Rappel des Graphiques - Matplotlib

1. Création d'un graphique simple

```
import matplotlib.pyplot as plt

# Données
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 35]

# Tracer un graphique
plt.plot(x, y)
plt.title("Graphique simple")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

2. Histogramme

```
import matplotlib.pyplot as plt

# Données
data = [1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 5, 5]

# Tracer un histogramme
plt.hist(data, bins=5, color='skyblue', edgecolor='black')
plt.title("Histogramme")
plt.xlabel("Valeurs")
plt.ylabel("Fréquence")
plt.show()
```

3. Diagramme à barres

```
# Données
categories = ["A", "B", "C", "D"]
values = [4, 7, 2, 8]

# Tracer un diagramme à barres
plt.bar(categories, values, color='lightgreen')
plt.title("Diagramme à barres")
```

```
plt.xlabel("Catégories")
plt.ylabel("Valeurs")
plt.show()
```

4. Diagramme à barres horizontal

```
# Tracer un diagramme à barres horizontal
plt.barh(categories, values, color='orange')
plt.title("Diagramme à barres horizontal")
plt.xlabel("Valeurs")
plt.ylabel("Catégories")
plt.show()
```

5. Diagramme circulaire (camembert)

```
# Données
labels = ["Python", "Java", "C++", "JavaScript"]
sizes = [40, 25, 20, 15]
colors = ["gold", "lightblue", "lightgreen", "pink"]

# Tracer un diagramme circulaire
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title("Diagramme circulaire")
plt.show()
```

6. Nuage de points (scatter plot)

```
import numpy as np

# Données
x = np.random.rand(50)
y = np.random.rand(50)
sizes = np.random.rand(50) * 100

# Tracer un nuage de points
plt.scatter(x, y, s=sizes, alpha=0.5, color='purple')
plt.title("Nuage de points")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

7. Courbes multiples

```
# Données
x = [1, 2, 3, 4, 5]
y1 = [1, 4, 9, 16, 25]
y2 = [1, 8, 27, 64, 125]

# Tracer deux courbes
plt.plot(x, y1, label="Carrés", color="blue", marker="o")
plt.plot(x, y2, label="Cubes", color="red", linestyle="--")
plt.title("Courbes multiples")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.legend()
plt.show()
```

8. Personnalisation des graphiques

```
# Données
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 35]

# Tracer avec personnalisation
plt.plot(x, y, color="green", marker="o", linestyle="dashed", linewidth=2, markersize=8)
plt.title("Graphique personnalisé", fontsize=14, color="blue")
plt.xlabel("X-Axis", fontsize=12)
plt.ylabel("Y-Axis", fontsize=12)
plt.grid(True, linestyle="--", linewidth=0.5)
plt.show()
```

9. Graphique en 3D

```
from mpl_toolkits.mplot3d import Axes3D

# Données
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

# Tracer un graphique 3D
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.plot_surface(X, Y, Z, cmap="viridis")
ax.set_title("Graphique 3D")
plt.show()
```

10. Graphique en sous-graphiques

```
# Données
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Tracer deux sous-graphiques
fig, axs = plt.subplots(2, 1, figsize=(8, 6))
axs[0].plot(x, y1, color="blue")
axs[0].set_title("Sinus")
axs[1].plot(x, y2, color="red")
axs[1].set_title("Cosinus")
plt.tight_layout()
plt.show()
```