8. Gráficos

8.1. Gráficos 2D

8.1.1. Propiedades Líneas 2D
8.1.2. Modificadores de Estílo
8.1.3. Líneas y Disperción
8.1.4. Histograma

8.1. Gráficos 2D

En esta guía los gráficos se van a crear con el paquete Matplotlib, como se verá más adelante hay varios tipos de gráficos que se pueden crear, sin embargo aquí sólo se mostrarán algunos de ellos, si quiere investigar más lo invito a entrar a la galería del módulo.

Para crear un gráfico en 2D se usarán la funciones del módulo pyplot, cuya documentación puede encontrar aquí. En los apartados siguientes encontrará una recopilación de las funciones y métodos básicos para modificar el estilo y dos tipos de gráficos.

A continuación, el Mini_Ejemplo muestra cómo crear, mostrar y guardar un gráfico, argumento 'block' es False para que el código no deje de ejecutarse mientras se esté mostrando el gráfico.

```
# Mini_Ejemplo: Grafico_2D

import matplotlib.pyplot as plt

plt.plot([1,4,2,3,2,6,4,5,2,3,1])
plt.show(block=False)
plt.savefig("Grafico_Ejemplo.png")
plt.close()
```

```
# Pase aquí el mini ejemplo anterior, una vez ejecutado busque en la carpeta del notebook la im
```

8.1.1. Propiedades Líneas 2D

Las líneas tienen gran cantidad de propiedades con las que se pueden modificar su aspecto, a continuación se mencionaran algunas de ellas, sin embargo, para conocerlas todas diríjase aquí.

```
class matplotlib.lines.Line2D(xdata, ydata, linewidth=None, linestyle=None, color=None, marker=
```

- **linewidth**: Ancho de línea, se representa con un número entero.
- **linestyle**: Estilo de línea, puede tomar estos valores ['-', '--', '-.', ':'] entre otros.
- **color**: Color de la línea.
- **marker=None**: Pone un marcador en cada punto, algunos de los valores que puede tomar son ['.','o','s','*'], para ver la lista completa ingrese aquí
  - **markersize**: Tamaño del marcador
  - **markeredgecolor**: Color del contorno del marcador
  - **markerfacecolor**: Color del relleno del marcador

```
# Mini_Ejemplo: Linea_2D

import matplotlib.pyplot as plt

plt.plot([1,4,2,3,2,6,4,5,2,3,1],linewidth=4,linestyle=':',color='lightpink',marker='*',markers

plt.show(block=False)
plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

8.1.2. Modificadores de Estilo

En esta sección se van a ver las funciones para modificar el estilo del gráfico, como poner títulos, cuadriculas, leyendas, colores etc.

- figure()

```
matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, fram
```

Con esta función se puede modificar la figura donde se va dibujar el gráfico, los argumentos más importantes son:

- **num**: Número de gráficos que contendrá
- **figsize**: Tamaño de la figura, se representa como una lista [ancho x alto]
- **dpi**: Resolución.
- **facecolor**: color de la figura.
- **edgecolor**: color del borde
- **frameon**: Por defecto está inicializado con True, de lo contrario no se mostraría la figura.

```
import matplotlib.pyplot as plt

plt.figure(figsize=[10,2.5],dpi=150,facecolor="ivory")

plt.plot([1,4,2,3,2,6,4,5,2,3,1])
plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

- **axis()**

```
matplotlib.pyplot.axis(*args, **kwargs)
```

Con esta función se puede modificar los ejes del gráfico, además devuelve los límites de los ejes, sus argumentos son abiertos, entre los más comunes están:

- **limites**: lista con los límites de los ejes [xmin, xmax, ymin, ymax]
- **option**: Su valor puede ser booleano o str, si es False no se mostrarán los ejes, para conocer los valores que puede adoptar diríjase a la documentación.

```
import matplotlib.pyplot as plt

xmin, xmax, ymin, ymax=plt.axis([-2,15,-4,10])
print("La función devuelve los límites de los ejes: ",xmin, xmax, ymin, ymax,"\n")

plt.plot([1,4,2,3,2,6,4,5,2,3,1])
plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

- **xticks()**

```
matplotlib.pyplot.xticks(ticks=None, labels=None, **kwargs)
```

Con esta función se puede modificar etiquetas de las rejillas de los ejes, para el eje y se utiliza **yticks()**, además devuelve un array con las posiciones y una lista con las etiquetas.

- **ticks**: Es una lista con las posiciones de las etiquetas.
- **labels**: Es una lista con las etiquetas de la regla.

```
import matplotlib.pyplot as plt

p_x,e_x=plt.xticks(range(12),labels=['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'S

p_y,e_y=plt.yticks([1,1.5, 2, 2.3, 3, 3.3, 4.3, 4.7, 5, 6])

plt.plot([1.5,4.3,2,3.2,2,6,4.7,5,2,3,1,2.3])
plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

- **grid()**

```
matplotlib.pyplot.grid(b=None, which='major', axis='both', **kwargs)
```

Con esta función se puede mostrar y modificar una cuadrícula principal y otra secundaria, para esta última se debe activar los ticks secundarios con la función **minorticks_on()**.

- **b**: Booleano para activar o desactivar la cuadrícula.
- **which**: Tipo de cuadrícula, puede ser 'major' 'minor' o 'both'.
- **axis**: El eje al que se le va aplicar las líneas guía, puede ser 'x', 'y' o 'both'.
- **otros**: Propiedades de las líneas 2D

```
import matplotlib.pyplot as plt

plt.grid(b=True,color="darksalmon", alpha=0.5)

plt.minorticks_on()
p_y,e_y=plt.yticks([1,1.5, 2, 2.3, 3, 3.3, 4.3, 4.7, 5, 6])
plt.grid(True, which='minor', axis="y", color="linen")

plt.plot([1.5,4.3,2,3.2,2,6,4.7,5,2,3,1,2.3])
plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

- **title()**

```
matplotlib.pyplot.title(label, fontdict=None, loc='center', pad=None, **kwargs)[source]
```

Con esta función se le puede asignar y mostrar un título al gráfico, para hacer lo mismo con los ejes se utilizan las funciones **xlabel()** y **ylabel()**.

```
import matplotlib.pyplot as plt

plt.title("Gráfico de Ejemplo", fontsize=18)
plt.xlabel("Eje x", fontsize=14)
plt.ylabel("Eje y", fontsize=14)

plt.plot([1.5,4.3,2,3.2,2,6,4.7,5,2,3,1,2.3])
plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

- **legend()**

```
matplotlib.pyplot.legend(*args, **kwargs)
```

Con esta función se puede mostrar un cuadro con la leyenda de las series que están dibujadas en los gráficos.

```
import matplotlib.pyplot as plt

plt.title("Gráfico de Ejemplo", fontsize=18)
plt.xlabel("Eje x", fontsize=14)
plt.ylabel("Eje y", fontsize=14)

plt.plot([1.5,4.3,2,3.2,2,6,4.7,5,2,3,1,2.3], label="Serie 1")
plt.plot([9.91, 5.82, 0.86, 5.12, 9.28, 4.63, 2.61, 6.97, 8.48, 6.15, 4.21, 2.8], label="Serie

plt.legend()
plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

- **subplot()**

```
matplotlib.pyplot.subplot(*args, **kwargs)
```

Con esta función se puede dibujar varios gráficos en una sola figura.

- ***args**: Número entero de tres dígitos, donde el primero indica el número de filas, el segundo el de columnas y el tercero el índice para ubicar el gráfico.

```
import matplotlib.pyplot as plt

plt.figure(num=2,figsize=[10,4])

plt.subplot(121)
plt.title("Ejemplo Gráfico 1")
plt.plot([1.5,4.3,2,3.2,2,6,4.7,5,2,3,1,2.3])

plt.subplot(122)
plt.title("Ejemplo Gráfico 2")
plt.plot([9.91, 5.82, 0.86, 5.12, 9.28, 4.63, 2.61, 6.97, 8.48, 6.15, 4.21, 2.8],color="salmon"

plt.show(block=False)
plt.close()
```

```
# Pase el código anterior aquí
```

8.1.3. Líneas y Dispersión

La función plot() sirve para crear diferentes gráficos entre los más importantes los de línea y dispersión, aquí puede encontrar la documentación oficial de esta función.

```
matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)
```

- ***arg**: Listas de coordenadas 'x' y 'y'.

Cuando se ingresan dos listas la función las interpreta como las coordenadas de una serie de puntos, la primera siempre va a ser 'x' y la segunda 'y'.

```
# Mini_Ejemplo: Cuadrado.
import matplotlib.pyplot as plt

plt.figure(figsize=[5,5])
plt.axis([-1,5,-1,5])

plt.plot([0,0,4,4,0],[0,4,4,0,0],color='dodgerblue',marker='o')

plt.show(block=False)

plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

Gráfico de Líneas

Cuando solo se ingresa una lista de valores, la función hace de cuenta que se ha ingresado la coordenada 'y' y que la coordenada x está compuesta por una lista de 0 hasta la cantidad de datos que tenga la lista ingresada.

```
# Mini_Ejemplo: Ventas Mensuales
import matplotlib.pyplot as plt
```

```
plt.title("Ventas Mensuales 2006")
plt.ylabel("Millones de pesos")

plt.xticks(range(12),labels=['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep','Oct
plt.plot([209,135,181,250,230,300,267,185,148,240,170,310],marker="o",color='darkmagenta')

plt.show(block=False)
plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

Gráfico de Dispersión

Cuando solo se usan los marcadores y se omite la línea, se puede obtener un diagrama de dispersión.

```
# Mini_Ejemplo: Gráfico de Dispersión

import matplotlib.pyplot as plt

x=[1.16, 6.43, 8.63, 2.62, 2.91, 9.54, -0.02, 4.75, 6.89, 0.15, 0.98, 4.69, 0.84, 9.54, 0.57, 2
y=[6.45, 17.72, 21.96, 9.4, 10.47, 23.17, 5.34, 14.45, 18.48, 5.17, 6.33, 13.0, 6.64, 23.69, 6.

plt.title("Gráfico de Dispersión")
plt.ylabel("Eje x")
plt.ylabel("Eje y")

plt.plot(x,y,"bo",color='darkmagenta')

plt.show(block=False)
plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

8.1.4. Histograma

Con la función hist() se puede crear un histograma, además devuelve las frecuencias , las clases, y una lista usada para crearlo, para más información de la función ingrese aquí para ver la documentación oficial.

```
matplotlib.pyplot.hist(x, bins=None, range=None, density=None, weights=None, cumulative=False,
```

- **x**: Lista de datos.
- **bins**: Número de clases.
- **density**: Si es True se crea un histograma normalizado, es decir, a partir de las frecuencias.
- **cumulative**: Si es True se crea un histograma acumulativo.
- **color**: Color del histograma.
- **label**: Nombre de la serie.

```
# Mini_Ejemplo: Histograma.
import matplotlib.pyplot as plt

plt.title("Hitograma",fontsize=18)
plt.ylabel("Frecuencia",fontsize=14)
plt.xlabel("Datos",fontsize=14)

histograma=plt.hist([2,1,3,4,2,3,4,3,5,3],5,edgecolor='dodgerblue',color='dodgerblue',alpha=0.5

frecuencias=histograma[0]
clases=histograma[1]
print("Las frecuencias de los datos son: ",frecuencias)
print("Las clases del histograma son: ",clases,"\n")

plt.xticks(clases)
plt.show(block=False)

plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

Histograma Normalizado

```
# Mini_Ejemplo: Histograma Normalizado.
import matplotlib.pyplot as plt

plt.title("Hitograma Normalizado",fontsize=18)
plt.ylabel("Frec. Relativa",fontsize=14)
plt.xlabel("Datos",fontsize=14)
```

```
histograma=plt.hist([2,1,3,4,2,3,4,3,5,3],5,density=True,edgecolor='orange',color='orange',alph

clases=histograma[1]

plt.xticks(clases)
plt.show(block=False)

plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

Histograma Acumulativo

```
# Mini_Ejemplo: Histograma Acumulativo.
import matplotlib.pyplot as plt

plt.title("Hitograma Acumulativo",fontsize=18)
plt.ylabel("Frec. Relativa",fontsize=14)
plt.xlabel("Datos",fontsize=14)

histograma=plt.hist([2,1,3,4,2,3,4,3,5,3],5,cumulative=True,edgecolor='g',color='g',alpha=0.5)

clases=histograma[1]

plt.xticks(clases)
plt.show(block=False)

plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

Histograma Comparativo

```
# Mini_Ejemplo: Histograma Comparativo.
import matplotlib.pyplot as plt

plt.title("Hitograma Comparativo",fontsize=18)
plt.ylabel("Frec. Relativa",fontsize=14)
plt.xlabel("Datos",fontsize=14)

plt.hist([2,1,3,4,2,3,4,3,5,3],5,edgecolor='dodgerblue',color='dodgerblue',alpha=0.5)
```

```
plt.hist([2,1,3,4,2,3,4,3,5,3],5,cumulative=True,edgecolor='yellowgreen',color='yellowgreen',al

plt.xticks(clases)
plt.show(block=False)

plt.close()
```

```
# Pase aquí el mini ejemplo anterior
```

Anterior - *Siguiente*

Home