Método orientado a objetos

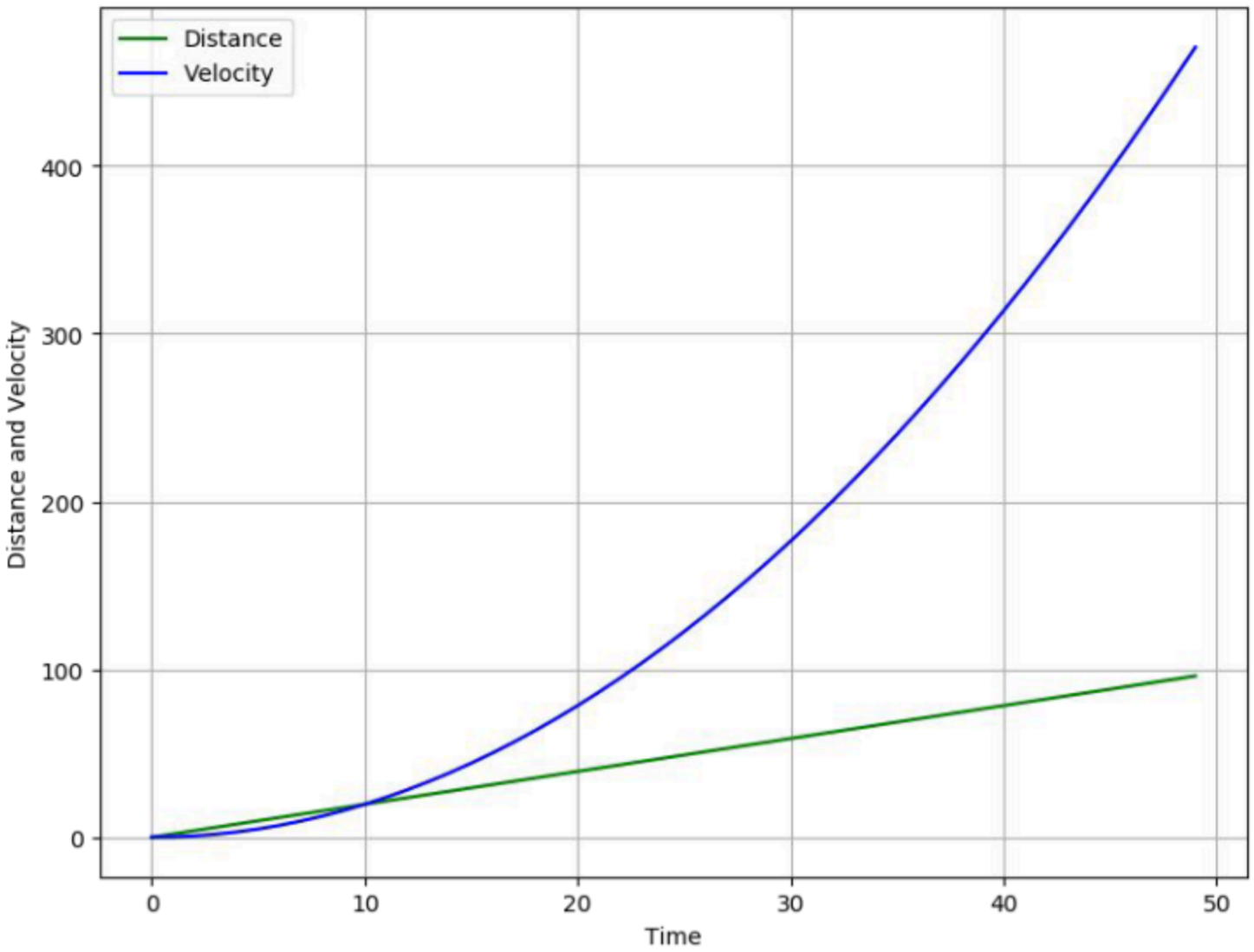
Hay distintas maneras de hacer gráficas dentro de Matplotlib.

- **Pytplot:** Rápido, bastante sencillo y poco código, pero es difícil de personalizar.
- **Orientado a Objetos:** Más difícil de usar y con más código, pero la personalización es mucho más amigable y al detalle. También puedo crear distintos tipos de gráfico y dsitribución en 1 solo

pyplot()

Podemos hacer esto:

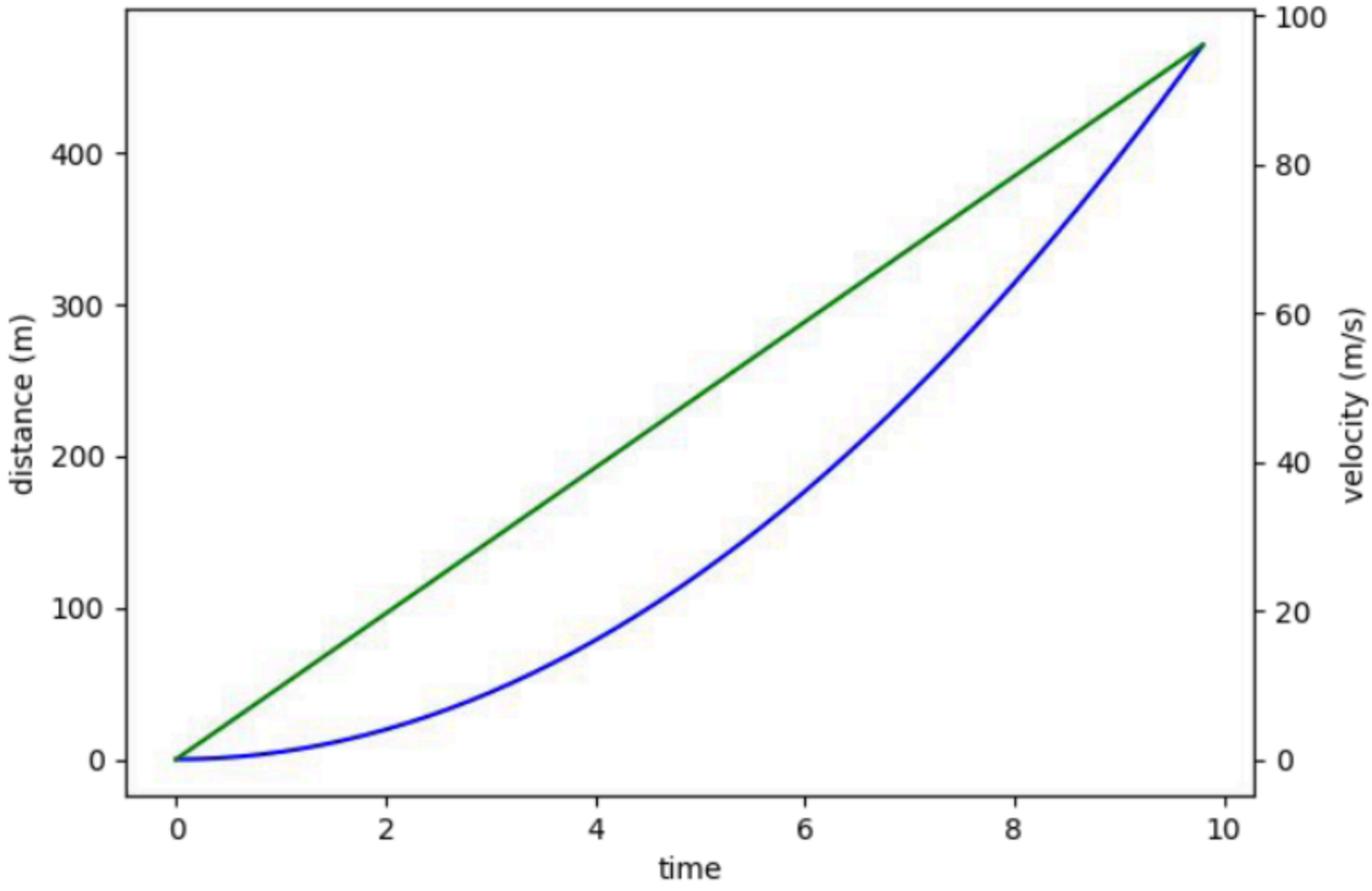
```
plt.figure(figsize=(9,7),dpi=100)
plt.plot(time,velocity,'g-')
plt.plot(time,distance,'b-')
plt.xlabel('Tiempo (s)')
plt.ylabel('Velocidad (m/s) y Distancia (m)')
plt.grid(True)
```



método orientado a objetos

Podemos hacer algo como esto:

```
fig, ax1 = plt.subplots()
ax1.set_ylabel("distance (m)")
ax1.set_xlabel("time")
ax1.plot(time, distance, "blue")
ax2 = ax1.twinx()
ax2.set_ylabel("velocity (m/s)")
ax2.set_xlabel("time")
ax2.plot(time, velocity, "green")
fig.set_size_inches(7,5)
fig.set_dpi(100)
```



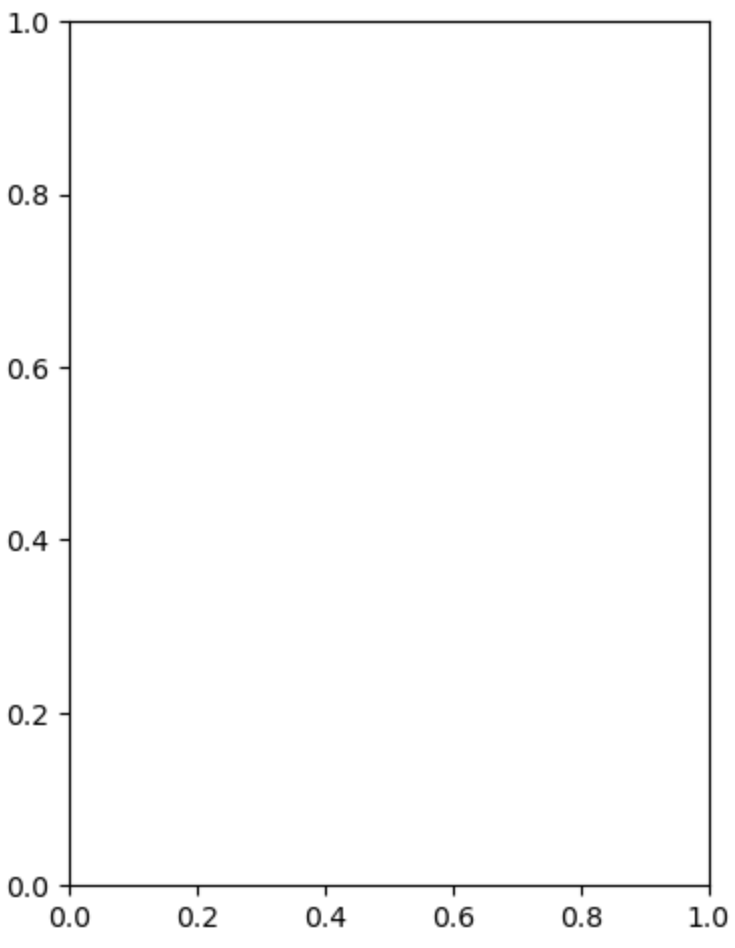
Tiene mayor grado de personalización, al punto de de poder modificar los valor entre los ejes coordenados, pero se requiere de mayor código.

Implementación:

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

In [ ]:
```
x = np.linspace(0,5,11)
y = x**2
```

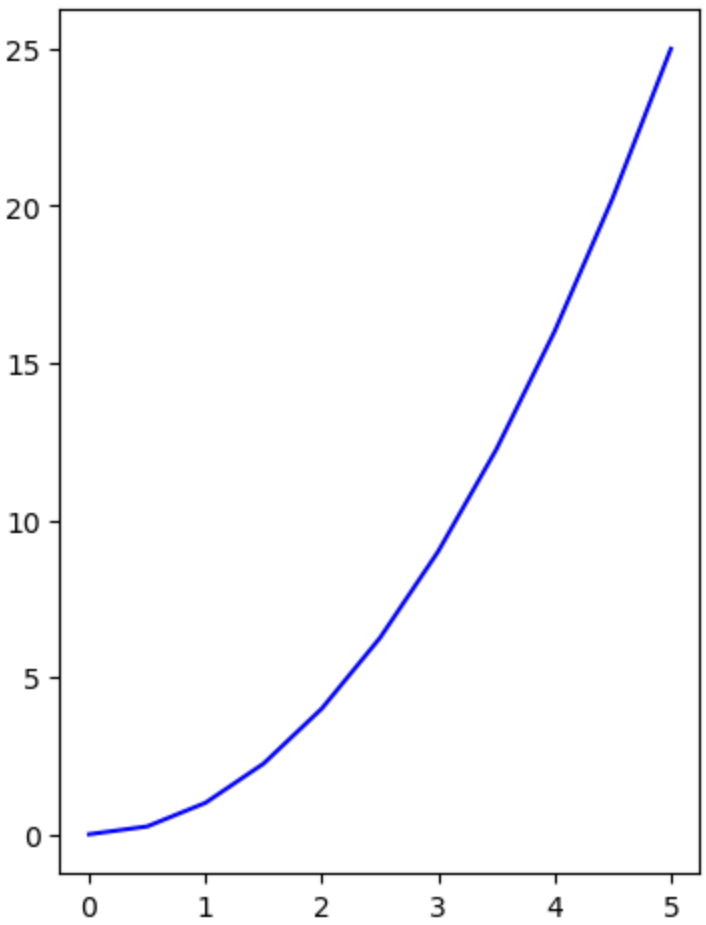
In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo

#Graficas = Axes
#Es igual a mi objeto de tipo figura
axes = fig.add_axes([0.1,0.1,0.5,0.9])
#En nuestro lienzo acabo de crear una figura
```



In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo
```

```
#Graficas = Axes
#Es igual a mi objeto de tipo figura
axes = fig.add_axes([0.1,0.1,0.5,0.9])
#En nuestro lienzo acabo de crear una figura

#Que voy a hacer con el axes - Plot
axes.plot(x,y,'b')
fig.show()
```

```
/tmp/ipykernel_80725/2585781117.py:12: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```



La ventaja es cuando quiero hacer algo con más detalle y personalizado.

In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo

#Graficas = Axes
#Es igual a mi objeto de tipo figura
axes = fig.add_axes([0.1,0.1,0.5,0.9])
#En nuestro lienzo acabo de crear una figura

#Que voy a hacer con el axes - Plot
axes.plot(x,y,'b')
fig.show()
```

Explicación:

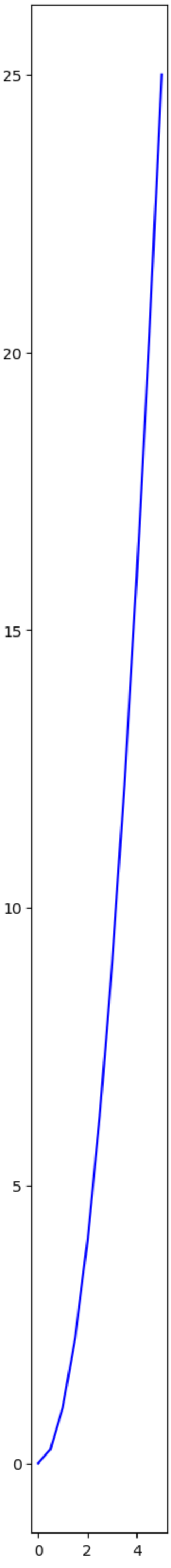
```
axes = fig.add_axes([0.1,0.1,0.5,0.9])
```

- Los 2 primeros parametros `[0.1,0.1]` hacen referencia a la posición de la figura en el lienzo.
- Los 2 siguientes parametros `[0.5,0.9]` hacen referencia al tamaño que van a tener en el lienzo

Veamos un ejemplo

In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo

#Graficas = Axes
#Es igual a mi objeto de tipo figura
axes = fig.add_axes([0.1,0.1,0.2,3])
#En nuestro lienzo acabo de crear una figura

#Que voy a hacer con el axes - Plot
axes.plot(x,y,'b')
fig.show()
```

```
/tmp/ipykernel_80725/4083603943.py:12: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```


25
20
15
10
5
0
0
2
4

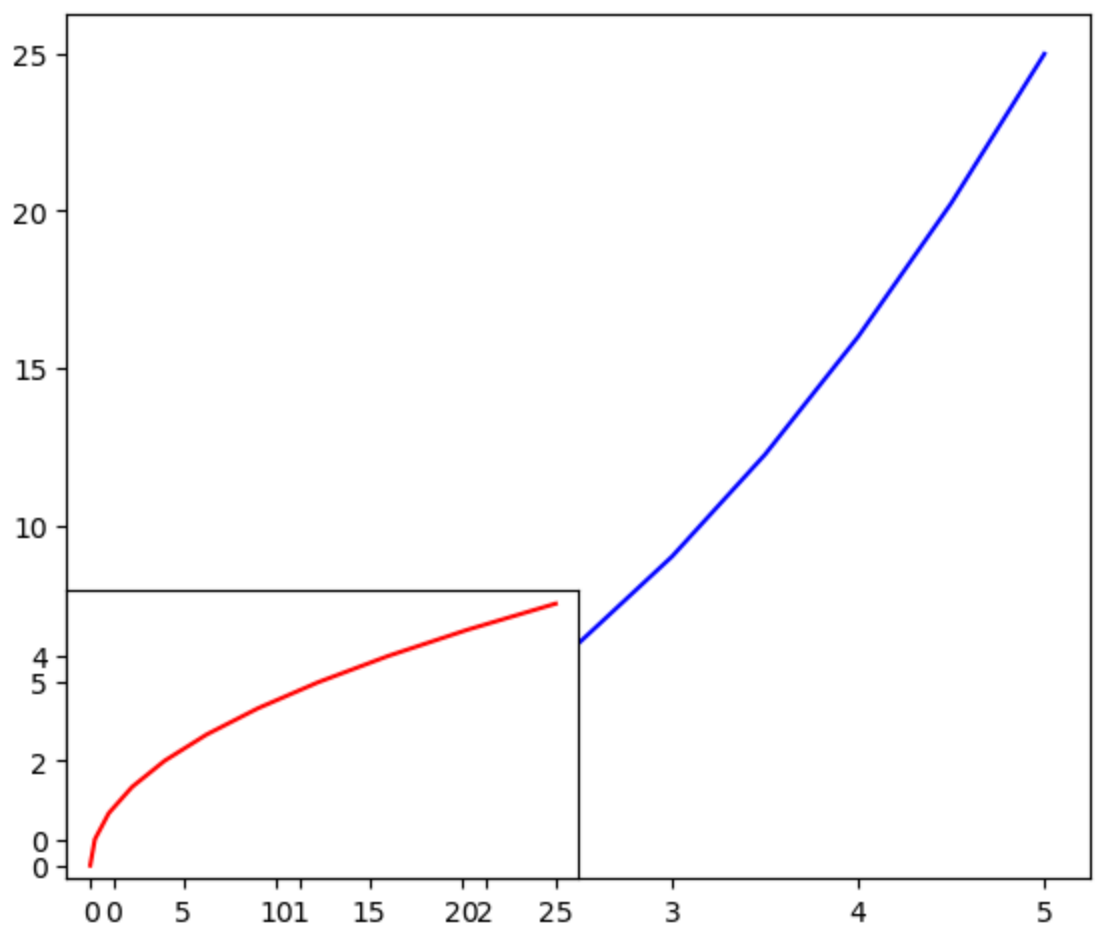
Explicación:

`axes = fig.add_axes([0.1,0.1,0.2,3])`

- Los 2 primeros parametros `[0.1,0.1]` hacen referencia a la posición de la figura en el lienzo.
- Los 2 siguientes parametros `[0.2,3]` hacen referencia al tamaño que van a tener en el lienzo
  - Aquí de ancho lo definí como `0.2`
  - El alto como `3`. El resultado es que hay una figura bastante alta

In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo

#Graficas = Axes
#Es igual a mi objeto de tipo figura
axes = fig.add_axes([0.1,0.1,0.8,0.9])
#En nuestro lienzo acabo de crear una figura

#Ahora voy a especificar que no solo quiero un
#grafico
#2da gráfica
axes2 = fig.add_axes([0.1,0.1,0.4,0.3])

#Que voy a hacer con el axes - Plot
axes.plot(x,y,'b')      #Color blue
axes2.plot(y,x,'r')     #Color red

#Graficando
fig.show()
```

```
/tmp/ipykernel_80725/3286355147.py:20: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```

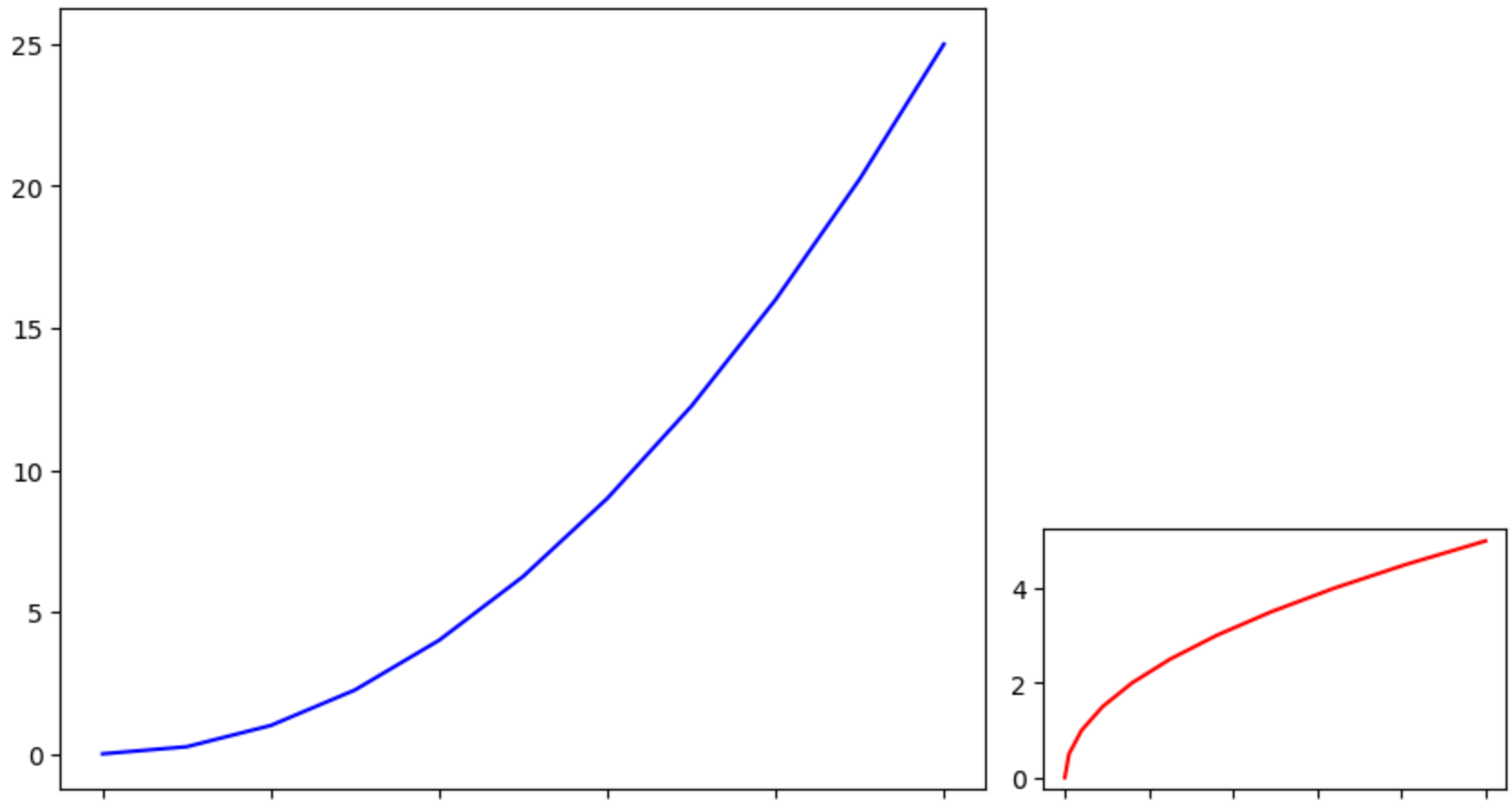


Como se puede observar, las gráficas no salen adecuadamente

In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo

#Graficas = Axes
#Es igual a mi objeto de tipo figura

#1a gráfica GRAFICA AZUL
axes = fig.add_axes([0.1,0.1,0.8,0.9])
#En nuestro lienzo acabo de crear una figura

#Ahora voy a especificar que no solo quiero un
#grafico

#2da gráfica GRAFICA ROJA
axes2 = fig.add_axes([.95,0.1,0.4,0.3])

#Que voy a hacer con el axes - Plot
axes.plot(x,y,'b')      #Color blue
axes2.plot(y,x,'r')     #Color red
```

```
#Graficando
fig.show()
```

```
/tmp/ipykernel_80725/3197660575.py:20: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```

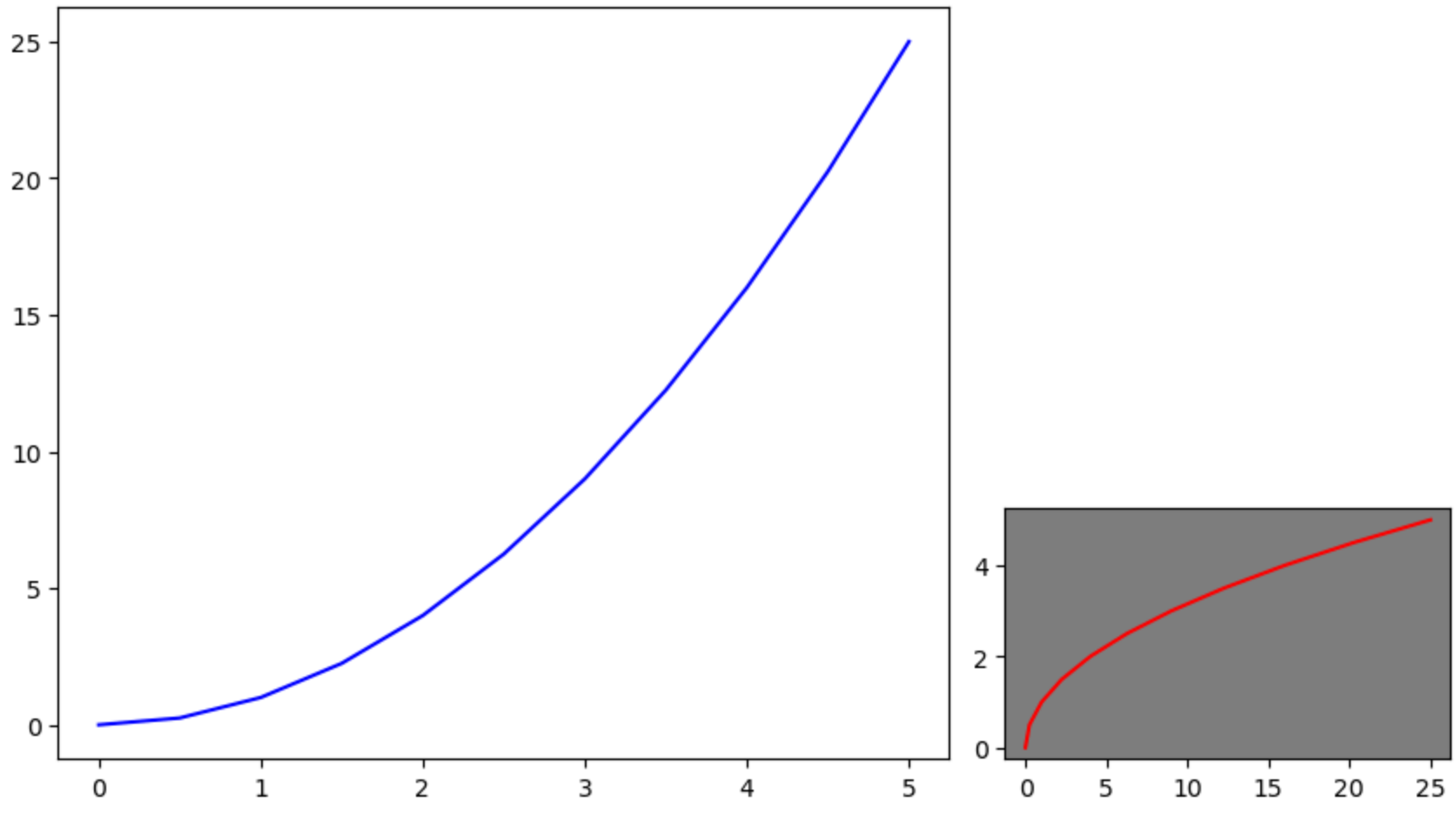


Explicación

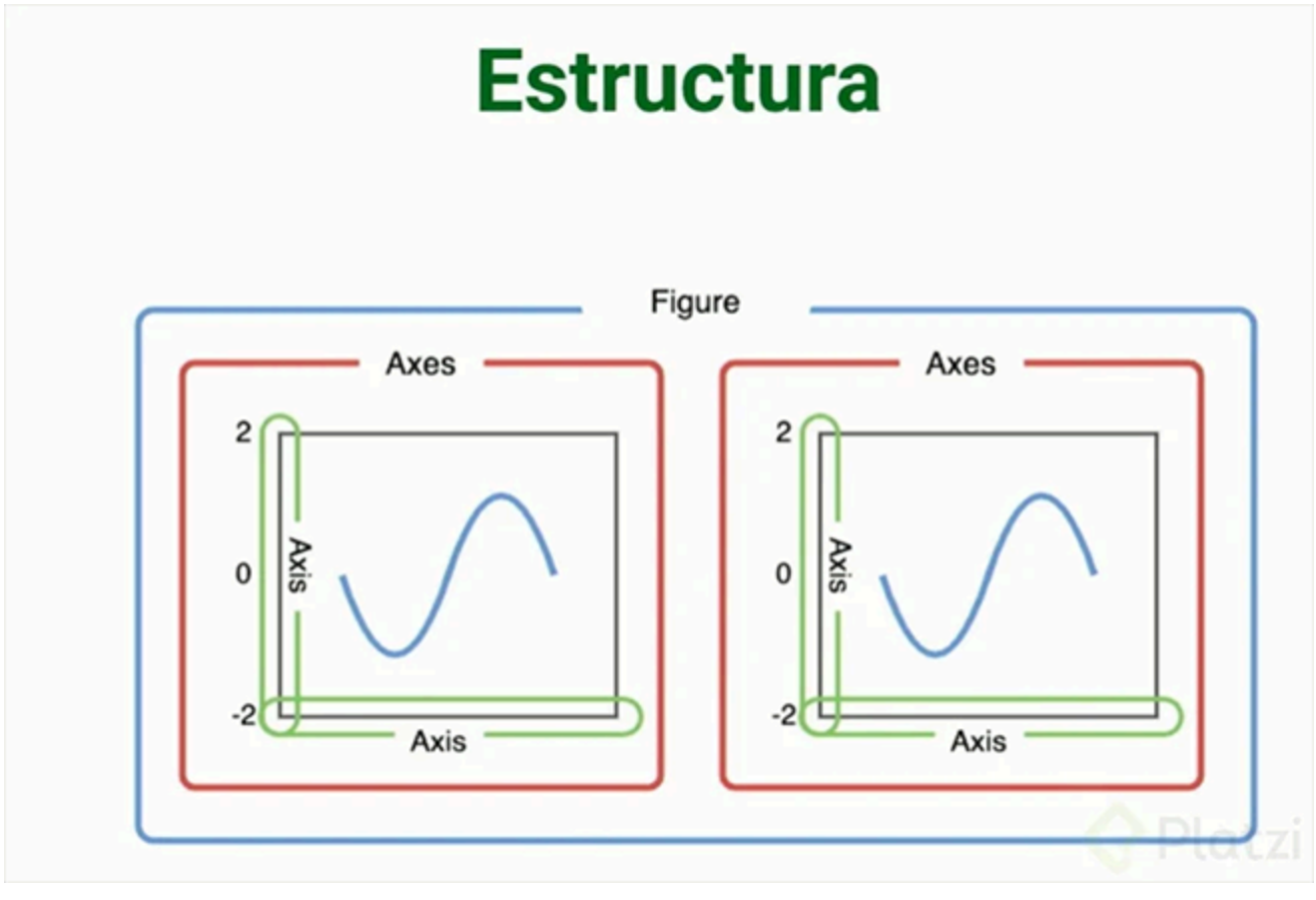
Los 2 son objetos totalmente distintos, solo que reposan sobre un mismo lienzo. Con sus parámetros independientes.

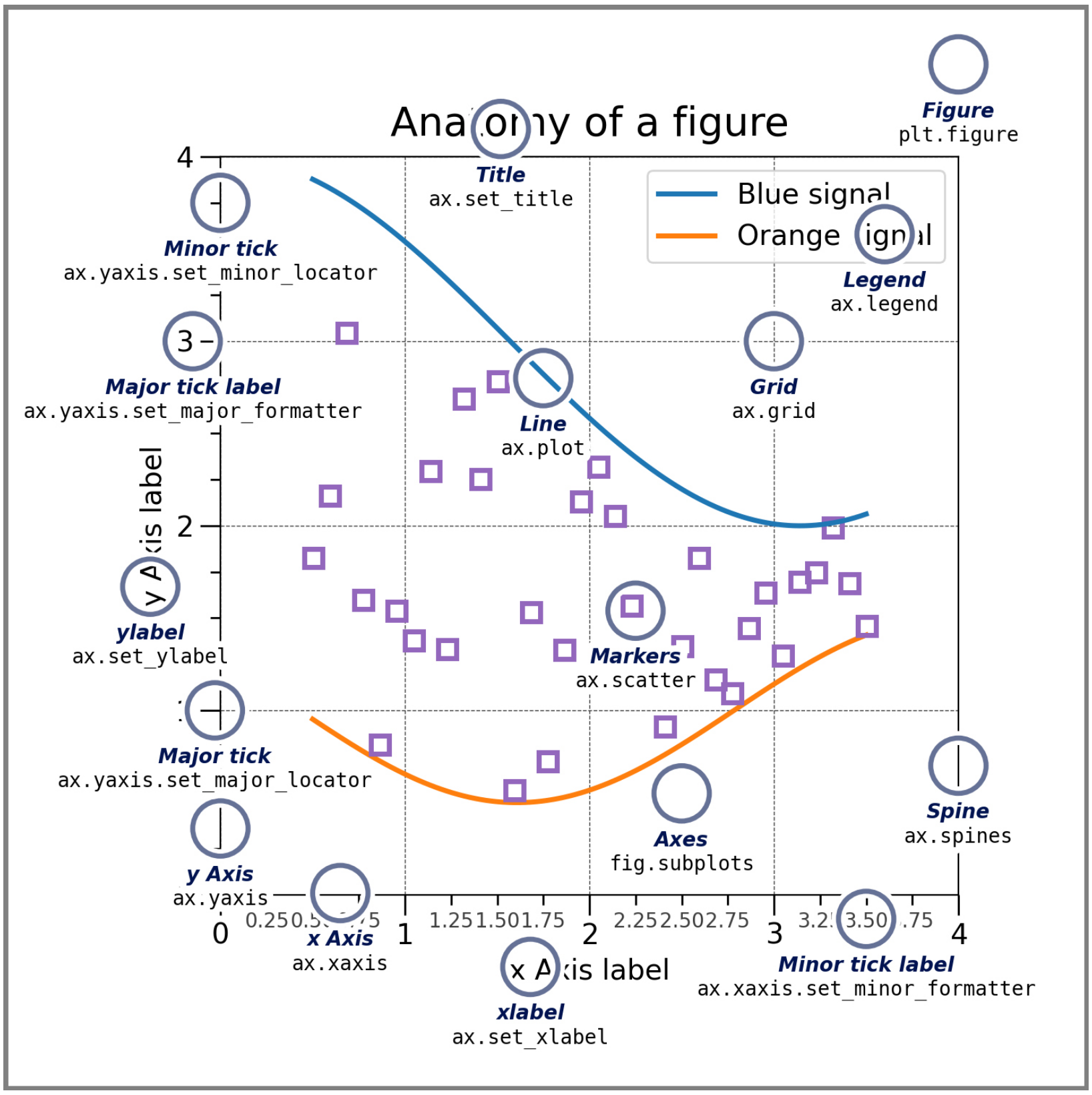
In [ ]:
```
#Orientado a objetos
#Creando FIGURA O lienzo donde van las gráficas
fig = plt.figure()  #Definir lienzo

#Graficas = Axes
#Es igual a mi objeto de tipo figura

#1a gráfica GRAFICA AZUL
axes = fig.add_axes([0.1,0.1,0.8,0.9])
#En nuestro lienzo acabo de crear una figura

#Ahora voy a especificar que no solo quiero un
#grafico

#2da gráfica GRAFICA ROJA
axes2 = fig.add_axes([.95,0.1,0.4,0.3])

#Que voy a hacer con el axes - Plot
axes.plot(x,y,'b')      #Color blue
axes2.plot(y,x,'r')     #Color red
#Alterando parametros de Figura2
axes2.set_facecolor('gray')

#Graficando
fig.show()
```

```
/tmp/ipykernel_80725/2216434554.py:25: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  fig.show()
```



Cómo se puede deducir, el método por objectos es ***Altamente personalizable*** y nos ayuda a ajustar lo que realmente estemos buscando.





Recursos adicionales:

Pyplot vs Object Oriented