Matplotlib

Gráficos básicos

CertiDevs

Índice de contenidos

1. Gráficos de líneas

Los **gráficos de líneas** son útiles para representar datos en función del tiempo u otras variables secuenciales.

Para crear un gráfico de líneas en Matplotlib, utilizamos la función `plot()`:

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y)
plt.show()
```

2. Gráficos de dispersión (scatter plots)

Los **gráficos de dispersión** se utilizan para visualizar la relación entre dos variables.

Para crear un gráfico de dispersión en Matplotlib, utilizamos la función `scatter()`:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.scatter(x, y)
plt.show()
```

3. Gráficos de barras

Los **gráficos de barras** son útiles para comparar categorías o grupos.

Para crear un gráfico de barras en Matplotlib, utilizamos la función `bar()`:

```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D', 'E']
values = [3, 5, 2, 7, 4]
plt.bar(categories, values)
plt.show()
```

4. Histogramas

Los **histogramas** son útiles para visualizar la distribución de una variable numérica.

Para crear un histograma en Matplotlib, utilizamos la función `hist()`:

```
import matplotlib.pyplot as plt

data = [1, 1, 2, 3, 3, 3, 4, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9]
plt.hist(data, bins=9)
plt.show()
```

5. Gráficos circulares (pie charts)

Los **gráficos circulares** o de tarta son útiles para mostrar proporciones o porcentajes de un total.

Para crear un gráfico circular en Matplotlib, utilizamos la función `pie()`:

```
import matplotlib.pyplot as plt

labels = ['A', 'B', 'C', 'D']
sizes = [20, 30, 40, 10]
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.show()
```

6. Personalización de gráficos en Matplotlib

También es posible **personalizar gráficos** en Matplotlib. Esto incluye agregar títulos y etiquetas a los ejes, ajustar los límites de los ejes, agregar leyendas y anotaciones, y modificar estilos y colores de líneas, marcadores y rellenos.

6.1. Establecer títulos y etiquetas para los ejes

Para agregar un **título** al gráfico, utilizamos la función `title()`.

Para agregar etiquetas a los ejes X e Y, utilizamos las funciones `xlabel()` e `ylabel()`, respectivamente:

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y)
plt.title('Gráfico de línea')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
plt.show()
```

6.2. Ajustar límites de los ejes

Para establecer los **límites de los ejes** X e Y, utilizamos las funciones `xlim()` e `ylim()`:

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y)
plt.xlim(0, 6)
plt.ylim(-2, 12)
plt.show()
```

6.3. Añadir leyendas y anotaciones

Para agregar una **leyenda** al gráfico, utilizamos la función `legend()`.

Primero, debemos asignar una etiqueta a cada serie de datos utilizando el argumento `label` en la función de trazado:

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y1 = [0, 2, 4, 6, 8, 10]
y2 = [0, 1, 4, 9, 16, 25]
plt.plot(x, y1, label='Lineal')
plt.plot(x, y2, label='Cuadrática')
plt.legend()
plt.show()
```

Para agregar anotaciones al gráfico, utilizamos la función `annotate()`:

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y)
plt.annotate('Punto de interés', xy=(2, 4), xytext=(3, 4),
arrowprops=dict(facecolor='black', shrink=0.05))
plt.show()
```

6.4. Modificar estilos y colores de líneas, marcadores y rellenos

Podemos **personalizar el estilo** y el color de las líneas, marcadores y rellenos utilizando

argumentos adicionales en las funciones de trazado:

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y, linestyle='--', color='red', marker='o', markersize=8)
plt.show()
```

6.5. Utilizar diferentes estilos y paletas de colores

Matplotlib admite diferentes estilos y paletas de colores que pueden ayudar a mejorar la estética y legibilidad de sus gráficos. Para cambiar el estilo de su gráfico, utilice la función `style.use()`.

Para ver una lista de estilos disponibles, utilice style.available:

```
import matplotlib.pyplot as plt
from matplotlib import style

print(style.available)

style.use('ggplot')

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y)
plt.show()
```

https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html

```
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['figure.figsize'] = (5,4)
plt.rcParams['figure.dpi'] = 120

x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
plt.plot(x, y)
plt.show()
```

Para utilizar paletas de colores personalizadas, puede importar el módulo colors de Matplotlib y utilizar el objeto Colormap para asignar colores a sus gráficos:

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

```
cmap = colors.ListedColormap(['red', 'green', 'blue', 'yellow'])
norm = colors.BoundaryNorm(range(5), cmap.N)

x = np.linspace(0, 4, 100)
y = np.sin(x)
plt.scatter(x, y, c=y, cmap=cmap, norm=norm)
plt.colorbar()
plt.show()
```

7. Interfaz pyplot vs orientada a objetos

7.1. Interfaz pyplot

Interfaz Pyplot: Esta es la forma más simple de usar matplotlib y es muy popular porque su sintaxis es similar a la de MATLAB. La interfaz pyplot maneja automáticamente la creación y gestión de figuras y ejes. Aunque es simple de usar, puede ser difícil de manejar cuando se realizan gráficos más complejos. Aquí tienes un ejemplo de cómo usarlo:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

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

7.2. Interfaz orientada a objetos

Esta forma de usar matplotlib ofrece más control y flexibilidad sobre tus gráficos.

Se centra en la creación de objetos Figure y Axes, y en la manipulación de estos para crear gráficos.

Aunque puede ser un poco más complicada al principio, esta es generalmente la forma recomendada de usar matplotlib cuando se necesitan gráficos más personalizados o complicados. Aquí tienes un ejemplo de cómo usarlo:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```

La interfaz orientada a objetos de matplotlib proporciona más control sobre las figuras y los ejes, permitiéndote crear gráficos más complejos y personalizados.

- Figure: La clase Figure es un objeto contenedor de alto nivel para todos los elementos del gráfico. Puedes pensar en ella como una única ventana que contiene varios gráficos. Una Figure puede contener varios objetos Axes, pero un objeto Axes solo puede estar en una Figure.
- Axes: Esta es la clase que contiene la mayoría de los elementos de los gráficos: Axis, Tick, Line2D, Text, Polygon, etc., y establece las coordenadas. En términos simples, es lo que comúnmente llamamos un 'gráfico'.

Aquí hay un ejemplo de cómo usar la interfaz orientada a objetos para crear un gráfico simple:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

fig, ax = plt.subplots()  # Crea una figura y unos ejes
ax.plot(x, y)  # Dibuja algunos datos en los ejes
plt.show()  # Muestra la figura
```

También puedes trabajar con más de un gráfico a la vez.

```
import matplotlib.pyplot as plt

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

fig, (ax1, ax2) = plt.subplots(2)  # Crea una figura y dos conjuntos de ejes

ax1.plot(x, y1)  # Dibuja algunos datos en los primeros ejes
ax1.set_title('Square Numbers')  # Establece un título para los primeros ejes

ax2.plot(x, y2)  # Dibuja algunos datos en los segundos ejes
ax2.set_title('Cubic Numbers')  # Establece un título para los segundos ejes

plt.tight_layout()  # Asegura que los gráficos no se superpongan
plt.show()  # Muestra la figura
```

Además, con la interfaz orientada a objetos, tienes un control completo sobre los aspectos individuales de tu gráfico.

Por ejemplo, puedes configurar fácilmente las etiquetas de los ejes y los límites de los ejes:

```
import matplotlib.pyplot as plt
```

```
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_xlabel('X-Axis')  # Establece la etiqueta del eje X
ax.set_ylabel('Y-Axis')  # Establece la etiqueta del eje Y
ax.set_xlim(0, 6)  # Establece los límites del eje X
ax.set_ylim(0, 30)  # Establece los límites del eje Y
plt.show()
```