GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN

CIENCIA DE DATOS

Agenda
Sesión 9/18

Matplotlib

- ¿Qué es **Matplotlib** ?
- Uso de Matplotlib
- Pyplot
- Dispersión
- Barras
- Pastel

¿Qué es Matplotlib?

Matplotlib es una biblioteca de trazado de gráficos de bajo nivel en Python que sirve como una utilidad de visualización.

Matplotlib fue creado por John D. Hunter.

Matplotlib es de código abierto y podemos usarlo libremente.

Matplotlib está escrito principalmente en Python, algunos segmentos están escritos en C, Objective-C y Javascript para compatibilidad con plataformas.

¿Qué es Matplotlib?

¿Dónde está la base de código de Matplotlib?

El código fuente de Matplotlib se encuentra en este repositorio de github

https://github.com/matplotlib/matplotlib

¿Qué es Matplotlib?

Instalación de Matplotlib

Si ya tiene Python y PIP instalados en un sistema, la instalación de Matplotlib es muy fácil.

Instálelo usando este comando:

pip install matplotlib

Uso de Matplotlib

Una vez que Matplotlib esté instalado, impórtelo en sus aplicaciones agregando la declaración:

```
import module
```

Ahora Matplotlib está importado y listo para usar:

```
import matplotlib
print(matplotlib.__version__)
```

Pyplot

La mayoría de las utilidades de Matplotlib se encuentran en el submódulo pyplot y, por lo general, se importan con el alias plt:

import matplotlib.pyplot as plt

Ahora se puede hacer referencia al paquete Pyplot como plt.

Pyplot

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

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

Pyplot

Graficar puntos X y Y

La función plot() se utiliza para dibujar puntos (marcadores) en un diagrama.

De forma predeterminada, la función plot() dibuja una línea de un punto a otro.

La función toma parámetros para especificar puntos en el diagrama.

Pyplot

Graficar puntos X y Y

El parámetro 1 es una matriz que contiene los puntos en el eje x .

El parámetro 2 es una matriz que contiene los puntos en el eje y .

Si necesitamos trazar una línea de (1, 3) a (8, 10), tenemos que pasar dos matrices [1, 8] y [3, 10] a la función de trazado.

Pyplot

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

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

El eje x es el eje horizontal.

El eje y es el eje vertical.

Pyplot

Trazado sin línea

Para trazar solo los marcadores, puede usar el parámetro de notación de cadena de método abreviado 'o', que significa 'anillos'.

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

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, 'o')
plt.show()
```

Pyplot

Varios puntos

Puede trazar tantos puntos como desee, solo asegúrese de tener el mismo número de puntos en ambos ejes.

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

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

Pyplot

Marcadores

Puede utilizar el argumento de la palabra clave markerpara enfatizar cada punto con un marcador específico:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o')
plt.show()
```

Pyplot

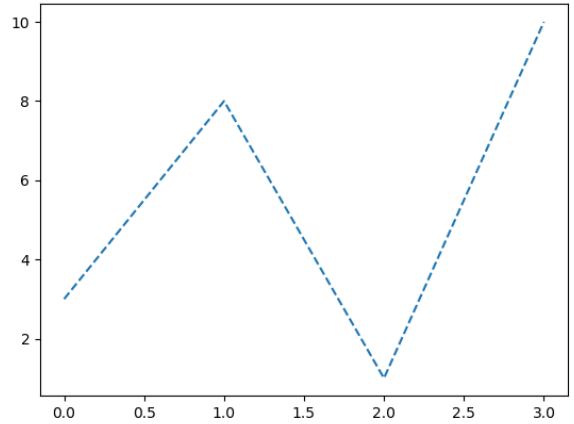
Estilo de línea

Puede usar el argumento de palabra clave linestyle, o más corto ls, para cambiar el estilo de la línea trazada:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

Pyplot

plt.plot(ypoints, linestyle = 'dashed')



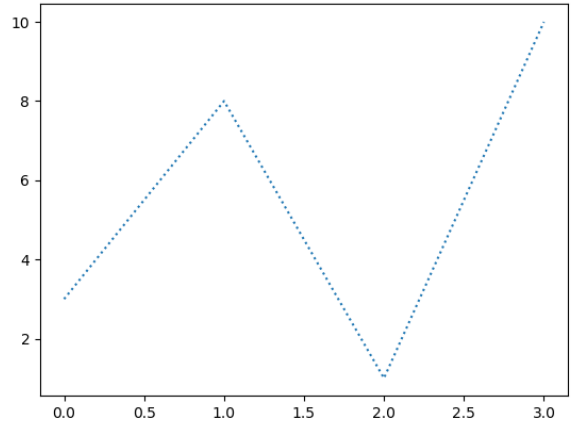
Sintaxis más corta

El estilo de línea se puede escribir en una sintaxis más corta:

Linestyle se puede escribir como ls.

Dotted se puede escribir como :.

Dashed se puede escribir como --



Pyplot

Color de linea

Puede usar el argumento de palabra clave coloro el más corto cpara establecer el color de la línea:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```

Pyplot

Color de linea

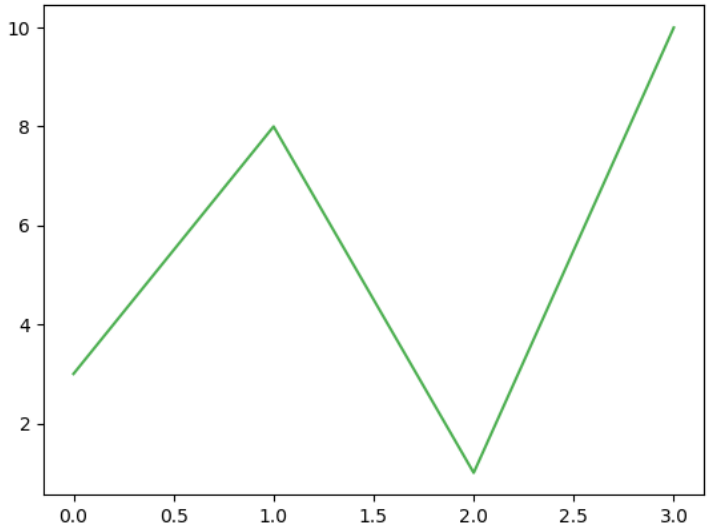
Puede usar el argumento de palabra clave coloro el más corto cpara establecer el color de la línea:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color = 'r')
plt.show()
```

Pyplot

Color de linea

plt.plot(ypoints, c = '#4CAF50')



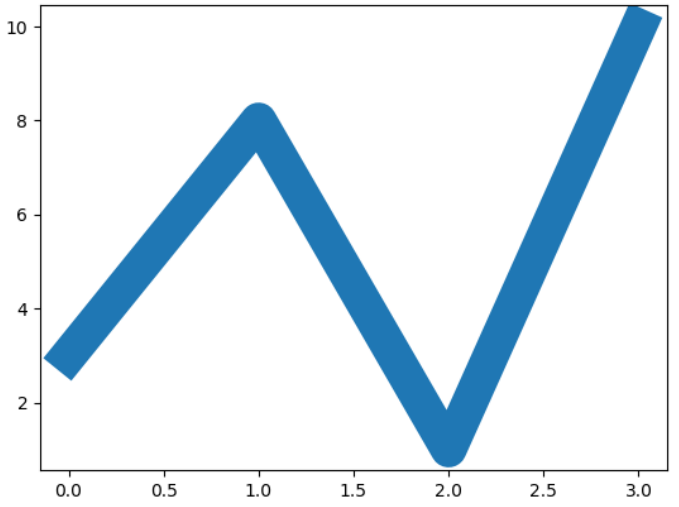
Pyplot

Grosor de línea

Puede usar el argumento de palabra clave linewidtho el más corto lwpara cambiar el ancho de la línea.

El valor es un número flotante, en puntos:

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

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linewidth = '20.5')
plt.show()
```



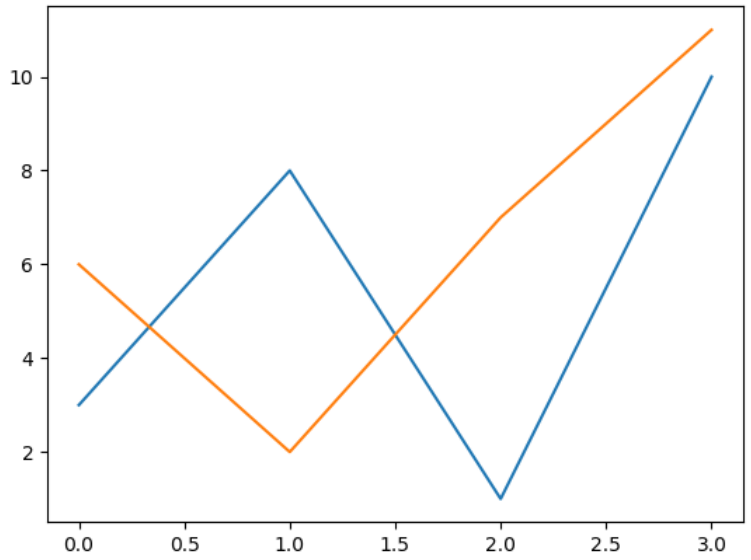
Pyplot

Varias líneas

Puede trazar tantas líneas como desee simplemente agregando más funciones plt.plot() :

Dibuja dos líneas especificando una función plt.plot() para cada línea:

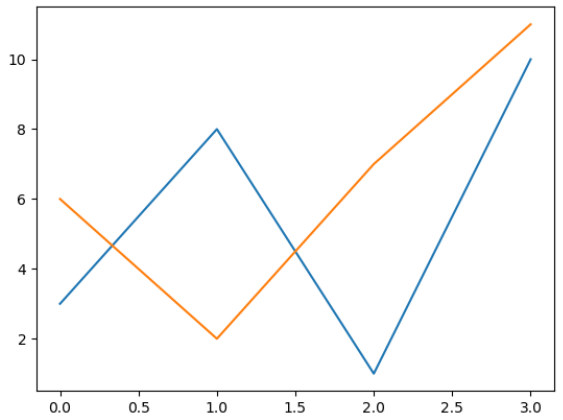
```
import matplotlib.pyplot as plt
import numpy as np
y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])
plt.plot(y1)
plt.plot(y2)
plt.show()
```



Pyplot

También puede trazar muchas líneas agregando los puntos para el eje xey para cada línea en la misma función plt.plot().

(En los ejemplos anteriores, solo especificamos los puntos en el eje y, lo que significa que los puntos en el eje x obtuvieron los valores predeterminados (0, 1, 2, 3)).



Pyplot

Crear etiquetas para una parcela

Con Pyplot, puede usar las funciones xlabel() y ylabel() para establecer una etiqueta para los ejes xey.

```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.show()
```

Pyplot

Agregar líneas de cuadrícula a una gráfica

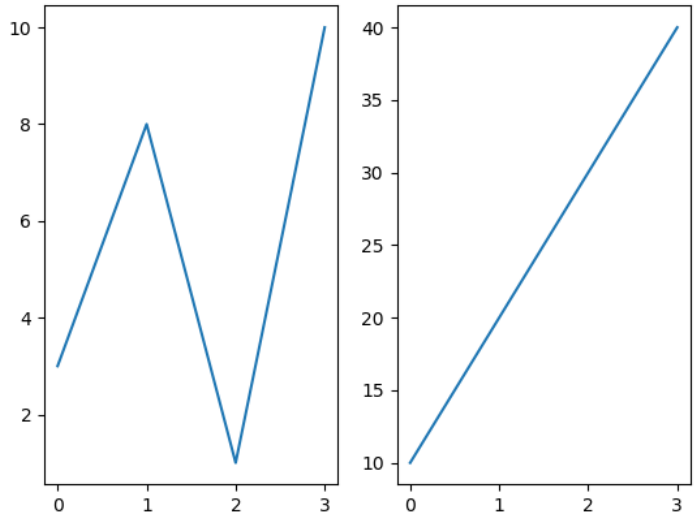
Con Pyplot, puede usar la función grid() para agregar líneas de cuadrícula al gráfico.

```
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

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

Pyplot

Mostrar múltiples gráficos

Con la función subplots() puede dibujar múltiples gráficos en una figura:



Pyplot

Mostrar múltiples gráficos

```
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```

Pyplot

La función subplots ()

La función subplots() toma tres argumentos que describen el diseño de la figura.

El diseño está organizado en filas y columnas, que están representadas por el primer y segundo argumento.

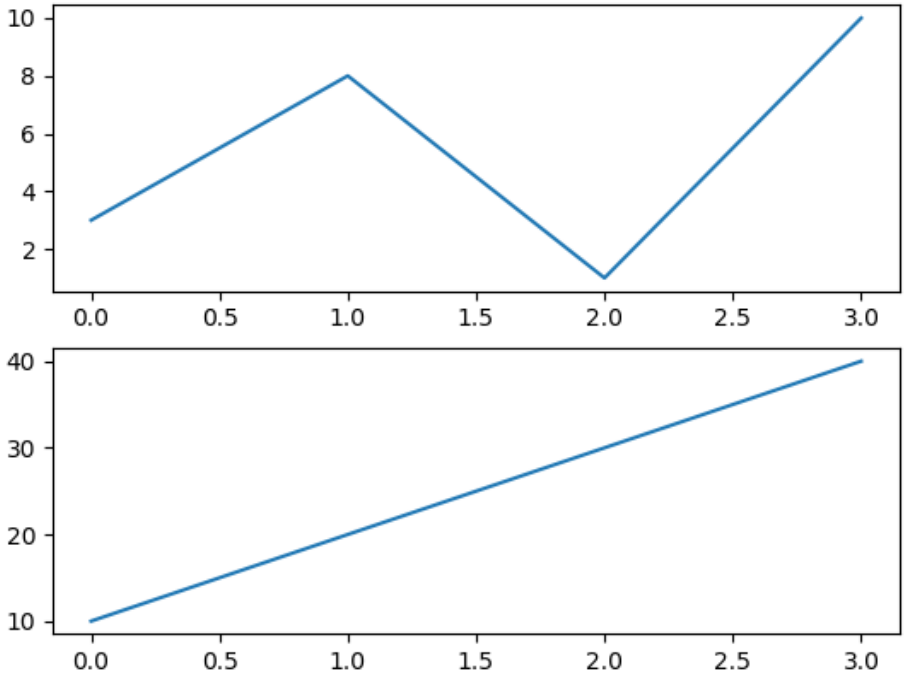
El tercer argumento representa el índice de la trama actual.

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
```

Pyplot

La función subplots ()

```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 1, 1)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y)

plt.show()
```



Dispersión

Crear gráficos de dispersión

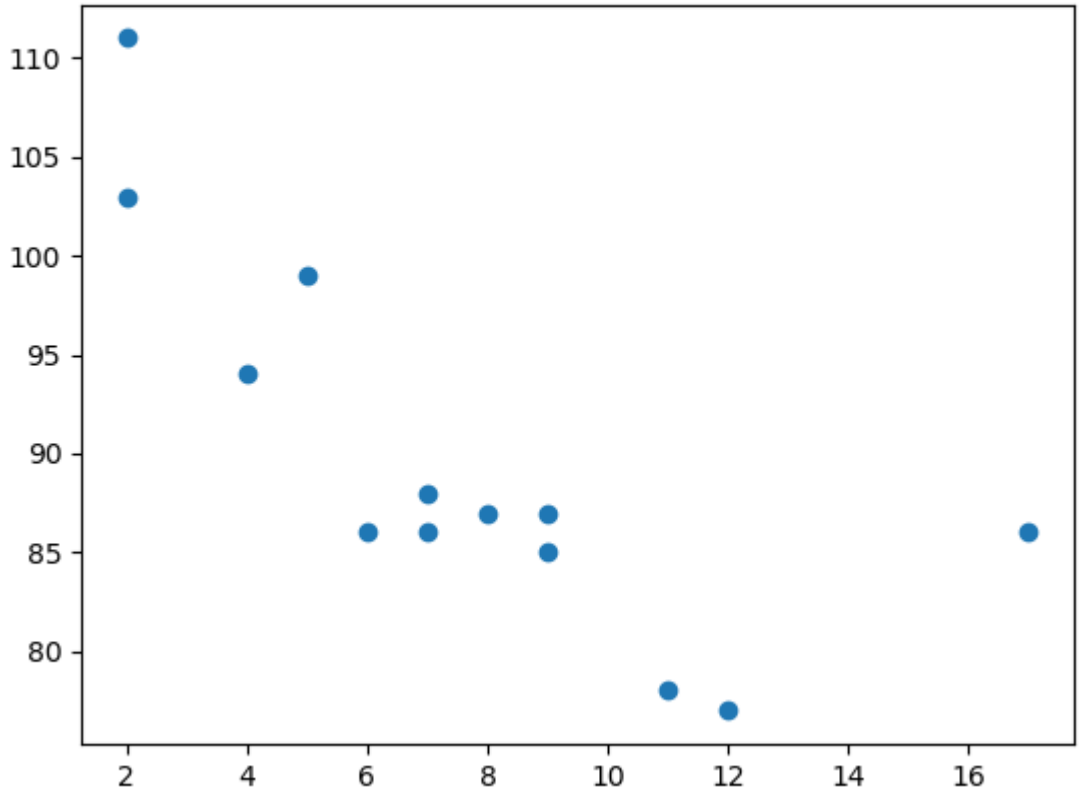
Con Pyplot, puede usar la scatter()función para dibujar un diagrama de dispersión.

La función scatter() traza un punto para cada observación. Necesita dos matrices de la misma longitud, una para los valores del eje xy otra para los valores del eje y:

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
plt.show()
```

Dispersión

Crear gráficos de dispersión



Dispersión

Crear gráficos de dispersión

La observación del ejemplo anterior es el resultado del paso de 13 automóviles.

El eje X muestra la antigüedad del automóvil.
El eje Y muestra la velocidad del automóvil cuando pasa.

¿Existe alguna relación entre las observaciones?

Parece que cuanto más nuevo es el automóvil, más rápido conduce, pero eso podría ser una coincidencia, después de todo, solo registramos 13 automóviles.

Dispersión

Comparar parcelas

En el ejemplo anterior, parece haber una relación entre la velocidad y la edad, pero ¿qué pasa si también graficamos las observaciones de otro día? ¿El diagrama de dispersión nos dirá algo más?

Al comparar los dos gráficos, creo que es seguro decir que ambos nos dan la misma conclusión:

cuanto más nuevo es el automóvil, más rápido conduce.

Dispersión

Colores

Puede establecer su propio color para cada diagrama de dispersión con el color c argumento o :

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y, color = 'hotpink')

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y, color = '#88c999')

plt.show()
```

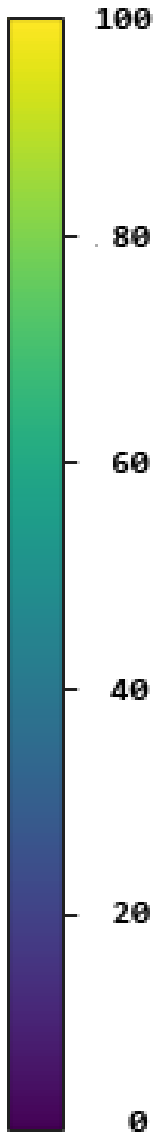
Dispersión

Mapa de colores

El módulo Matplotlib tiene varios mapas de colores disponibles.

Un mapa de colores es como una lista de colores, donde cada color tiene un valor que va de 0 a 100.

A continuación, se muestra un ejemplo de mapa de colores:



Dispersión

Cómo utilizar ColorMap

Puede especificar el mapa de colores con el argumento de palabra clave cmapcon el valor del mapa de colores, en este caso, 'viridis'que es uno de los mapas de colores incorporados disponibles en Matplotlib.

Además, debe crear una matriz con valores (de 0 a 100), un valor para cada uno de los puntos en el diagrama de dispersión

Dispersión

Cómo utilizar ColorMap

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='viridis')

plt.show()
```

Dispersión

Tamaño

Puede cambiar el tamaño de los puntos con el argumento s.

Al igual que los colores, asegúrese de que la matriz de tamaños tenga la misma longitud que las matrices de los ejes x e y:

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes)

plt.show()
```

Dispersión

Alfa

Puede ajustar la transparencia de los puntos con el argumento alpha.

Al igual que los colores, asegúrese de que la matriz de tamaños tenga la misma longitud que las matrices de los ejes x e y:

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes, alpha=0.5)
plt.show()
```

Dispersión

Alfa

Puede ajustar la transparencia de los puntos con el argumento alpha.

Al igual que los colores, asegúrese de que la matriz de tamaños tenga la misma longitud que las matrices de los ejes x e y:

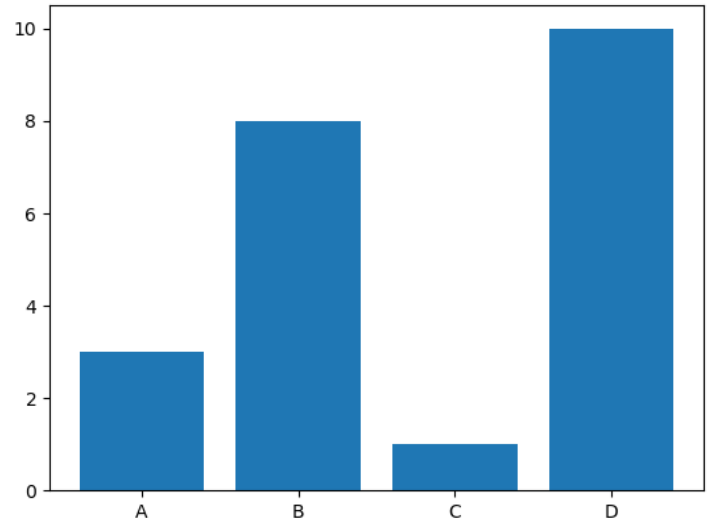
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes, alpha=0.5)
plt.show()
```

Barras

Creando barras

Con Pyplot, puede usar la función bar() para dibujar gráficos de barras:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

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



Barras

La función bar() toma argumentos que describen el diseño de las barras.

Las categorías y sus valores representados por el primer y segundo argumento como matrices.

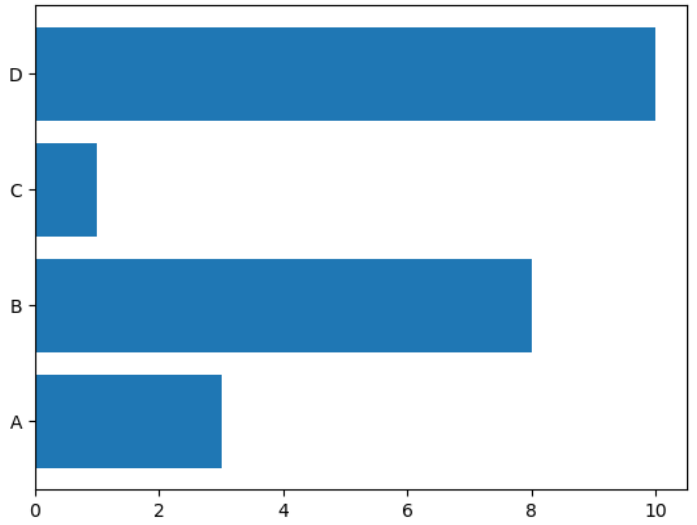
```
x = ["APPLES", "BANANAS"]
y = [400, 350]
plt.bar(x, y)
```

Barras

Barras horizontales

Si desea que las barras se muestren horizontalmente en lugar de verticalmente, use la función barh() :

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

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

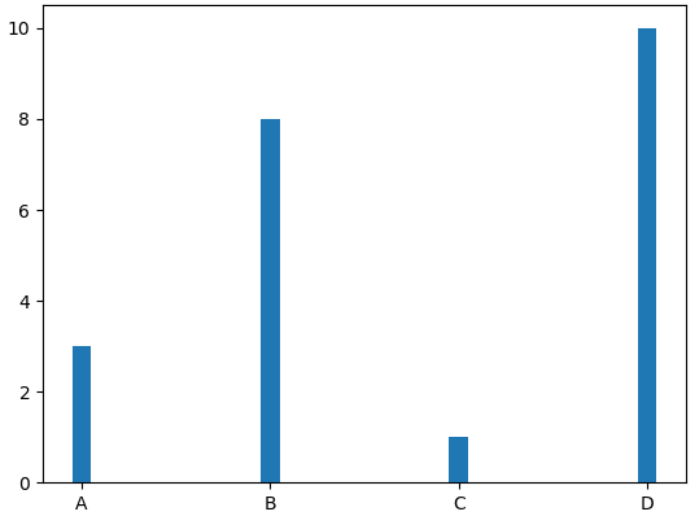


Barras

Ancho de barra

El bar() toma el argumento de palabra clave width para definir el ancho de las barras:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, width = 0.1)
plt.show()
```



Barras

Histograma

Un histograma es un gráfico que muestra distribuciones de frecuencia .

Es un gráfico que muestra el número de observaciones dentro de cada intervalo dado.

Ejemplo: digamos que pregunta por la altura de 250 personas, podría terminar con un histograma

Barras

Ejemplo: digamos que pregunta por la altura de 250 personas, podría terminar con un histograma:

2 personas de 140 a 145cm

5 personas de 145 a 150cm

15 personas de 151 a 156cm

31 personas de 157 a 162cm

46 personas de 163 a 168cm

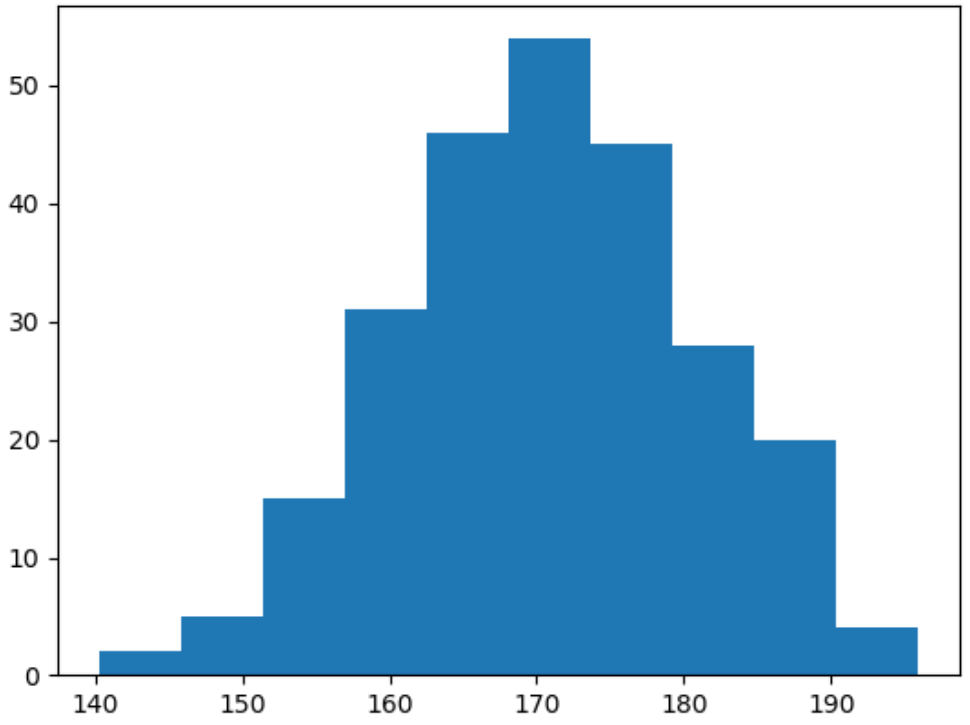
53 personas de 168 a 173cm

45 personas de 173 a 178cm

28 personas de 179 a 184cm

21 personas de 185 a 190cm

4 personas de 190 a 195cm



Barras

Crear histograma

En Matplotlib, usamos la función hist() para crear histogramas.

La función hist() usará una matriz de números para crear un histograma, la matriz se envía a la función como un argumento.

Para simplificar, usamos NumPy para generar aleatoriamente una matriz con 250 valores, donde los valores se concentrarán alrededor de 170 y la desviación estándar es 10.

Barras

Una distribución de datos normal por NumPy:

```
import numpy as np
x = np.random.normal(170, 10, 250)
print(x)
#Imprimirá 250 valores aleatorios


plt.hist(x)
plt.show()
```

Pastel

Crear gráficos circulares

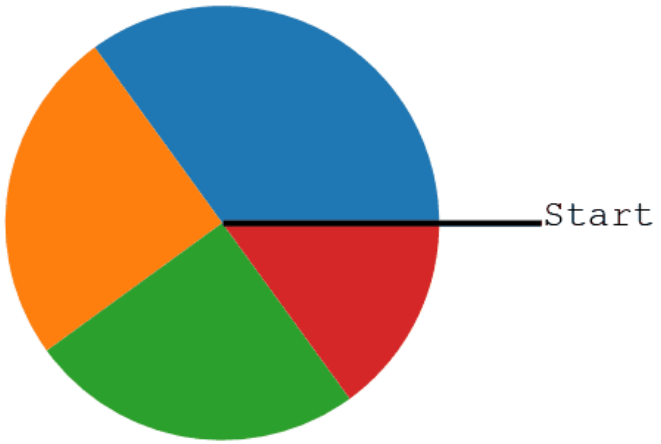
Con Pyplot, puede usar la función pie() para dibujar gráficos circulares:

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

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

Pastel

Como puede ver, el gráfico circular dibuja una pieza (llamada cuña) para cada valor en la matriz (en este caso [35, 25, 25, 15]).

De forma predeterminada, el trazado de la primera cuña comienza desde el eje x y se mueve en sentido antihorario :



Pastel

Etiquetas

Agregue etiquetas al gráfico circular con el parámetro label.

El parámetro label debe ser una matriz con una etiqueta para cada sector:

```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```

Pastel

Leyenda con encabezado

Para agregar un encabezado a la leyenda, agregue el titleparámetro a la legend función.

```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.legend(title = "Four Fruits:")
plt.show()
```

RESUMEN DE CLASE

GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN