SESIÓN LIBRERÍA MATPLOTLIB

CONTENIDOS:

- Características de la librería matplotlib: importación de la librería.
- Componentes principales de un gráfico.
- Crear un gráfico con las funciones pyplot.
- Crear un gráfico con modelo de orientación a objetos.
- Figuras y subgráficos.
- Posición y tamaño de los gráficos.
- Colores, marcadores y estilos.
- Ticks, etiquetas y leyendas.
- Fijando los límites de un gráfico.
- Anotaciones y dibujos en un gráfico.
- Guardando los gráficos en un archivo.
- Tipos de gráficos:
    - Diagrama de línea.
    - Histograma.
    - Boxplot.
    - Diagrama de dispersión.
    - Diagrama de barras.
    - Diagrama de torta.

CARACTERÍSTICAS DE LA LIBRERÍA MATPLOTLIB: IMPORTACIÓN DE LA LIBRERÍA.

La librería Matplotlib es una de las bibliotecas más populares en Python para la creación de gráficos y visualización de datos. Se utiliza principalmente para generar gráficos estáticos, animados e interactivos. Su principal objetivo es facilitar la representación visual de datos numéricos, permitiendo a los usuarios comprender y analizar patrones, tendencias y comportamientos dentro de sus datos.

Principales características de Matplotlib:

1. Versatilidad en la creación de gráficos

- Gráficos básicos: Permite crear gráficos de líneas, barras, dispersión, histogramas, diagramas de caja, entre otros.
- Gráficos complejos: Soporta gráficos de subgráficos, gráficos 3D y gráficos de contornos.
- Gráficos interactivos: Se pueden crear gráficos interactivos usando matplotlib en combinación con otras librerías como mpl_toolkits o plotly.

2. Personalización detallada

- Colores y estilos: Permite cambiar colores, estilos de líneas, tamaños de marcadores, y más.
- Etiquetas y títulos: Ofrece control sobre las etiquetas de los ejes, títulos de los gráficos, leyendas y otros textos.
- Escalas: Se pueden usar escalas logarítmicas, escalas de tiempo, entre otras.

3. Compatibilidad con otros entornos

- Matplotlib se integra fácilmente con otras librerías de Python, como NumPy, Pandas, y SciPy, lo que facilita la visualización de datos numéricos complejos.
- También se puede usar junto con Jupyter Notebooks, lo que facilita la visualización de gráficos dentro de un entorno interactivo.

4. Soporte para gráficos estáticos, animados y 3D

- Gráficos estáticos: Los gráficos generados con Matplotlib son de alta calidad, lo que los hace adecuados para informes y presentaciones.
- Animaciones: Puedes crear animaciones en tus gráficos, lo cual es útil para visualizar cómo cambian los datos con el tiempo.
- Gráficos 3D: A través del módulo mpl_toolkits.mplot3d, puedes generar gráficos en tres dimensiones.

5. Fácil exportación

- Los gráficos generados se pueden guardar en varios formatos de imagen, como PNG, PDF, SVG y EPS, lo que facilita su inclusión en otros documentos o presentaciones.

6. Escalabilidad

- Matplotlib es capaz de manejar grandes cantidades de datos, aunque su rendimiento puede ser menos eficiente comparado con otras librerías específicas de visualización como Seaborn o Plotly cuando se trata de grandes volúmenes de información. Aun así, sigue siendo muy útil en la mayoría de los casos.

7. Soporte para gráficos en varios subgráficos

- Se pueden crear varias visualizaciones en un solo gráfico mediante la función subplots, lo que es útil para comparar diferentes conjuntos de datos o representar distintas dimensiones de estos.

Importación de la Librería

Para usar Matplotlib, primero debemos importarla. La convención común es importar el módulo pyplot como plt, que es el submódulo más utilizado para la creación de gráficos. Este módulo proporciona una serie de funciones que simplifican la creación de gráficos de alta calidad en Python.

La importación estándar de Matplotlib es la siguiente:

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

Ilustración 1 Importación de matplotlib

COMPONENTES PRINCIPALES DE UN GRÁFICO

Un gráfico en Matplotlib está compuesto por varios elementos:

- **Figura (Figure):** El contenedor principal que puede incluir uno o más gráficos. Una figura puede tener múltiples ejes, cada uno con un gráfico diferente.
- **Ejes (Axes):** El área donde se dibuja el gráfico, incluyendo los ejes X e Y. En Matplotlib, un gráfico generalmente se dibuja en un conjunto de ejes.

- **Ticks:** Son las marcas o divisiones en los ejes, que indican las unidades o valores que corresponden a cada punto en el gráfico.
- **Etiquetas (Labels):** Texto que describe los ejes, como “Eje X” o “Eje Y”, indicando qué datos representan esos ejes.
- **Leyenda (Legend):** Explica los elementos del gráfico. Se utiliza para identificar las diferentes series de datos o elementos representados en el gráfico.
- **Título (Title):** Describe el gráfico. Usualmente se coloca en la parte superior del gráfico para indicar de qué trata o cuál es el objetivo de la visualización.

CREAR UN GRÁFICO CON LAS FUNCIONES PYPLOT

El enfoque de pyplot es sencillo y está diseñado para crear gráficos rápidamente. Con un conjunto de funciones simples, es posible generar gráficos como líneas, barras, histrogramas, etc., sin necesidad de configurar detalles avanzados.

Ejemplo: Gráfico de Línea

```python
import matplotlib.pyplot as plt

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

# Crear el gráfico
plt.plot(x, y)

# Agregar título y etiquetas
plt.title('Gráfico de Línea')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar el gráfico
plt.show()
```

Ilustración 2 Ejemplo gráfico de línea

En este ejemplo:

- plt.plot(x, y) crea una línea conectando los puntos definidos por las listas x e y.
- plt.title(), plt.xlabel(), y plt.ylabel() permiten añadir el título y las etiquetas a los ejes X e Y.
- plt.show() muestra el gráfico en la pantalla.

CREAR UN GRÁFICO CON MODELO DE ORIENTACIÓN A OBJETOS.

El modelo orientado a objetos (OO) de Matplotlib ofrece más control y flexibilidad. A diferencia de la interfaz de pyplot, que está basada en funciones, en el enfoque orientado a objetos trabajamos directamente con los objetos Figure y Axes, lo que permite un manejo más explícito de cada uno de los componentes del gráfico.

Ejemplo: Gráfico de Línea con Modelo OO

```python
import matplotlib.pyplot as plt

# Crear una figura y un conjunto de ejes
fig, ax = plt.subplots()

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

# Dibujar la línea en los ejes
ax.plot(x, y)

# Agregar título y etiquetas
ax.set_title('Gráfico de Línea')
ax.set_xlabel('Eje X')
ax.set_ylabel('Eje Y')

# Mostrar el gráfico
plt.show()
```

Ilustración 3 Ejemplo gráfico de línea con modelo OO

En este ejemplo:

- Se crea un objeto Figure y un conjunto de ejes ax utilizando plt.subplots().
- Los métodos de ax se utilizan para agregar elementos como el título y las etiquetas de los ejes, así como para dibujar la línea.

FIGURAS Y SUBGRÁFICOS

Las figuras y subgráficos te permiten crear múltiples gráficos en una sola figura. Usando plt.subplots(), puedes definir una cuadrícula de subgráficos y agregar gráficos en cada uno de ellos. Esto es útil cuando deseas comparar múltiples gráficos en una sola visualización.

Ejemplo: Dos Subgráficos

```python
import matplotlib.pyplot as plt

# Crear una figura con dos subgráficos (1 fila, 2 columnas)
fig, (ax1, ax2) = plt.subplots(1, 2)

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

# Graficar en el primer subgráfico
ax1.plot(x, y1)
ax1.set_title('Gráfico 1')

# Graficar en el segundo subgráfico
ax2.plot(x, y2)
ax2.set_title('Gráfico 2')

# Mostrar el gráfico
plt.show()
```

Ilustración 4 Ejemplo dos subgráficos

En este ejemplo, se crean dos subgráficos, y en cada uno se dibuja una línea diferente.

POSICIÓN Y TAMAÑO DE LOS GRÁFICOS

Puedes ajustar el tamaño y la posición de los gráficos utilizando figsize para definir el tamaño de la figura y subplots_adjust() para modificar el espacio entre los subgráficos.

Ejemplo: Ajustar Tamaño y Espaciado

```python
import matplotlib.pyplot as plt

# Crear una figura con un tamaño específico
fig, ax = plt.subplots(figsize=(8, 6))

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

# Graficar
ax.plot(x, y)
ax.set_title('Gráfico Ajustado')
ax.set_xlabel('Eje X')
ax.set_ylabel('Eje Y')

# Ajustar el espaciado entre subgráficos
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)

# Mostrar el gráfico
plt.show()
```

Ilustración 5 Ejemplo ajustar tamaño y espaciado

Este ejemplo ajusta tanto el tamaño de la figura (con figsize) como el espaciado entre los elementos de la figura utilizando subplots_adjust().

COLORES, MARCADORES Y ESTILOS.

Puedes personalizar el estilo de los gráficos usando diferentes colores, tipos de marcadores y estilos de línea. Esto te permite crear gráficos más visualmente atractivos y adecuados a tus necesidades de presentación.

Ejemplo: Personalización

```python
import matplotlib.pyplot as plt

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

# Crear gráfico con color, marcador y estilo de línea personalizados
plt.plot(x, y, color='red', marker='o', linestyle='--')

# Agregar título y etiquetas
plt.title('Gráfico Personalizado')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar el gráfico
plt.show()
```

Ilustración 6 Ejemplo personalización

En este ejemplo, se utiliza un color rojo (color='red'), un marcador circular (marker='o') y una línea discontinua (linestyle='--') para personalizar el gráfico. Puedes experimentar con diferentes combinaciones de estos atributos para mejorar la presentación de tus gráficos.

TICKS, ETIQUETAS Y LEYENDAS.

Los ticks son las marcas en los ejes que indican los valores correspondientes. Puedes personalizar los ticks, las etiquetas de los ejes y agregar una leyenda que describa los elementos del gráfico.

Ejemplo: Personalización de Ticks y Leyenda

```python
import matplotlib.pyplot as plt

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

# Crear gráfico
plt.plot(x, y, label='Curva cuadrática', color='blue')

# Personalizar los ticks
plt.xticks([1, 2, 3, 4, 5], ['Uno', 'Dos', 'Tres', 'Cuatro', 'Cinco'])
plt.yticks([1, 4, 9, 16, 25], ['Uno', 'Cuatro', 'Nueve', 'Dieciséis', 'Veinticinco'])

# Añadir leyenda
plt.legend()

# Títulos y etiquetas
plt.title('Gráfico con Ticks Personalizados')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar gráfico
plt.show()
```

Ilustración 7 Ejemplo de ticks y leyenda

En este ejemplo, los ticks de los ejes X e Y han sido personalizados y se ha añadido una leyenda para describir la línea del gráfico.

FIJANDO LOS LÍMITES DE UN GRÁFICO

Puedes establecer límites para los ejes X e Y de tu gráfico utilizando set_xlim() y set_ylim(), lo que te permite controlar el rango de valores que se mostrarán.

Ejemplo: Límites de Ejes

```python
import matplotlib.pyplot as plt

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

# Crear gráfico
plt.plot(x, y)

# Fijar los límites de los ejes
plt.xlim(0, 6)  # Limitar el eje X entre 0 y 6
plt.ylim(0, 30)  # Limitar el eje Y entre 0 y 30

# Títulos y etiquetas
plt.title('Gráfico con Límites de Ejes')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar gráfico
plt.show()
```

Ilustración 8 Ejemplo límites de ejes

Este ejemplo muestra cómo fijar los límites de los ejes X e Y para controlar el área visual del gráfico.

ANOTACIONES Y DIBUJOS EN UN GRÁFICO

Puedes añadir anotaciones y formas a tus gráficos para resaltar información importante. Matplotlib te permite colocar texto en cualquier posición del gráfico y dibujar formas como líneas, círculos, rectángulos, entre otros-

Ejemplo: Anotación

```python
import matplotlib.pyplot as plt

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

# Crear gráfico
plt.plot(x, y)

# Añadir anotación
plt.annotate('Punto máximo', xy=(5, 25), xytext=(4, 20),
             arrowprops=dict(facecolor='black', arrowstyle='->'),
             fontsize=12, color='red')

# Títulos y etiquetas
plt.title('Gráfico con Anotación')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar gráfico
plt.show()
```

Ilustración 9 Ejemplo anotación

En este ejemplo, se utiliza annotate() para agregar una anotación con una flecha que señala un punto específico en el gráfico.

GUARDANDO LOS GRÁFICOS EN UN ARCHIVO

Puedes guardar los gráficos generados en varios formatos como PNG, PDF, SVG, y más, utilizando el método savefig(). Esto te permite guardar tus visualizaciones para compartirlas o usarlas en otros proyectos.

Ejemplo: Guardar Gráfico

```python
import matplotlib.pyplot as plt

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

# Crear gráfico
plt.plot(x, y)

# Guardar gráfico en un archivo PNG
plt.savefig('grafico.png')

# Títulos y etiquetas
plt.title('Gráfico Guardado')
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

# Mostrar gráfico
plt.show()
```

Ilustración 10 Ejemplo guardar gráfico

Este ejemplo guarda el gráfico como un archivo PNG con el nombre gráfico.png en el directorio actual. También puedes cambiar el formato cambiando la extensión, por ejemplo, .pdf o .svg según lo necesites.

TIPOS DE GRÁFICOS

Diagrama de Línea.

Un diagrama de línea se utiliza para mostrar la evolución de los datos a través del tiempo o una secuencia. Cada punto en el gráfico está conectado por una línea, lo que permite observar tendencias y patrones en los datos.

Ejemplo Python:

```python
import matplotlib.pyplot as plt

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

# Crear diagrama de línea
plt.plot(x, y)

# Títulos y etiquetas
plt.title('Diagrama de Línea')
plt.xlabel('X')
plt.ylabel('Y')

# Mostrar gráfico
plt.show()
```

Ilustración 11 Ejemplo diagrama de línea

Histograma

Un histograma es un gráfico que muestra la distribución de un conjunto de datos dividiendo los datos en intervalos (bins) y representando la frecuencia de los valores en cada intervalo. Es útil para visualizar la distribución y dispersión de los datos.

Ejemplo Python:

```python
import matplotlib.pyplot as plt

# Datos
data = [1, 2, 2, 3, 3, 3, 4, 5, 5, 6, 6, 6, 6, 7, 8]

# Crear histograma
plt.hist(data, bins=5, edgecolor='black')

# Títulos y etiquetas
plt.title('Histograma')
plt.xlabel('Valores')
plt.ylabel('Frecuencia')

# Mostrar gráfico
plt.show()
```

Ilustración 12 Ejemplo histograma

Boxplot.

El boxplot o diagrama de caja es útil para visualizar la distribución de los datos a través de sus cuartiles y detectar posibles valores atípicos. Muestra el rango intercuartílico, la mediana y los valores atípicos de un conjunto de datos.

Ejemplo Python:

```python
import matplotlib.pyplot as plt

# Datos
data = [1, 2, 5, 7, 8, 9, 11, 14, 15, 16, 17]

# Crear boxplot
plt.boxplot(data)

# Títulos y etiquetas
plt.title('Boxplot')
plt.ylabel('Valores')

# Mostrar gráfico
plt.show()
```

Ilustración 13 Ejemplo boxplot

Diagrama de Dispersión

Un diagrama de dispersión muestra la relación entre dos variables. Cada punto en el gráfico representa un par de valores, lo que permite observar posibles correlaciones entre las variables.

Ejemplo Python:

```python
import matplotlib.pyplot as plt

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

# Crear diagrama de dispersión
plt.scatter(x, y)

# Títulos y etiquetas
plt.title('Diagrama de Dispersión')
plt.xlabel('X')
plt.ylabel('Y')

# Mostrar gráfico
plt.show()
```

Ilustración 14 Ejemplo diagrama de dispersión

Diagrama de Barras

El diagrama de barras se utiliza para representar datos categóricos, donde cada categoría está representada por una barra cuya altura corresponde al valor de la categoría.

Ejemplo Python:

```python
import matplotlib.pyplot as plt

# Datos
categorias = ['A', 'B', 'C', 'D']
valores = [3, 7, 5, 8]

# Crear diagrama de barras
plt.bar(categorias, valores)

# Títulos y etiquetas
plt.title('Diagrama de Barras')
plt.xlabel('Categorías')
plt.ylabel('Valores')

# Mostrar gráfico
plt.show()
```

Ilustración 15 Ejemplo diagrama de barras

Diagrama de Torta

El diagrama de torta o gráfico de pastel muestra cómo se divide un conjunto de datos en categorías. Es útil para visualizar las proporciones de cada categoría en relación con el total.

Ejemplo Python:

```python
import matplotlib.pyplot as plt

# Datos
labels = ['A', 'B', 'C', 'D']
sizes = [15, 30, 45, 10]

# Crear diagrama de torta
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)

# Títulos y etiquetas
plt.title('Diagrama de Torta')

# Mostrar gráfico
plt.show()
```

Ilustración 16 Ejemplo diagrama de torta

ACTIVIDAD PRÁCTICA GUIADA: CREAR UN GRÁFICO DE LÍNEAS CON ETIQUETAS PERSONALIZADAS Y UNA LEYENDA

En este ejercicio práctico, aprenderás a crear un gráfico de líneas utilizando la librería Matplotlib en Python. Además, personalizarás las etiquetas de los ejes y añadirás una leyenda para mejorar la claridad del gráfico.

Paso 1: Importar Librerías

Primero, importamos la librería necesaria: Matplotlib.

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

Ilustración 17 Ejemplo importar librerías

Paso 2: Crear un Conjunto de Datos

Vamos a crear un conjunto de datos simple que represente las ventas mensuales de una empresa durante un año.

```
# Datos de ejemplo
meses = ['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic']
ventas = [200, 220, 250, 300, 350, 400, 450, 430, 400, 380, 350, 320]
```

Ilustración 18 Ejemplo de creación conjunto de datos

Paso 3: Crear el Gráfico de Líneas

Usamos la función plot de Matplotlib para crear un gráfico de líneas.

```
# Crear gráfico de líneas
plt.plot(meses, ventas)
plt.show()
```

Ilustración 19 Ejemplo creación gráfico de líneas

Salida: Un gráfico de líneas básico que muestra las ventas mensuales.

Paso 4: Personalizar las Etiquetas de los Ejes

Añadimos etiquetas a los ejes X e Y para que el gráfico sea más informativo.

```
# Crear gráfico de líneas con etiquetas
plt.plot(meses, ventas)
plt.xlabel('Meses')  # Etiqueta del eje X
plt.ylabel('Ventas (en miles)')  # Etiqueta del eje Y
plt.show()
```

Ilustración 20 Ejemplo para agregar etiquetas a los ejes X e Y

Salida: El gráfico ahora tiene etiquetas en los ejes X e Y.

Paso 5: Añadir un Título al Gráfico

Un título ayuda a entender rápidamente el propósito del gráfico.

```
# Crear gráfico de líneas con título
plt.plot(meses, ventas)
plt.xlabel('Meses')
plt.ylabel('Ventas (en miles)')
plt.title('Ventas Mensuales en 2023')
plt.show()
```

Ilustración 21 Ejemplo para añadir un título al gráfico

Salida: El gráfico ahora incluye un título.

Paso 6: Añadir una Leyenda

Si tienes múltiples líneas en el gráfico, una leyenda es esencial para diferenciarlas. En este caso, aunque solo tenemos una línea, añadiremos una leyenda para practicar.

```python
# Crear gráfico de líneas con leyenda
plt.plot(meses, ventas, label='Ventas 2023')
plt.xlabel('Meses')
plt.ylabel('Ventas (en miles)')
plt.title('Ventas Mensuales en 2023')
plt.legend()  # Añadir leyenda
plt.show()
```

Ilustración 22 Ejemplo para añadir leyenda a un gráfico

Salida: El gráfico ahora incluye una leyenda que describe la línea.

Paso 7: Personalizar la Línea

Puedes personalizar el estilo de la línea para mejorar la visualización.

```python
# Crear gráfico de líneas personalizado
plt.plot(meses, ventas, label='Ventas 2023', color='blue', marker='o', linestyle='--', linewidth=2)
plt.xlabel('Meses')
plt.ylabel('Ventas (en miles)')
plt.title('Ventas Mensuales en 2023')
plt.legend()
plt.show()
```

Ilustración 23 Ejemplo para personalizar el estilo de la línea

Explicación de Parámetros:

- color='blue': Color de la línea.
- marker='o': Marcadores en cada punto de datos.
- linestyle='--': Estilo de la línea (en este caso, línea discontinua).
- linewidth=2: Grosor de la línea.

Paso 8: Ajustar los Ticks del Eje X

Puedes personalizar los ticks del eje X para mejorar la claridad.

```
# Crear gráfico de líneas con ticks personalizados
plt.plot(meses, ventas, label='Ventas 2023', color='blue', marker='o', linestyle='--', linewidth=2)
plt.xlabel('Meses')
plt.ylabel('Ventas (en miles)')
plt.title('Ventas Mensuales en 2023')
plt.xticks(rotation=45)  # Rotar los ticks del eje X para mejor legibilidad
plt.legend()
plt.show()
```

Ilustración 24 Ejemplo para personalizar ticks

Salida: Los ticks del eje X están rotados para evitar superposiciones.

Paso 9: Guardar el Gráfico

Puedes guardar el gráfico en un archivo para usarlo en informes o presentaciones.

```
# Guardar gráfico en un archivo
plt.plot(meses, ventas, label='Ventas 2023', color='blue', marker='o', linestyle='--', linewidth=2)
plt.xlabel('Meses')
plt.ylabel('Ventas (en miles)')
plt.title('Ventas Mensuales en 2023')
plt.xticks(rotation=45)
plt.legend()
plt.savefig('ventas_mensuales.png', dpi=300)  # Guardar como PNG con alta resolución
plt.show()
```

Ilustración 25 Ejemplo para guardar el gráfico en un archivo

Salida: El gráfico se guarda como ventas_mensuales.png en el directorio actual.

Conclusión del Ejercicio

- Aprendiste a crear un gráfico de líneas con Matplotlib.
- Personalizaste las etiquetas de los ejes, añadiste un título y una leyenda.
- Ajustaste el estilo de la línea y los ticks del eje X para mejorar la claridad.
- Guardaste el gráfico en un archivo para su uso posterior.