TP Matplotib

Matplotlib est une bibliothèque Python pour créer des graphiques et des visualisations de données.

Partie 1: Introduction à Matplotlib

1. Installation et configuration

-Importez la bibliothèque Matplotlib.
-Utilisez également la commande magique “%matplotlib inline” pour permettre l'affichage des graphiques directement dans le notebook.

2. Créer un graphique simple

Générez une série de 50 nombres allant de 0 à 10, et tracez leur graphique de ligne.

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

# Générer une série de 50 nombres allant de 0 à 10
x = np.linspace(0, 10, 50)

# Générer des données pour l'axe des ordonnées (dans cet exemple, des carrés des nombres x)
y = x ** 2

# Tracer le graphique de ligne
plt.plot(x, y, '-o', label='y = x^2')  # '-o' pour spécifier le style de ligne et de marqueurs
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Graphique de ligne')
plt.grid(True)  # Activer la grille
plt.show()
```

3. Personnalisation de graphiques

Modifiez le graphique précédent en ajoutant un titre, des labels pour les axes et changez la couleur de la ligne en rouge.

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

# Générer une série de 50 nombres allant de 0 à 10
x = np.linspace(0, 10, 50)

# Générer des données pour l'axe des ordonnées (dans cet exemple, des
carrés des nombres x)
y = x ** 2

# Tracer le graphique de ligne avec une couleur rouge
plt.plot(x, y, '-o', color='red', label='y = x^2')  # '-o' pour
spécifier le style de ligne et de marqueurs
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Graphique de ligne')
plt.grid(True)  # Activer la grille
plt.show()
```

4. Tracer plusieurs séries de données

Tracez deux séries de données dans le même graphique. Utilisez des styles de ligne différents pour chaque série.

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

# Générer une série de 50 nombres allant de 0 à 10
x = np.linspace(0, 10, 50)

# Générer des données pour les deux séries (dans cet exemple, des
carrés et des cubes des nombres x)
y1 = x ** 2
y2 = x ** 3

# Tracer le graphique avec deux séries de données et des styles de
ligne différents
plt.plot(x, y1, '-o', color='blue', label='y = x^2')  # style de ligne
'-o' et couleur bleue
plt.plot(x, y2, '--s', color='red', label='y = x^3')  # style de ligne
'--s' et couleur rouge
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Graphique de deux séries de données')
```

```
plt.grid(True)  # Activer la grille
plt.show()
```

5. Ajouter une légende

Ajoutez une légende au graphique précédent, placez-la dans le coin supérieur droit.

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

# Générer une série de 50 nombres allant de 0 à 10
x = np.linspace(0, 10, 50)

# Générer des données pour les deux séries (dans cet exemple, des carrés et des cubes des nombres x)
y1 = x ** 2
y2 = x ** 3

# Tracer le graphique avec deux séries de données et des styles de ligne différents
plt.plot(x, y1, '-o', color='blue', label='y = x^2')  # style de ligne '-o' et couleur bleue
plt.plot(x, y2, '--s', color='red', label='y = x^3')  # style de ligne '--s' et couleur rouge
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Graphique de deux séries de données')
plt.legend(loc='upper right')  # Ajouter la légende et la placer dans le coin supérieur droit
plt.grid(True)  # Activer la grille
plt.show()
```

6. Travailler avec des figures et des axes

Créez une figure contenant deux subplots (graphiques) et tracez des données différentes dans chacun.

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

# Générer une série de 50 nombres allant de 0 à 10
x = np.linspace(0, 10, 50)
```

```
# Générer des données pour les deux séries (dans cet exemple, des
carrés et des cubes des nombres x)
y1 = x ** 2
y2 = x ** 3

# Créer une nouvelle figure et des sous-graphiques
fig, (ax1, ax2) = plt.subplots(1, 2)

# Tracer les données dans le premier sous-graphique
ax1.plot(x, y1, '-o', color='blue', label='y = x^2')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_title('Premier subplot')
ax1.legend()
ax1.grid(True)

# Tracer les données dans le deuxième sous-graphique
ax2.plot(x, y2, '--s', color='red', label='y = x^3')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_title('Deuxième subplot')
ax2.legend()
ax2.grid(True)

# Ajuster la disposition pour éviter le chevauchement
plt.tight_layout()

# Afficher la figure
plt.show()
```

7. Sauvegarder un graphique

Sauvegardez le dernier graphique créé en format PNG avec une résolution de 300 dpi.

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

# Générer une série de 50 nombres allant de 0 à 10
x = np.linspace(0, 10, 50)

# Générer des données pour les deux séries (dans cet exemple, des
carrés et des cubes des nombres x)
y1 = x ** 2
```

```
y2 = x ** 3

# Créer une nouvelle figure et des sous-graphiques
fig, (ax1, ax2) = plt.subplots(1, 2)

# Tracer les données dans le premier sous-graphique
ax1.plot(x, y1, '-o', color='blue', label='y = x^2')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_title('Premier subplot')
ax1.legend()
ax1.grid(True)

# Tracer les données dans le deuxième sous-graphique
ax2.plot(x, y2, '--s', color='red', label='y = x^3')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_title('Deuxième subplot')
ax2.legend()
ax2.grid(True)

# Ajuster la disposition pour éviter le chevauchement
plt.tight_layout()

# Sauvegarder le graphique en format PNG avec une résolution de 300 dpi
fig.savefig('graphique.png', dpi=300)

# Afficher la figure
plt.show()
```

Partie 2: Graphiques avancés

8. Histogrammes

Créez un histogramme pour visualiser la distribution de 100 valeurs aléatoires tirées d'une distribution normale.

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

# Générer 100 valeurs aléatoires tirées d'une distribution normale
data = np.random.normal(loc=0, scale=1, size=100)
```

```
# Créer un histogramme
plt.hist(data, bins=10, color='skyblue', edgecolor='black', alpha=0.7)
plt.xlabel('Valeurs')
plt.ylabel('Fréquence')
plt.title('Histogramme de la distribution normale')
plt.grid(True)

# Afficher l'histogramme
plt.show()
```

9. Graphiques à barres

Créez un graphique à barres pour comparer le nombre d'articles vendus dans 5 catégories différentes.

```
import matplotlib.pyplot as plt

# Catégories
categories = ['Catégorie 1', 'Catégorie 2', 'Catégorie 3', 'Catégorie
4', 'Catégorie 5']

# Nombre d'articles vendus dans chaque catégorie
articles_vendus = [120, 190, 150, 210, 180]

# Créer un graphique à barres
plt.bar(categories, articles_vendus, color='skyblue')

# Ajouter des étiquettes et un titre
plt.xlabel('Catégories')
plt.ylabel('Nombre d\'articles vendus')
plt.title('Nombre d\'articles vendus par catégorie')

# Afficher le graphique
plt.show()
```

10. Graphiques de dispersion (Scatter Plots)

Générez deux séries de 100 points chacune, suivant une distribution normale, et affichez-les dans un graphique de dispersion.

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

# Générer deux séries de 100 points chacune suivant une distribution
normale
x = np.random.normal(loc=0, scale=1, size=100)
```

```
y = np.random.normal(loc=0, scale=1, size=100)

# Créer un graphique de dispersion
plt.scatter(x, y, color='orange', alpha=0.7)

# Ajouter des étiquettes et un titre
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Graphique de dispersion de deux séries de données')

# Afficher le graphique
plt.show()
```

11. Tracer des erreurs

Ajoutez des barres d'erreur verticales à un graphique à barres simple pour indiquer l'incertitude sur chaque barre.

```
import matplotlib.pyplot as plt

# Catégories
categories = ['Catégorie 1', 'Catégorie 2', 'Catégorie 3', 'Catégorie
4', 'Catégorie 5']

# Nombre moyen d'articles vendus dans chaque catégorie
articles_vendus_moyen = [120, 190, 150, 210, 180]

# Incertitude associée à chaque moyenne
incertitude = [15, 20, 10, 25, 18]

# Créer un graphique à barres
plt.bar(categories, articles_vendus_moyen, color='skyblue',
yerr=incertitude, capsize=5)

# Ajouter des étiquettes et un titre
plt.xlabel('Catégories')
plt.ylabel('Nombre moyen d\'articles vendus')
plt.title('Nombre moyen d\'articles vendus par catégorie avec
incertitude')

# Afficher le graphique
plt.show()
```

12. Graphiques à empilement (Stack Plots)

Créez un graphique à empilement qui montre comment trois groupes de données évoluent avec le temps.

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

# Temps
temps = np.arange(1, 11)  # Par exemple, 10 points de données

# Données pour les trois groupes
groupe1 = np.random.randint(1, 10, size=len(temps))
groupe2 = np.random.randint(1, 10, size=len(temps))
groupe3 = np.random.randint(1, 10, size=len(temps))

# Créer un graphique à empilement
plt.stackplot(temps, groupe1, groupe2, groupe3, labels=['Groupe 1',
'Groupe 2', 'Groupe 3'])

# Ajouter des étiquettes et un titre
plt.xlabel('Temps')
plt.ylabel('Valeurs')
plt.title('Évolution des trois groupes avec le temps')
plt.legend()

# Afficher le graphique
plt.show()
```

Partie 3: Exploration d'autres bibliothèques

13. Seaborn: Heatmaps

Utilisez Seaborn pour créer un heatmap à partir d'une matrice de corrélations.

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

```
# Générer des données aléatoires pour créer une matrice de corrélations
np.random.seed(0)
data = np.random.rand(10, 10)

# Calculer la matrice de corrélations
corr_matrix = np.corrcoef(data)

# Tracer le heatmap avec Seaborn
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")

# Ajouter un titre
plt.title('Heatmap de la matrice de corrélations')

# Afficher le heatmap
plt.show()
```

14. Seaborn: Pairplot

Générez un pairplot avec Seaborn pour visualiser les relations entre quatre variables aléatoires.

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Générer des données aléatoires pour quatre variables
np.random.seed(0)
data = np.random.randn(100, 4)
df = pd.DataFrame(data, columns=['Variable 1', 'Variable 2', 'Variable 3', 'Variable 4'])

# Tracer le pairplot avec Seaborn
sns.pairplot(df)

# Afficher le pairplot
plt.show()
```

15. Plotly: Graphiques interactifs tous droits réservés

Créez un graphique interactif avec Plotly pour explorer les données sur les passagers du Titanic.

```
import plotly.express as px
import seaborn as sns

```

```
# Charger les données sur les passagers du Titanic (par exemple, avec
Seaborn)
titanic_data = sns.load_dataset('titanic')

# Créer un graphique interactif avec Plotly
fig = px.scatter(titanic_data, x='age', y='fare', color='survived',
hover_data=['sex', 'class'],
                 title='Exploration des données sur les passagers du
Titanic',
                 labels={'age': 'Âge', 'fare': 'Fare', 'survived':
'Survived'},
                 category_orders={'survived': ['No', 'Yes']})

# Afficher le graphique interactif
fig.show()
```

16. Graphiques de boîtes (Boxplot) avec Seaborn

Utilisez Seaborn pour tracer un boxplot montrant la distribution des valeurs dans plusieurs groupes.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Exemple de données : trois groupes avec des valeurs aléatoires
import numpy as np
np.random.seed(0)
group1 = np.random.normal(0, 1, 100)
group2 = np.random.normal(1, 1.5, 100)
group3 = np.random.normal(2, 0.5, 100)

# Créer un dataframe pandas à partir des données
import pandas as pd
data = pd.DataFrame({'Group 1': group1, 'Group 2': group2, 'Group 3':
group3})

# Tracer le boxplot avec Seaborn
sns.boxplot(data=data)

# Ajouter des étiquettes et un titre
plt.xlabel('Groupes')
plt.ylabel('Valeurs')
plt.title('Boxplot de la distribution des valeurs dans plusieurs
groupes')
```

```
# Afficher le boxplot
plt.show()
```

17. Graphiques en camembert (Pie Chart)

Créez un graphique en camembert pour illustrer la proportion de différentes catégories d'articles vendus.

```
import matplotlib.pyplot as plt

# Catégories d'articles vendus
categories = ['Catégorie 1', 'Catégorie 2', 'Catégorie 3', 'Catégorie 4', 'Catégorie 5']

# Proportions de ventes dans chaque catégorie
ventes = [20, 30, 15, 25, 10]

# Créer un graphique en camembert
plt.figure(figsize=(8, 8))  # Réglage de la taille de la figure
plt.pie(ventes, labels=categories, autopct='%1.1f%%', startangle=140, colors=plt.cm.tab10.colors)

# Ajouter un titre
plt.title('Proportion des ventes par catégorie')

# Afficher le graphique en camembert
plt.show()
```

18. Utilisation avancée des subplots

Générez 4 subplots dans une même figure pour comparer différentes distributions de données.

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

# Créer des données pour les 4 distributions
data1 = np.random.normal(loc=0, scale=1, size=1000)
data2 = np.random.normal(loc=2, scale=1.5, size=1000)
data3 = np.random.uniform(low=-1, high=1, size=1000)
data4 = np.random.exponential(scale=1, size=1000)

# Créer une figure et des sous-graphiques
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
```

```
# Tracer les données sur chaque sous-graphique
axs[0, 0].hist(data1, bins=30, color='blue', alpha=0.7)
axs[0, 0].set_title('Distribution normale')

axs[0, 1].hist(data2, bins=30, color='red', alpha=0.7)
axs[0, 1].set_title('Distribution normale (2, 1.5)')

axs[1, 0].hist(data3, bins=30, color='green', alpha=0.7)
axs[1, 0].set_title('Distribution uniforme')

axs[1, 1].hist(data4, bins=30, color='purple', alpha=0.7)
axs[1, 1].set_title('Distribution exponentielle')

# Ajuster l'espacement entre les sous-graphiques
plt.tight_layout()

# Afficher la figure
plt.show()
```

19. Animations dans Matplotlib

Créez une animation simple où un point se déplace le long d'une courbe sinusoïdale.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Créer des données pour la courbe sinusoïdale
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

# Initialiser la figure et les axes
fig, ax = plt.subplots()
ax.set_xlim(0, 2*np.pi)
ax.set_ylim(-1.5, 1.5)

# Créer un point initial
point, = ax.plot([], [], marker='o', color='red')

# Fonction d'initialisation de l'animation
def init():
    point.set_data([], [])
    return point,
```

```
# Fonction d'animation
def update(frame):
    x_point = x[frame]
    y_point = y[frame]
    point.set_data(x_point, y_point)
    return point,

# Créer l'animation
ani = FuncAnimation(fig, update, frames=len(x), init_func=init,
blit=True, interval=50)

# Afficher l'animation
plt.show()
```

20. Graphiques 3D

Utilisez Matplotlib pour tracer un graphique 3D montrant une surface z=f(x, y).

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# Définir la fonction z = f(x, y)
def f(x, y):
    return np.sin(np.sqrt(x**2 + y**2))

# Créer des données pour les variables x et y
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = f(x, y)

# Créer une figure 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Tracer la surface
surf = ax.plot_surface(x, y, z, cmap='viridis')

# Ajouter une barre de couleur
fig.colorbar(surf)
```

```
# Ajouter des étiquettes d'axes
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Ajouter un titre
ax.set_title('Surface 3D de z = f(x, y)')

# Afficher le graphique
plt.show()
```