10.Matplotlib

September 2, 2024

1 Matplotlib

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

Matplotlib es de código abierto y fue creado por John D. Hunter.

Está escrito principalmente en Python, algunos segmentos están escritos en C, Objective-C y Javascript.

1.1 Comprobar versión de matplotlib

Una vez que `matplotlib` esté instalado es necesario importarlo. Se puede verificar la versión del paquete con el comando.

[ ]:
```
import matplotlib
print(matplotlib.__version__)
```

```
3.8.3
```

1.2 Pyplot

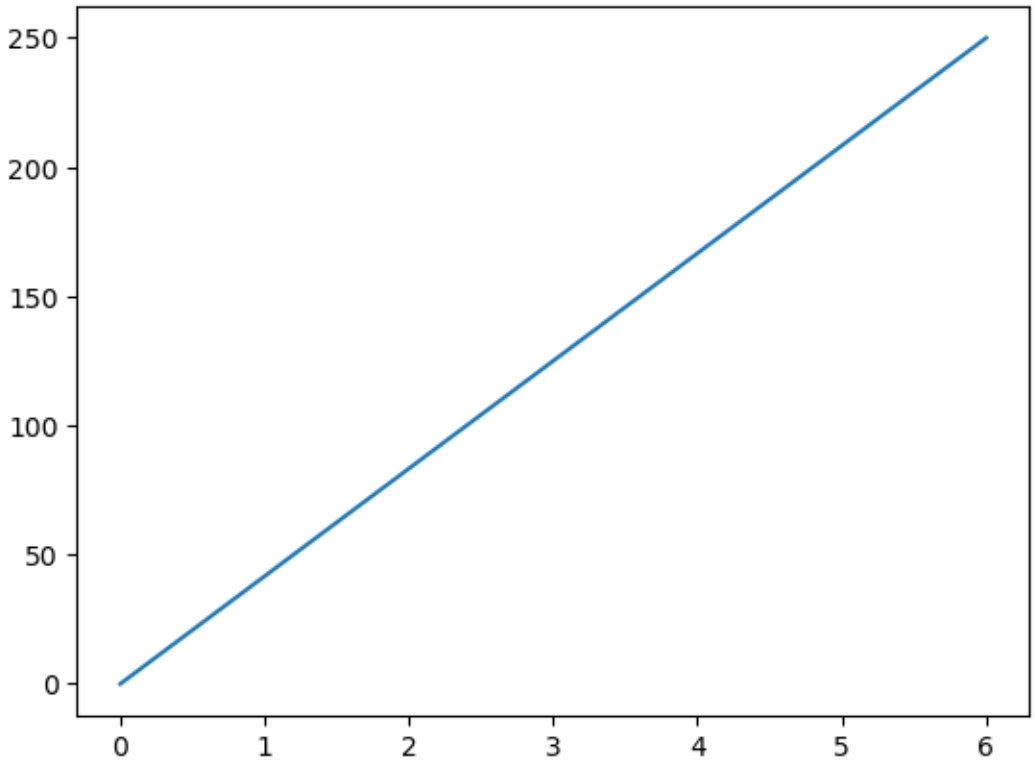
La mayoría de las utilidades de `Matplotlib` se encuentran bajo el submódulo `pyplot`, y generalmente se importan bajo el alias `plt`:

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

Ahora el paquete `Pyplot` se puede denominar `plt`.

Ejemplo. Dibuja una línea en un diagrama desde la posición $(0, 0)$ hasta la posición $(6, 250)$.

[ ]:
```
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()
```



1.3 Graficas con matplotlib

1.3.1 Gráfica puntos (x, y)

La función `plot()` se utiliza para dibujar puntos en un diagrama. Por defecto, `plot()` dibuja una línea de punto a punto.

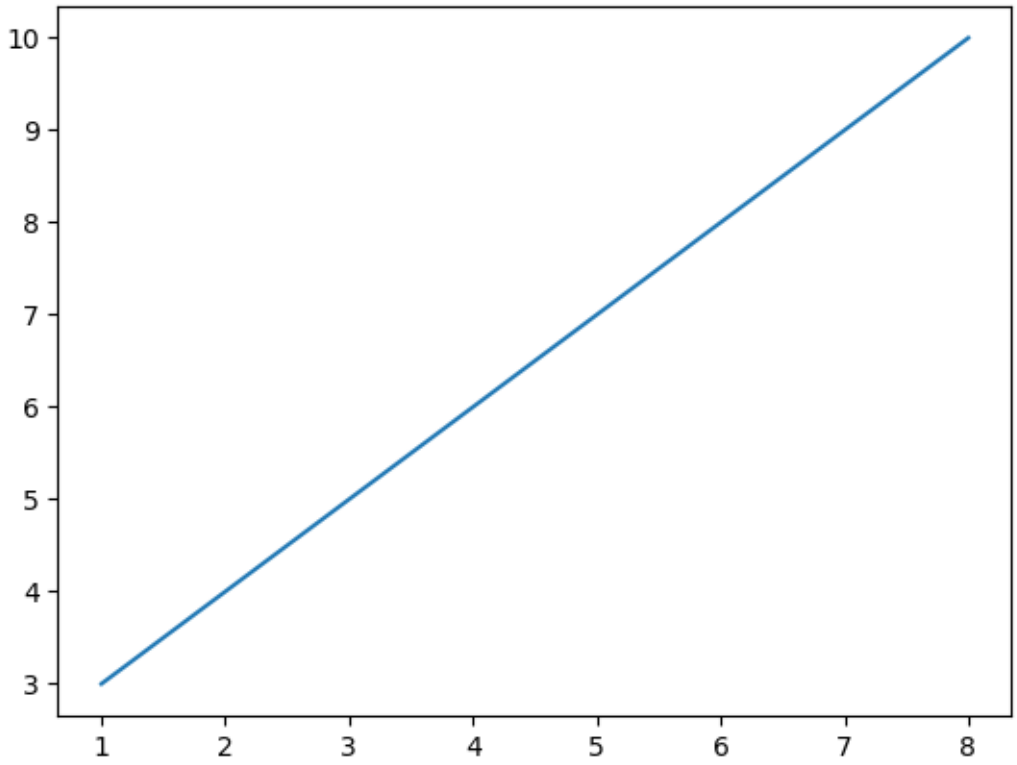
La función toma parámetros para especificar puntos en el diagrama.

- 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.

Ejemplo. Dibuje una línea en un diagrama desde la posición (1, 3) a la posición (8, 10)

[ ]:
```
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()
```

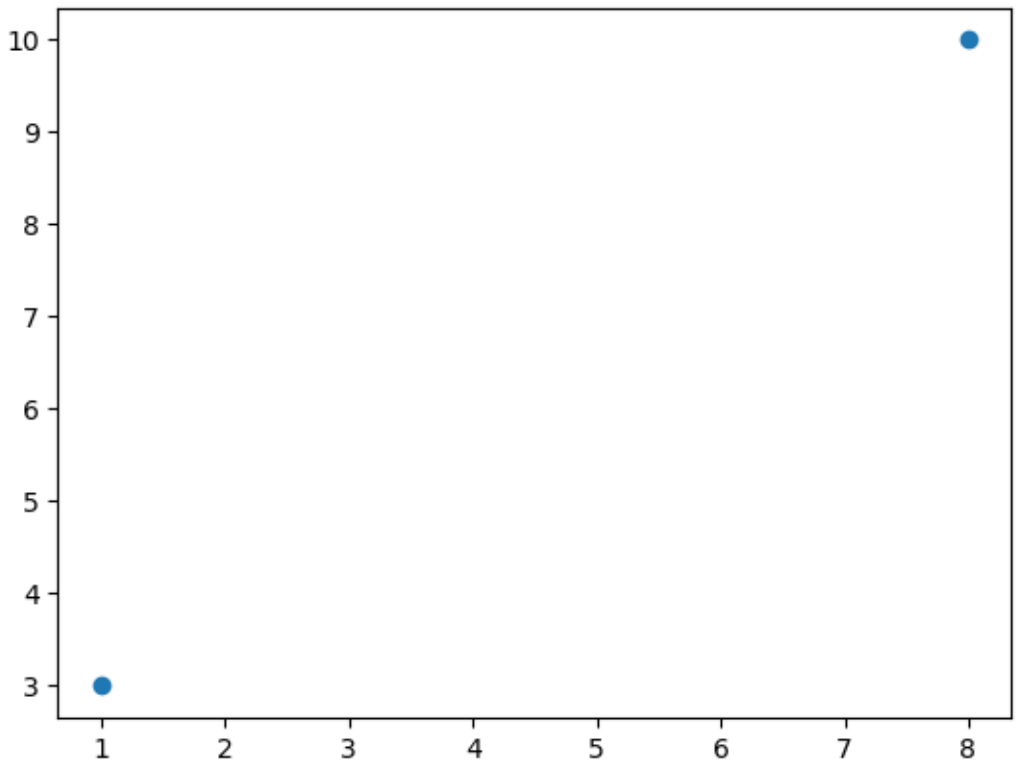


1.4 Gráfica Sin Línea

Para trazar solo los marcadores, puede usar notación de cadena de acceso directo parámetro `'o'`, que significa *'anillos'*.

Ejemplo. Dibuja dos puntos en el diagrama, uno en posición (1, 3) y otro en posición (8, 10).

[ ]:
```
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()
```

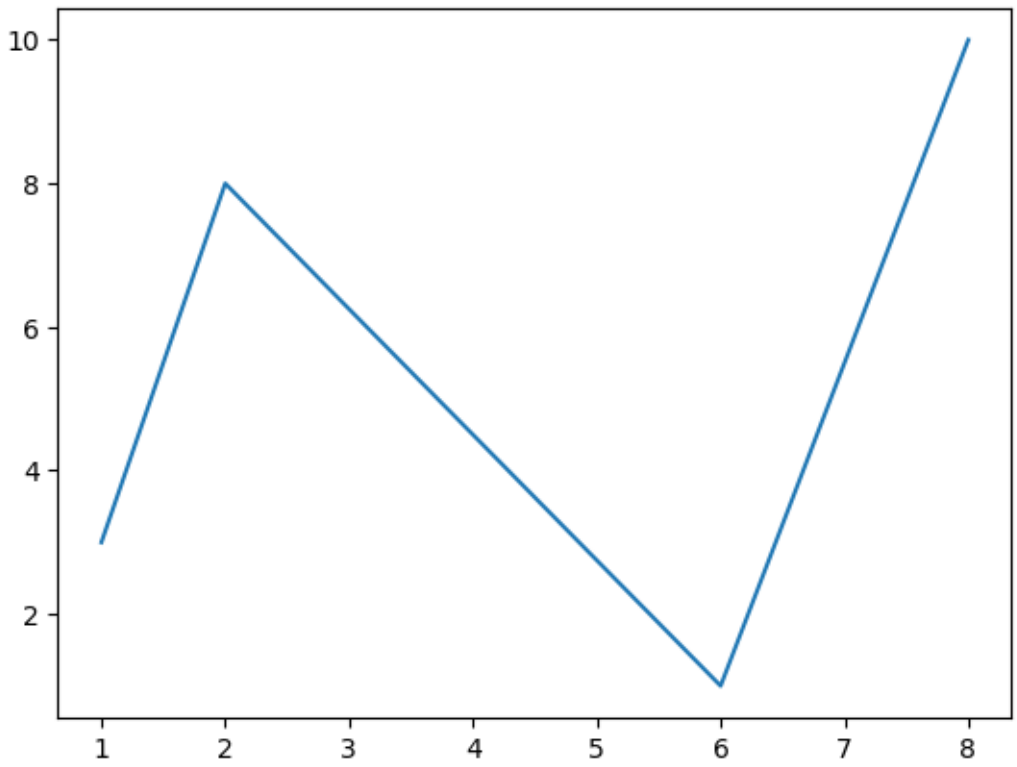


1.5 Múltiples Puntos

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

Ejemplo. Dibuja una línea en un diagrama desde la posición $(1, 3)$ a $(2, 8)$ luego a $(6, 1)$ y finalmente a la posición $(8, 10)$

[ ]:
```
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()
```



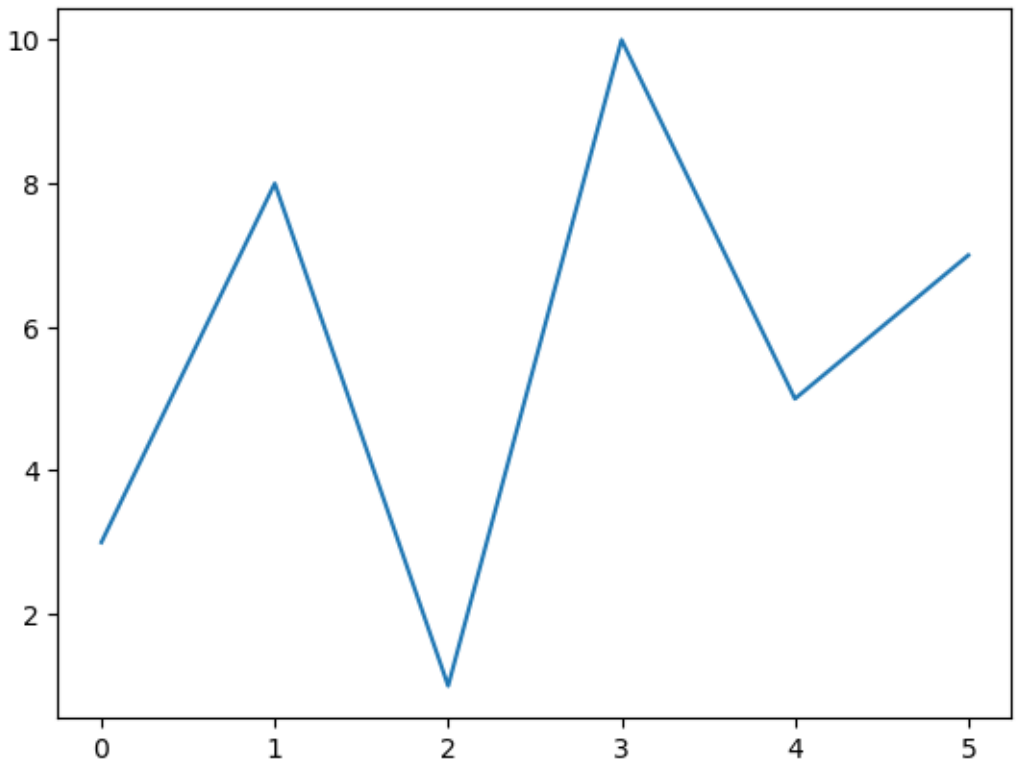
1.6 Valores x predeterminados

Si no especificamos los puntos en el eje x, obtendrán los valores predeterminados $0, 1, 2, 3, ...$, dependiendo de la longitud de los puntos y.

Entonces, si tomamos el mismo ejemplo que el anterior y dejamos de lado los puntos x, el diagrama se verá así:

Ejemplo. Gráfica sin valores x.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

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

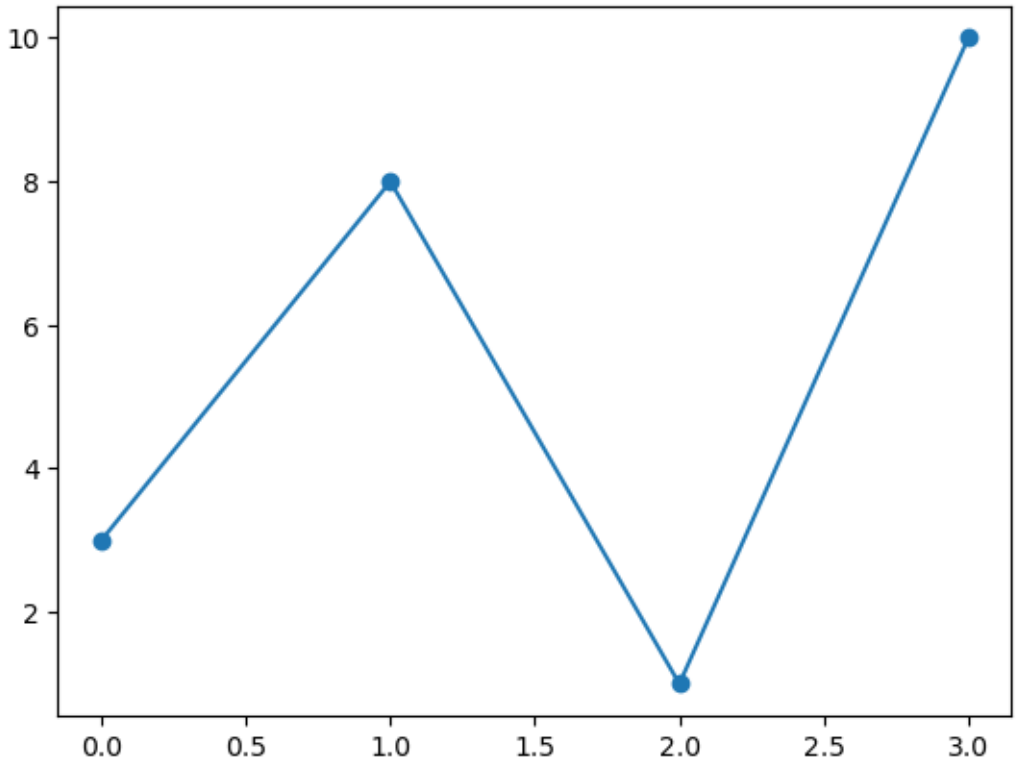


1.7 Marcadores

Puede usar el argumento `marker` para enfatizar cada punto con un marcador especifico.

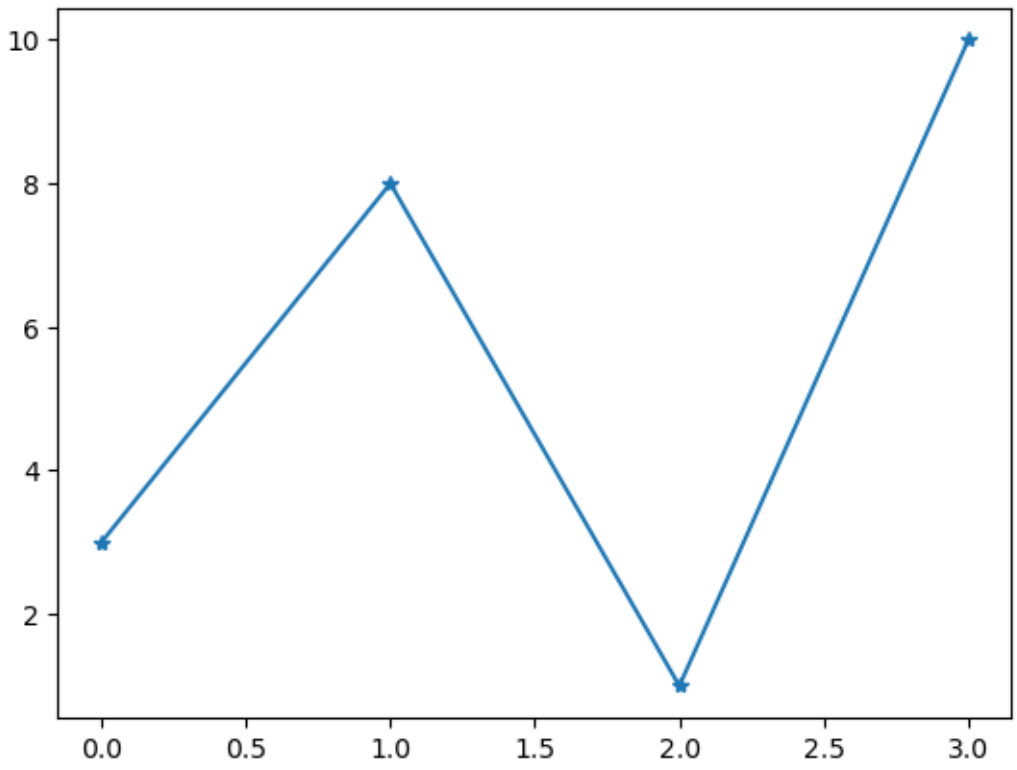
Ejemplo. Marcar cada punto con un círculo.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

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



Ejemplo. Marca cada punto con una estrella.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

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



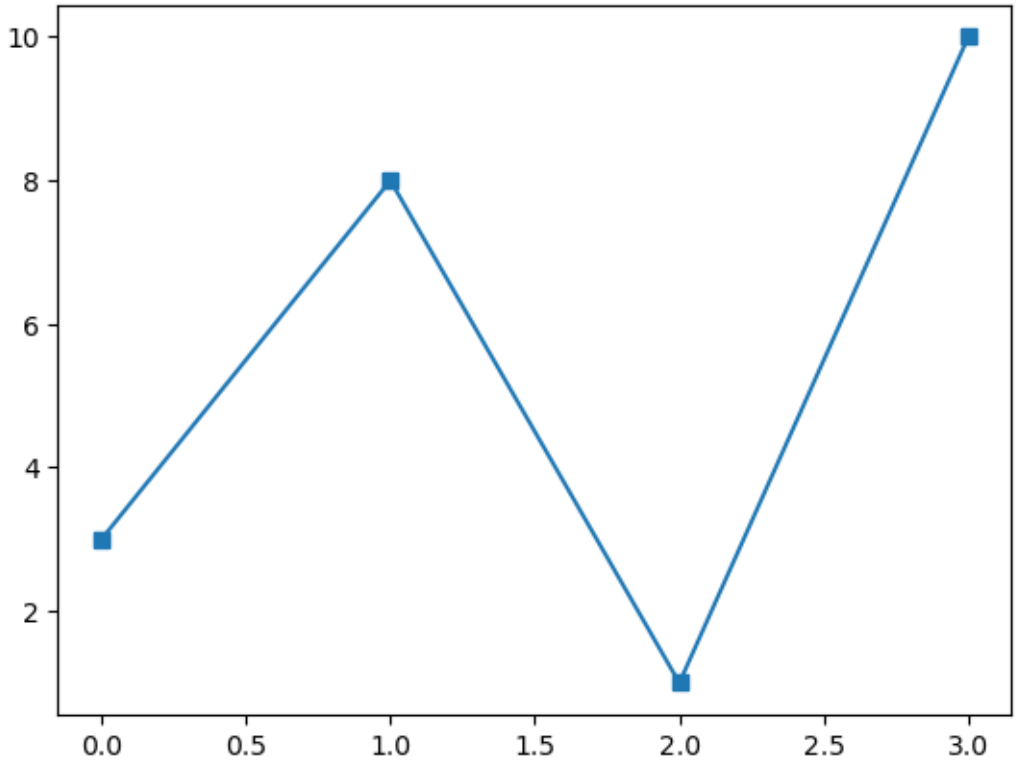
Referencia de los marcadores

Puede elegir cualquiera de estos marcadores:

Marker	Description
‘o’	Circle
’*’	Star
‘.’	Point
‘,’	Pixel
‘x’	X
‘X’	X (filled)
‘+’	Plus
‘P’	Plus (filled)
‘s’	Square
‘D’	Diamond
‘d’	Diamond (thin)
‘p’	Pentagon
‘H’	Hexagon
‘h’	Hexagon
‘v’	Triangle Down
‘^’	Triangle Up
‘<’	Triangle Left

Marker	Description
'>'	Triangle Right
'1'	Tri Down
'2'	Tri Up
'3'	Tri Left
'4'	Tri Right
' '	' '
'_'	Hline

```
[ ]: plt.plot(ypoints, marker = 's')
     plt.show()
```



1.8 Formato cadena `fmt`

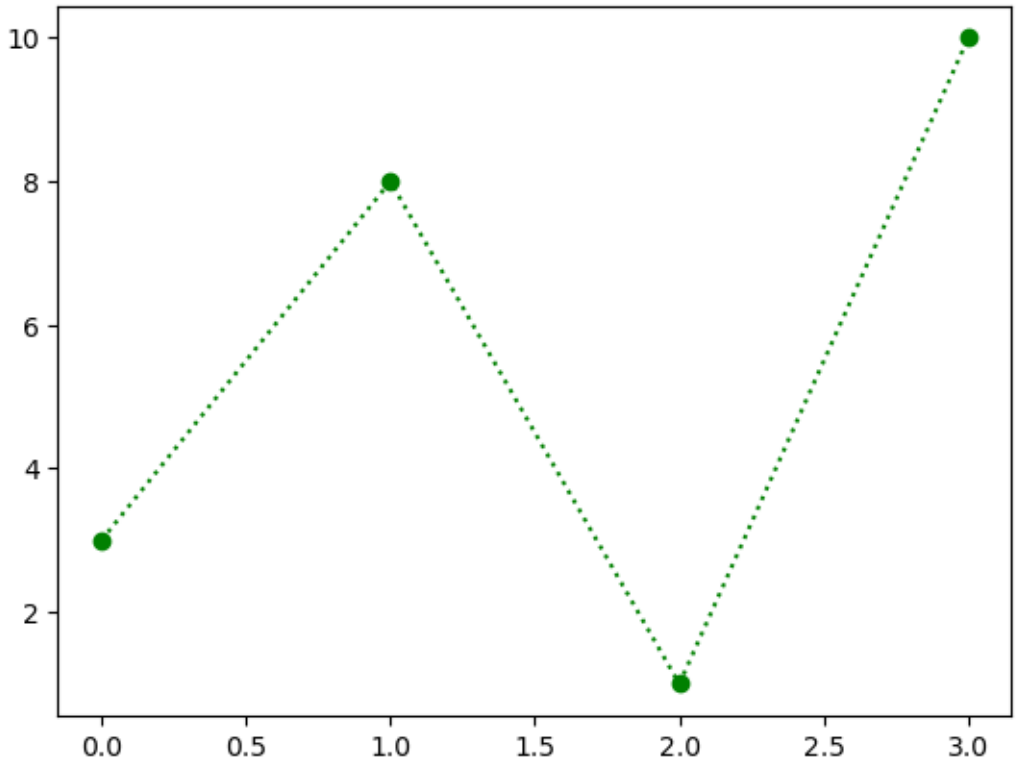
También puedes usar el notación de cadena de acceso directo parámetro para especificar el marcador.

Este parámetro también se llama `fmt` y está escrito con esta sintaxis:

`marker|line|color`

Ejemplo. Marque cada punto con un círculo.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

plt.plot(ypoints, 'o:g')
plt.show()
```



El valor del marcador puede ser cualquier cosa de la Referencia del Marcador anterior.

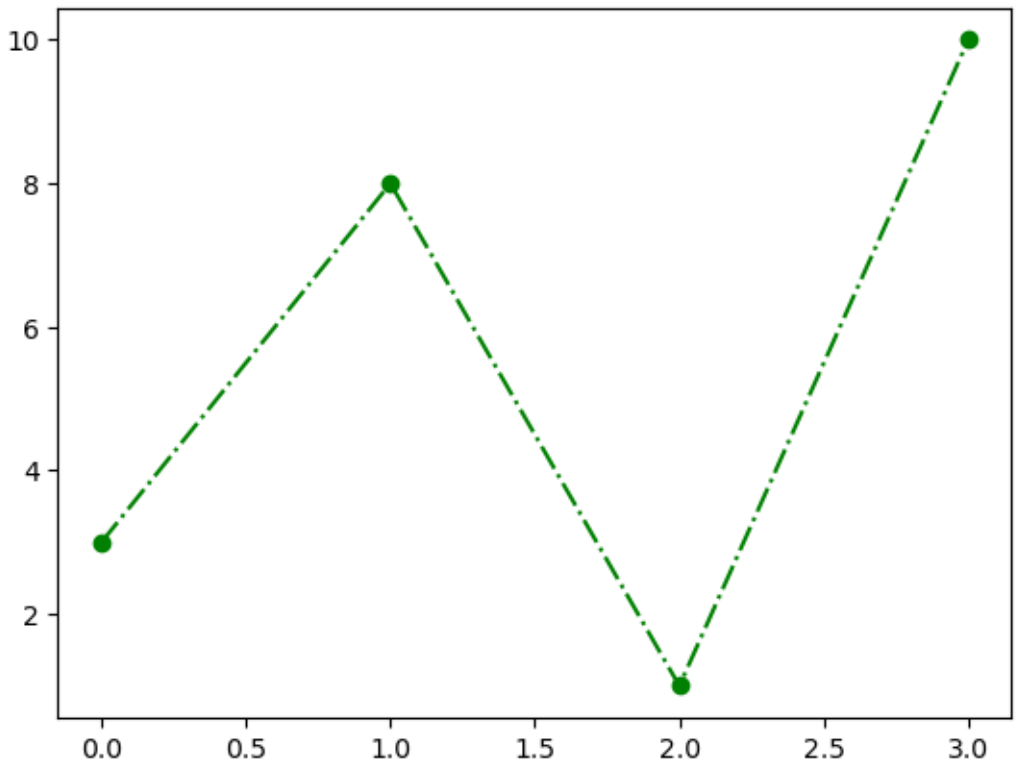
El valor de la línea puede ser uno de los siguientes:

1.8.1 Referencia de Línea

Sintaxis línea	Descripción
'-'	Solid line
':'	Dotted line 1

'–' Dashed line '-.' Dashed/dotted line

[ ]:
```
plt.plot(ypoints, 'o-.g')
plt.show()
```



1.9 Referencia de Color

Sintaxis	Descripción
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

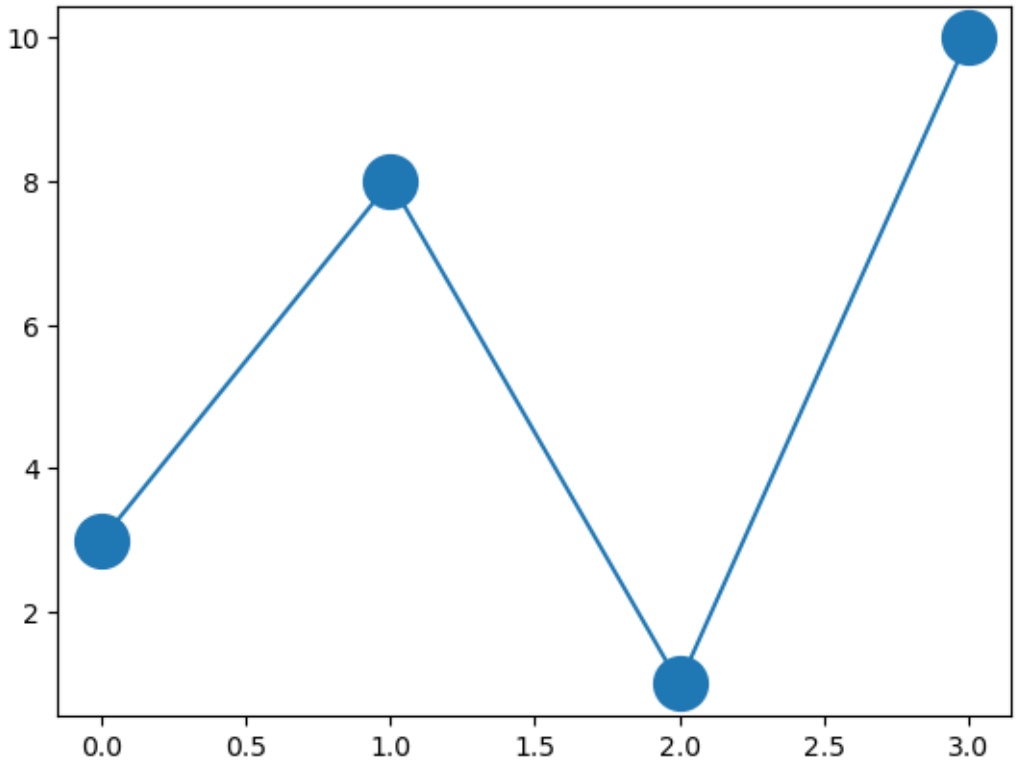
1.10 Tamaño del Marcador

Puede usar el argumento de palabra clave `markersize` o la versión más corta, `ms` para establecer el tamaño de los marcadores.

Ejemplo. Establezca el tamaño de los marcadores en 20.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

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

plt.plot(ypoints, marker = 'o', ms = 20)
plt.show()
```

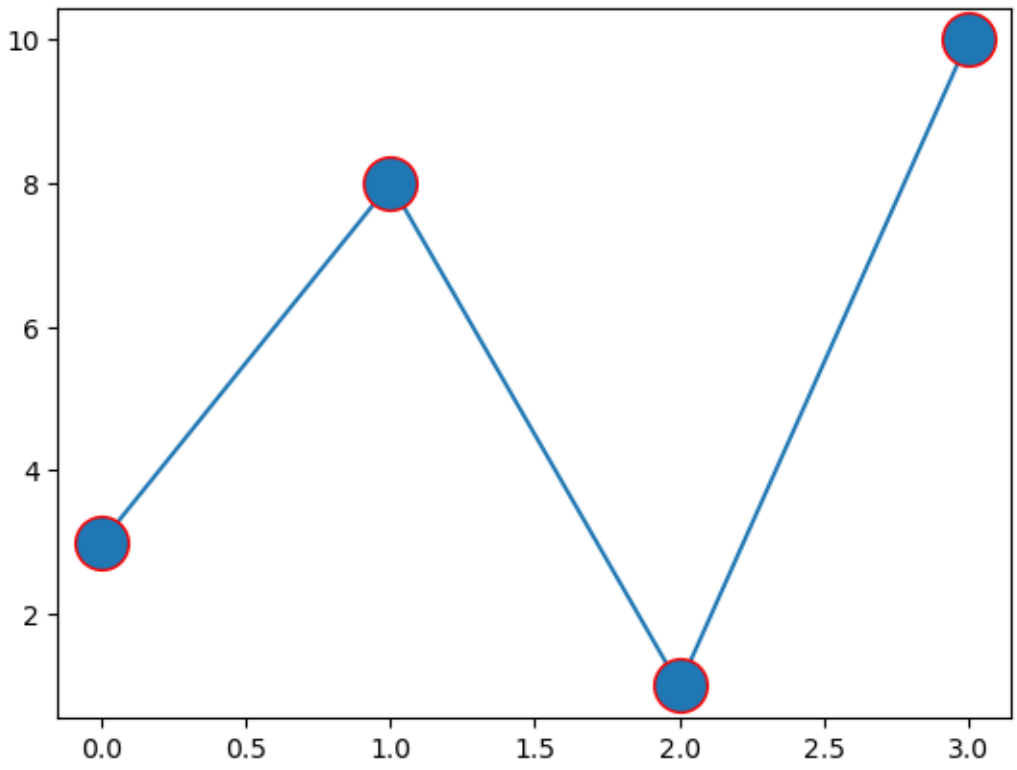


1.11 Color del Marcador

Puede usar el argumento de palabra clave `markeredgecolor` o el más corto `mec` para establecer el color de la borde de los marcadores.

Ejemplo. Establezca el color EDGE en rojo.

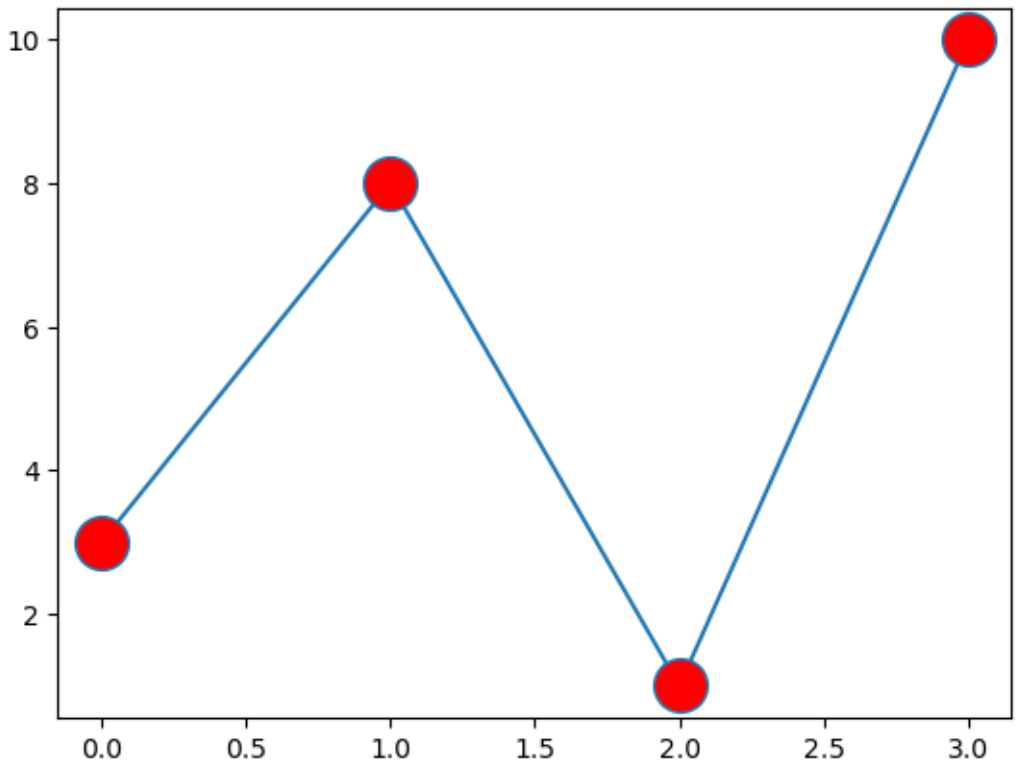
[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r')
plt.show()
```



Puede usar el argumento de palabra clave `markerfacecolor` o el más corto `mfc` para establecer el color dentro del borde de los marcadores:

Ejemplo. Coloque el color FACE en rojo.

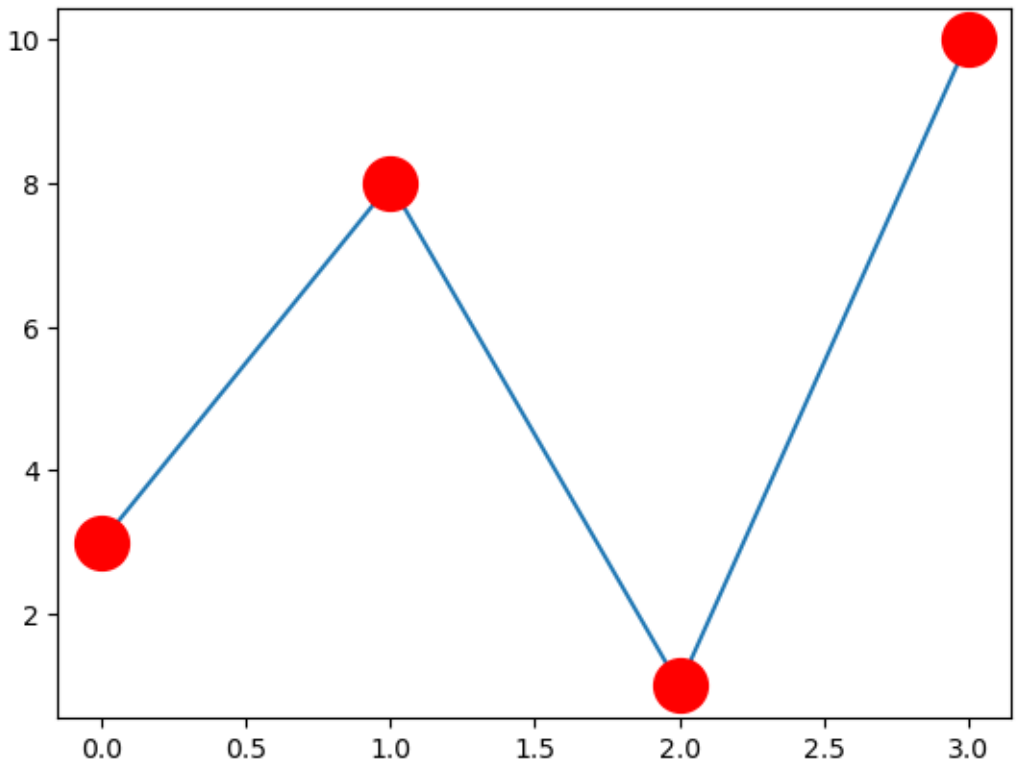
[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

plt.plot(ypoints, marker = 'o', ms = 20, mfc = 'r')
plt.show()
```



Usar ambos el `mec` y `mfc` argumentos para colorear todo el marcador:

Ejemplo. Establezca el color de ambos borde y el cara a rojo.

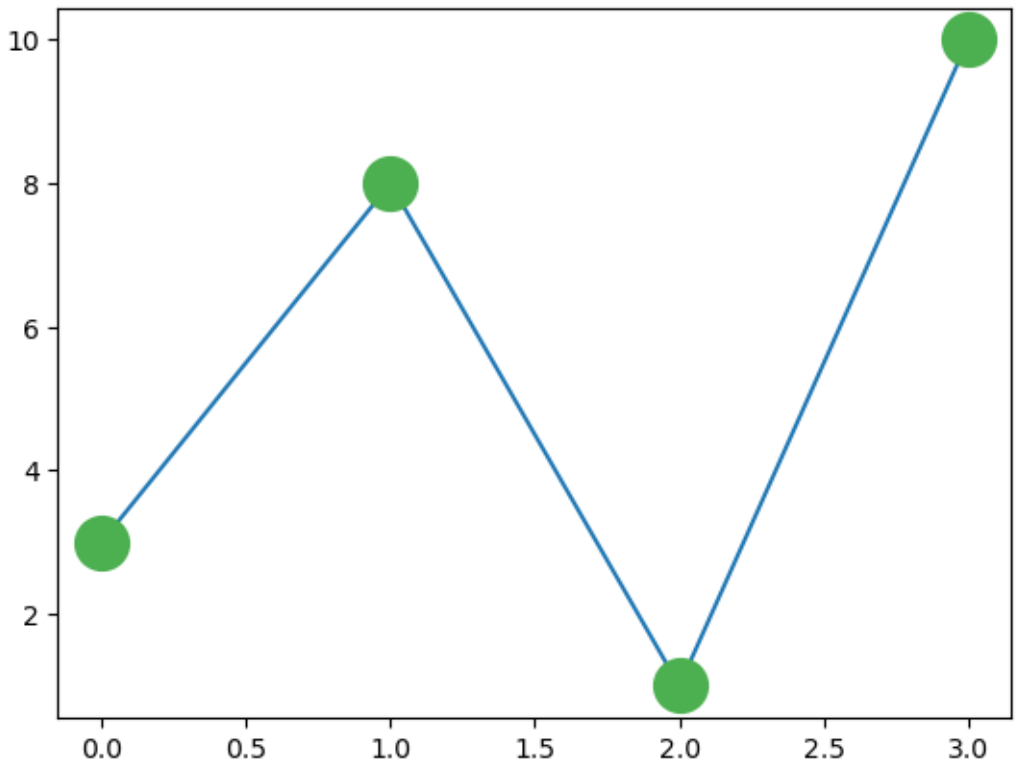
[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r', mfc = 'r')
plt.show()
```



También puedes usar valores de color hexadecimal, o cualquiera de los 140 nombres de color compatibles.

Ejemplo. Marque cada punto con un hermoso color verde.

[ ]:
```
plt.plot(ypoints, marker = 'o', ms = 20, mec = '#4CAF50', mfc = '#4CAF50')
plt.show()
```

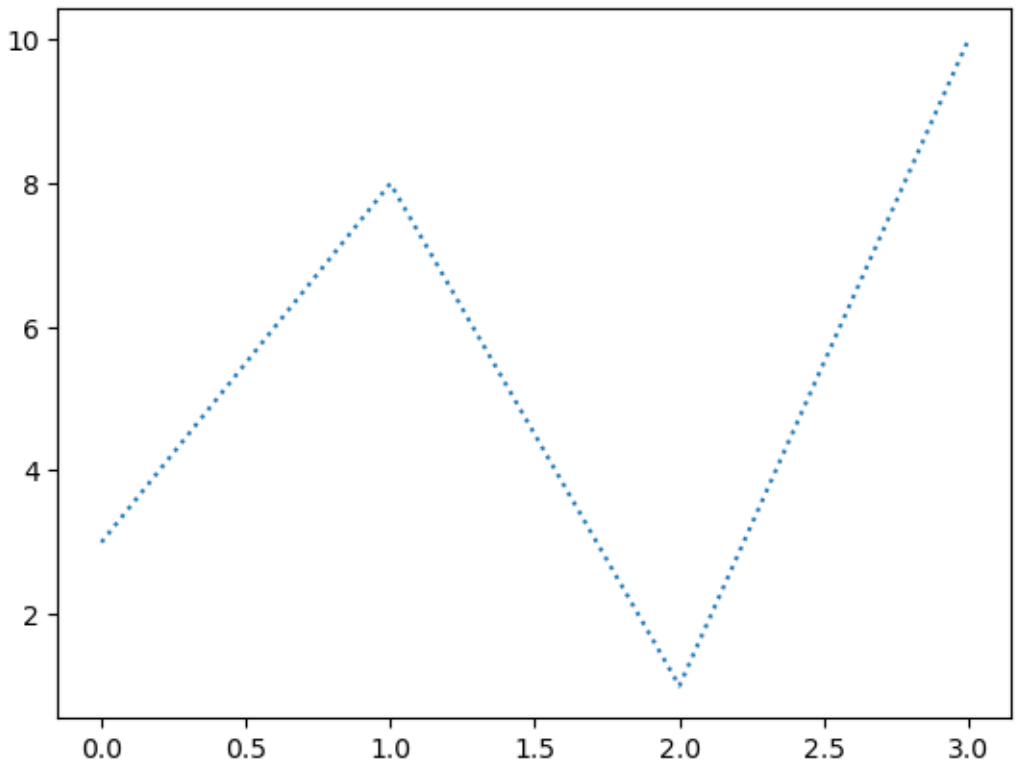


1.12 Estilo de línea

Puede usar el argumento de palabra clave `linestyle`, o más corto `ls`, a cambiar el estilo de la línea trazada.

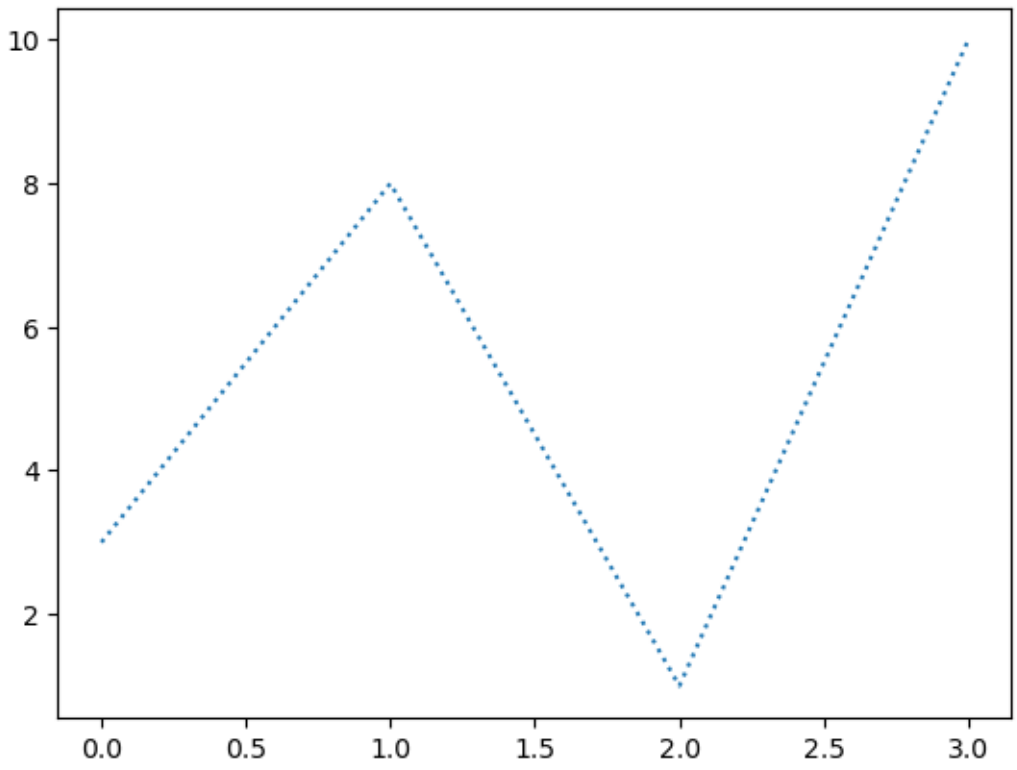
Ejemplo. Usa una línea punteada.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

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



Ejemplo. Usa una línea discontinua.

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



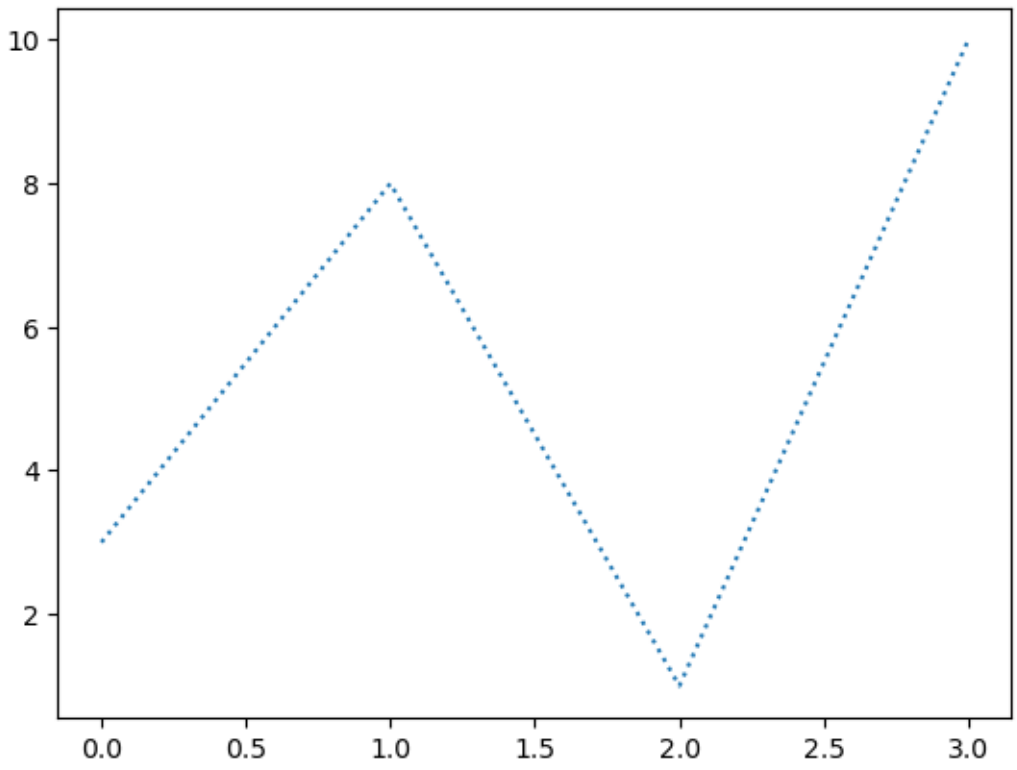
1.13 Sintaxis más corta

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 `--`.

Ejemplo. Sintaxis más corta.

[ ]:
```
plt.plot(ypoints, ls = ':')
plt.show()
```



1.14 Estilos de Línea

Puedes elegir cualquiera de estos estilos:

Style | Or |

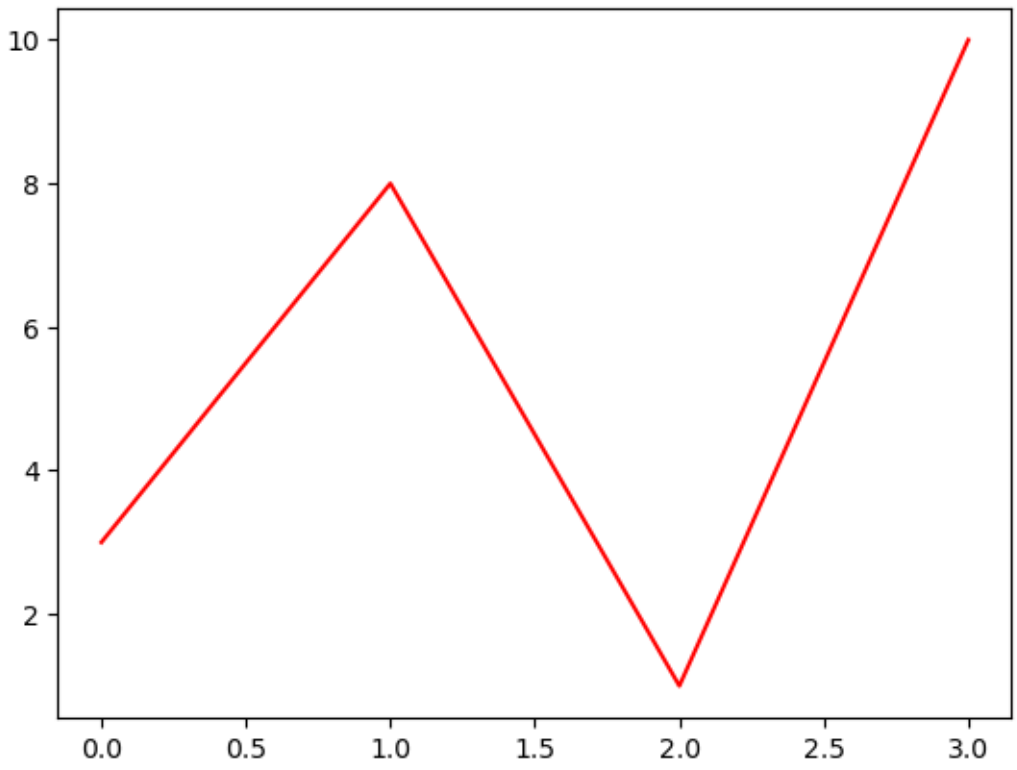
|:-…-:|:–:| | ‘solid’ | (default) ‘-’ | | ‘dotted’ | ‘:’ | | ‘dashed’ | ‘–’ | | ‘dashdot’ | ‘-.’ | | ‘None’ | ’ ’ or ’ ’ |

1.15 Color de Línea

Puede usar el argumento de palabra clave `color` o el más corto `c` para establecer el color de la línea.

Ejemplo. Establecer el color de la línea en rojo.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

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



También se puede utilizar el color en su valor hexadecimal o los nombres de color compatibles.

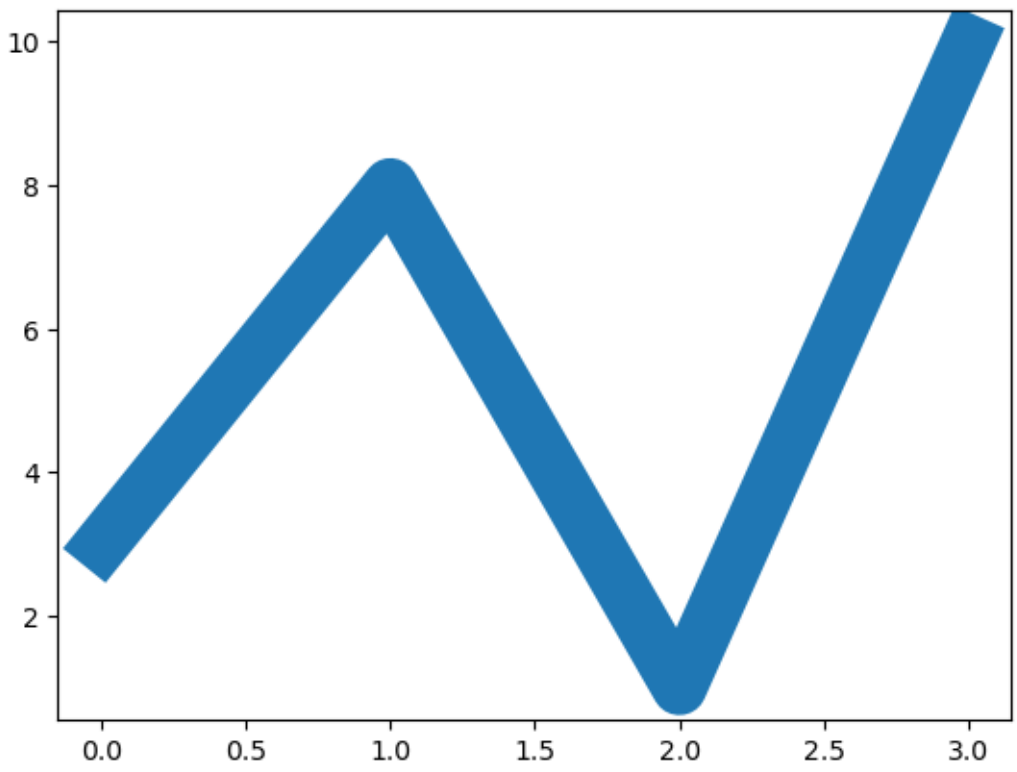
1.16 Ancho de Línea

Puede usar el argumento de palabra clave `linewidth` o el más corto `lw` para cambiar el ancho de la línea.

El valor es un número flotante, en puntos.

Ejemplo. Parcela con una línea ancha de 20.5pt.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

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

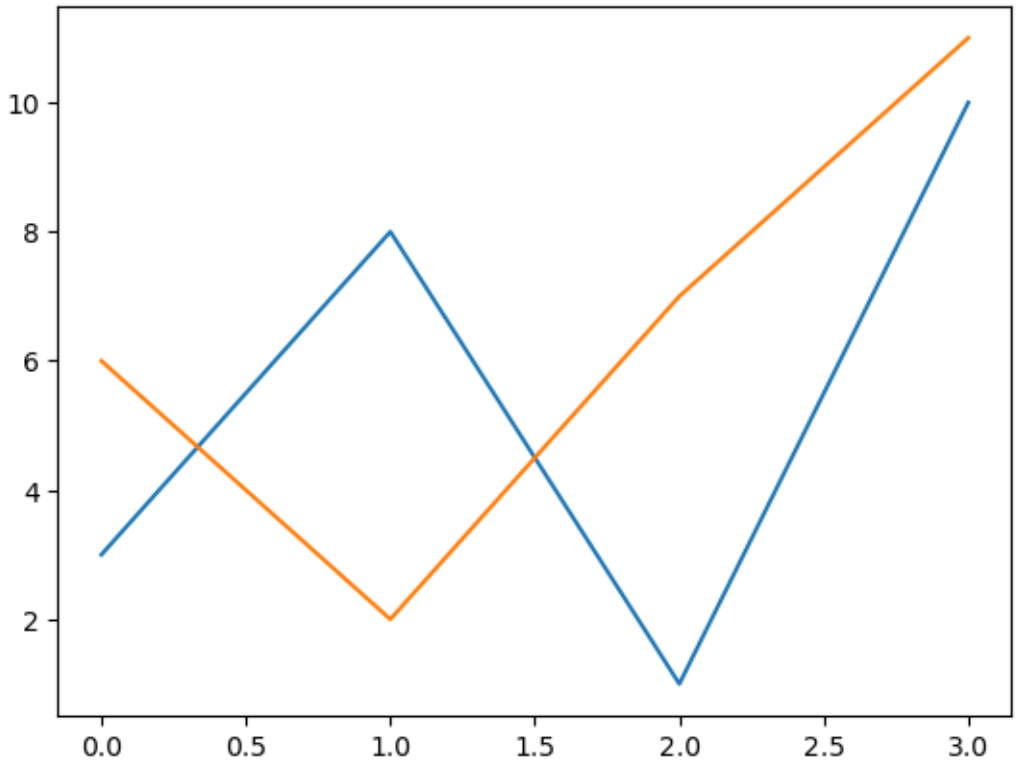


1.17 Múltiples Líneas

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

Ejemplo. Dibuja dos líneas especificando un `plt.plot()` función 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()
```



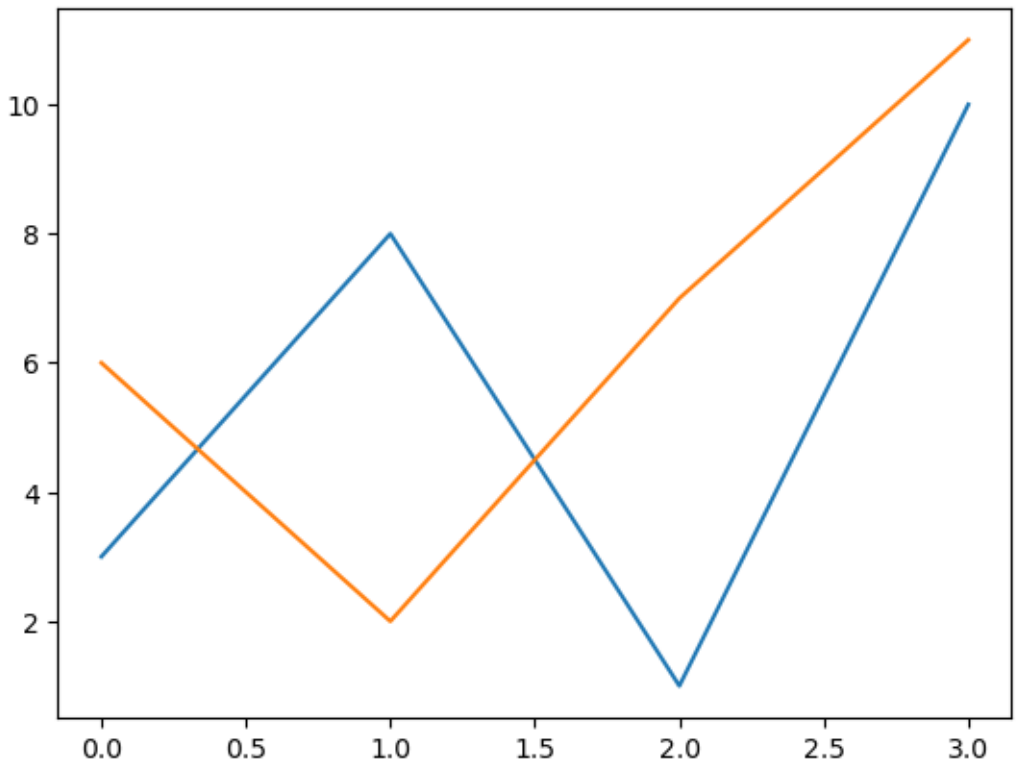
También puede trazar muchas líneas agregando los puntos para el eje x y y para cada línea en la misma función `plt.plot()`.

(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).)

Los valores x y y vienen en pares.

Ejemplo. Dibuje dos líneas especificando los valores x y y para ambas líneas.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1, x2, y2)
plt.show()
```

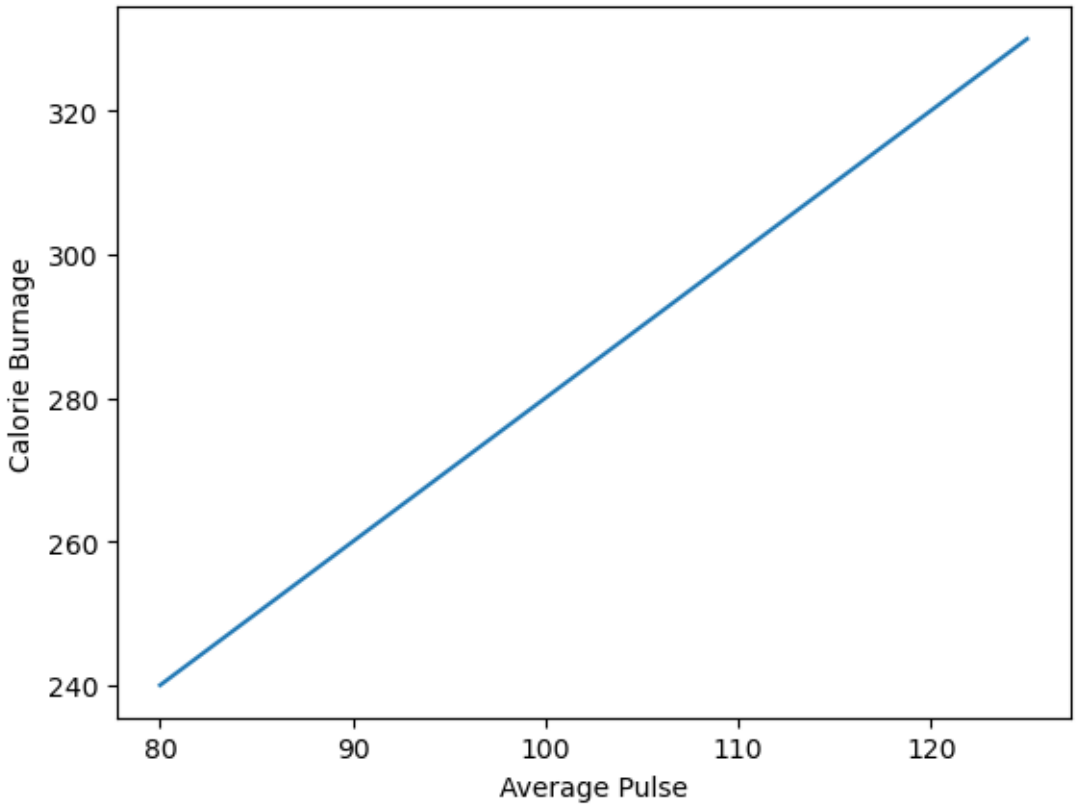


1.18 Crear Etiquetas para una gráfica

Con `Pyplot`, puede usar las funciones `xlabel()` y `ylabel()` para establecer una etiqueta para el eje x y y.

Ejemplo. Agregar etiquetas al eje x y y.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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()
```

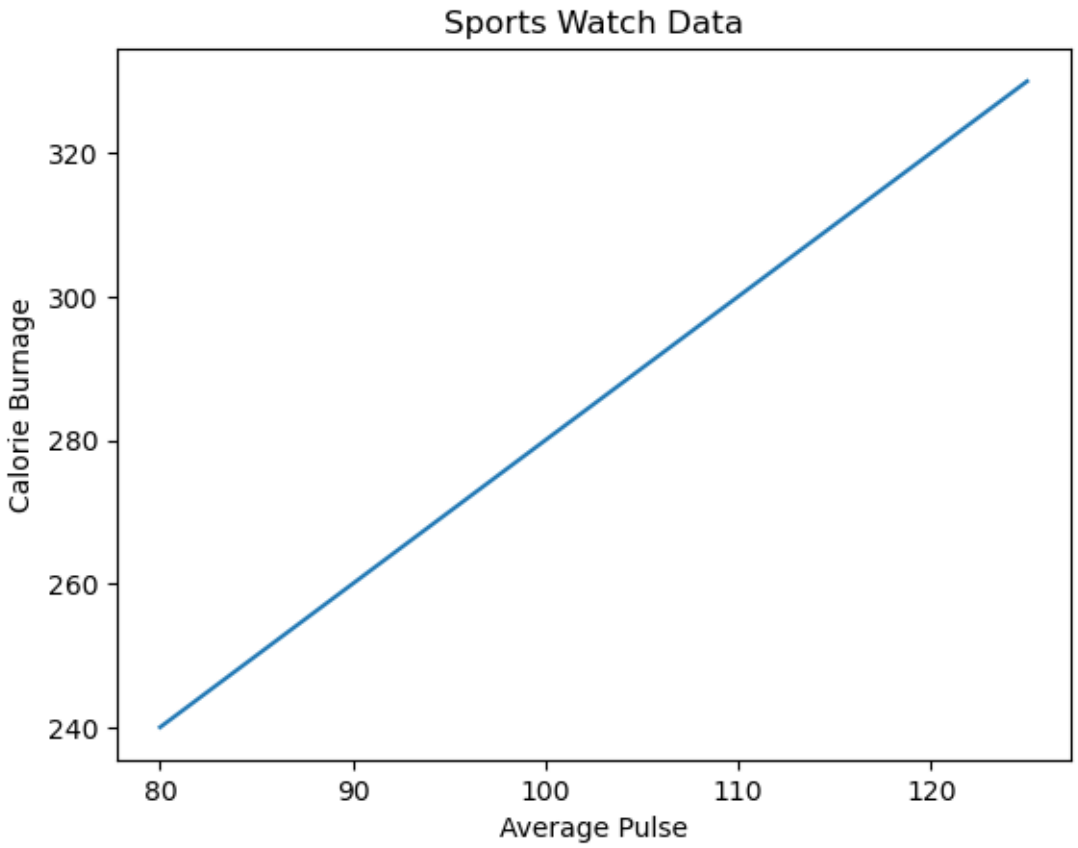


1.19 Crear un título para una trama

Con `Pyplot`, puedes utilizar la función `title()` para establecer un título para el gráfico.

Ejemplo. Agregue un título de gráfico y etiquetas para los ejes x e y.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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

plt.show()
```



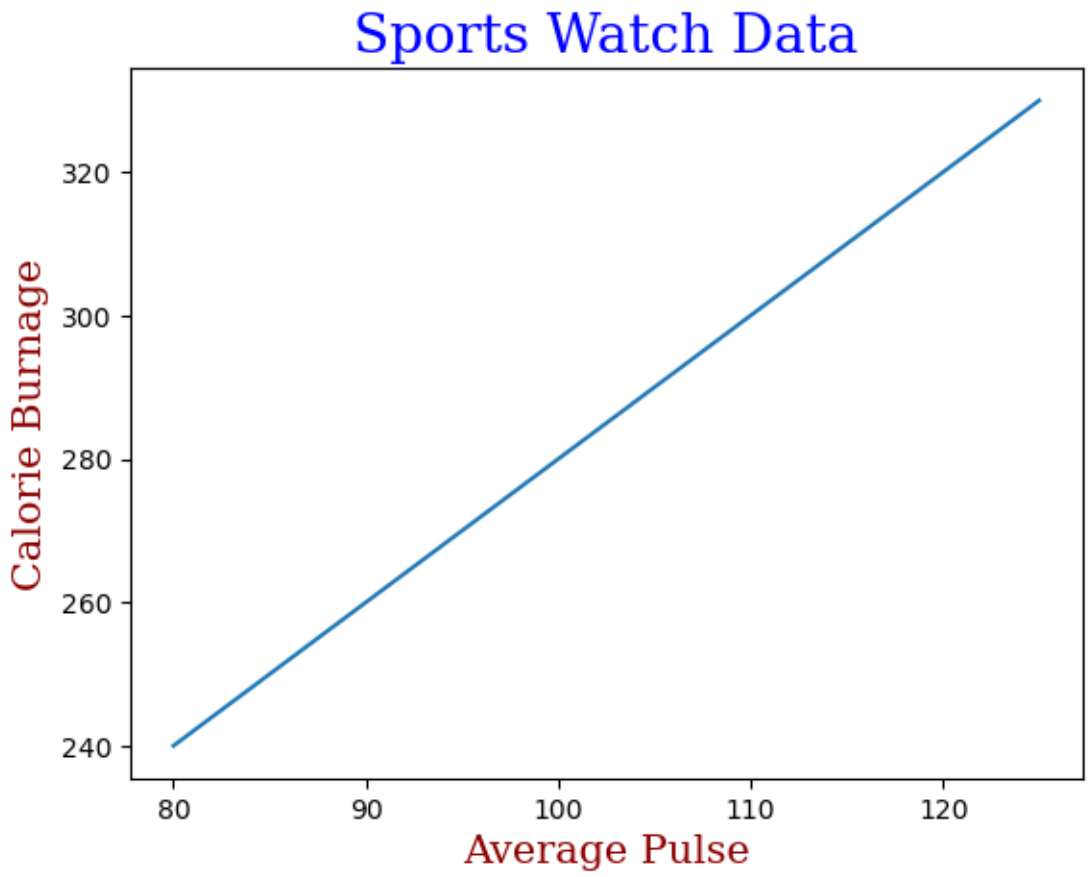
1.20 Establecer fuente para títulos y etiquetas

Puede utilizar el parámetro `fontdict` en `xlabel()`, `ylabel()` y `title()` para establecer las propiedades de fuente para el título y las etiquetas.

Ejemplo. Establecer propiedades de fuente para el título y las etiquetas.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)
```

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



1.21 Posicionar el título

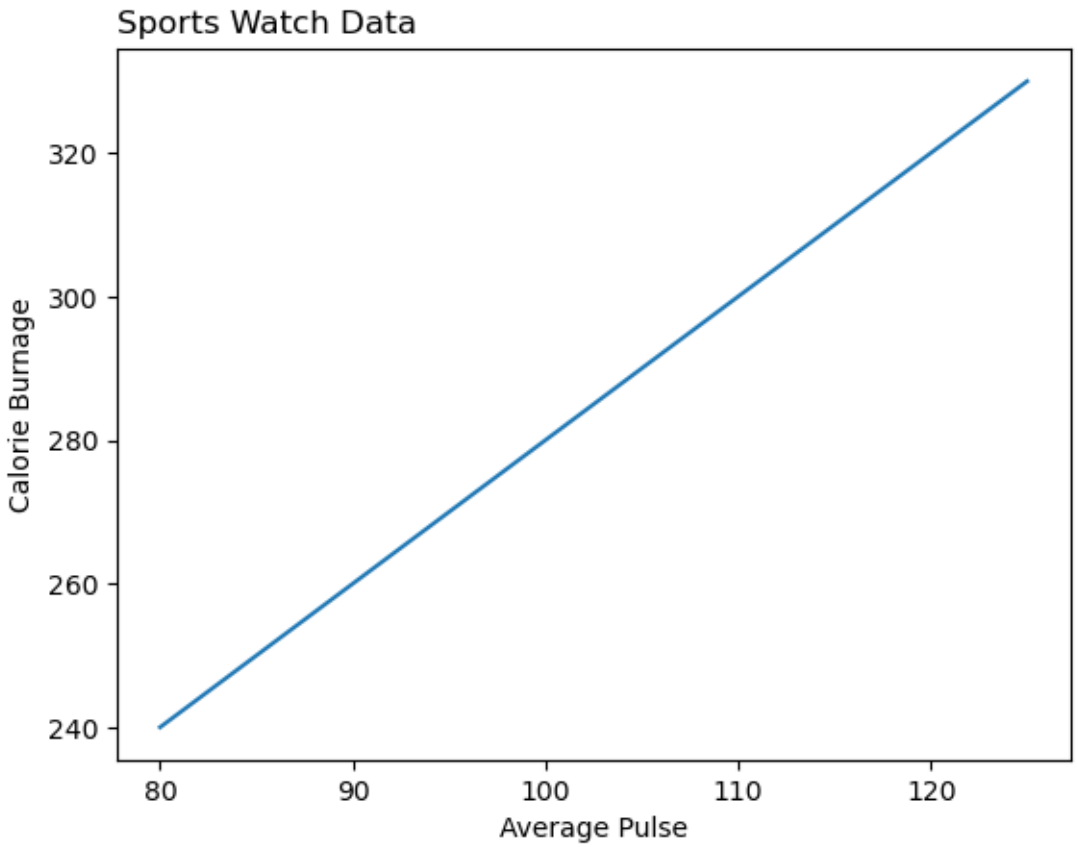
Puedes utilizar el parámetro `loc` en `title()` para posicionar el título.

Los valores admitidos son: *'left'*, *'right'* y *'center'*. El valor predeterminado es *'center'*.

`Ejemplo`. Coloque el título a la izquierda.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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.title("Sports Watch Data", loc = 'left')
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
```

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



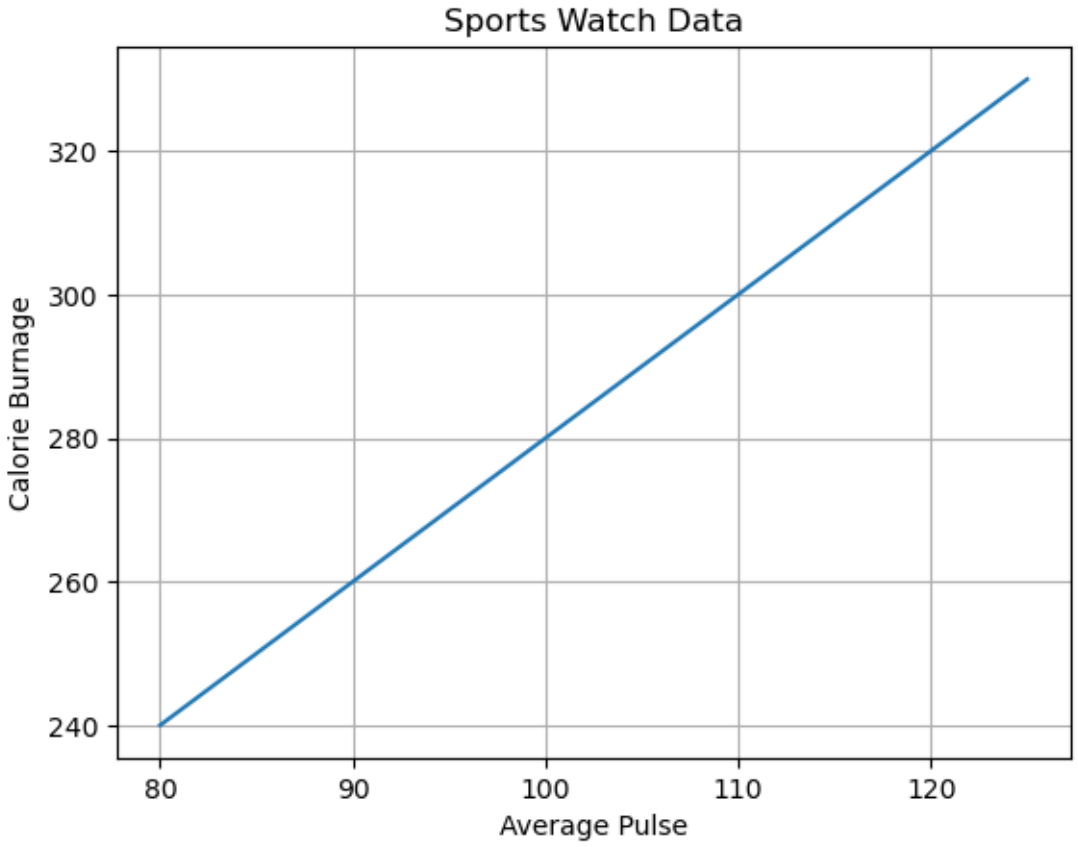
1.22 Agregar cuadrícula a un gráfico

Con `Pyplot`, puedes usar la función `grid()` para agregar cuadrícula al gráfico.

Ejemplo. Añadir líneas de cuadrícula al gráfico.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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

```
plt.plot(x, y)

plt.grid()

plt.show()
```



1.23 Especificar qué líneas de cuadrícula se mostrarán

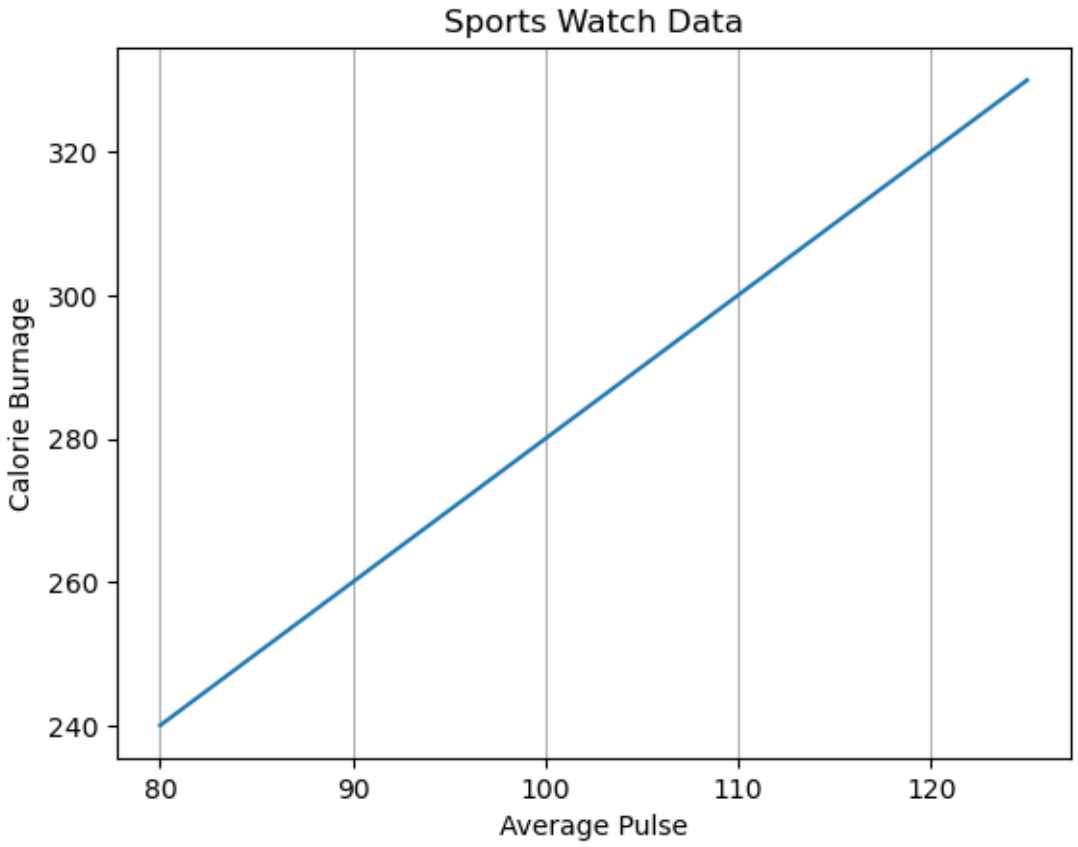
Puede utilizar el parámetro `axis` en la función `grid()` para especificar qué líneas de cuadrícula mostrar.

Los valores admitidos son: *'x'*, *'y'* y *'both'*. El valor predeterminado es *'both'*.

Ejemplo. Mostrar solo líneas de cuadrícula para el eje x.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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])
```

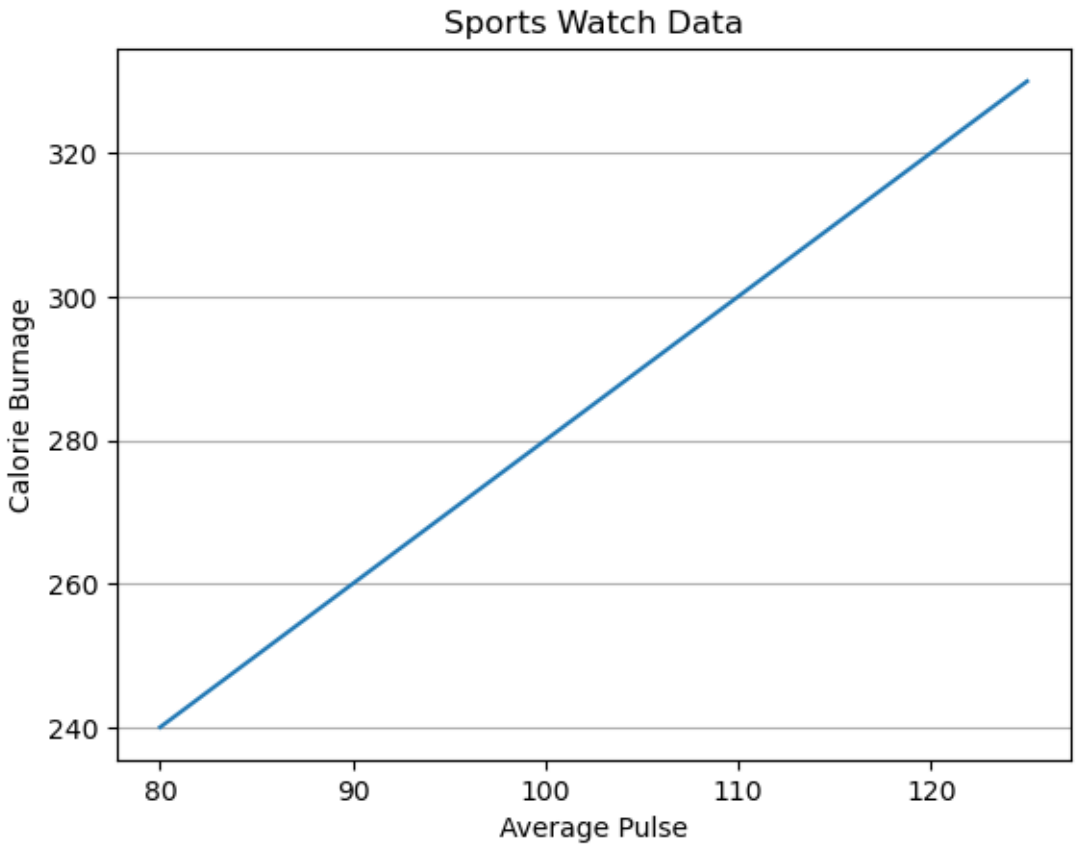
```
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid(axis = 'x')

plt.show()
```



Ejemplo. Mostrar solo líneas de cuadrícula para el eje y.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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

```
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid(axis = 'y')

plt.show()
```



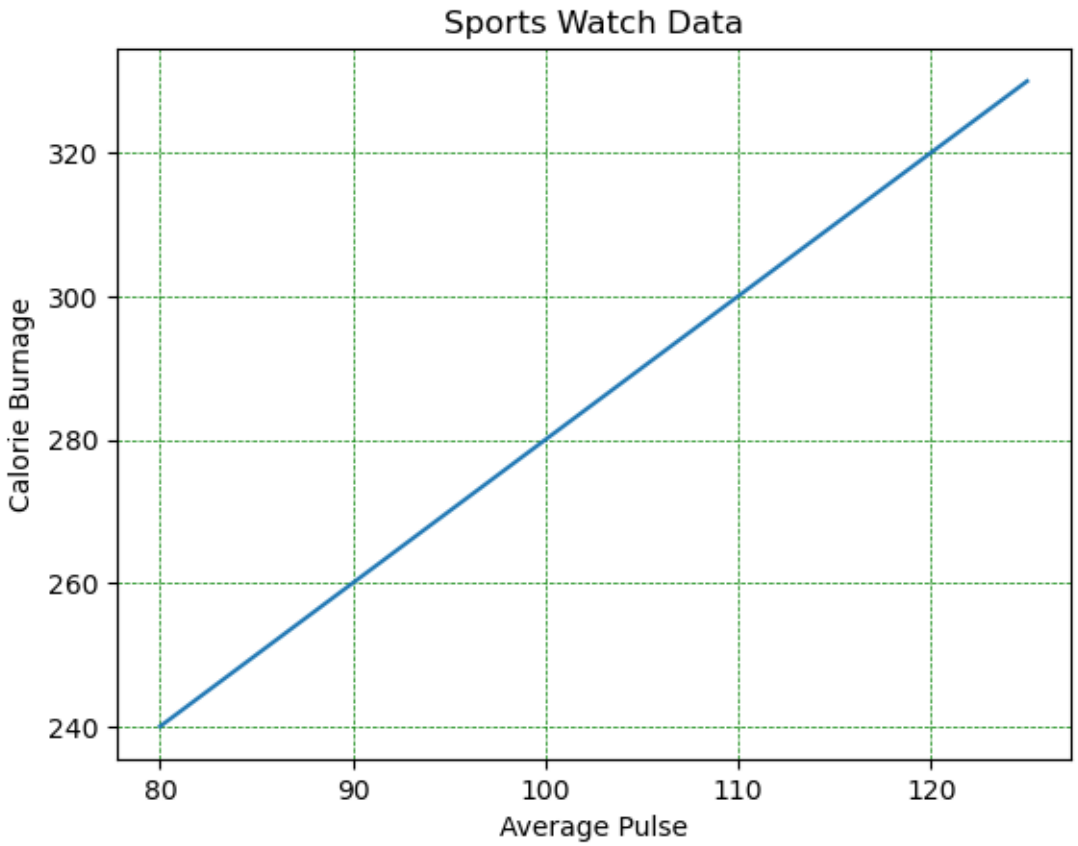
1.24 Establecer propiedades de línea para la cuadrícula

También puede configurar las propiedades de línea de la cuadrícula, de esta manera: `grid(color = '_color_', linestyle = '_linestyle_', linewidth = _number_)`.

Ejemplo. Establecer las propiedades de línea de la cuadrícula.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

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

plt.plot(x, y)

plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```



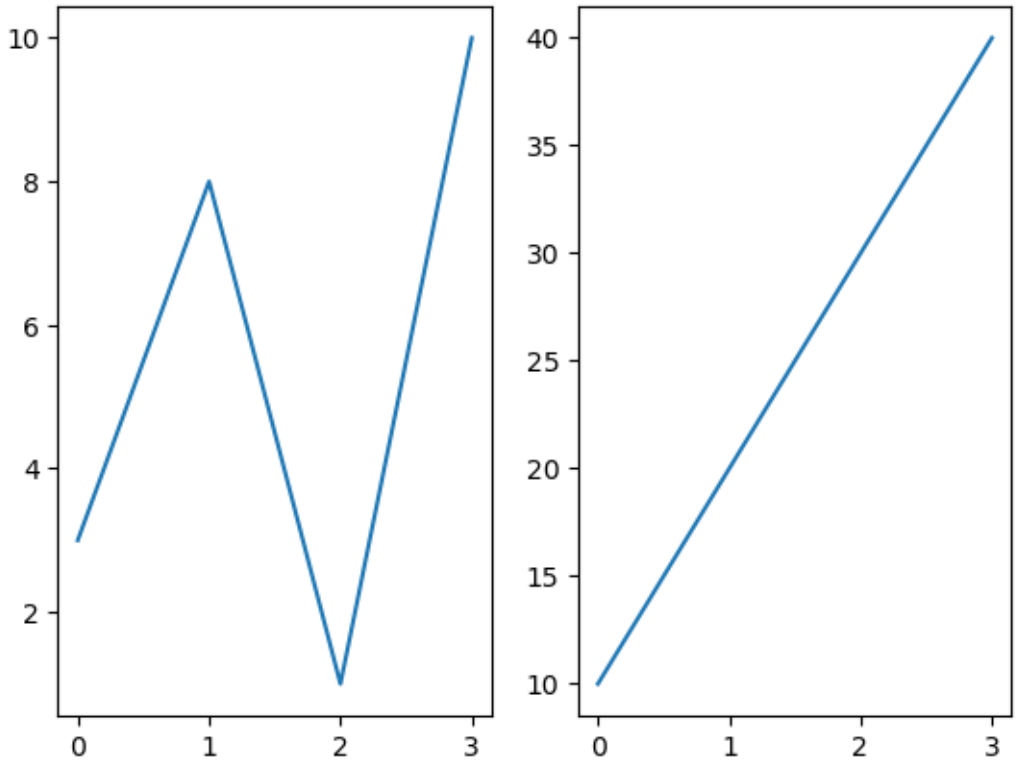
1.25 Mostrar múltiples gráficos

Con esta función `subplot()` puedes dibujar múltiples gráficos en una figura.

Ejemplo. Dibuje 2 gráficos.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

```
#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()
```



1.26 La función subplot()

La función `subplot()` 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.

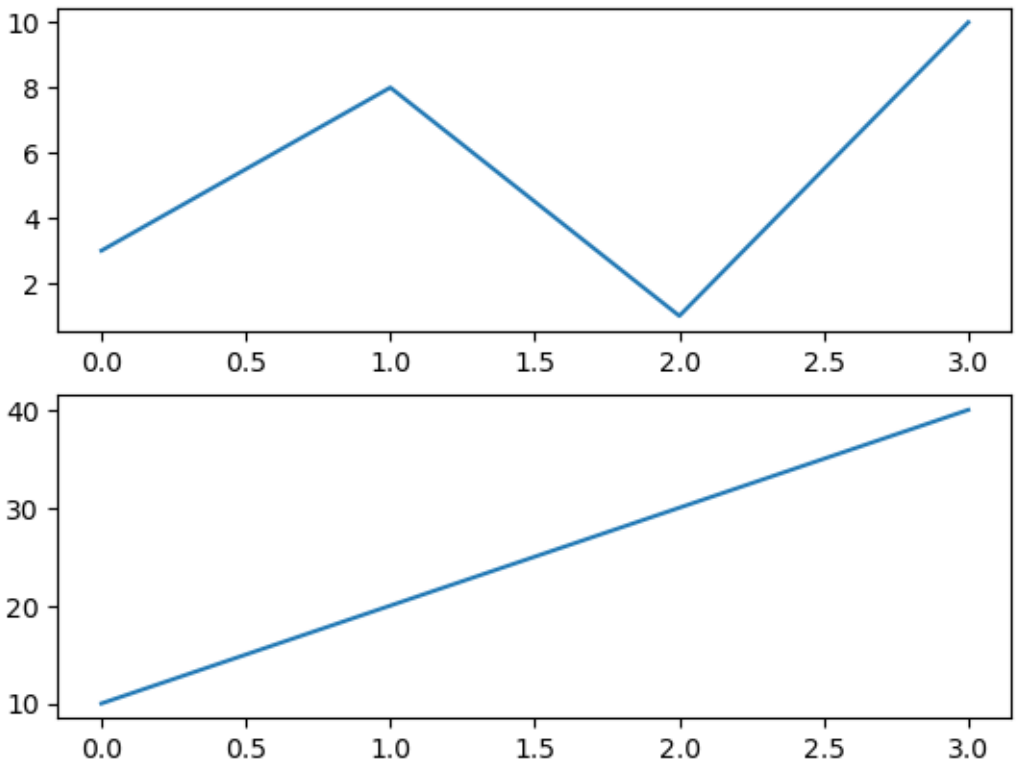
El tercer argumento representa el índice de la malla actual.

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

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

Entonces, si queremos una figura con 2 filas y 1 columna (lo que significa que los dos gráficos se mostrarán uno encima del otro en lugar de uno al lado del otro), podemos escribir la sintaxis de esta manera.

Ejemplo. Dibuje 2 gráficos uno encima del otro.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
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()
```

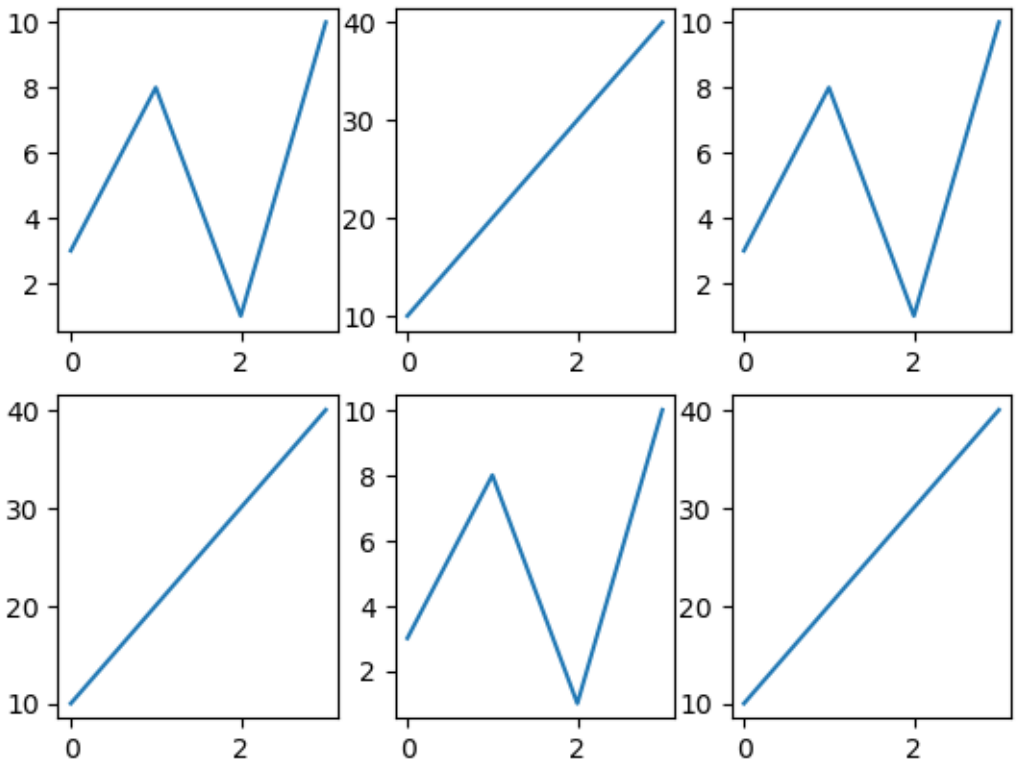


Puede dibujar tantos gráficos como quiera en una figura, simplemente describa el número de filas, columnas y el índice del gráfico.

Ejemplo. Dibuje 6 gráficos.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

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

plt.subplot(2, 3, 1)
plt.plot(x,y)

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

plt.subplot(2, 3, 2)
plt.plot(x,y)

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

```
plt.subplot(2, 3, 3)
plt.plot(x,y)

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

plt.subplot(2, 3, 4)
plt.plot(x,y)

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

plt.subplot(2, 3, 5)
plt.plot(x,y)

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

plt.subplot(2, 3, 6)
plt.plot(x,y)

plt.show()
```

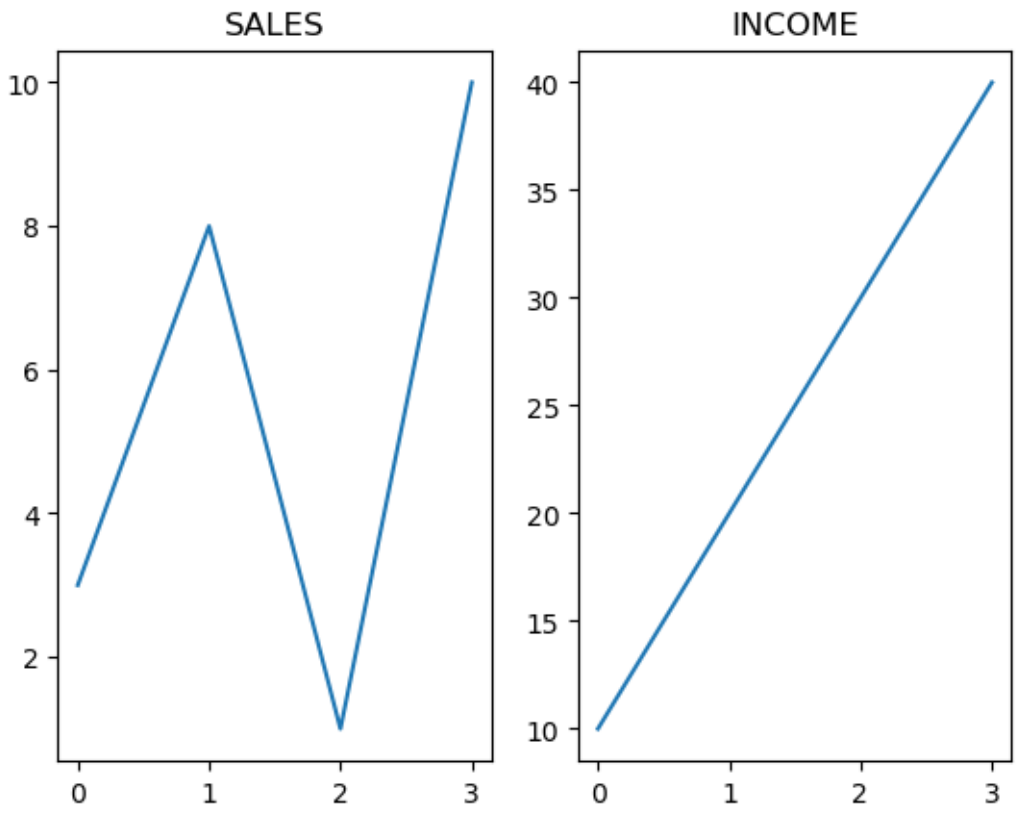


1.27 Título

Puede agregar un título a cada gráfico con la función `title()`.

Ejemplo. 2 parcelas, con títulos.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

#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)
plt.title("SALES")

#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.title("INCOME")

plt.show()
```



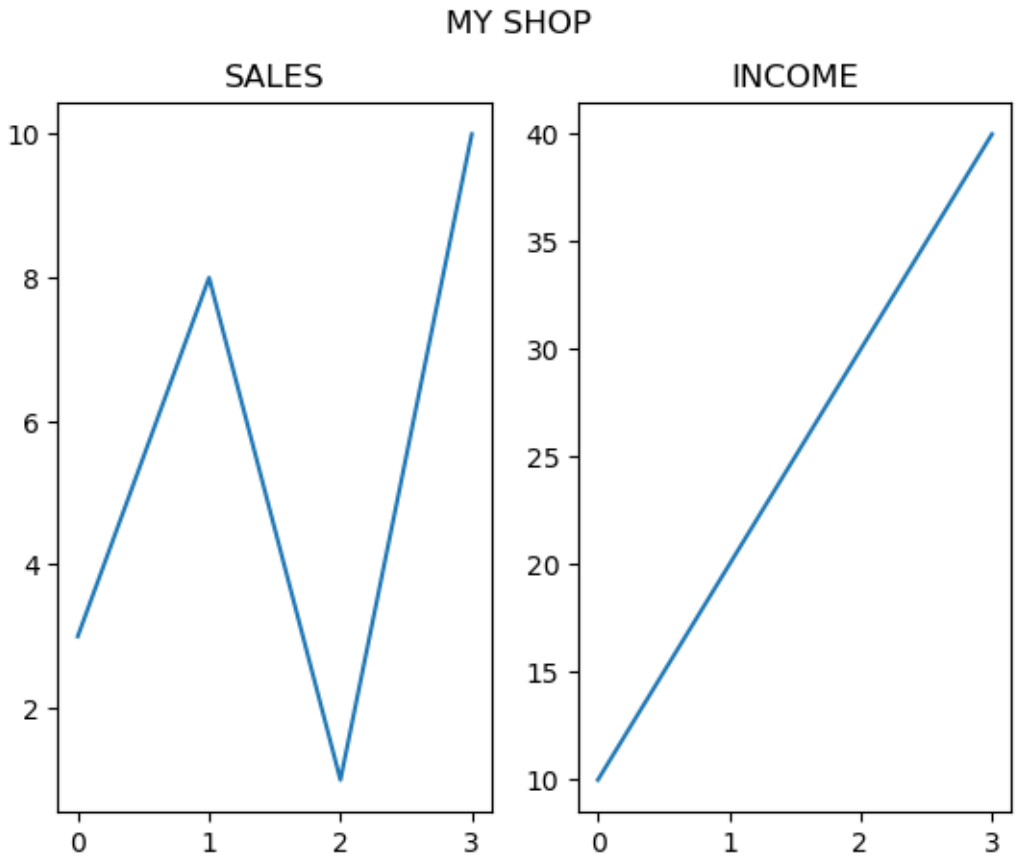
1.28 Súper título

Puede agregar un título a toda la figura con la función `suptitle()`.

Ejemplo. Añade un título para toda la figura.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

#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)
plt.title("SALES")

#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.title("INCOME")

plt.suptitle("MY SHOP")
plt.show()
```



1.29 Gráficos de dispersión

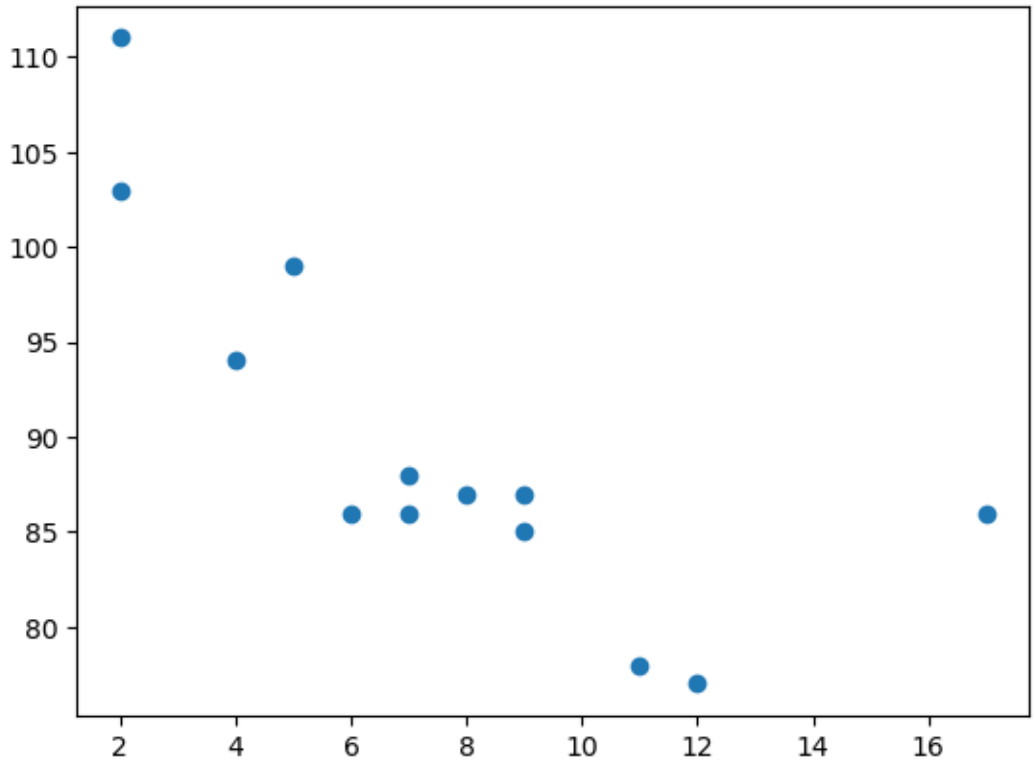
Con `Pyplot`, puede utilizar la función `scatter()` para dibujar un diagrama de dispersión.

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

Ejemplo. Un diagrama de dispersión simple.

[ ]:
```
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])
```

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



La observación en el ejemplo anterior es el resultado de 13 automóviles.

El eje x muestra la antigüedad del coche y el eje y muestra la velocidad del automóvil cuando pasa.

¿Existe alguna relación entre las observaciones?. Parece que cuanto más nuevo es el coche, más rápido va, pero eso podría ser una coincidencia, después de todo sólo registramos 13 coches.

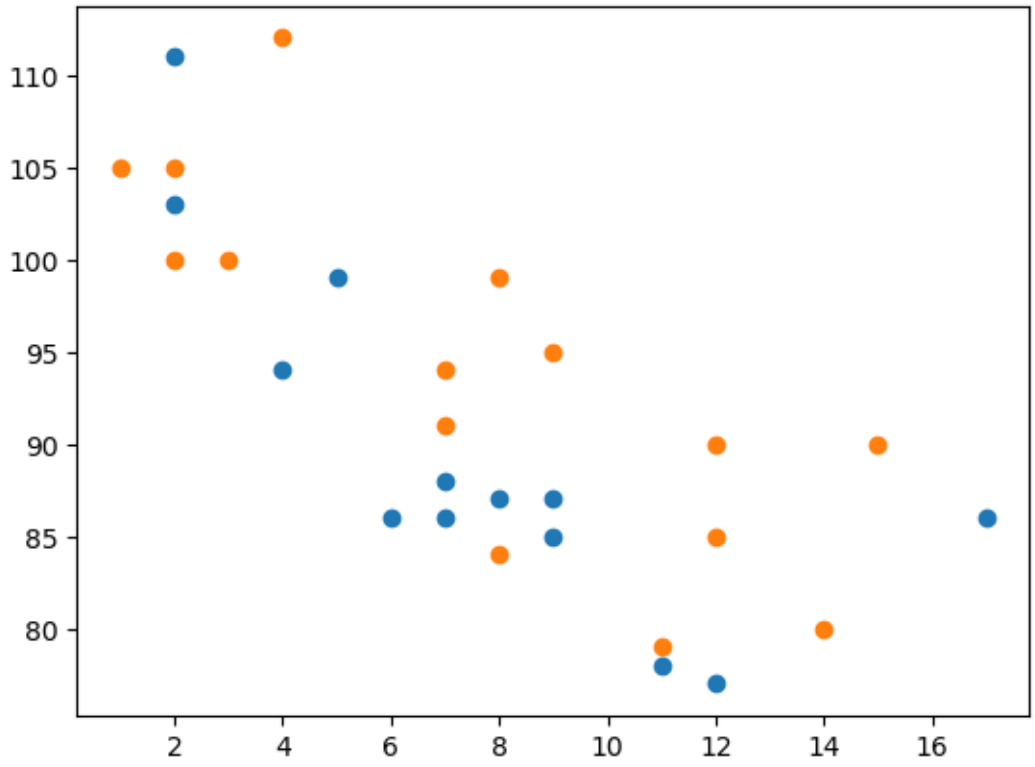
1.30 Comparar gráficos

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

Ejemplo. Dibuje dos gráficos en la misma figura.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

#day one, the age and speed of 13 cars:
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)

#day two, the age and speed of 15 cars:
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)

plt.show()
```



Nota: Los dos gráficos están representados con dos colores diferentes, por defecto azul y naranja.

Comparando ambos gráficos, creo que es seguro decir que ambos nos llevan a la misma conclusión: cuanto más nuevo es el coche, más rápido va.

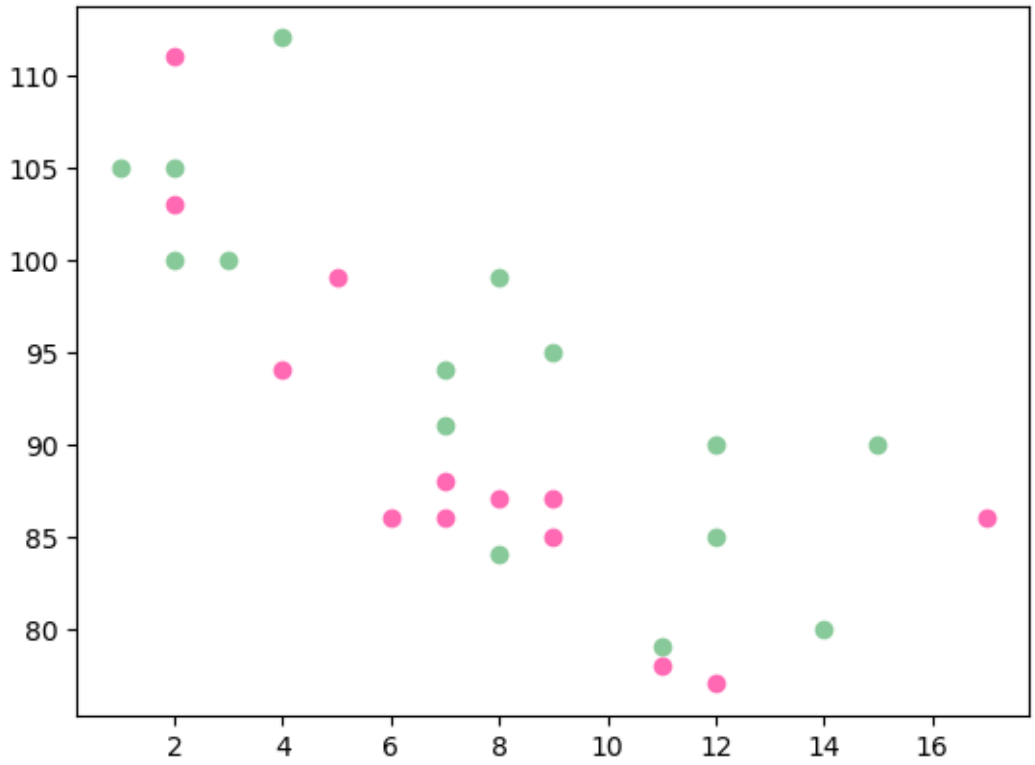
1.31 Colores

Puede establecer su propio color para cada gráfico de dispersión con `color` o el argumento `c`.

Ejemplo. Establezca su propio color de marcadores.

[ ]:
```
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])
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()
```



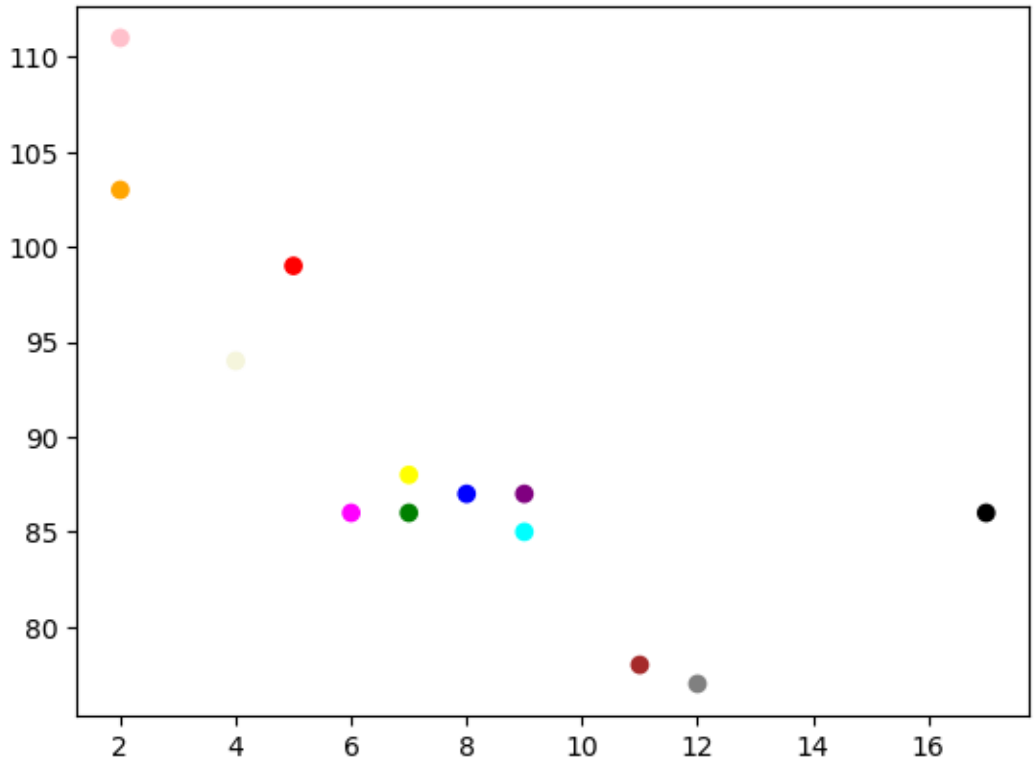
1.32 Colorear cada punto

Es posible establecer un color específico para cada punto utilizando una matriz de colores como valor para el cargumento.

Nota: No puede usar el argumento `color` para esto, solo el argumento `c`.

[ ]:
```
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(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray
```

```
plt.scatter(x, y, c=colors)

plt.show()
```



1.33 Mapa de colores

El módulo `Matplotlib` tiene varios mapas de colores disponibles.

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

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

Este mapa de colores se llama *'viridis'* y como puedes ver, varía desde 0, que es un color púrpura, hasta 100, que es un color amarillo.

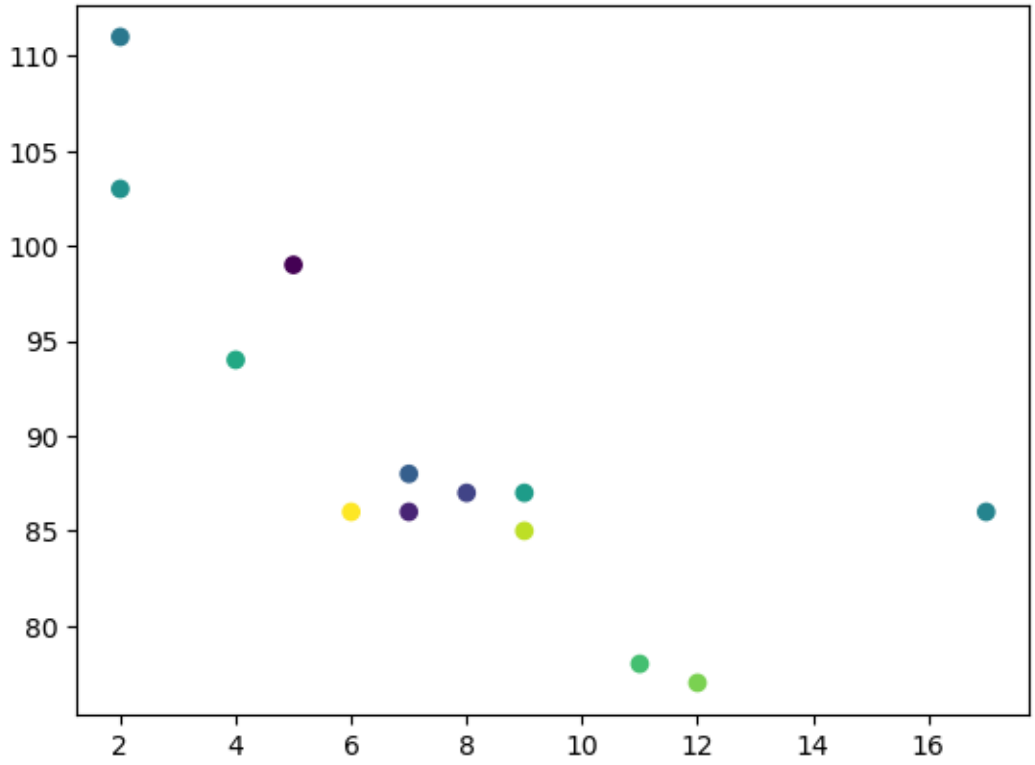
1.33.1 Cómo utilizar el mapa de colores

Puede especificar el mapa de colores con el argumento de palabra clave `cmap`, en este caso *'viridis'* que es uno de los mapas de colores integrados disponibles en `Matplotlib`.

Además, debes crear una matriz con valores (de 0 a 100), un valor para cada punto en el diagrama de dispersión.

Ejemplo. Cree una matriz de colores y especifique un mapa de colores en el gráfico de dispersión.

```
[ ]: 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()
```

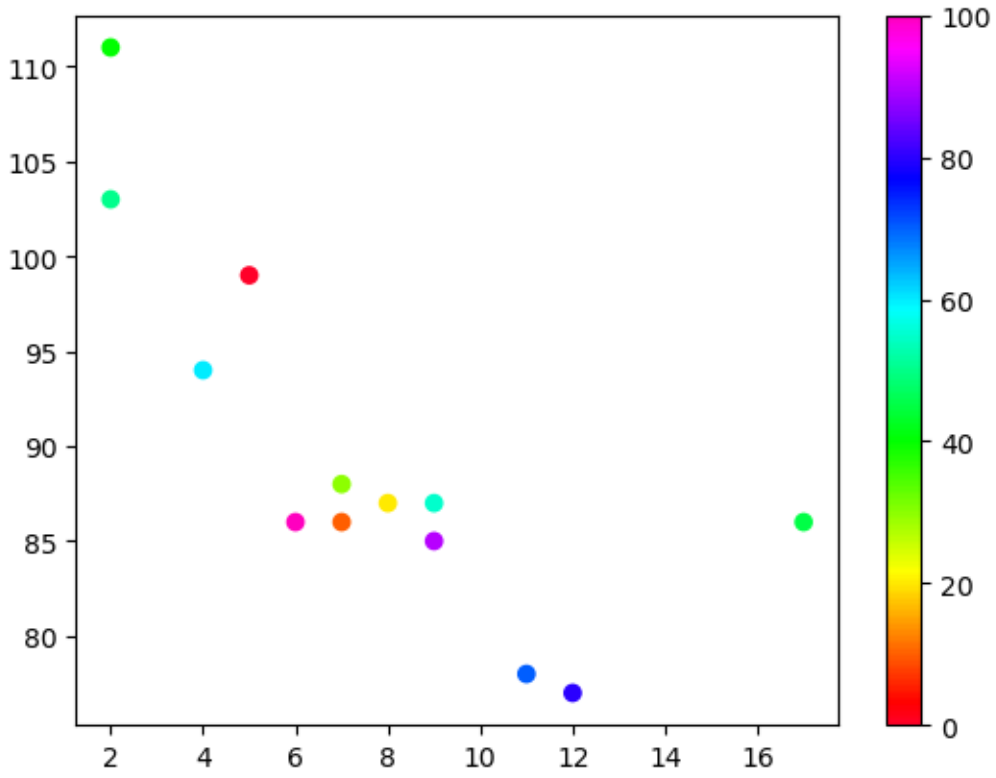


Puede incluir el mapa de colores en el dibujo incluyendo la declaración `plt.colorbar()`.

Ejemplo. Incluya el mapa de colores real.

```
[ ]: 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='gist_rainbow')
```

```
plt.colorbar()

plt.show()
```



1.34 Mapas de colores disponibles

Puede elegir cualquiera de los mapas de colores incorporados.

[ ]:
```
from matplotlib import colormaps
list(colormaps)
```

[ ]:
```
['magma',
 'inferno',
 'plasma',
 'viridis',
 'cividis',
 'twilight',
 'twilight_shifted',
 'turbo',
 'Blues',
 'BrBG',
```

```
'BuGn',
'BuPu',
'CMRmap',
'GnBu',
'Greens',
'Greys',
'OrRd',
'Oranges',
'PRGn',
'PiYG',
'PuBu',
'PuBuGn',
'PuOr',
'PuRd',
'Purples',
'RdBu',
'RdGy',
'RdPu',
'RdYlBu',
'RdYlGn',
'Reds',
'Spectral',
'Wistia',
'YlGn',
'YlGnBu',
'YlOrBr',
'YlOrRd',
'afmhot',
'autumn',
'binary',
'bone',
'brg',
'bwr',
'cool',
'coolwarm',
'copper',
'cubehelix',
'flag',
'gist_earth',
'gist_gray',
'gist_heat',
'gist_ncar',
'gist_rainbow',
'gist_stern',
'gist_yarg',
'gnuplot',
'gnuplot2',
```

```
'gray',
'hot',
'hsv',
'jet',
'nipy_spectral',
'ocean',
'pink',
'prism',
'rainbow',
'seismic',
'spring',
'summer',
'terrain',
'winter',
'Accent',
'Dark2',
'Paired',
'Pastel1',
'Pastel2',
'Set1',
'Set2',
'Set3',
'tab10',
'tab20',
'tab20b',
'tab20c',
'grey',
'gist_grey',
'gist_yerg',
'Grays',
'magma_r',
'inferno_r',
'plasma_r',
'viridis_r',
'cividis_r',
'twilight_r',
'twilight_shifted_r',
'turbo_r',
'Blues_r',
'BrBG_r',
'BuGn_r',
'BuPu_r',
'CMRmap_r',
'GnBu_r',
'Greens_r',
'Greys_r',
'OrRd_r',
```

```
'Oranges_r',
'PRGn_r',
'PiYG_r',
'PuBu_r',
'PuBuGn_r',
'PuOr_r',
'PuRd_r',
'Purples_r',
'RdBu_r',
'RdGy_r',
'RdPu_r',
'RdYlBu_r',
'RdYlGn_r',
'Reds_r',
'Spectral_r',
'Wistia_r',
'YlGn_r',
'YlGnBu_r',
'YlOrBr_r',
'YlOrRd_r',
'afmhot_r',
'autumn_r',
'binary_r',
'bone_r',
'brg_r',
'bwr_r',
'cool_r',
'coolwarm_r',
'copper_r',
'cubehelix_r',
'flag_r',
'gist_earth_r',
'gist_gray_r',
'gist_heat_r',
'gist_ncar_r',
'gist_rainbow_r',
'gist_stern_r',
'gist_yarg_r',
'gnuplot_r',
'gnuplot2_r',
'gray_r',
'hot_r',
'hsv_r',
'jet_r',
'nipy_spectral_r',
'ocean_r',
'pink_r',
```

```
'prism_r',
'rainbow_r',
'seismic_r',
'spring_r',
'summer_r',
'terrain_r',
'winter_r',
'Accent_r',
'Dark2_r',
'Paired_r',
'Pastel1_r',
'Pastel2_r',
'Set1_r',
'Set2_r',
'Set3_r',
'tab10_r',
'tab20_r',
'tab20b_r',
'tab20c_r']
```

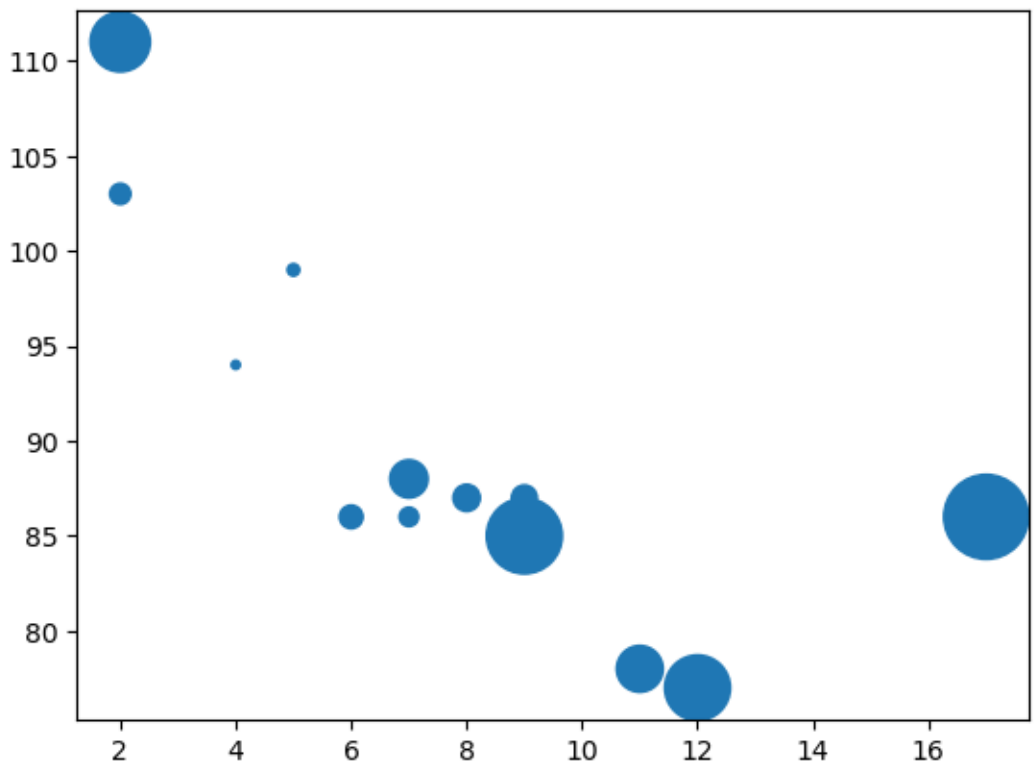
1.35 Tamaño

Puede cambiar el tamaño de los puntos con el sargumento.

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

Ejemplo. Establezca su propio tamaño para los marcadores.

[ ]:
```
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])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

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

plt.show()
```



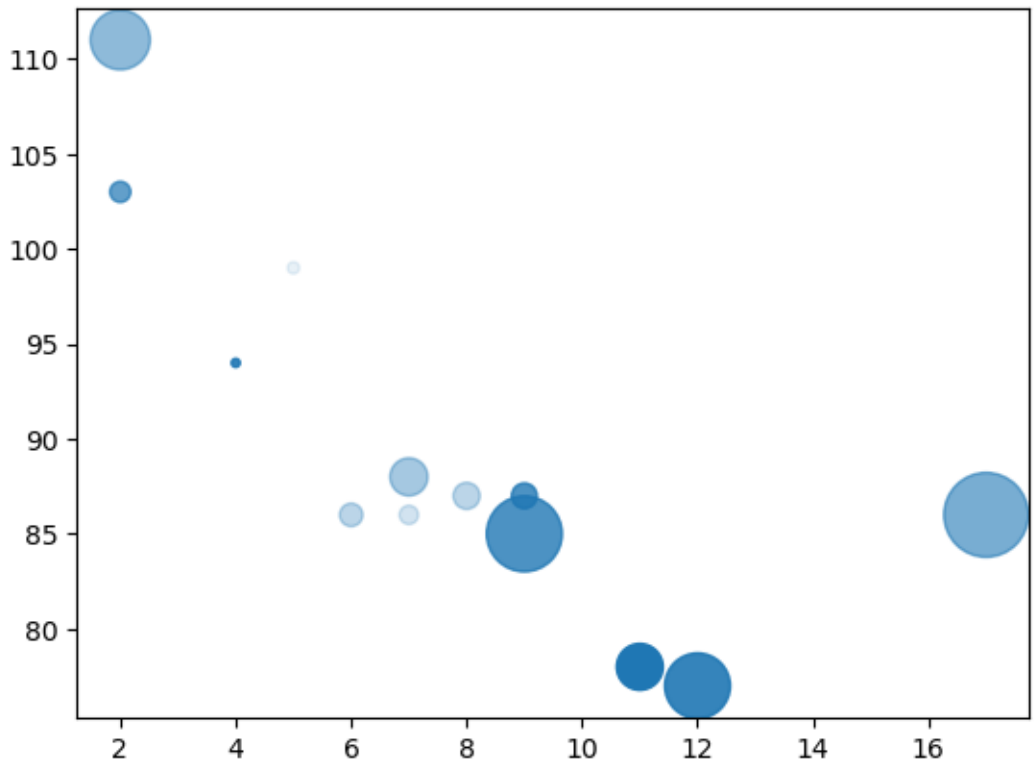
1.36 Transparencia

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

Al igual que con los colores, es posbile ajustar la transparencia a los puntos individualmente, sólo asegúrese de que la matriz de tamaños tenga la misma longitud que las matrices de ejes x y y.

Ejemplo. Establezca su propio tamaño para los marcadores.

[ ]:
```
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])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])
transparency = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 0.9, 0.
↪8, 0.3])

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

plt.show()
```

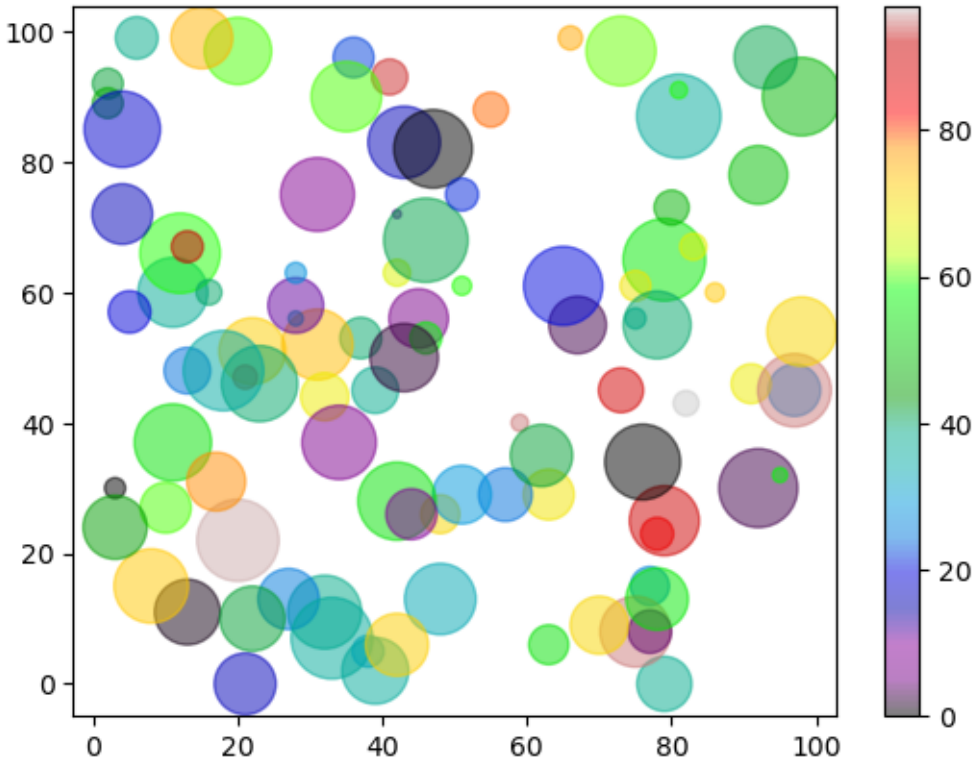


1.37 Combinar color, tamaño y alfa

Se puede combinar un mapa de colores con diferentes tamaños de puntos. Esto se visualiza mejor si los puntos son transparentes.

Ejemplo. Crear matrices aleatorias con 100 valores para puntos x, puntos y, colores y tamaños.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100))
sizes = 10 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')

plt.colorbar()

plt.show()
```

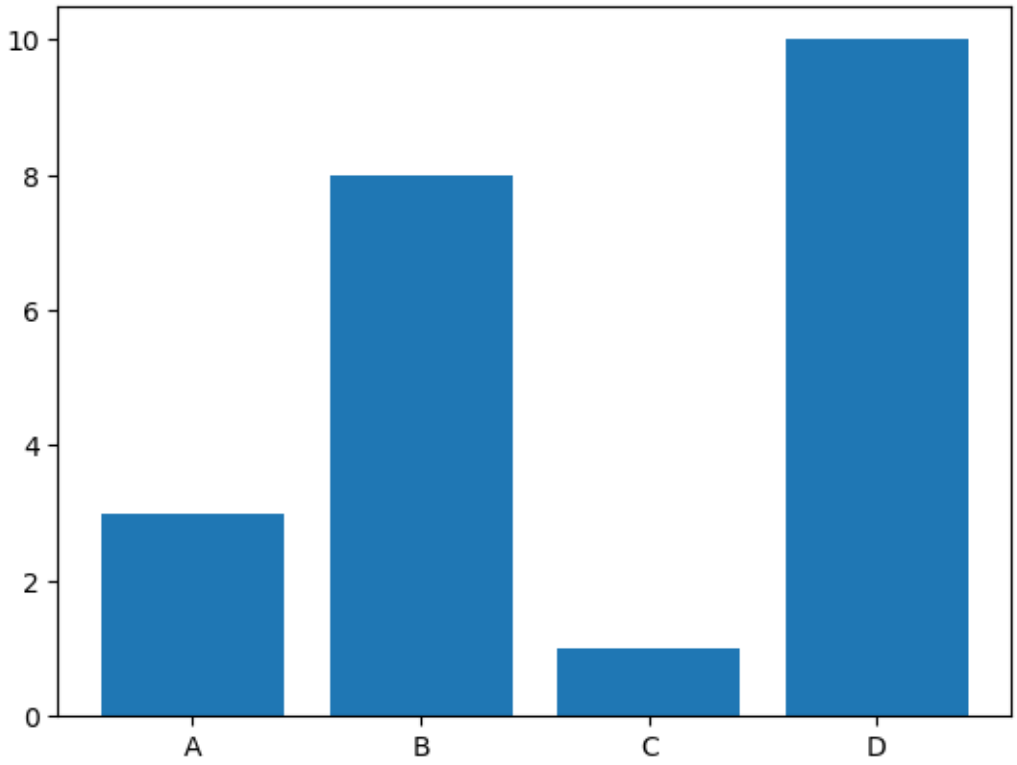


2 Diagrama de barras

Con `Pyplot`, se puede utilizar la función `bar()` para dibujar gráficos de barras.

Ejemplo. Dibuja 4 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()
```



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

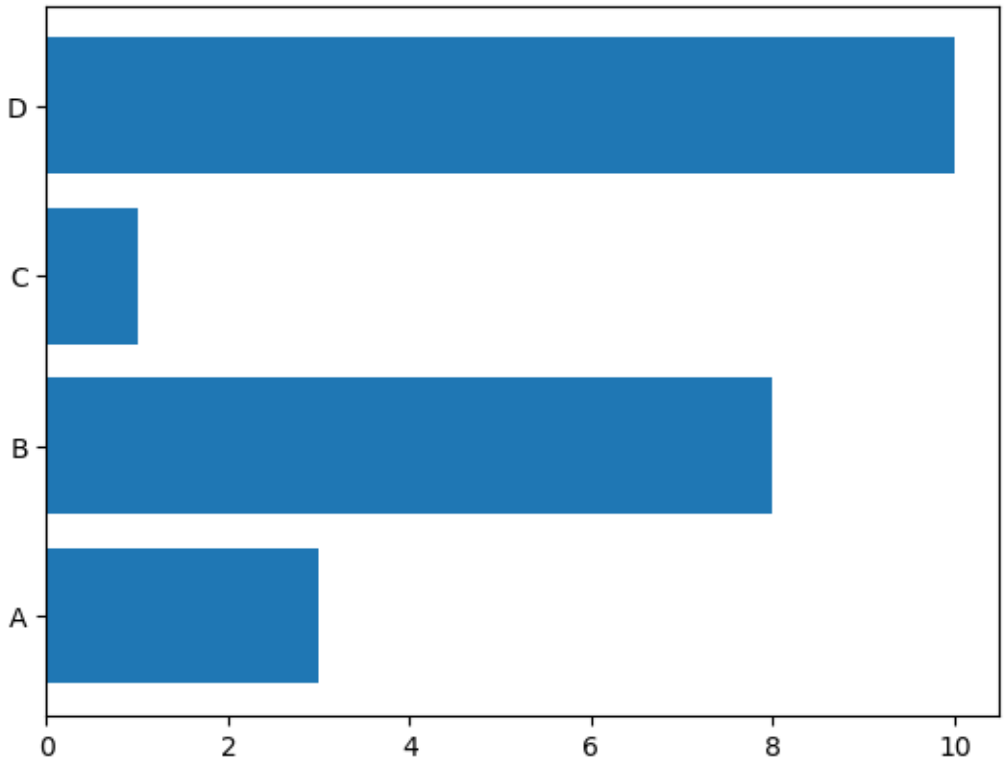
Las categorías y sus valores representados por el primer y segundo argumento como matrices.

2.1 Barras horizontales

Si desea que las barras se muestren horizontalmente en lugar de verticalmente, utilice la función `barh()`.

Ejemplo. Dibuja 4 barras horizontales.

[ ]:
```
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()
```

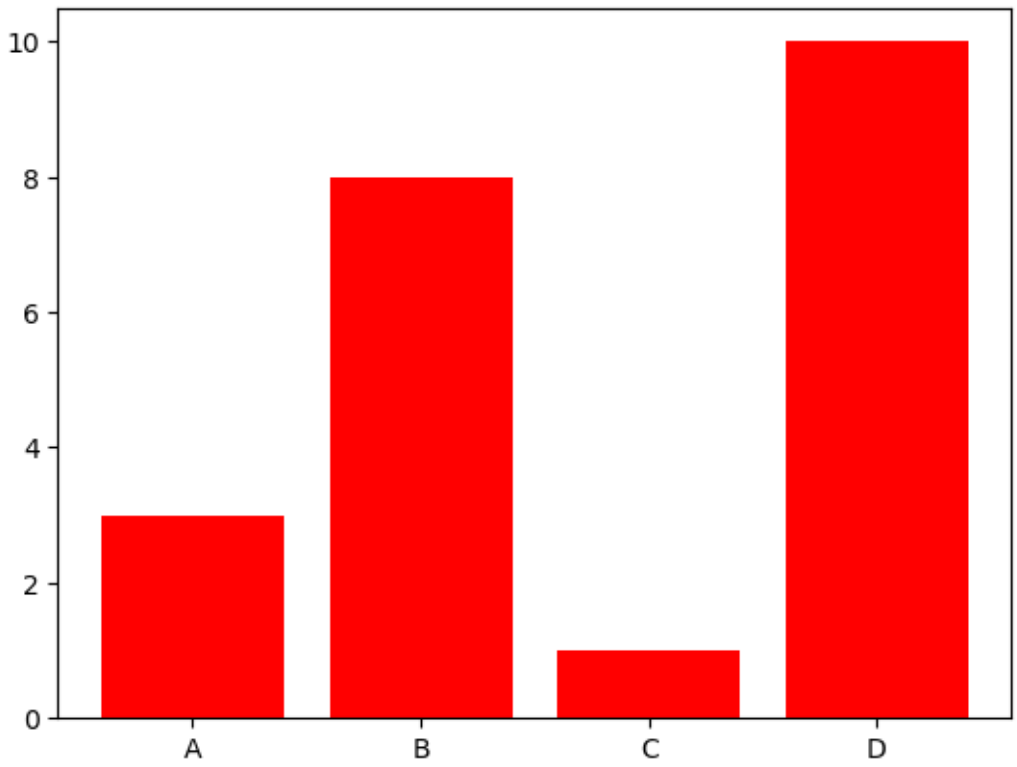


2.2 Color de la barra

El argumento `color` de palabra clave `bar()` y `barh()` se utiliza para establecer el color de las barras.

Ejemplo. Dibuja 4 barras rojas.

[ ]:
```
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, color = "red")
plt.show()
```



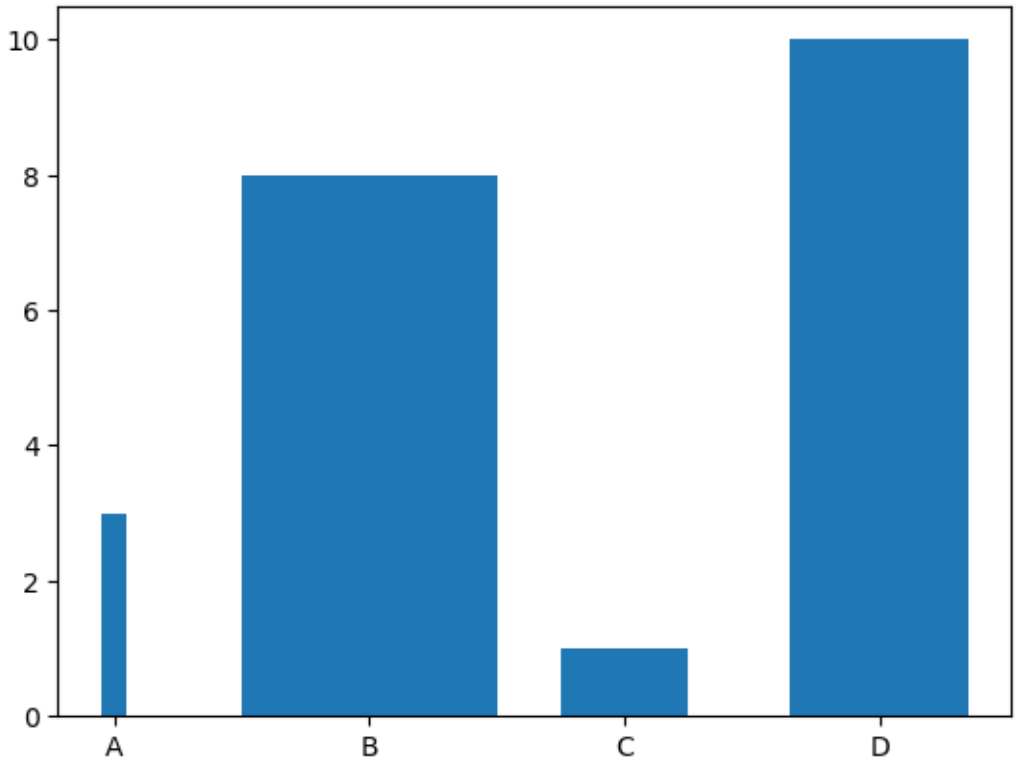
Nota. Puede utilizar cualquiera de los 140 nombres de colores admitidos o valores de color hexadecimales.

2.3 Ancho de barra

El argumento `width` de palabra clave `bar()` permite establecer el ancho de las barras.

Ejemplo. Dibuja 4 barras muy finas.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
width = np.array([0.1, 1, 0.5, 0.7])

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



El valor de ancho predeterminado es 0.8.

Nota: Para barras horizontales, utilice `height` en lugar de `width`.

3 Histograma

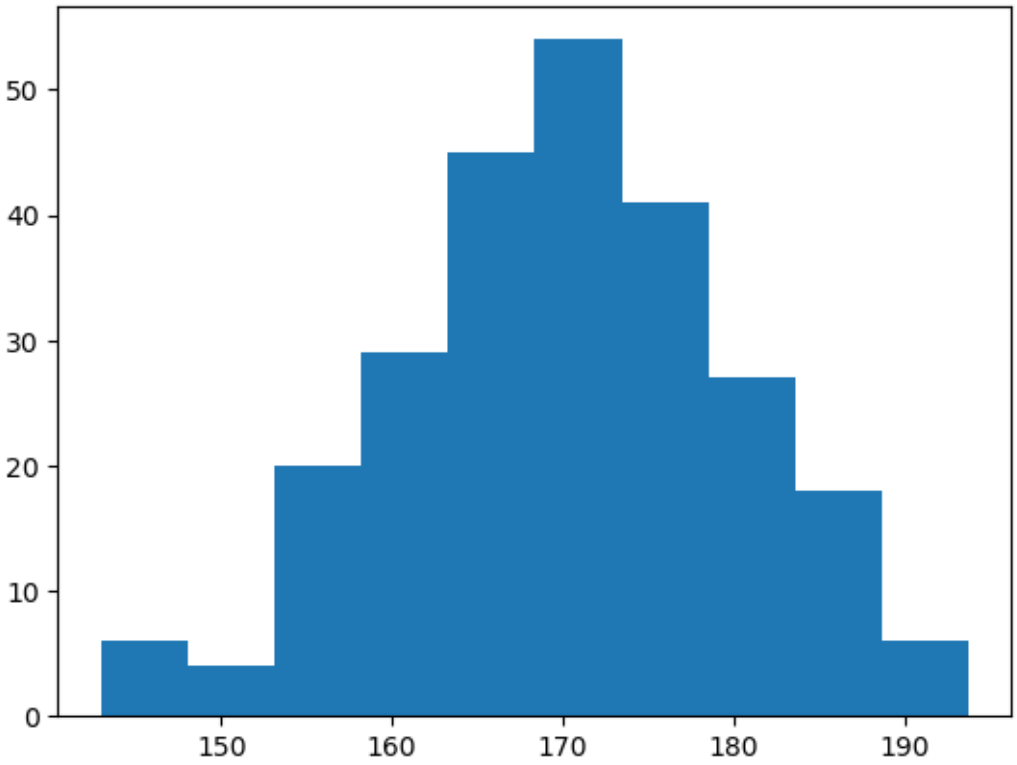
En Matplotlib, se utiliza la función `hist()` para crear histogramas.

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

A manera de ejemplo, se utiliza `NumPy` para generar una matriz con 250 valores aleatorios, donde los valores se concentrarán alrededor de 170 y la desviación estándar es 10.

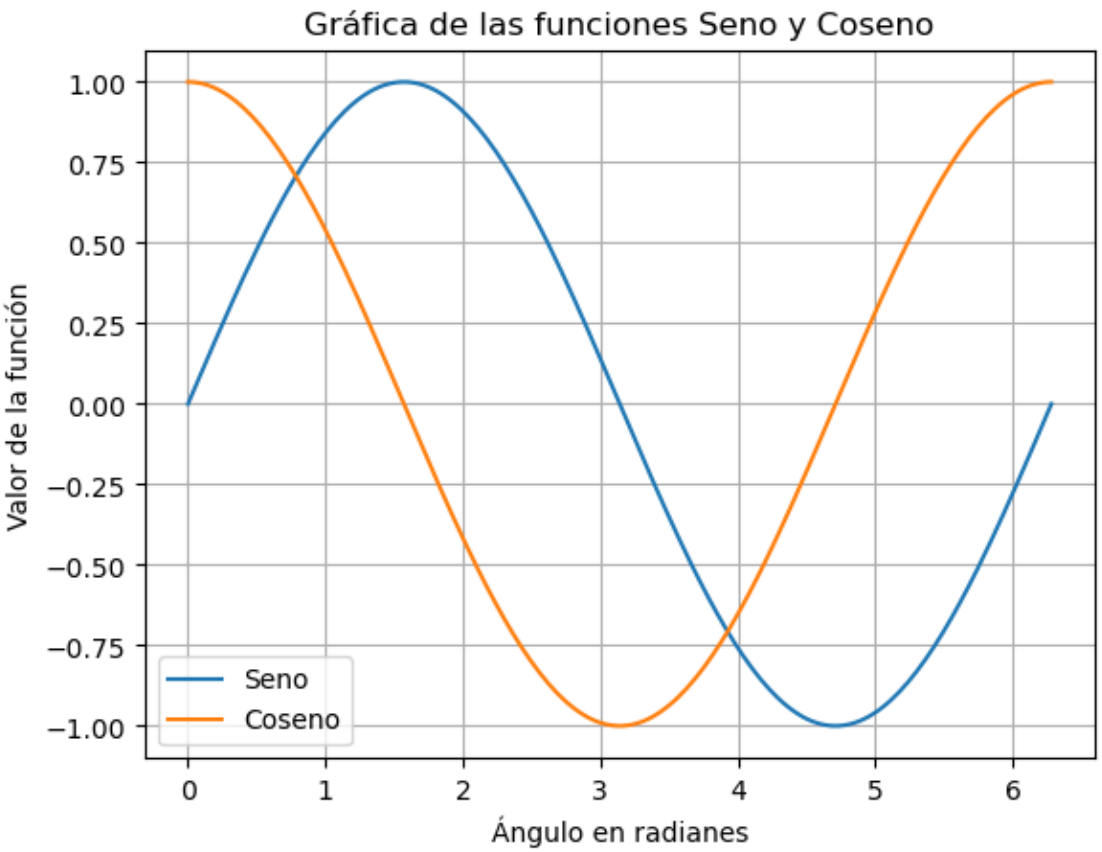
Ejemplo. Una distribución de datos normal por NumPy.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

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



Ejemplo. Graficar las funciones Seno y Coseno.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib

x = np.linspace(0, 2 * np.pi, 200)
y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y1, label="Seno")
plt.plot(x, y2, label="Coseno")

plt.legend()
plt.grid()
plt.title('Gráfica de las funciones Seno y Coseno')
plt.xlabel('Ángulo en radianes')
plt.ylabel('Valor de la función')

plt.show()
```



4 Referencias

- Matplotlib en GitHub