Visualización con Matplotlib: Líneas Sencillas

Quizás el más simple de todos los gráficos es la visualización de una sola función $y = f(x)$.

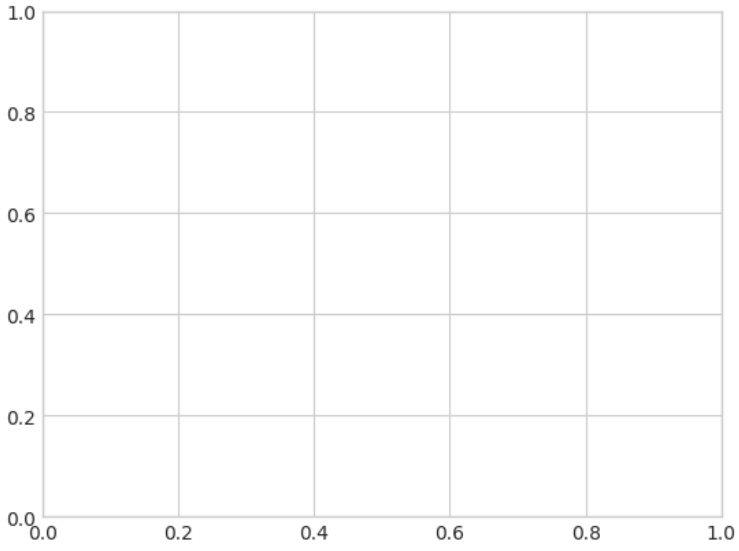
En esta sesión daremos un primer vistazo a la creación de un gráfico simple de este tipo. Comenzaremos por configurar el cuaderno para trazar e importar los paquetes que usaremos:

```
# recuerda que si lo necesitas puedes incluir: %matplotlib inline
import matplotlib.pyplot as plt
plt.style.use("seaborn-v0_8-whitegrid")
import numpy as np
```

Anatomía de una Figura y un Axe

Para todos los gráficos de Matplotlib, comenzamos creando una figura y unos ejes (objeto tipo *axes*, ojo eje en inglés se dice *axis*). [recuerda que una figura es un conjunto de una o más gráficas que llamamos ejes o mejor axes] En su forma más simple, una figura y unos axes (*ejes*) se pueden crear de la siguiente manera:

```
fig=plt.figure()
ax=plt.axes()
```



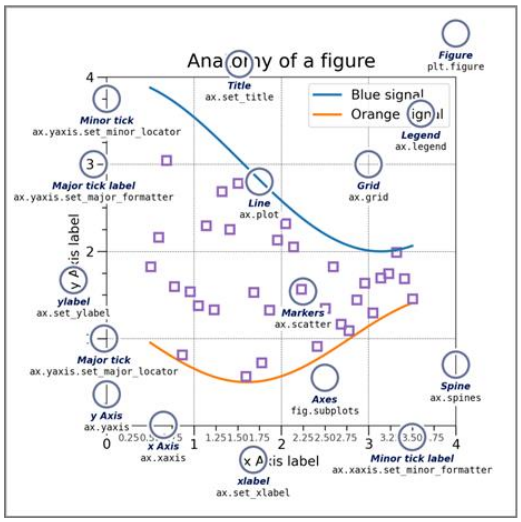
En Matplotlib, la *figura* (una instancia de la clase plt.Figure) puede ser considerada como un solo contenedor que contiene todos los objetos que representan ejes, gráficos, texto y etiquetas.

Los *ejes* (una instancia de la clase plt.Axes) es lo que vemos arriba: un cuadro delimitador con marcas y etiquetas, que eventualmente contendrá los elementos del gráfico que conforman nuestra visualización.

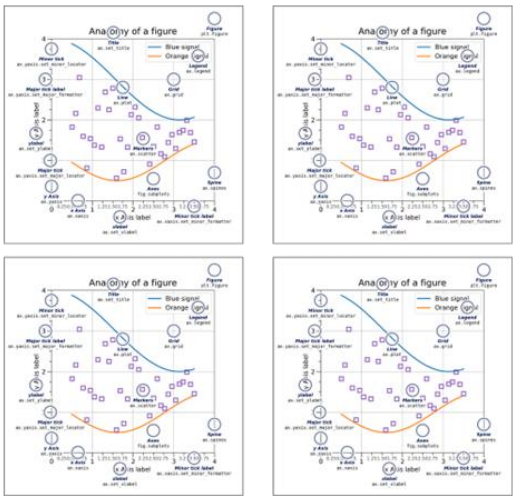
Usaremos el nombre de la variable fig para referirnos a una instancia de figura, y ax para referirnos a una instancia de ejes o grupo de instancias de ejes.

Estas son las partes de un objeto de la clase Figure de matplotlib:

Un objeto Figure matplotlib con un Axe



Un objeto Figure matplotlib con 4 Axes dispuestos en 2x2

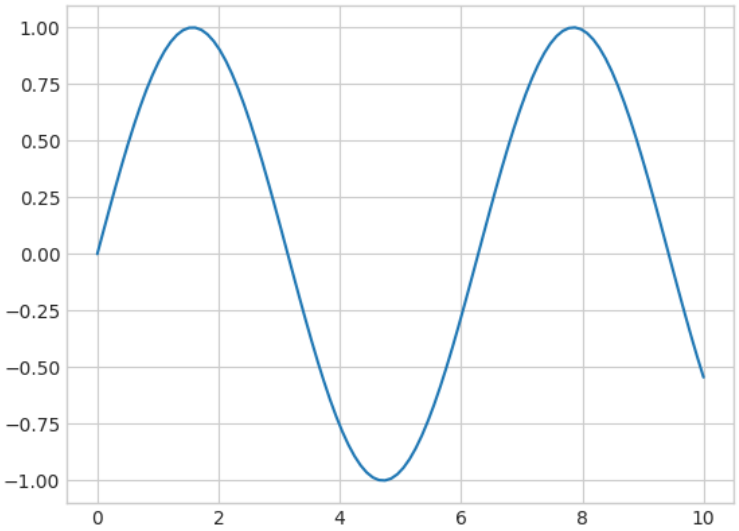


En realidad lo que ves son los elementos de un axe (eje) y una figura es un conjunto de axes

Primeros gráficos (axes)

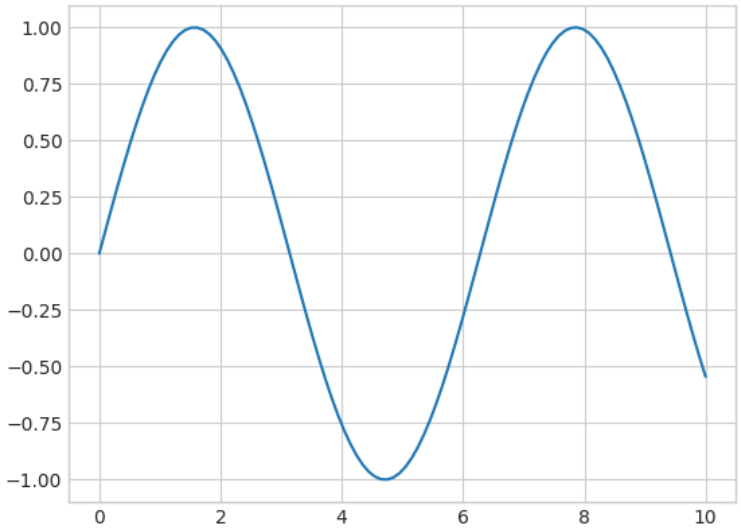
Una vez que hemos creado unos ejes, podemos usar la función ax.plot para trazar algunos datos. Comencemos con la función seno:

```
fig=plt.figure()
ax=plt.axes()
x=np.linspace(0,10,100)
ax.plot(x,np.sin(x));
```



Alternativamente, podemos usar la interfaz de pylab y dejar que la figura y los ejes se creen automáticamente en segundo plano

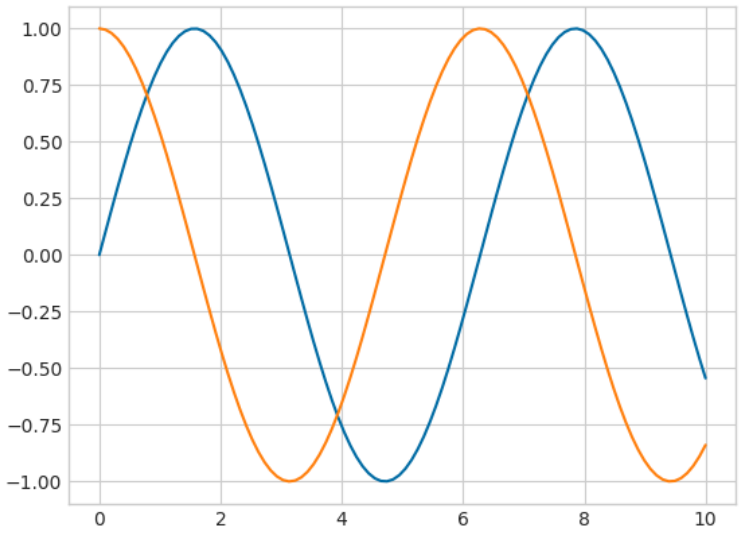
```
plt.plot(x,np.sin(x));
```



De esta segunda manera, no se pueden volver a tocar la gráfica mientras que mediante la primera alternativa sí.

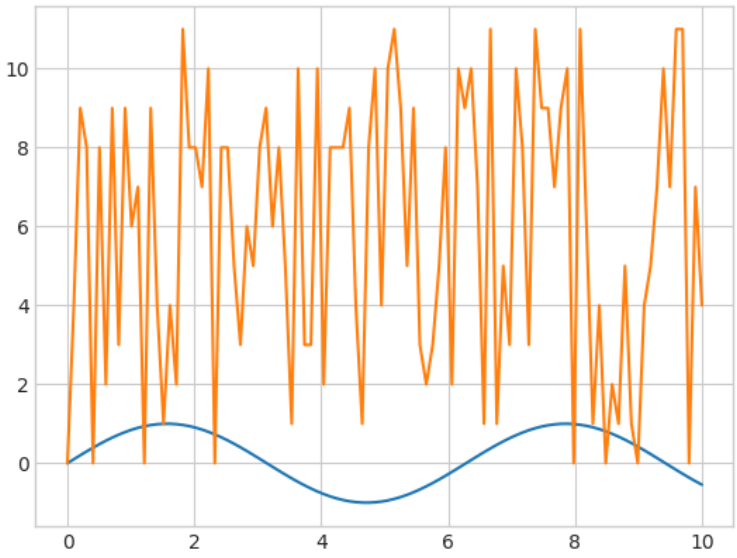
Si queremos crear una sola figura con múltiples líneas, simplemente podemos llamar a la función plot varias veces:

```
plt.style.use("tableau-colorblind10")
plt.plot(x,np.sin(x))
plt.plot(x,np.cos(x));
```



Eso es todo lo que hay que hacer para trazar funciones simples en Matplotlib. Recuerda que en realidad lo que hace es pintar puntos asociados y unirlos.

```
y=np.random.randint(0,12,len(x))
ax.plot(x,y)
display(fig)
```

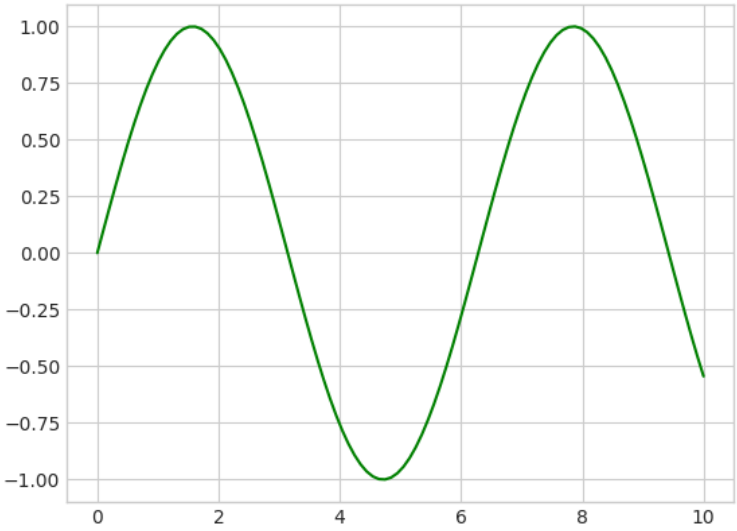


Veamos ahora cómo controlar la apariencia de los ejes, de las líneas y como introducir etiquetas

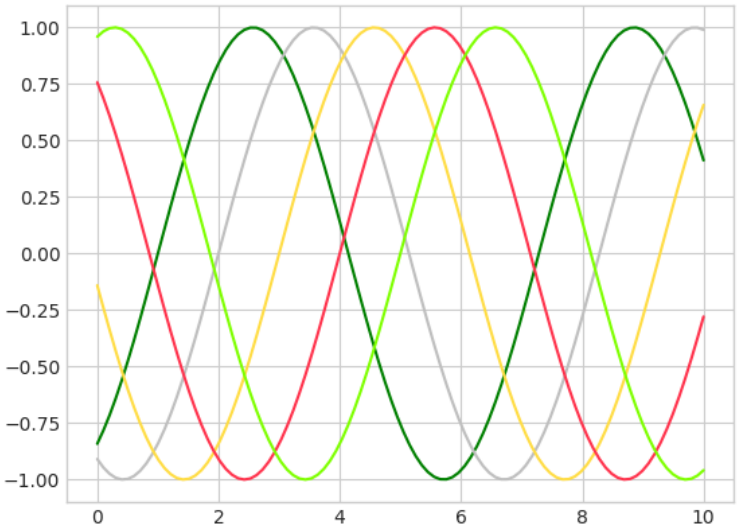
Colores y Estilos de Línea

La función plt.plot() toma argumentos adicionales que se pueden usar para especificar estos. Para ajustar el color, puedes usar la palabra clave color, que acepta un argumento de tipo string que representa virtualmente cualquier color imaginable. El color se puede especificar de varias maneras:

```
# specify color by name
plt.plot(x,np.sin(x-0),color="green");
```



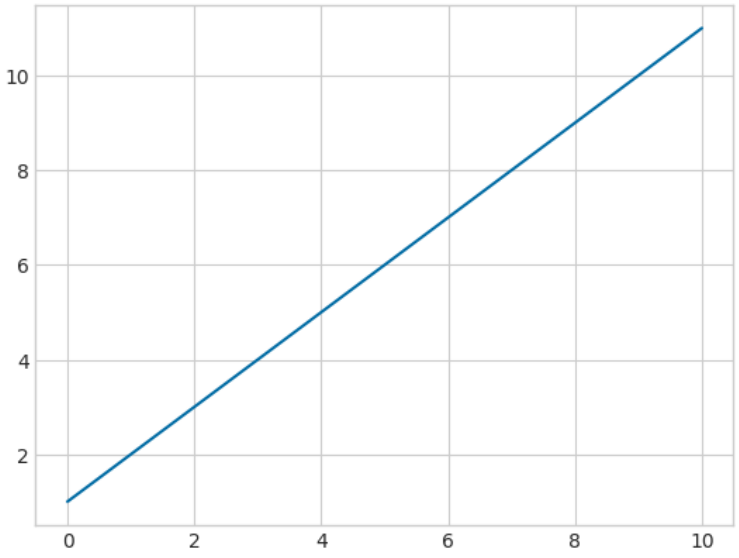

```
plt.plot(x, np.sin(x - 1), color='g')              # short color code (rgbcmyk)
plt.plot(x, np.sin(x - 2), color='0.75')           # Grayscale between 0 and 1
plt.plot(x, np.sin(x - 3), color='#FFDD44')        # Hex code (RRGGBB from 00 to FF)
plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3))  # RGB tuple, values 0 to 1
plt.plot(x, np.sin(x - 5), color='chartreuse');  # all HTML color names supported
```



Si no se especifica color, Matplotlib automáticamente toma un conjunto de colores predeterminados.

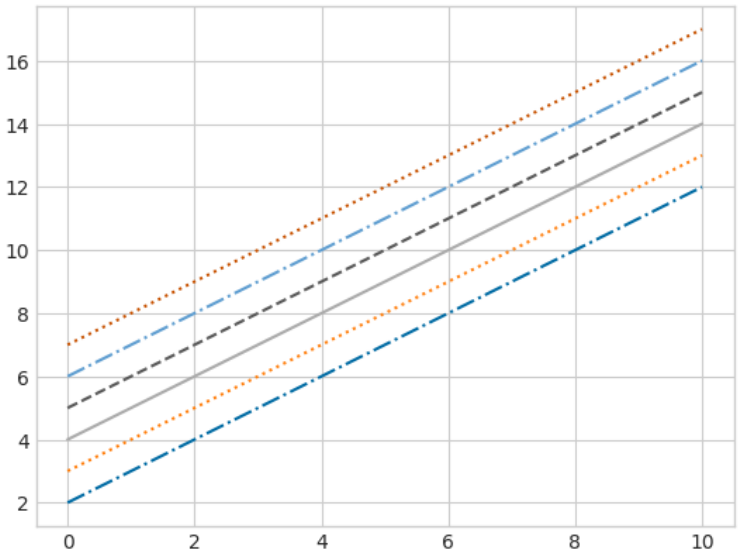
Del mismo modo, el estilo de línea se puede ajustar utilizando la palabra clave linestyle:

plt.plot(x,x+1,linestyle="solid") *#Si no pongo punto y como al final me sale la especificación del objeto:*
[<matplotlib.lines.Line2D at 0x7f121aeb3e80>]



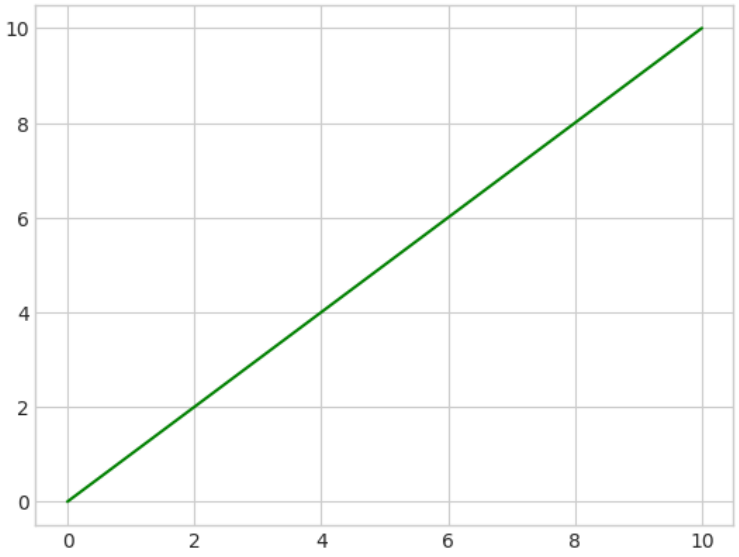

```
plt.plot(x, x + 2, linestyle='dashdot')
plt.plot(x, x + 3, linestyle='dotted');

# For short, you can use the following codes:
plt.plot(x, x + 4, linestyle='-')  # solid
plt.plot(x, x + 5, linestyle='--') # dashed
plt.plot(x, x + 6, linestyle='-.') # dashdot
plt.plot(x, x + 7, linestyle=':');  # dotted
```

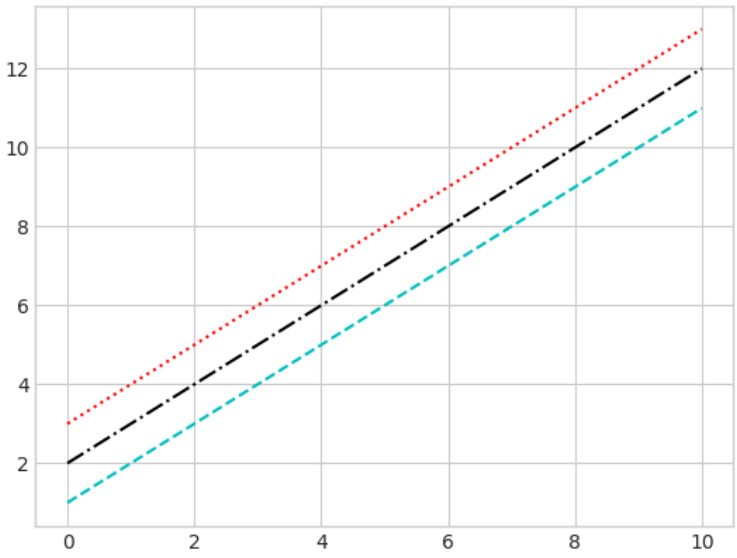


Si quisieras ser extremadamente conciso, estos códigos de linestyle y color pueden combinarse en un solo argumento no clave para la función plt.plot():

plt.plot(x,x,"-g");



plt.plot(x, x + 1, '--c') *# dashed cyan*
plt.plot(x, x + 2, '-.k') *# dashdot black*
plt.plot(x, x + 3, ':r'); *# dotted red*

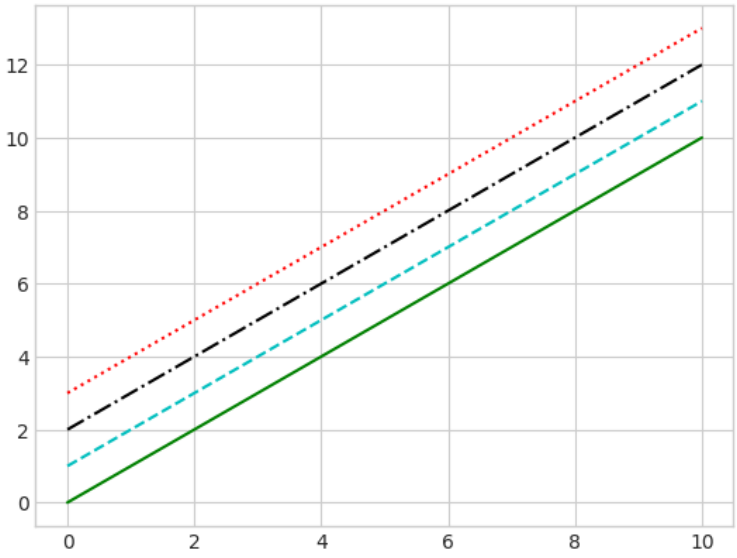


Estos códigos de color de un solo carácter reflejan las abreviaturas estándar en los sistemas de color RGB (Rojo/Verde/Azul) y CMYK (Cian/Magenta/Amarillo/negro), comúnmente utilizados para gráficos de color digital.

Hay muchos otros argumentos clave que se pueden utilizar para ajustar finamente la apariencia del gráfico; para más detalles, como siempre, ir a la documentación de plt.plot

Usarlo con objetos axe es "directo" (por resumir sólo en la versión concentrada):

```
fig = plt.Figure()
ax = plt.axes()
ax.plot(x, x + 0, '-g')  # solid green
ax.plot(x, x + 1, '--c') # dashed cyan
ax.plot(x, x + 2, '-.k') # dashdot black
ax.plot(x, x + 3, ':r');  # dotted red
```

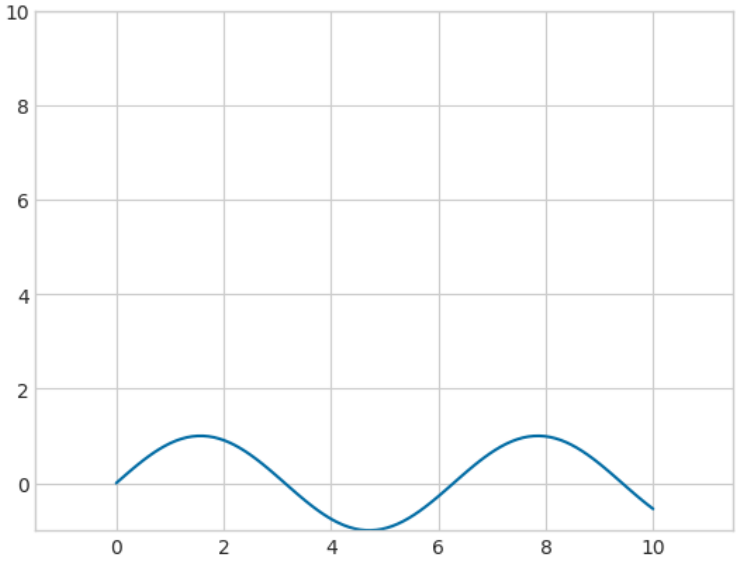


Límites de los Ejes

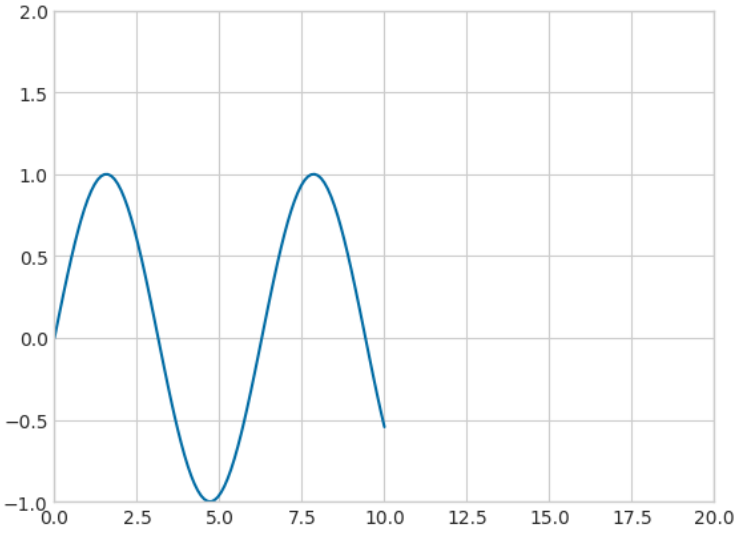
Matplotlib hace, en general, una buena estimación al elegir los límites predeterminados de los ejes de tu gráfico, pero a veces es bueno tener un control más preciso.

La forma más básica de ajustar los límites de los ejes es usar las funciones plt.xlim() y plt.ylim() o los métodos set_xlim y set_ylim caso de que estemos trabajando con objetos Axes:

```
plt.plot(x,np.sin(x))
plt.xlim(-1.5,11.5)
plt.ylim(-1,10);
```



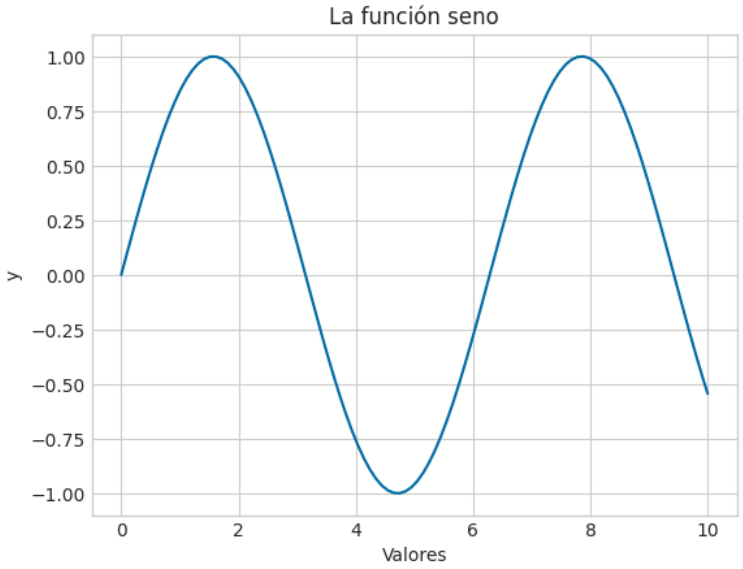

```
fig=plt.figure()
ax=plt.axes()
ax.set_xlim(0,20)
ax.set_ylim(-1,2)
ax.plot(x,np.sin(x));
```



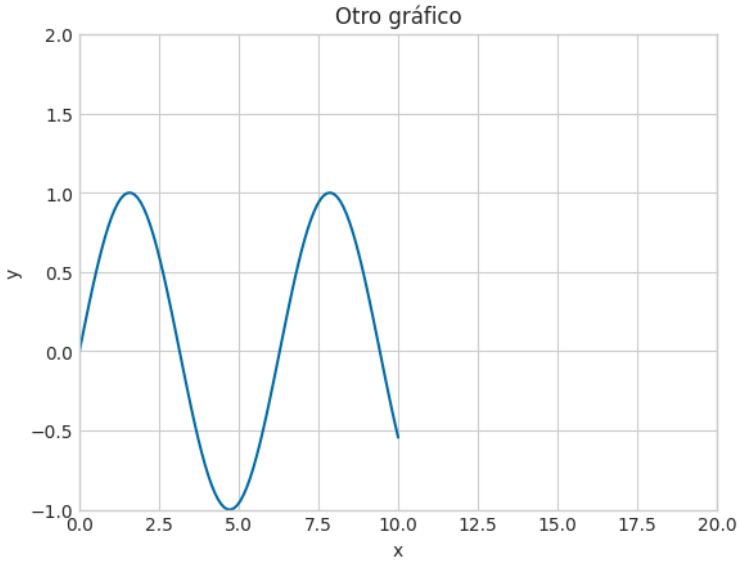
Etiquetado de gráficos (título, etiqueta x, etiqueta y,leyenda)

Los títulos y las etiquetas de ejes son las etiquetas más simples de este tipo:

```
plt.plot(x, np.sin(x))
plt.title("La función seno")
plt.xlabel("Valores")
plt.ylabel("y");
```



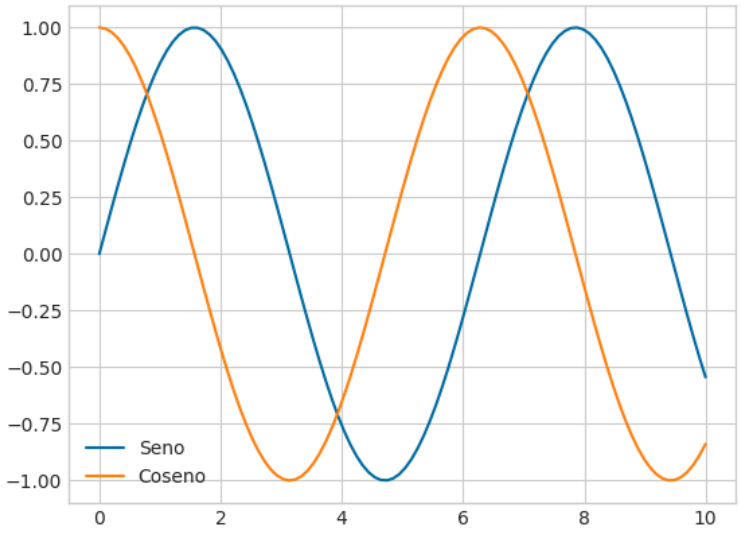

```
fig=plt.figure()
ax=plt.axes()
ax.set_xlim(0,20)
ax.set_ylim(-1,2)
ax.plot(x,np.sin(x))
ax.set_title("Otro gráfico")
ax.set_xlabel("x")
ax.set_ylabel("y");
```



La posición, tamaño y estilo de estas etiquetas se pueden ajustar utilizando argumentos opcionales en la función. Para obtener más información, consulte la documentación de Matplotlib.

Matplotlib tiene una forma incorporada de crear rápidamente una leyenda, cuando hay varias figuras:

```
plt.plot(x,np.sin(x),label="Seno")
plt.plot(x,np.cos(x),label="Coseno")
plt.legend();
```




```
fig = plt.figure()
ax = plt.axes()
ax.set_xlim(0,20)
ax.plot(x, np.sin(x), label = "Sen")
ax.plot(x, np.cos(x), label = "Cos", ls = "-.", color = "g")
ax.legend()
ax.set_title("Pruebas y no funciona")
ax.set_xlabel("Valores axe")
ax.set_ylabel("Valores y en axe");
```