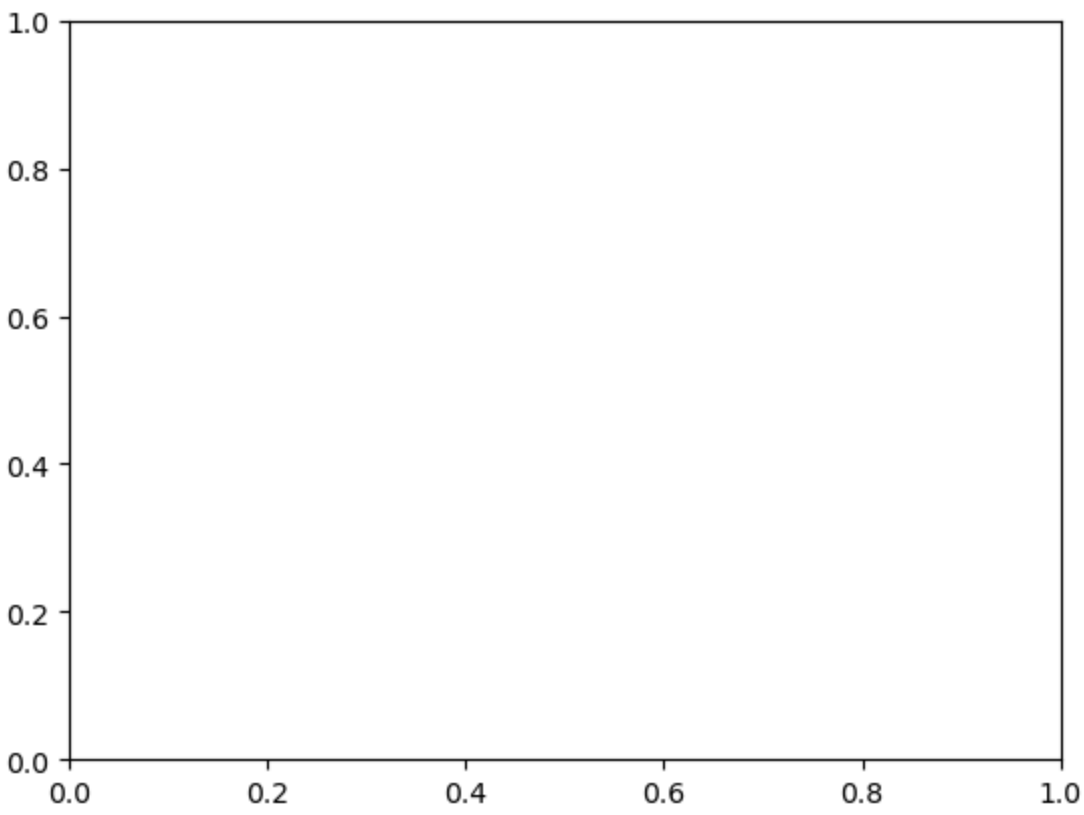
Subplots

Este es perteneciente al metodo por objetos y es algo similar al de pyplot.

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

In [ ]:
```
#Definiendo variables
x = np.linspace(0,5,11)
y = np.sin(x)
```

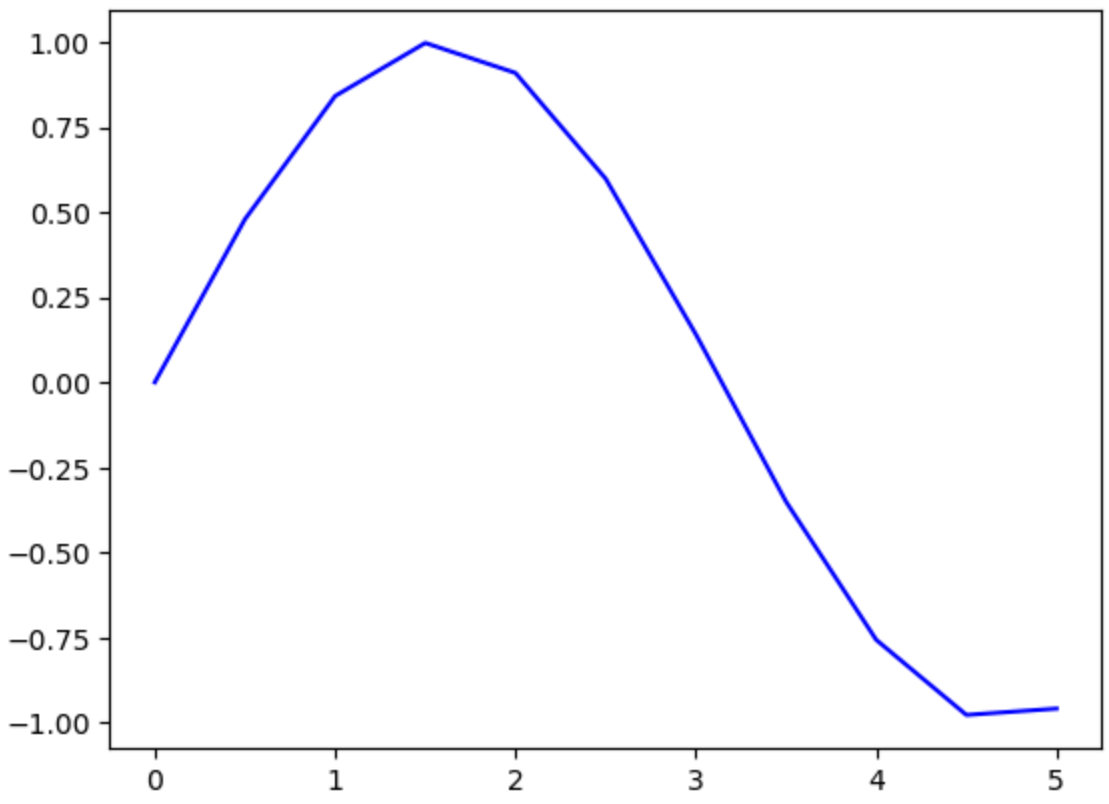
In [ ]:
```
#Metodo por objetos
fig, axes = plt.subplots()
#Vamos a ver que hace
```



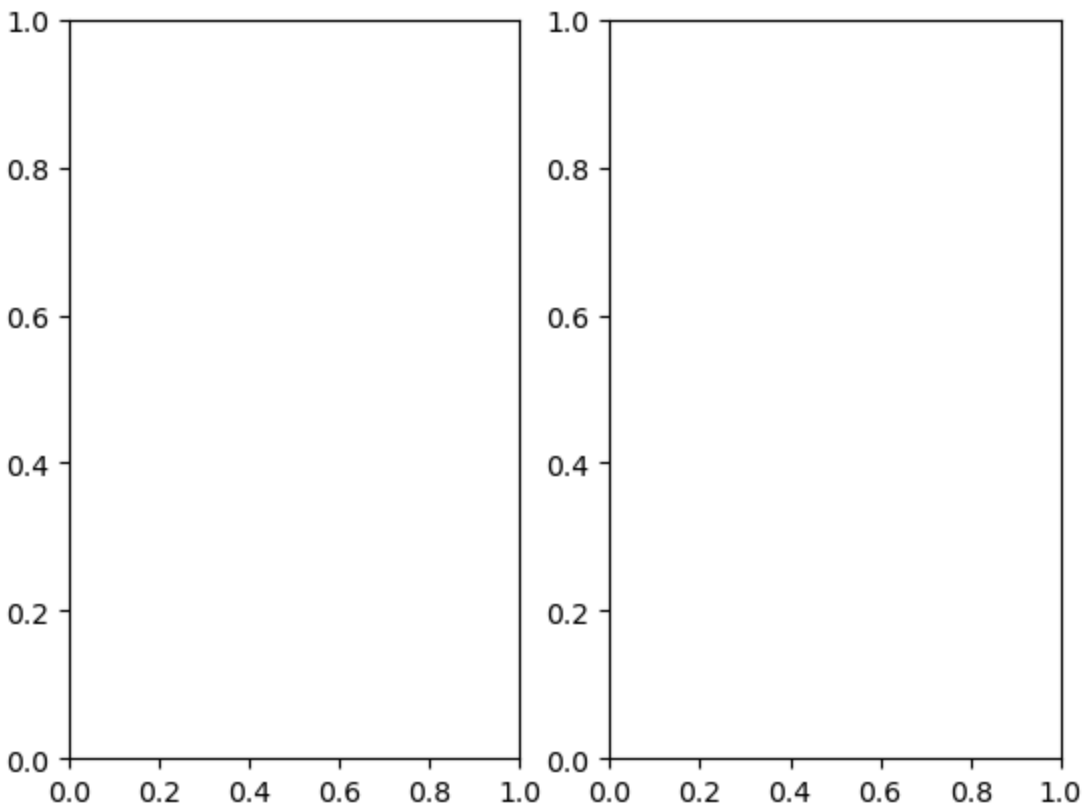
> Esto me crea una figura de lienzo, con un grafico adentro

In [ ]:
```
#Metodo por objetos
fig, axes = plt.subplots()

#Hagamos un plot
axes.plot(x,y,'b')  #Azul
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fa8a407d0a0>]
```



In [ ]:
```
#Metodo por objetos
#Definiendo parametros para subplots
fig, axes = plt.subplots(nrows=1,ncols=2)
```

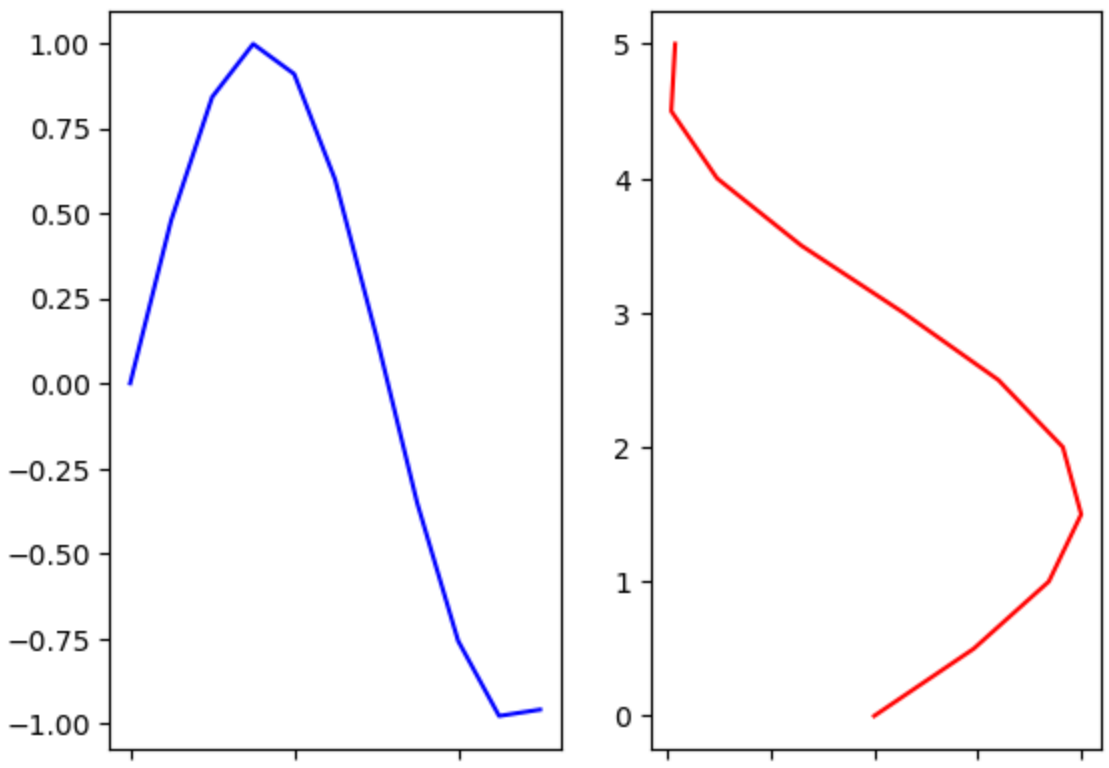


Como vemos el código anterior me arroja, este resultado. Creamos 1 fila y 2 columnas. Esto hace referencia a `axes` es decir **gráficas** que se encuentran dentro de la figura `fig`.

Vamos a ver cómo se puede acceder a esos gráficos y lo interesante de trabajar son `subplots()`.

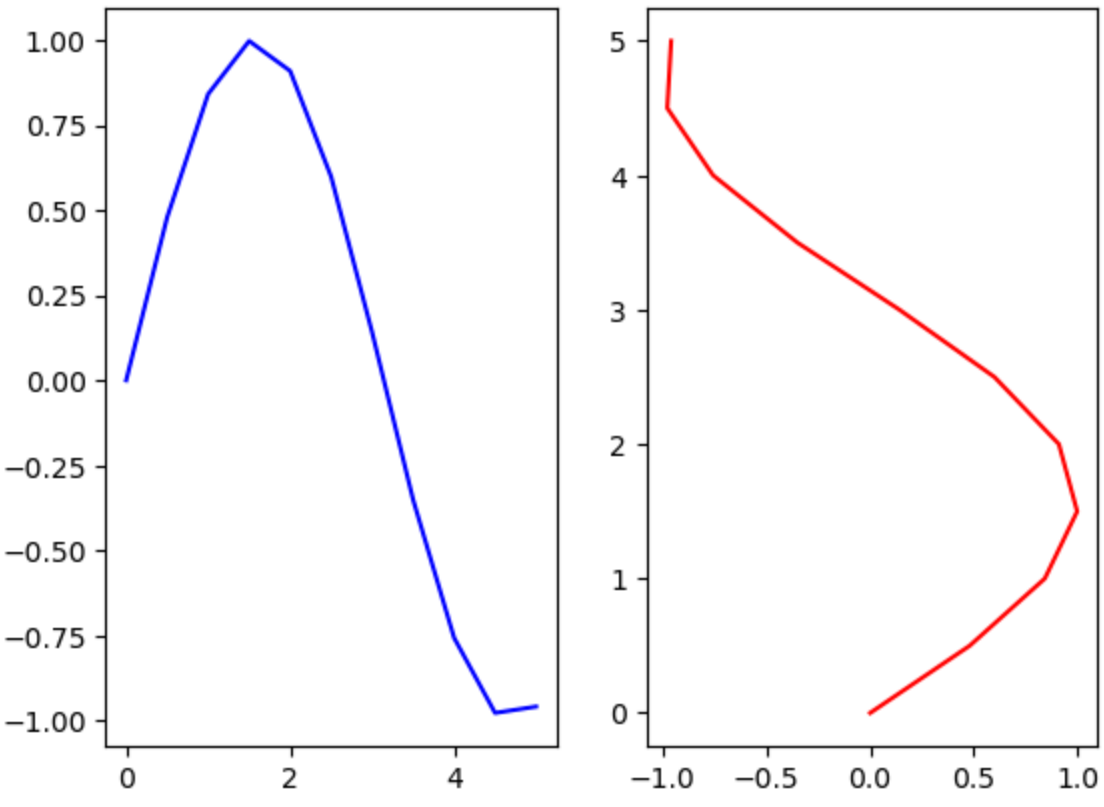
In [ ]:
```
#Metodo por objetos

#Definiendo parametros para subplots
fig, axes = plt.subplots(nrows=1,ncols=2)

#Trabajando con axes - graficos
axes[0].plot(x, y,'b')
axes[1].plot(y, x,'r')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fa8a3fdc6e0>]
```



In [ ]:
```
#Metodo por objetos

#Definiendo parametros para subplots
#Con axes como tuplas
fig, (ax1,ax2) = plt.subplots(nrows=1,ncols=2)

#Trabajando con axes - graficos COMO TUPLAS
ax1.plot(x, y,'b')
ax2.plot(y, x,'r')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fa89bd4b2c0>]
```



Conclusión:

Lo que me arroja es exactamente lo mismo:

```
fig, axes = plt.subplots(nrows=1,ncols=2)

#Trabajando con axes - graficos
axes[0].plot(x, y,'b')
axes[1].plot(y, x,'r')
```

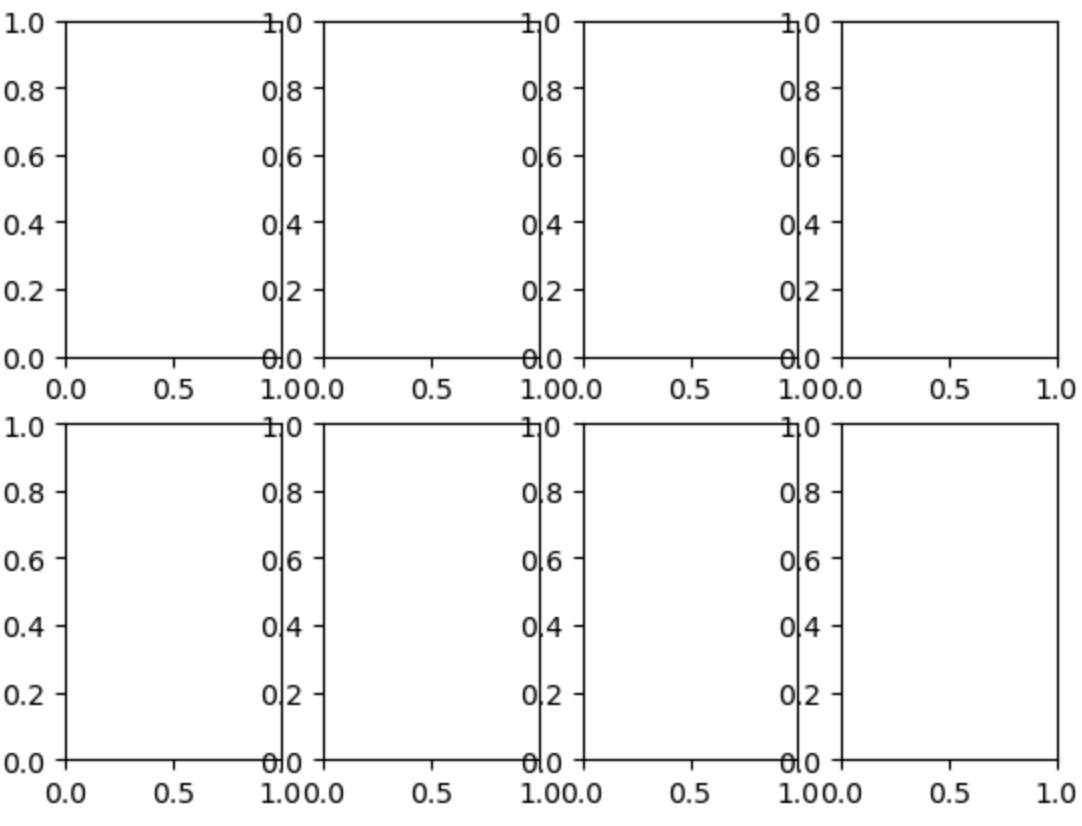
A esto:

Esto es trabajar con tuplas

```
#Definiendo parametros para subplots
#Con axes como tuplas
fig, (ax1,ax2) = plt.subplots(nrows=1,ncols=2)

#Trabajando con axes - graficos COMO TUPLAS
ax1.plot(x, y,'b')
ax2.plot(y, x,'r')
```

Estas son algunas ventajas de trabajar con métodos orientados a objetos, es decir tienes mayor grado de control

In [ ]:
```
#Metodo por objetos

#Definiendo parametros para subplots
fig,axes = plt.subplots(nrows=2,ncols=4)
```



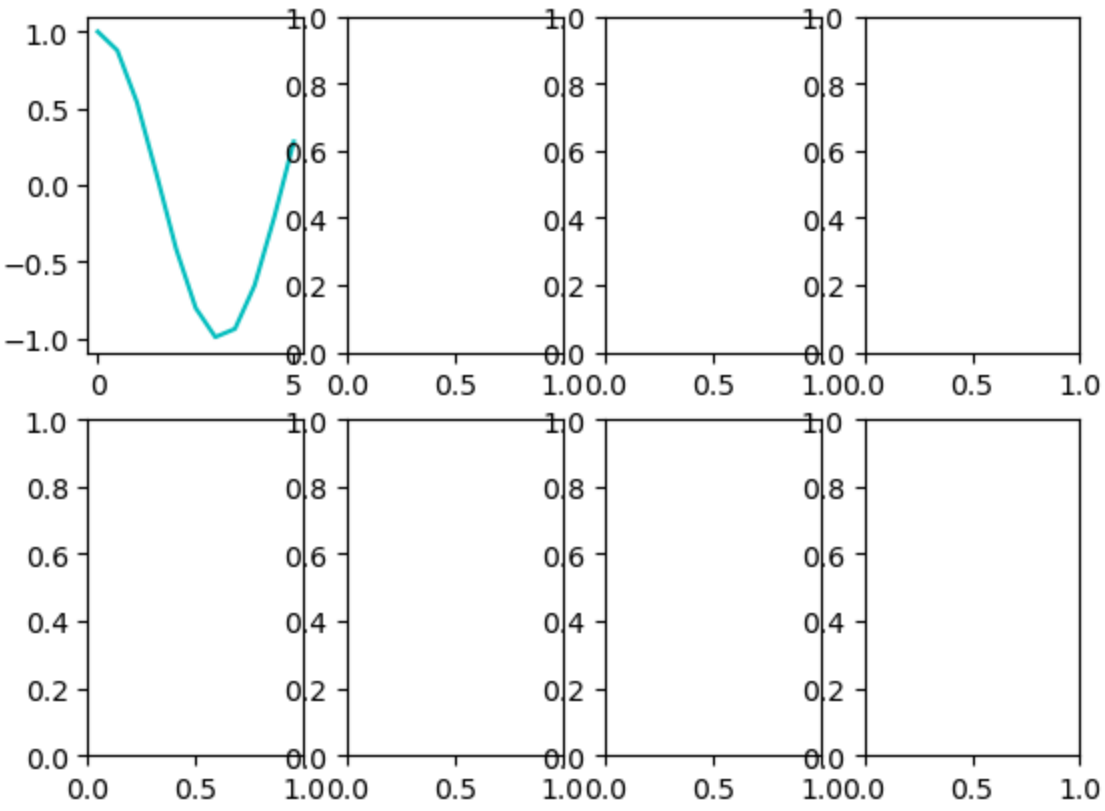
Tengo mi lienzo (canvas) como lo solicité.

- 2 filas
- 2 columnas

In [ ]:
```
#Accediendo a las gráficas
```

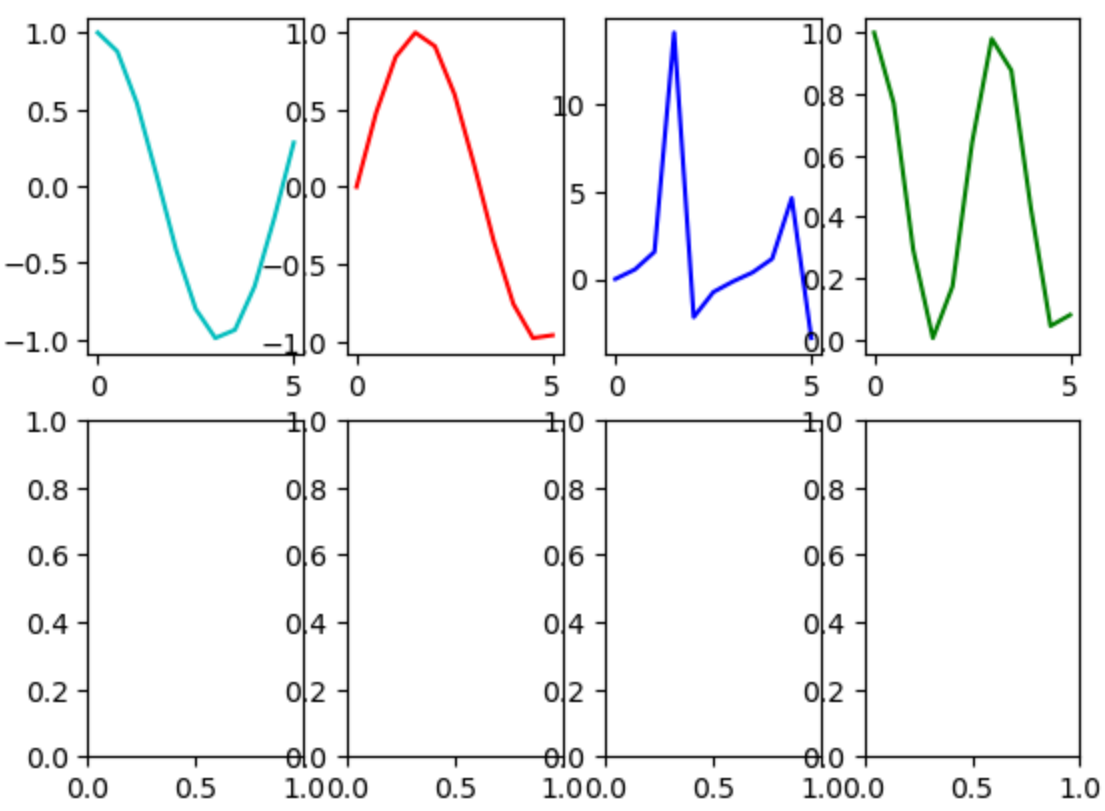
```
#Metodo por objetos

#Definiendo parametros para subplots
fig, axes = plt.subplots(nrows=2,ncols=4)

#Trabajando con axes
axes[0,0].plot(x,np.cos(x),'c')
```

Out[ ]: [<matplotlib.lines.Line2D at 0x7fa89b768080>]



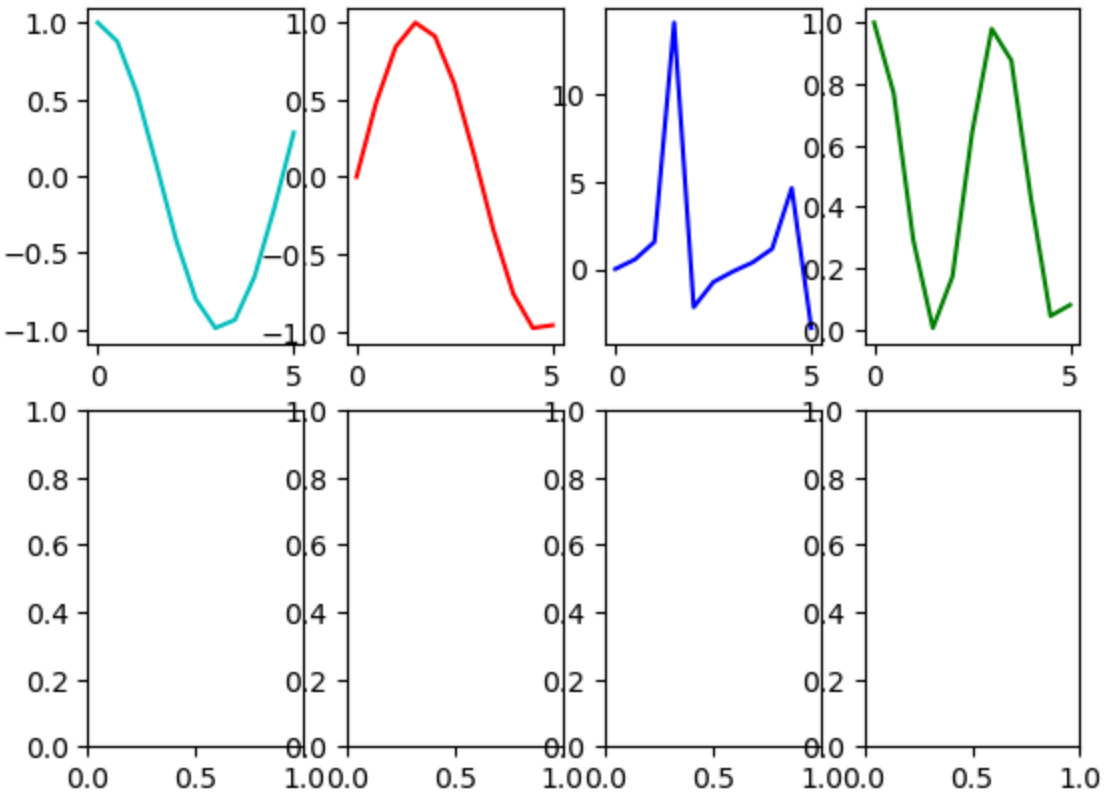
In [ ]:
```
#Accediendo a las gráficas

#Metodo por objetos

#Definiendo parametros para subplots
fig, axes = plt.subplots(nrows=2,ncols=4)

#Trabajando con axes
axes[0,0].plot(x,np.cos(x),'c') #cyan
axes[0,1].plot(x,np.sin(x),'r') #red
axes[0,2].plot(x,np.tan(x),'b') #blue
axes[0,3].plot(x,np.cos(x)**2,'g') #green
```

Out[ ]: [<matplotlib.lines.Line2D at 0x7fa89b179f40>]



In [ ]:
```
#Metodo por objetos

#Definiendo parametros para subplots
#Con axes como tuplas
fig, ((ax1,ax2,ax3,ax4),(ax5,ax6,ax7,ax8)) = plt.subplots(nrows=2,ncols=4)

#Trabajando con axes - graficos COMO TUPLAS
#Trabajando con axes
ax1.plot(x,np.cos(x),'c') #cyan
ax2.plot(x,np.sin(x),'r') #red
ax3.plot(x,np.tan(x),'b') #blue
ax4.plot(x,np.cos(x)**2,'g') #green
```

Out[ ]: [<matplotlib.lines.Line2D at 0x7fa89bc7b470>]

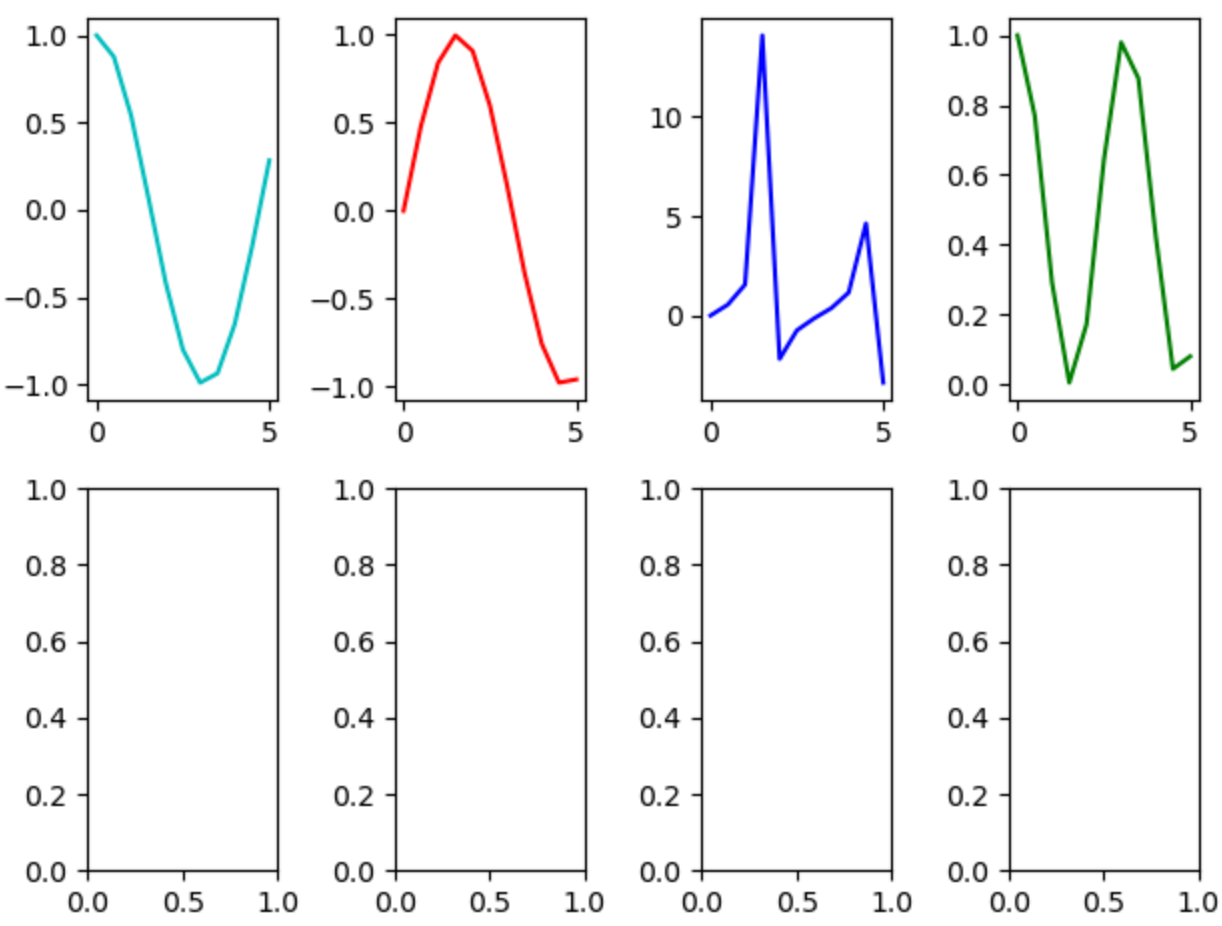


Conclusión:

El método se puede trabajar como un array o como una tupla, en la cual se asigne a cada elemento.
También algo que podemos ver es que las figuras están un poco amontonadas. Esto lo resolvemos con: `fig.tight_layout()`

Veamos

In [ ]:
```
#Metodo por objetos

#Definiendo parametros para subplots
#Con axes como tuplas
fig, ((ax1,ax2,ax3,ax4),(ax5,ax6,ax7,ax8)) = plt.subplots(nrows=2,ncols=4)

#Trabajando con axes - graficos COMO TUPLAS
#Trabajando con axes
ax1.plot(x,np.cos(x),'c') #cyan
ax2.plot(x,np.sin(x),'r') #red
ax3.plot(x,np.tan(x),'b') #blue
ax4.plot(x,np.cos(x)**2,'g') #green

#Acomodando los graficos
fig.tight_layout()
```

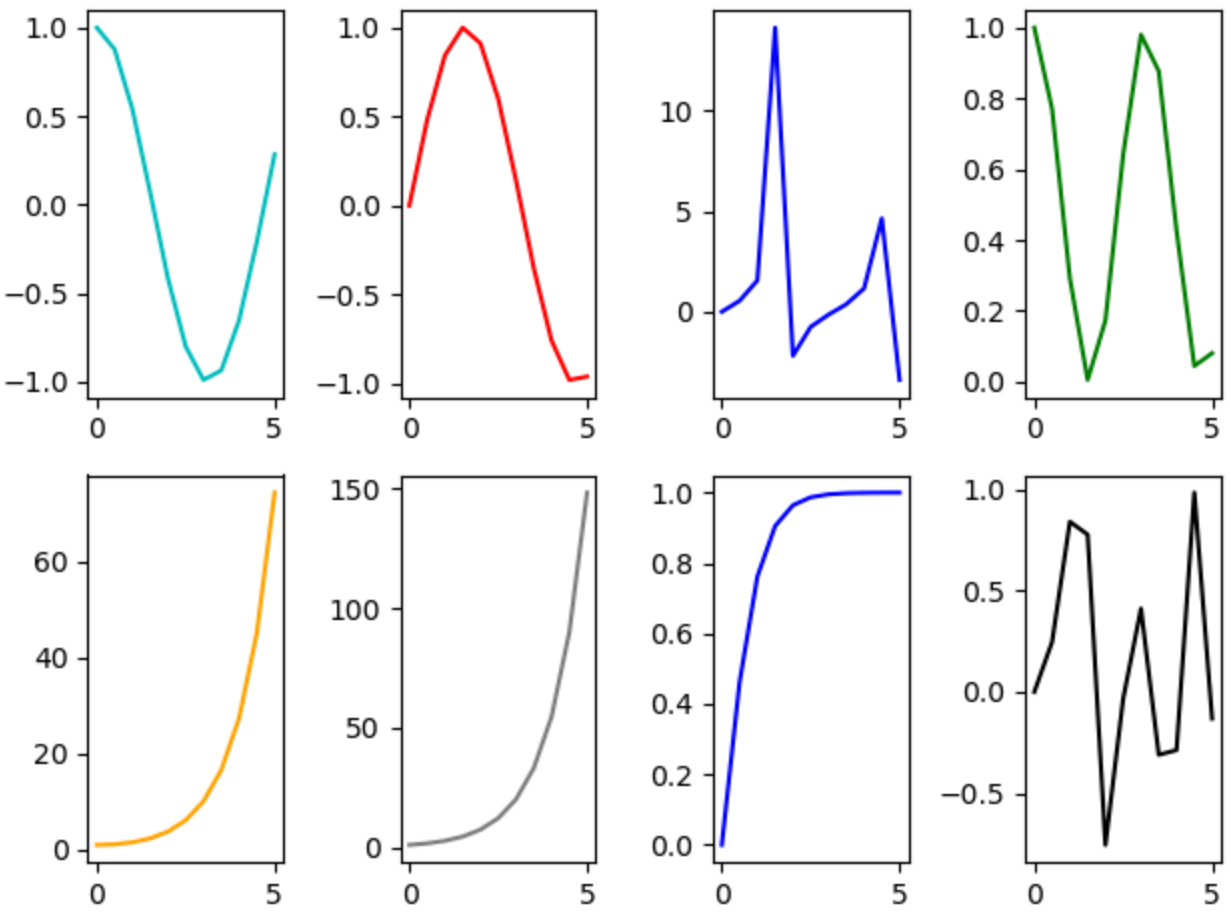


Reto:

Grafica la segunda línea de los subplots anteriormente mencionados. Usa el estilo de tu preferencia y compártenos el resultado en la sección de comentarios.

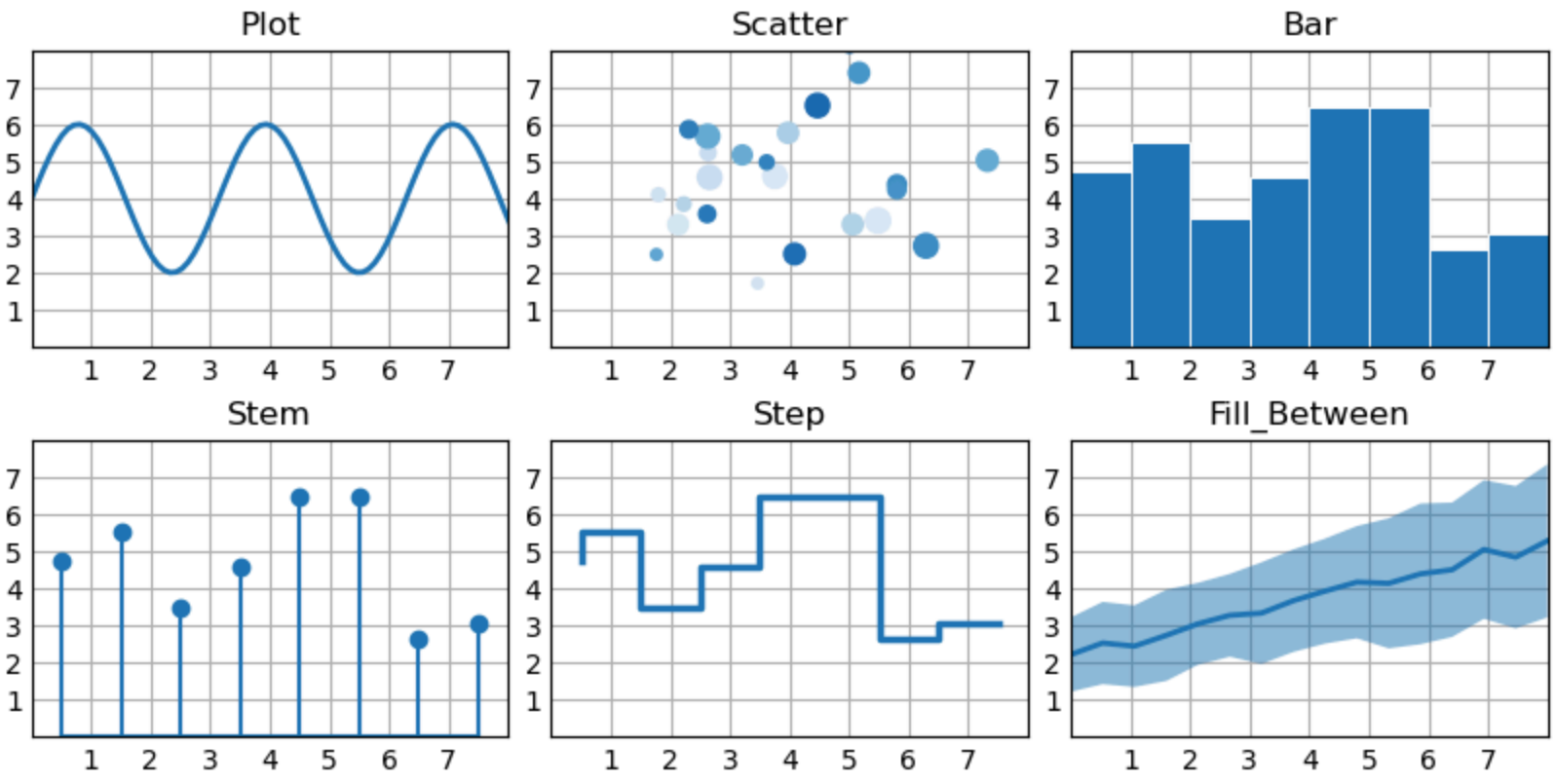
In [ ]:
```
#Metodo por objetos

#Definiendo parametros para subplots
```

```python
#Con axes como tuplas
fig, ((ax1,ax2,ax3,ax4),(ax5,ax6,ax7,ax8)) = plt.subplots(nrows=2,ncols=4)

#Trabajando con axes - graficos COMO TUPLAS
#Trabajando con axes
ax1.plot(x,np.cos(x),'c') #cyan
ax2.plot(x,np.sin(x),'r') #red
ax3.plot(x,np.tan(x),'b') #blue
ax4.plot(x,np.cos(x)**2,'g') #green
ax5.plot(x,np.cosh(x),'orange') #cyan
ax6.plot(x,np.exp(x),'gray') #red
ax7.plot(x,np.tanh(x),'b') #blue
ax8.plot(x,np.sin(x ** 2),'k') #green

#Acomodando los graficos
fig.tight_layout()
```



In [ ]:
```python
#Aporte de LUIS ANTONIO CALVO QUISPE
plt.style.use('_mpl-gallery')

fig, axes = plt.subplots(nrows=2, ncols=3, constrained_layout=True, figsize=(8,4))

# Funciones
x = np.linspace(0, 10, 100)
y = 4 + 2 * np.sin(2 * x)

# Plot
axes[0,0].title.set_text('Plot')
axes[0,0].plot(x, y, linewidth=2.0)
axes[0,0].set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 8), yticks=np.arange(1, 8))

# Funciones
x = 4 + np.random.normal(0, 2, 24)
y = 4 + np.random.normal(0, 2, len(x))
sizes = np.random.uniform(15, 80, len(x))
colors = np.random.uniform(15, 80, len(x))

# Scatter
axes[0,1].title.set_text('Scatter')
axes[0,1].scatter(x, y, s=sizes, c=colors, vmin=0, vmax=100)
axes[0,1].set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 8), yticks=np.arange(1, 8))

# Funciones
np.random.seed(3)
x = 0.5 + np.arange(8)
y = np.random.uniform(2, 7, len(x))

# Bar, Barh
axes[0,2].title.set_text('Bar')
axes[0,2].bar(x, y, width=1, edgecolor="white", linewidth=0.7)
axes[0,2].set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 8), yticks=np.arange(1, 8))

# Funciones
np.random.seed(3)
x = 0.5 + np.arange(8)
y = np.random.uniform(2, 7, len(x))

# Stem
```

```
axes[1,0].title.set_text('Stem')
axes[1,0].stem(x, y)
axes[1,0].set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 8), yticks=np.arange(1, 8))

# Funciones
np.random.seed(3)
x = 0.5 + np.arange(8)
y = np.random.uniform(2, 7, len(x))

# Step
axes[1,1].title.set_text('Step')
axes[1,1].step(x, y, linewidth=2.5)
axes[1,1].set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 8), yticks=np.arange(1, 8))

# Funciones
np.random.seed(1)
x = np.linspace(0, 8, 16)
y1 = 3 + 4*x/8 + np.random.uniform(0.0, 0.5, len(x))
y2 = 1 + 2*x/8 + np.random.uniform(0.0, 0.5, len(x))

# Fill_Between
axes[1,2].title.set_text('Fill_Between')
axes[1,2].fill_between(x, y1, y2, alpha=.5, linewidth=0)
axes[1,2].plot(x, (y1 + y2)/2, linewidth=2)
axes[1,2].set(xlim=(0, 8), xticks=np.arange(1, 8),
              ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```



Referencias:

- Creating multiple subplots using `plt.subplots()`
- Combining two subplots using subplots and GridSpec