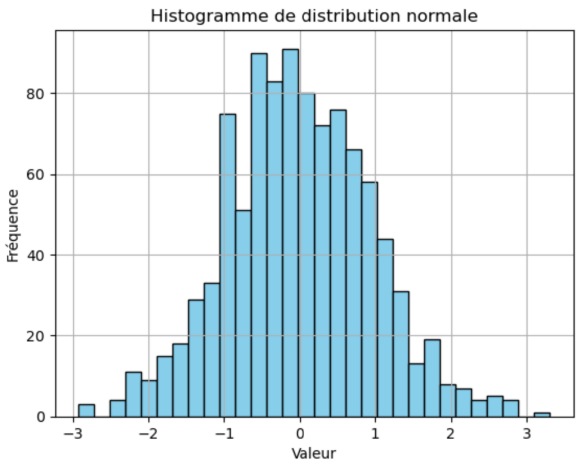

Figure 5: Histogramme de distribution

Exercice 6 : Graphiques en barres

Objectif : Créer un graphique en barres avec des données catégorielles.

1. Utilisez `plt.bar()` pour créer un graphique en barres.

2. Fournissez des données sous forme de catégories (produits) et de valeurs correspondantes.
3. Ajoutez un titre et des labels pour les axes.

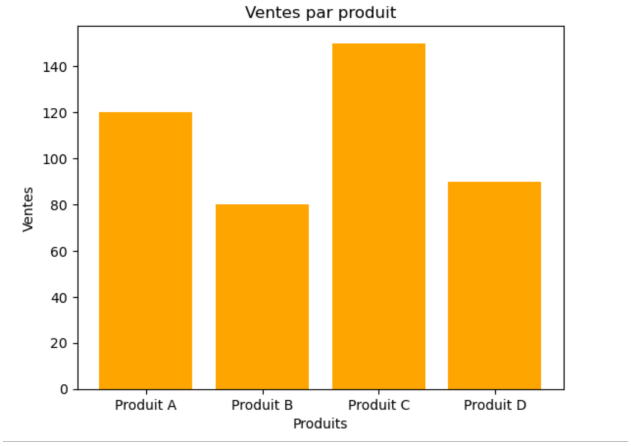

Figure 6: Graphique en barre

Exercice 7 : Tracer un graphique à secteurs (pie chart)

Objectif : Visualiser des proportions avec un graphique circulaire.

1. Utilisez `plt.pie()` pour créer un graphique à secteurs.
2. Fournissez une liste de valeurs proportionnelles et des étiquettes.
3. Ajoutez des pourcentages à chaque portion du graphique en définissant l'argument `autopct`.

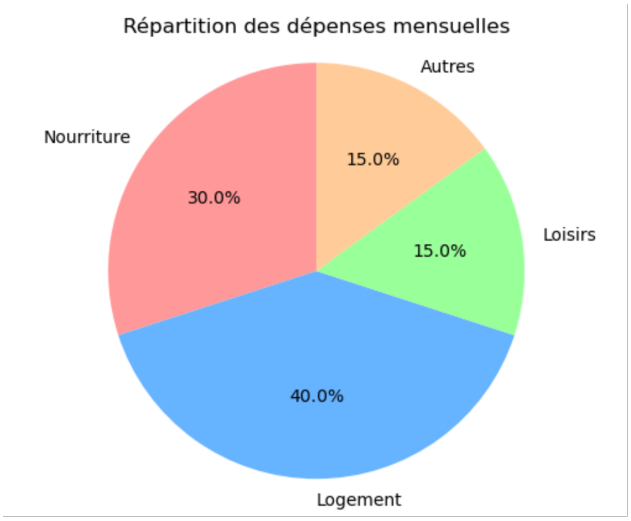

Figure 7: Caption

Exercice 8 : Tracer plusieurs sous-graphiques (subplots)

Objectif : Utiliser la fonctionnalité `subplot` pour créer plusieurs graphiques dans une seule figure.

1. Utilisez `plt.subplot()` pour diviser la figure en plusieurs sous-graphiques.
2. Tracez une fonction différente dans chaque sous-graphique.

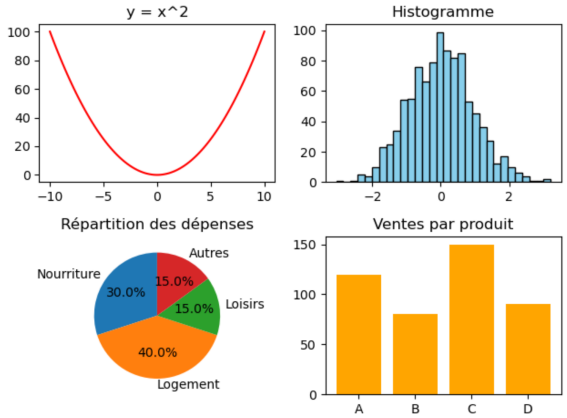

Figure 8: Plusieurs sous-graphiques

Exercice 9 : Personnalisation des styles de lignes et des marqueurs

Objectif : Personnaliser les lignes et les marqueurs d'un graphique.

1. Utilisez l'argument `linestyle` pour changer le type de ligne et `marker` pour ajouter des marqueurs sur la courbe.

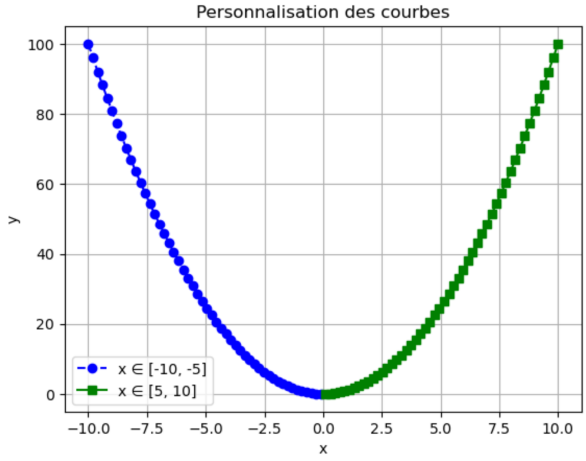

Figure 9: Fonction x^2

Exercice 10 : Ajouter des graphiques à axes multiples

Objectif : Créer un graphique avec plusieurs axes y.

1. Utilisez `plt.twinx()` pour ajouter un deuxième axe y.

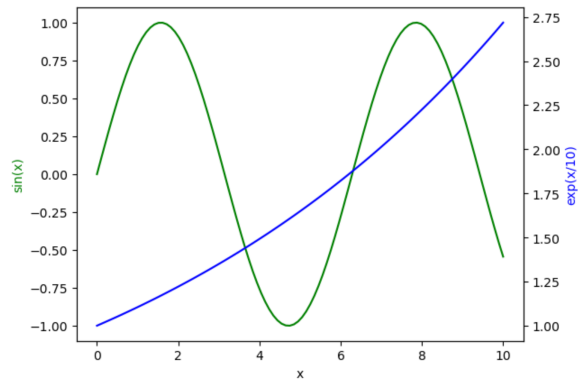

Figure 10: Graphique à axes multiples