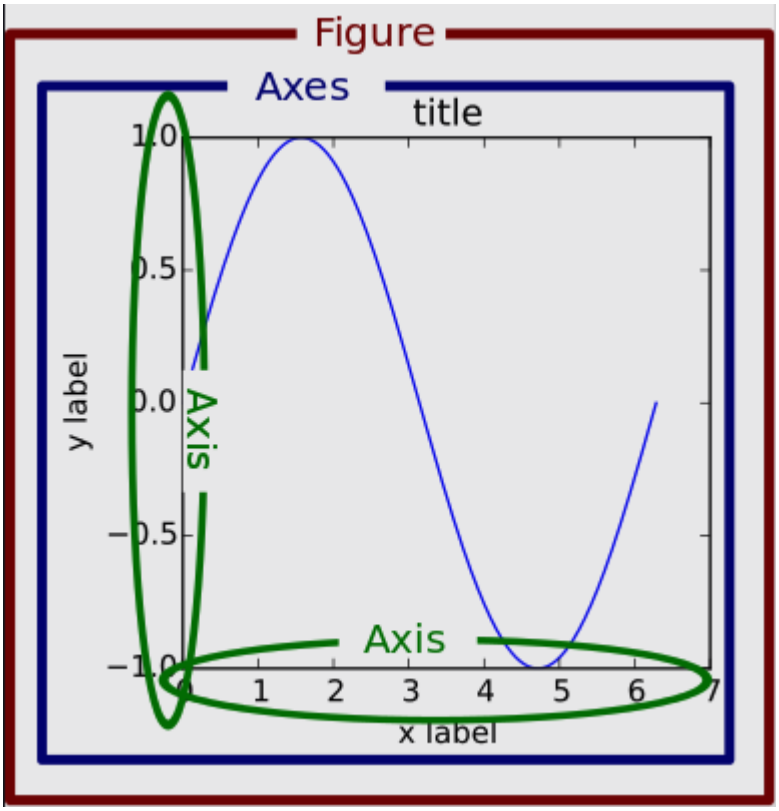
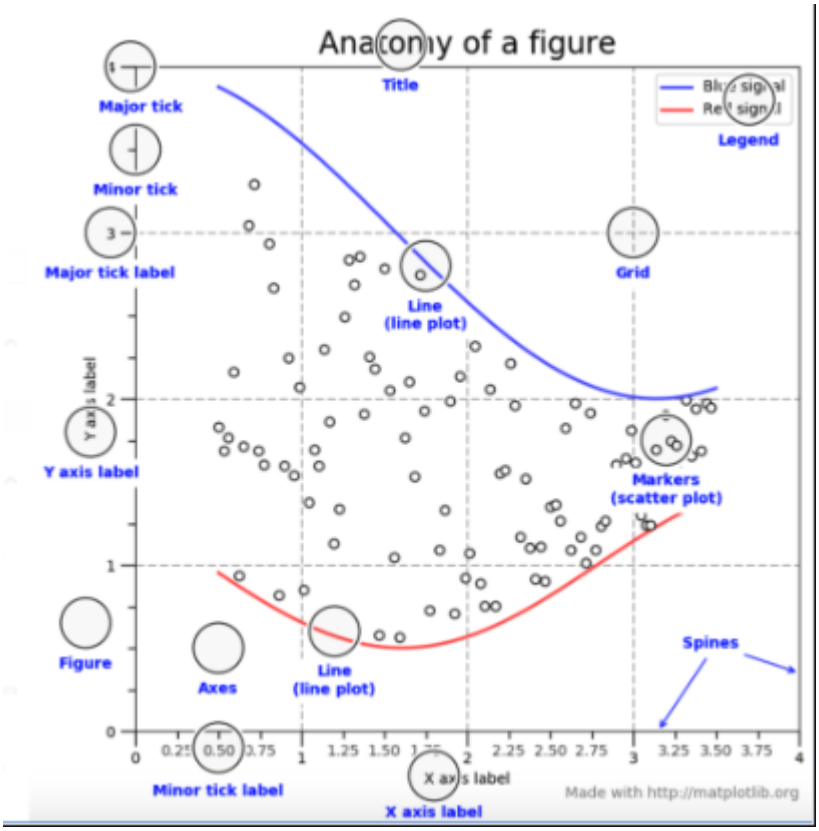
Basics of Matplotlib (Part 1)

Các thành phần cơ bản để vẽ đồ thị của Matplotlib

- Figure
- Axes
- Axis



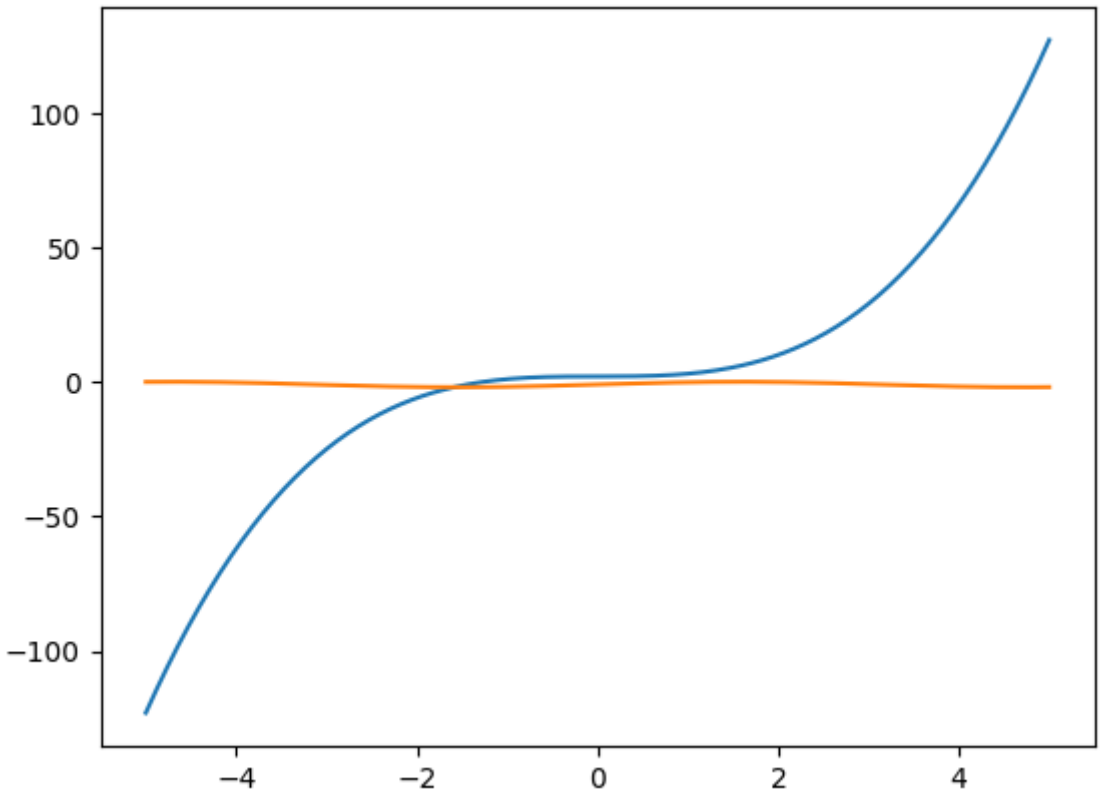
Artist



Vẽ thử hình đầu tiên

In [1]:
```
# Khởi tạo dataset
import numpy as np

points = np.linspace(-5, 5, 256)
y1 = np.power(points, 3) + 2.0
y2 = np.sin(points) - 1.0
```

In [2]:
```
# Tạo một figure để vẽ
import matplotlib.pyplot as plt
fig, axe = plt.subplots()

# Thêm data vào figure đã khởi tạo ở trên
axe.plot(points, y1)
axe.plot(points, y2)

# Hiển thị đồ thị vừa vẽ
plt.show()

# Muốn lưu đồ thị vừa vẽ được
# fig.savefig('graph.png')
# plt.close(fig)
```

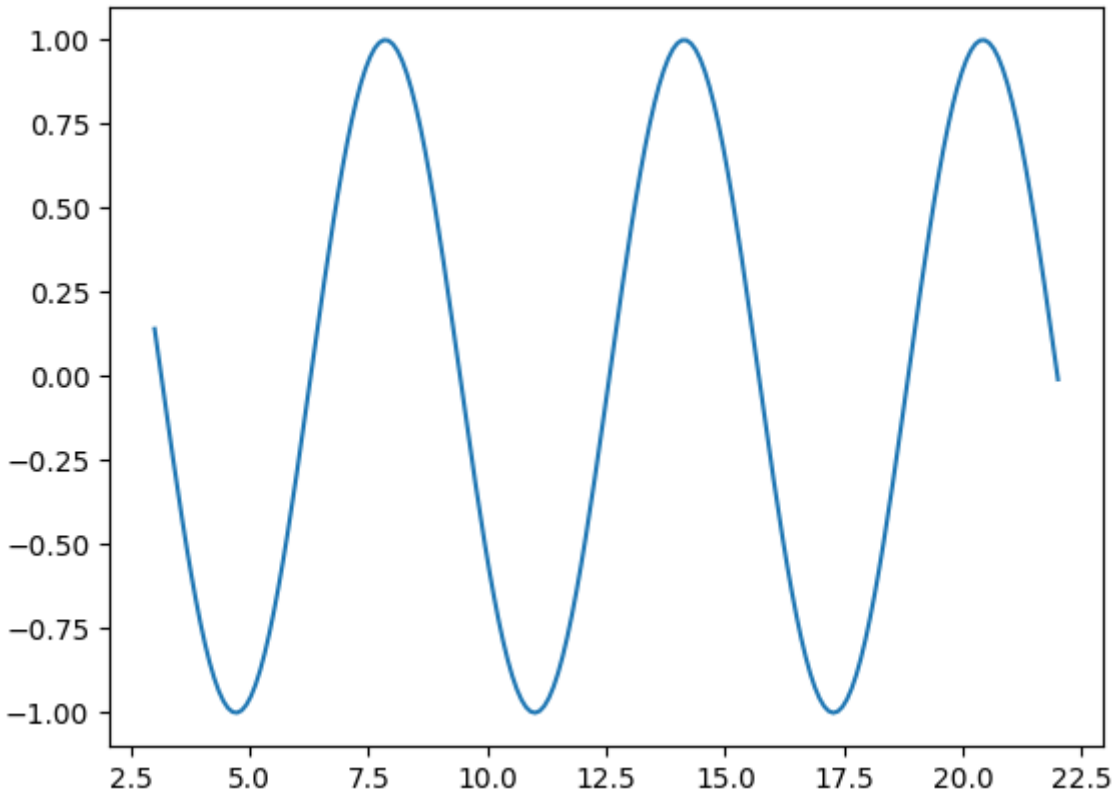


Một số cách khác để tạo figure

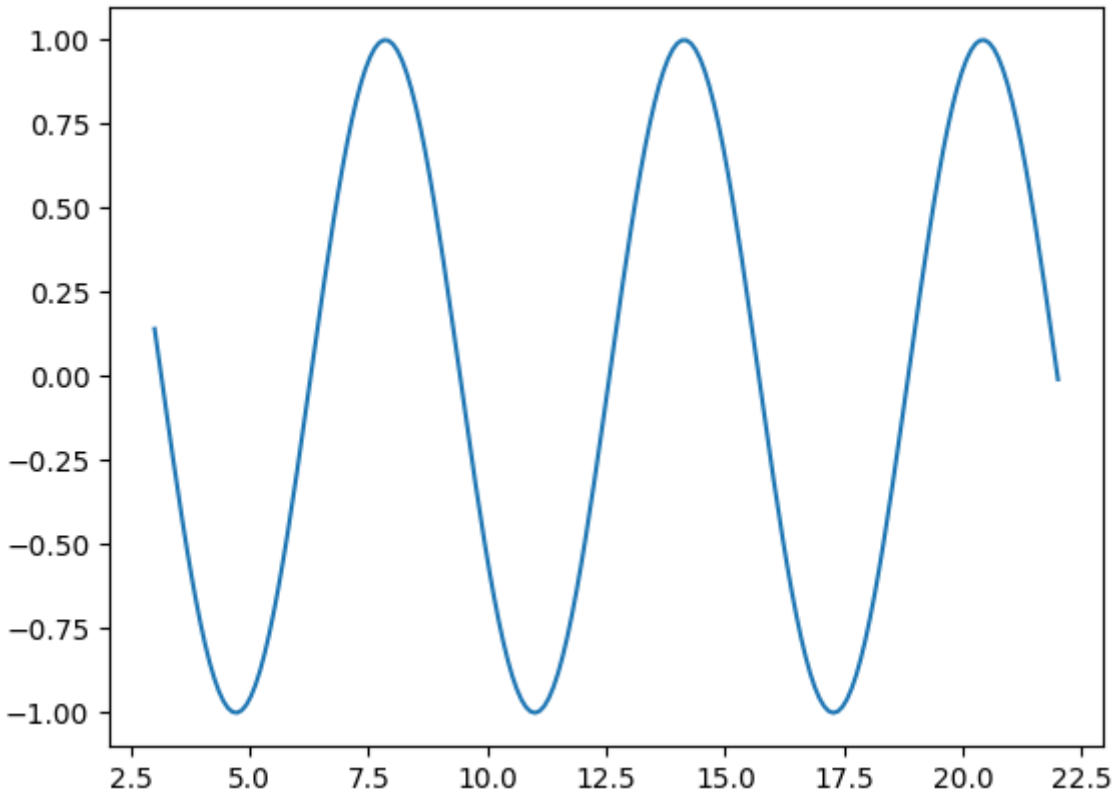
Ví dụ trên sử dụng suplots khởi tạo figure để vẽ đồ thị, đây là cách thông dụng nếu bạn muốn vẽ nhiều đồ thị trên cùng figure. Ngoài ra, có thể dụng một số cách khác như:

- Hàm plot()
- Hàm suplot()

In [3]:
```
# Sử dụng hàm plt.plot()
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(3, 22, 198)
y = np.sin(x)

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



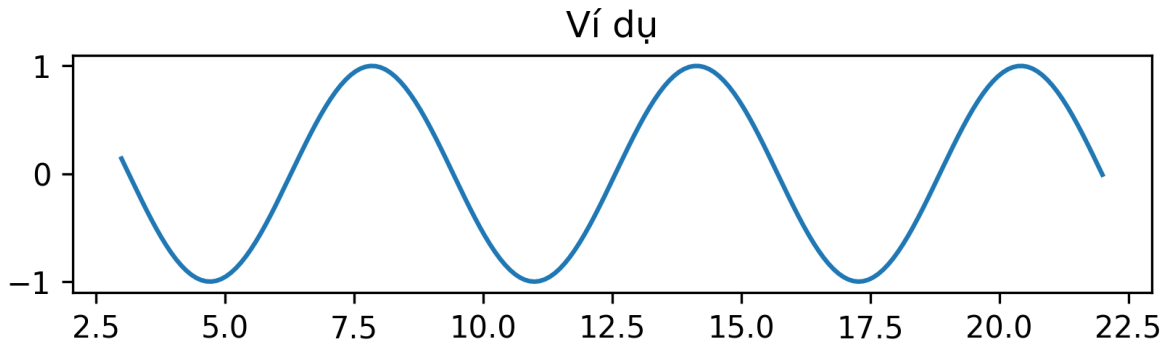
In [4]:
```
# Có thể sử dụng hàm plt.subplot() thay cho plt.subplots()
# khi muốn vẽ một đồ thị duy nhất trên figure

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(3, 22, 198)
y = np.sin(x)

axes = plt.subplot()
axes.plot(x, y)
plt.show()
```



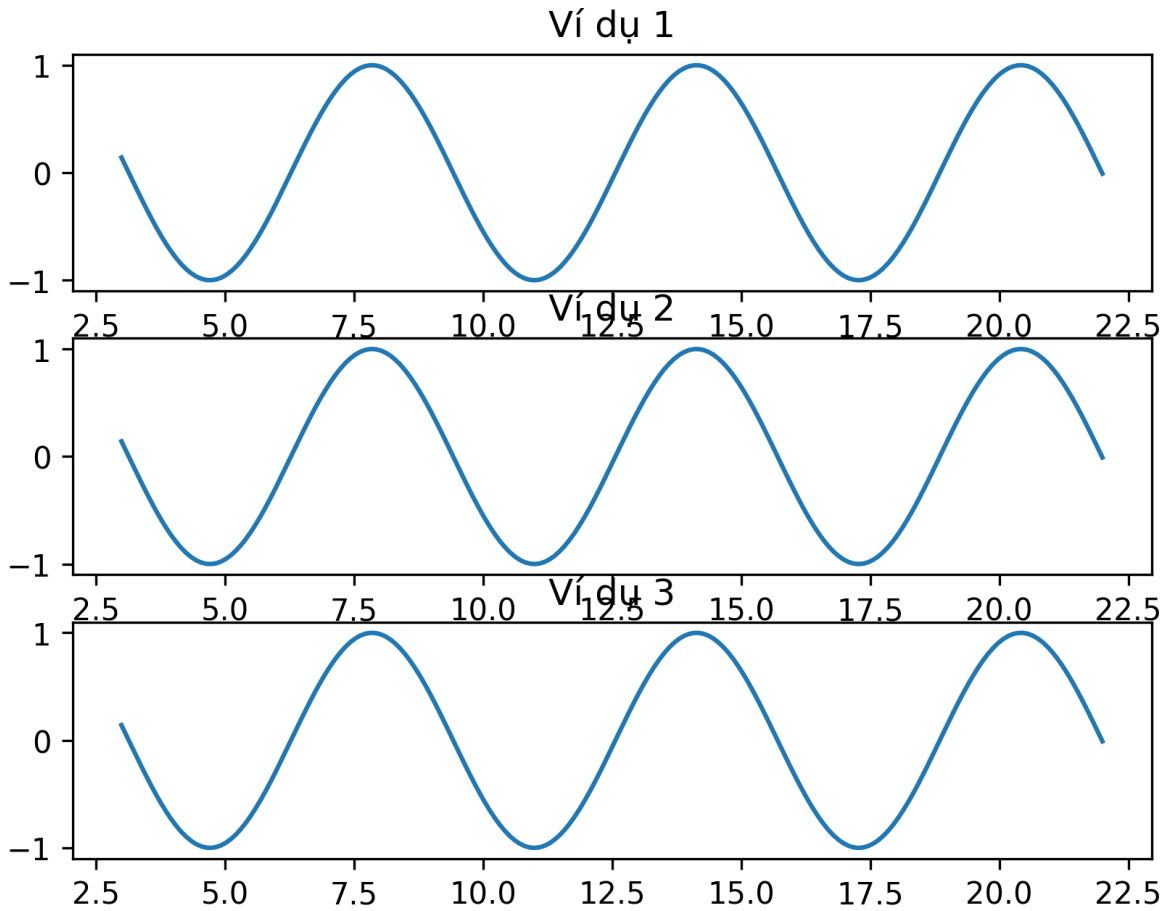
Biểu diễn nhiều đồ thị trên cùng một ảnh

Ngoài cách sử dụng trực tiếp 2 hàm trên, bạn còn có thể thêm các đồ thị vào một figure được khởi tạo trước như sau

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(22, 3, 198)
y = np.sin(x)

# Khởi tạo một figure
fig = plt.figure(dpi=300)  # dpi là độ phân giải của ảnh, cái này không cần

axes = fig.add_subplot(312)

# Lưu ý 111 chính là vị trí của đồ thị con trong ảnh.
# 111 nghĩa là ảnh chỉ có 1 đồ thị duy nhất
# Ví dụ:
# fig.add_subplot(312): Ảnh được chia thành 3 hàng, 1 cột
# Đồ thị của bạn nằm ở hàng 2
# .set_title("") đặt trên cho đồ thị con

axes.plot(x, y)
axes.set_title("Ví dụ")
plt.show()
```

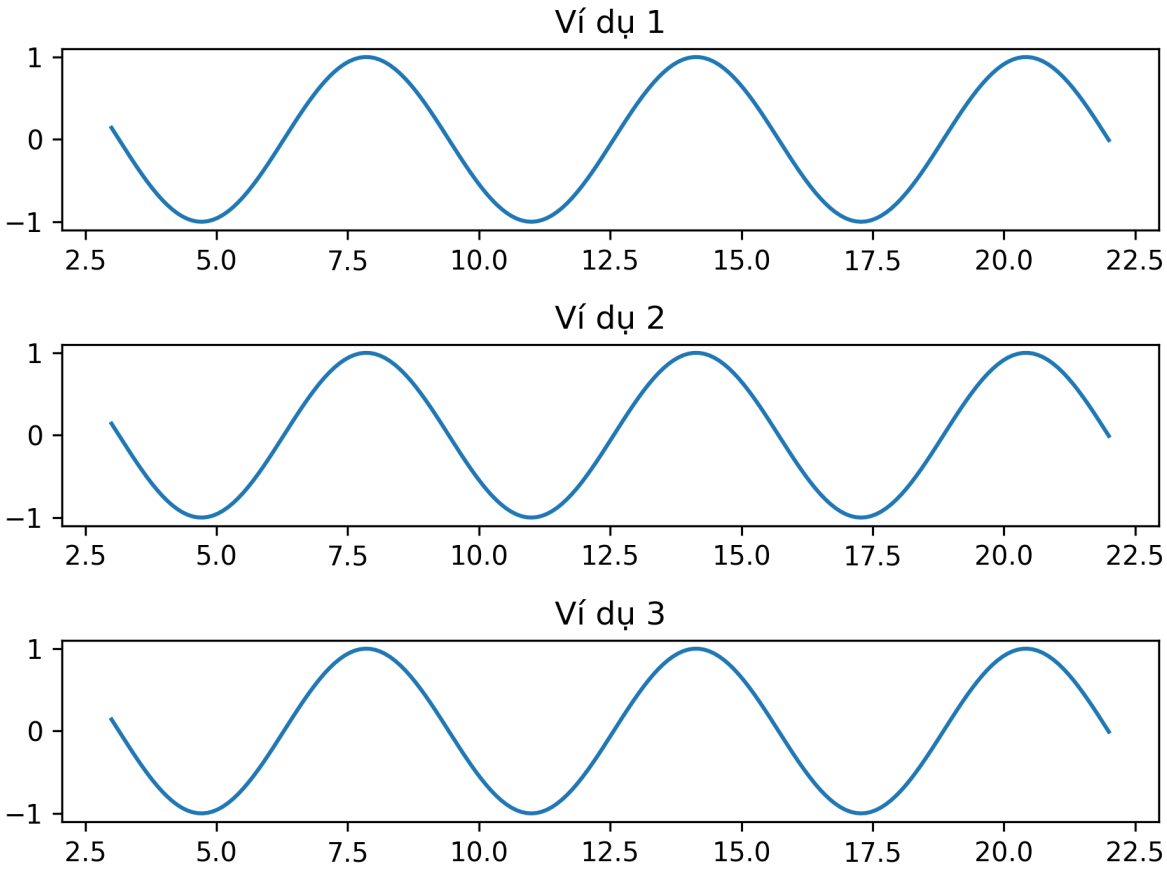


In [6]:
```
# Thử lại ví dụ trên nhưng ta vẽ 3 đồ thị trong cùng 1 figure

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(22, 3, 198)
y = np.sin(x)

# Khởi tạo một figure
fig = plt.figure(dpi=300)  # dpi là độ phân giải của ảnh, cái này không cần

axes = fig.add_subplot(312)
axes.plot(x, y)
axes.set_title("Ví dụ 2")

axes = fig.add_subplot(311)
axes.plot(x, y)
axes.set_title("Ví dụ 1")

axes = fig.add_subplot(313)
axes.plot(x, y)
axes.set_title("Ví dụ 3")

plt.show()
```



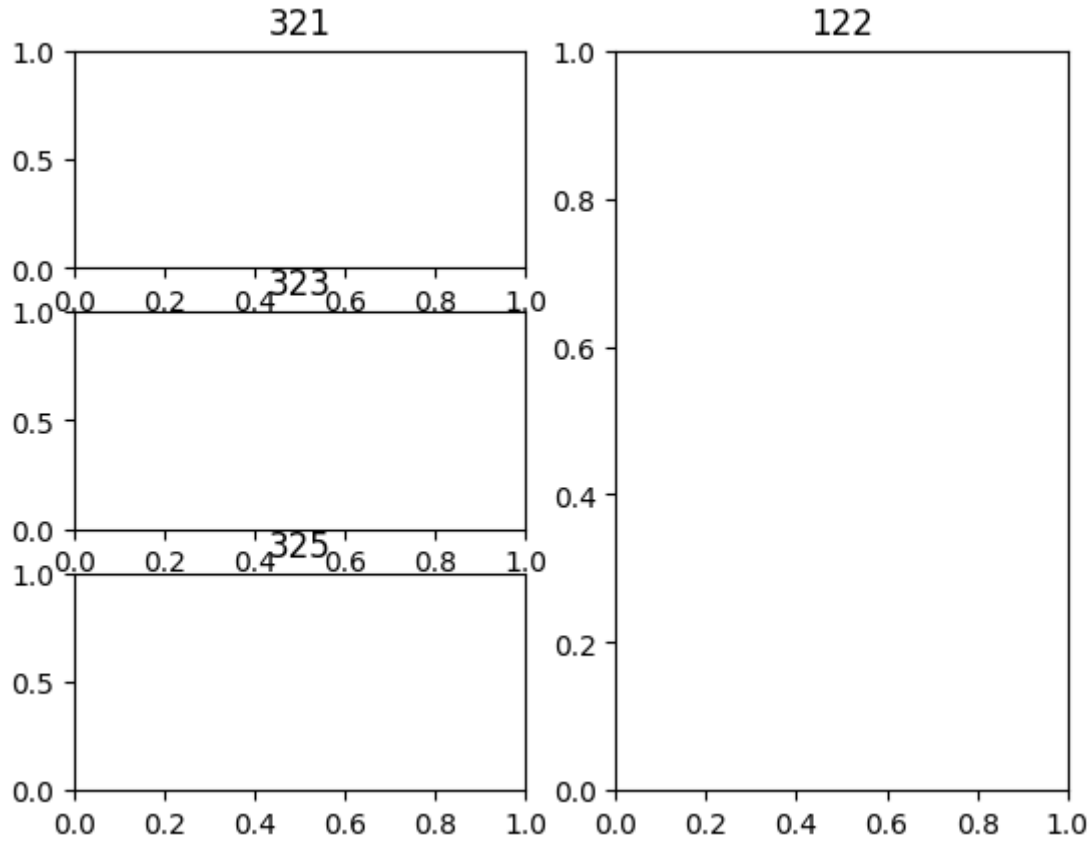
Nhìn hình trên có vẻ hơi xấu do các biểu đồ bị chồng lên nhau, rất khó nhìn. Hàm plt.tight_layout() sẽ căn chỉnh lại các biểu đồ trong cùng một ảnh như sau:

In [7]:
```
# Thử lại ví dụ trên nhưng ta vẽ 3 đồ thị trong cùng 1 figure

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(22, 3, 198)
y = np.sin(x)

# Khởi tạo một figure
fig = plt.figure(dpi=300)  # dpi là độ phân giải của ảnh, cái này không cần

axes = fig.add_subplot(312)
axes.plot(x, y)
axes.set_title("Ví dụ 2")

axes = fig.add_subplot(311)
axes.plot(x, y)
axes.set_title("Ví dụ 1")

axes = fig.add_subplot(313)
axes.plot(x, y)
axes.set_title("Ví dụ 3")

plt.tight_layout()
plt.show()
```



Ngoài ra, trên cùng một ảnh có thể tùy chỉnh bố cục của các đồ thị như sau

In [8]:
```
import matplotlib.pyplot as plt

# Khởi tạo figure
fig = plt.figure()

# Thêm subplots
fig.add_subplot(321).set_title('321')
fig.add_subplot(323).set_title('323')
fig.add_subplot(325).set_title('325')
fig.add_subplot(122).set_title('122')

plt.show()
```



Như ví dụ trên, ta thêm 3 đồ thị có kích thước 3x2 nằm ở vị trí 1, 3, 5. Một đồ thị khác có kích thước 1x2 nằm ở vị trí thứ 2 theo chiều ngang

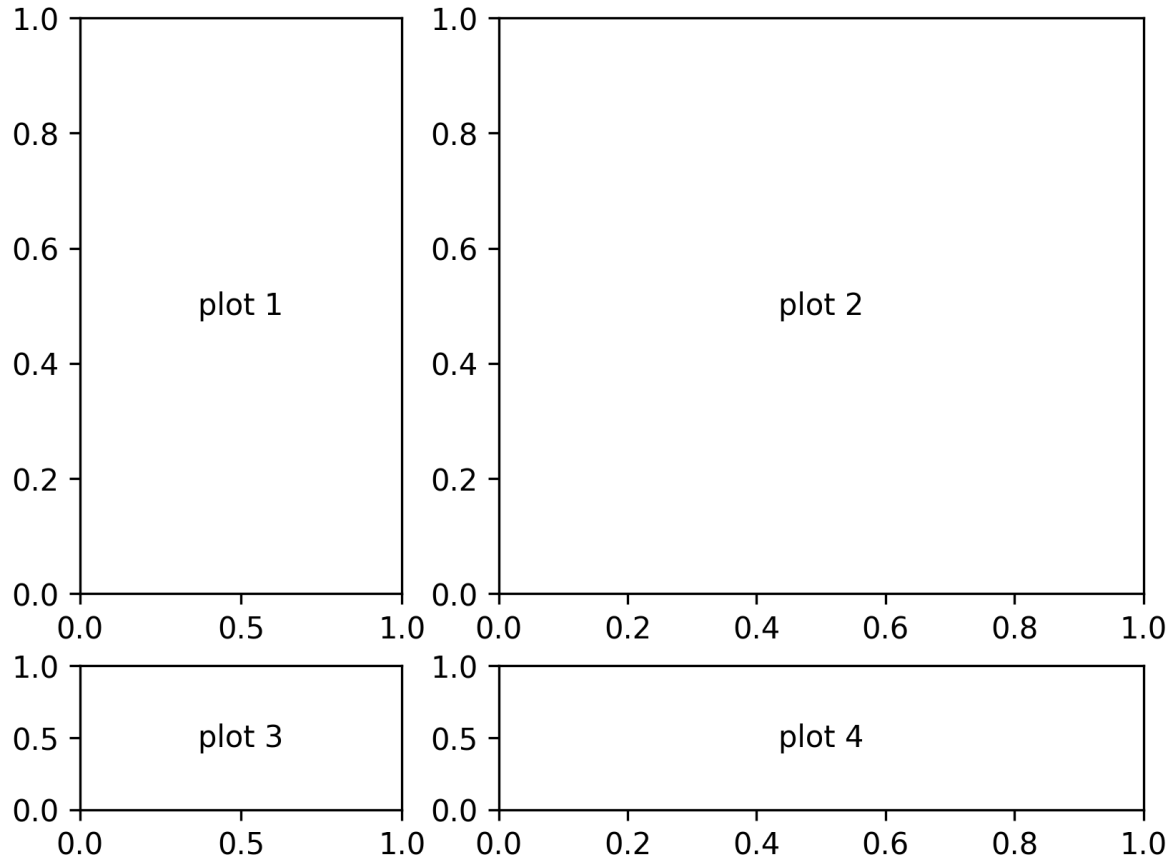
Tùy chỉnh các biểu đồ trong cùng figure

Trong các ví dụ trên, biểu đồ có kích thước bằng nhau. Ta có thể chỉnh sửa kích thước của từng biểu đồ trong figure theo ý muốn bằng các sử dụng các hàm trong class GridSpec mà thư viện cung cấp. GridSpec sẽ tạo một lưới để có thể chỉnh sửa kích thước của mỗi biểu đồ. Grid khởi tạo gồm các thông số:

- nrows: số hàng của lưới
- ncols: số cột của lưới
- width_ratio: độ rộng của mỗi cột, nếu không được thêm vào thì các cột sẽ có cùng độ rộng
- height_ratio: chiều cao của mỗi hàng

In [9]:
```
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec

fig = plt.figure(dpi=300)

# Khởi tạo gồm 2 hàng, 2 cột
gs = GridSpec(2, 2, width_ratios=[1, 2], height_ratios=[4, 1])

ax1 = fig.add_subplot(gs[0])
ax1.text(0.5, 0.5,"plot 1", verticalalignment='center', ha='center')

ax2 = fig.add_subplot(gs[1])
ax2.text(0.5, 0.5,"plot 2", verticalalignment='center', ha='center')
```

```
ax3 = fig.add_subplot(gs[2])
ax3.text(0.5, 0.5,"plot 3", verticalalignment='center', ha='center')

ax4 = fig.add_subplot(gs[3])
ax4.text(0.5, 0.5,"plot 4", verticalalignment='center', ha='center')

plt.show()
```



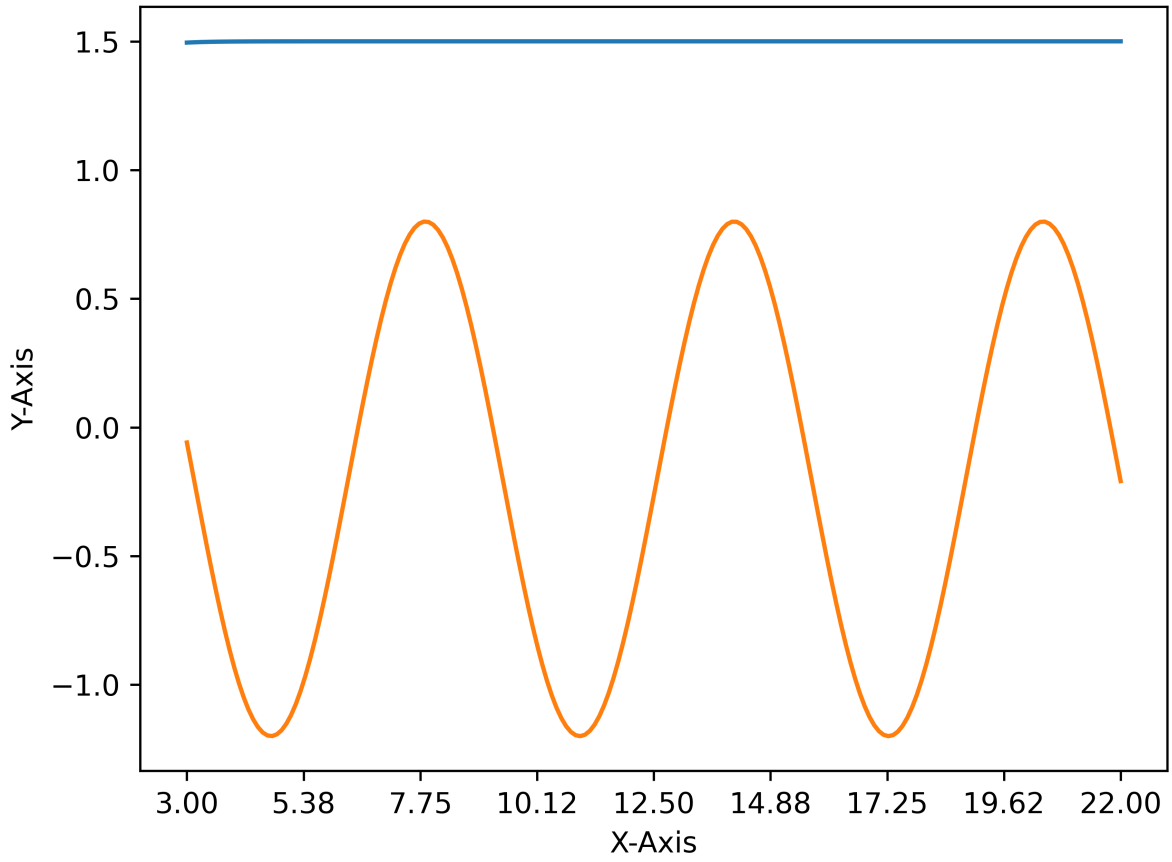
Tick

Tick là một khái niệm trong matplotlib dùng để đánh dấu các điểm trên trục tọa độ, có thể là số, chuỗi hoặc các kí tự khác. Matplolib cung cấp các tùy chọn mặc định trong các hàm vẽ, tuy nhiên ta có thể tùy chỉnh lại các thuộc tính theo yêu cầu:

- which: loại chỉ số áp dụng (major, minor, both), hiểu đơn giản thì có major là các mốc lớn như 0, 5, 10, 15,... minor là các mốc nhỏ hơn trong mỗi khoảng major
- color: màu chỉ số
- labelrotation: xoay chiều hiển thị tick
- width: độ rộng tick
- length: độ dài tick
- direction: hướng của các tick trên đồ thị

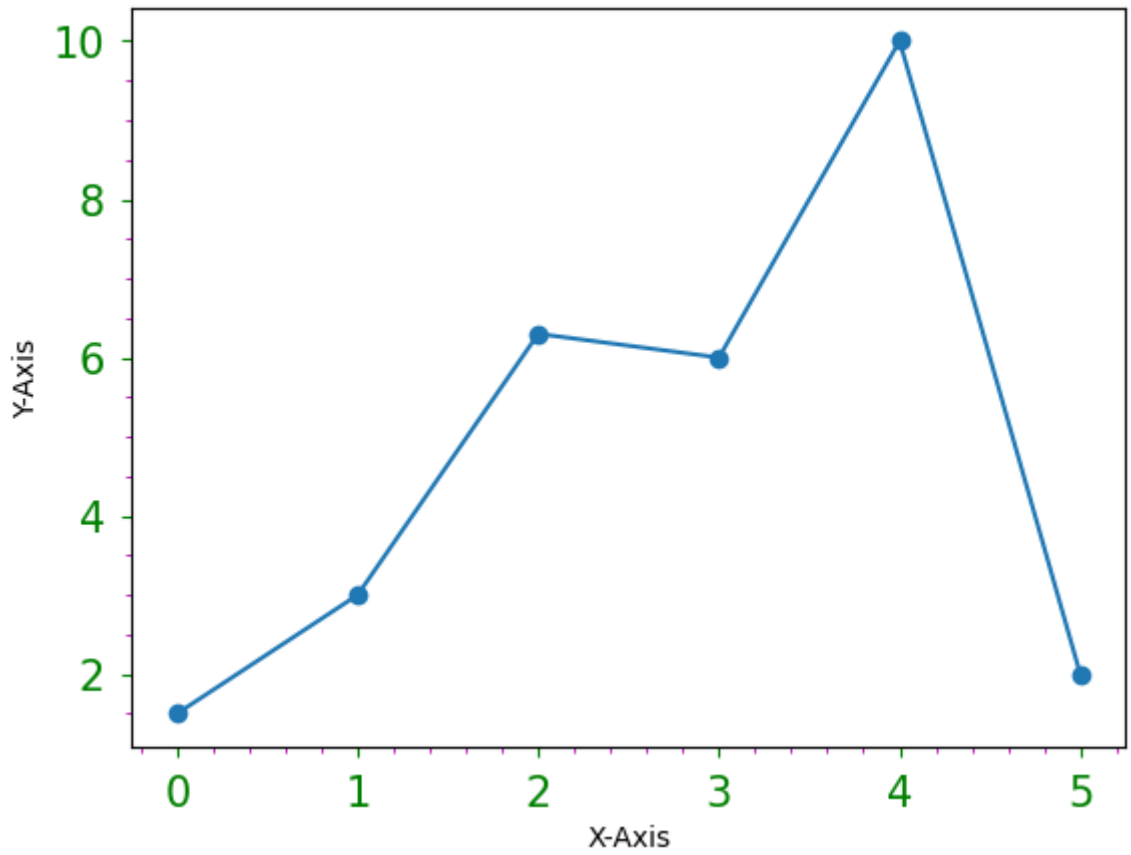
In [10]:
```
import numpy as np
import matplotlib.pyplot as plt

# Khởi tạo 2 hàm số
points = np.linspace(3, 22, 198)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2
```

```
fig, axes = plt.subplots(dpi=600)
axes.plot(points, y1)
axes.plot(points, y2)
# Tùy chỉnh chỉ số trên trục x
axes.set_xticks(np.linspace(3, 22, 9))

# Thêm label
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')

plt.show()
```



In [11]:
```
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
x = [0, 1, 2, 3, 4, 5]
y = [1.5, 3, 6.3, 6, 10, 2]

fig, axes = plt.subplots()
plt.plot(x, y, '-o')

# Tùy chỉnh các chỉ số loại minor
# Matplotlib cung cấp các hàm để tùy chỉnh chỉ số theo nhu cầu,
# Ở đây mình dùng AutoMinorLocator() để cấu hình cho chỉ số minor
# trong trường hợp các chỉ số là tuyến tính và chỉ số chính major cách đều n
# AutoMinorLocator() sẽ chia khoảng giữa 2 major thành 4 hoặc 5 khoảng
axes.xaxis.set_minor_locator(AutoMinorLocator())
axes.yaxis.set_minor_locator(AutoMinorLocator())

# Để chi tiết hơn, bạn có thể dùng các hàm formatter để hiển thị chỉ số
# https://matplotlib.org/3.1.1/gallery/ticks_and_spines/tick-formatters.html

# Thêm tên cho trục
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
```

```
# Tùy chỉnh các thông số khác cho chỉ số
# List các màu sắc của matplotlib để lựa chọn ở đây:
# https://matplotlib.org/stable/gallery/color/named_colors.html
plt.tick_params(axis ='both', which ='major',
                labelsize = 15, pad = 6,
                colors ='g')
plt.tick_params(axis ='both', which ='minor',
                labelsize = 6, colors ='m')

plt.show()
```

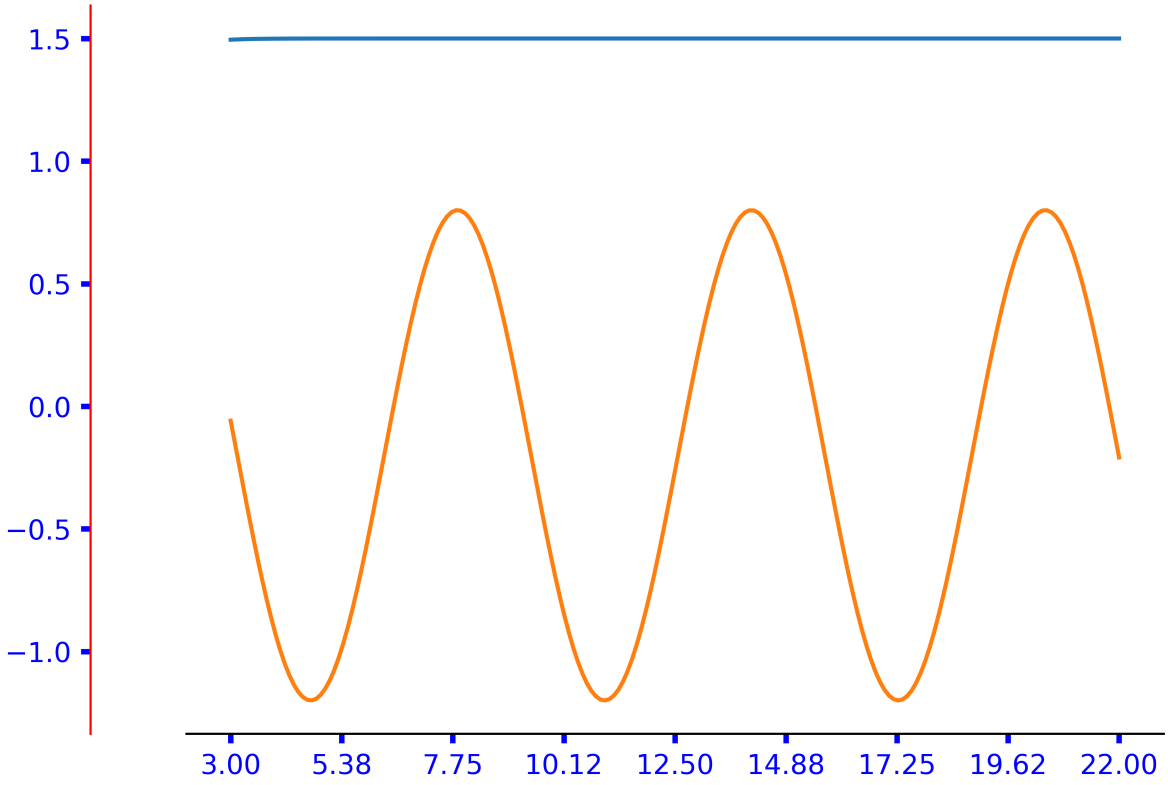


Spine

Spine trong matplotlib dùng để tùy chỉnh các yêu cầu liên quan đến trục tọa độ như vị trí, màu sắc,... Một ví dụ đơn giản

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt

points = np.linspace(3, 22, 198)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2

fig, axes = plt.subplots(dpi=600)
axes.plot(points, y1)
axes.plot(points, y2)
axes.set_xticks(np.linspace(3, 22, 9))
axes.tick_params(width=2, colors='b')

axes.spines['right'].set_color('none')
axes.spines['top'].set_color('none')
```

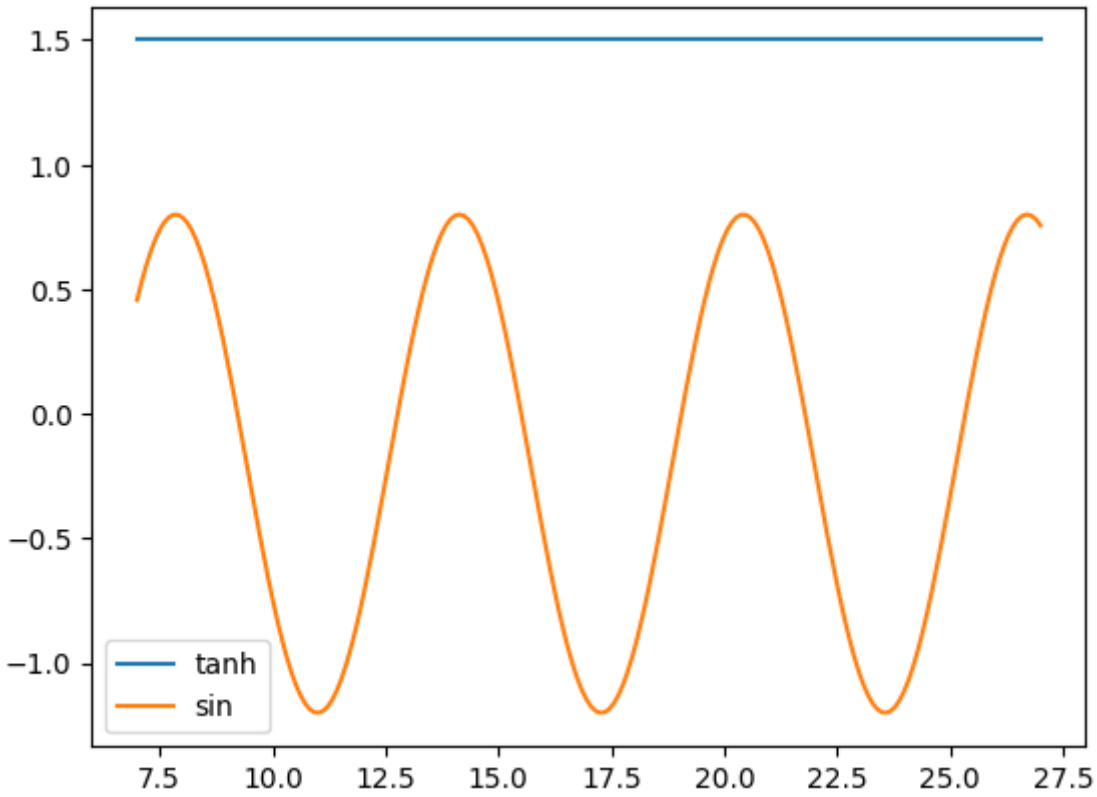
```
axes.spines['left'].set_color('red')
axes.spines['left'].set_position(('data', 0))  # Vị trí biên trái

plt.show()
```



Legend

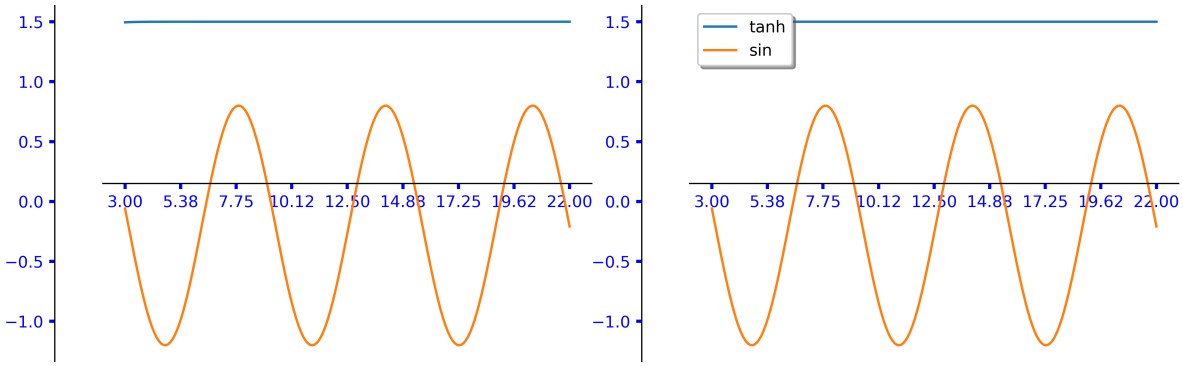
Legend cung cấp chú thích các thành phần trong đồ thị

In [13]:
```
# Thêm chú thích trong đồ thị
import numpy as np
import matplotlib.pyplot as plt

points = np.linspace(7, 27, 200)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2

fig, axes = plt.subplots(dpi=100)

# Cách 1
axes.plot(points, y1, label="tanh")
axes.plot(points, y2, label="sin")
axes.legend()

# Cách 2
# axes.plot(points, y1)
# axes.plot(points, y2)
# axes.legend(["tanh", "sin"])

plt.show()
```



Tổng hợp lại các chức năng đã note ở trên

In [14]:
```
import numpy as np
import matplotlib.pyplot as plt

points = np.linspace(3, 22, 198)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2

fig, axe = plt.subplots(nrows=1, ncols=2, dpi=500, figsize=(12, 4))
axe[0].plot(points, y1)
axe[0].plot(points, y2)
axe[0].set_xticks(np.linspace(3, 22, 9))
axe[0].tick_params(width=2, colors='b')
axe[0].spines['right'].set_color('none')
axe[0].spines['top'].set_color('none')
axe[0].spines['left'].set_position(('data', 0))
axe[0].spines['bottom'].set_position(('axes', 0.5))

axe[1].plot(points, y1, label="tanh")
axe[1].plot(points, y2, label="sin")
axe[1].set_xticks(np.linspace(3, 22, 9))
axe[1].tick_params(width=2, colors='b')
axe[1].spines['right'].set_color('none')
axe[1].spines['top'].set_color('none')
axe[1].spines['left'].set_position(('data', 0))
axe[1].spines['bottom'].set_position(('axes', 0.5))
# Tạo chú thích cho biểu đồ 1
# Thêm hiệu ứng và vị trí
axe[1].legend(loc="upper left", shadow=True)

plt.show()
```



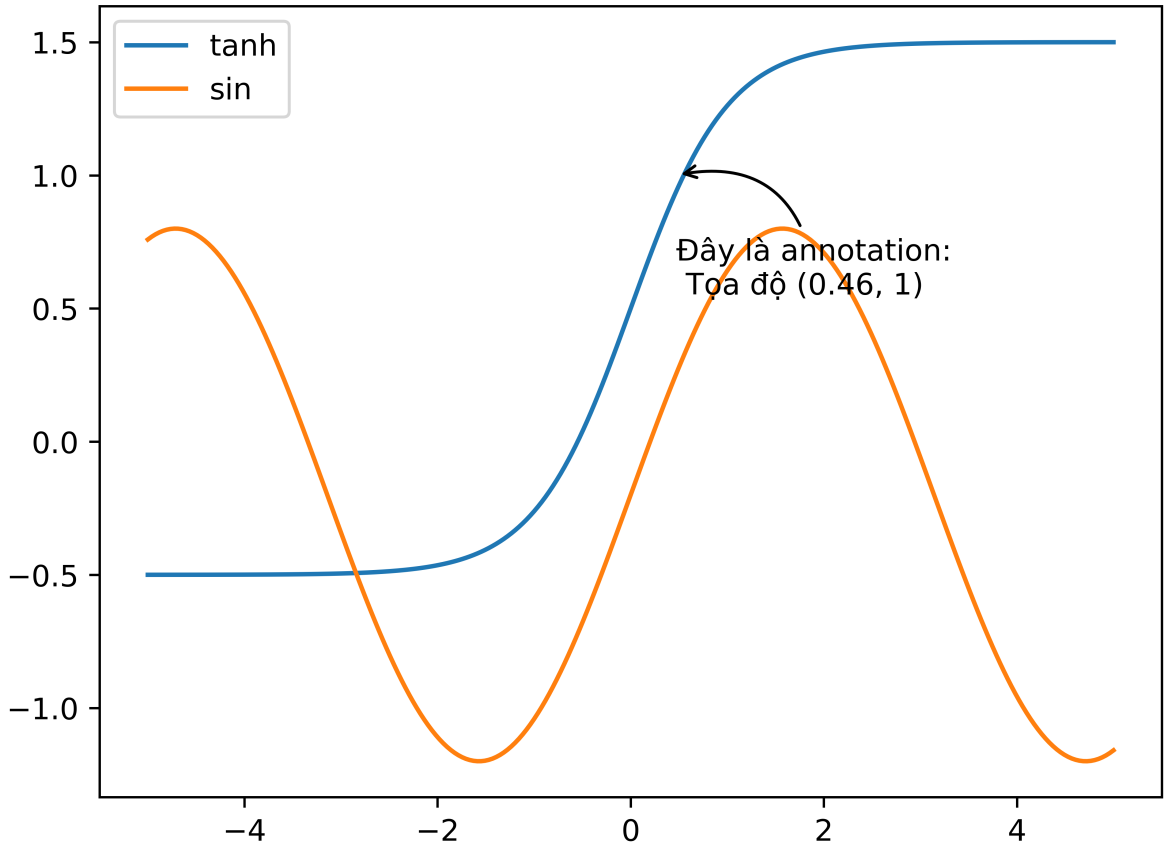
Annotation

Annotation dùng để thêm các chú thích chi tiết trên biểu đồ Matplotlib cung cấp hàm annotate() để thêm chú thích chi tiết cho các thành phần trong biểu đồ. Các tham số trong hàm này:

- xy (tuple): vị trí của điểm cần chú thích
- xytext: vị trí của text chú thích xuất hiện (nếu không có thì mặc định sử dụng giá trị xy
- xycoords: hệ tọa độ xy sử dụng (mặc định sử dụng hệ tọa độ tương ứng với data truyền vào)
- textcoords: hệ tọa độ xytext sử dụng
- arrowprops: thuộc tính sử dụng để vẽ đường liên kết giữ xy và xytext

In [15]:
```
import numpy as np
import matplotlib.pyplot as plt

points = np.linspace(-5, 5, 256)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2

fig, axe = plt.subplots(dpi=800)
axe.plot(points, y1)
axe.plot(points, y2)
axe.legend(["tanh", "sin"])
axe.annotate("Đây là annotation:\n Tọa độ (0.46, 1)", xy=(0.46, 1), xycoords
             xytext=(0.4, -40), textcoords='offset points',
             arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.5")
```

Out[15]:
```
Text(0.4, -40, 'Đây là annotation:\n Tọa độ (0.46, 1)')
```

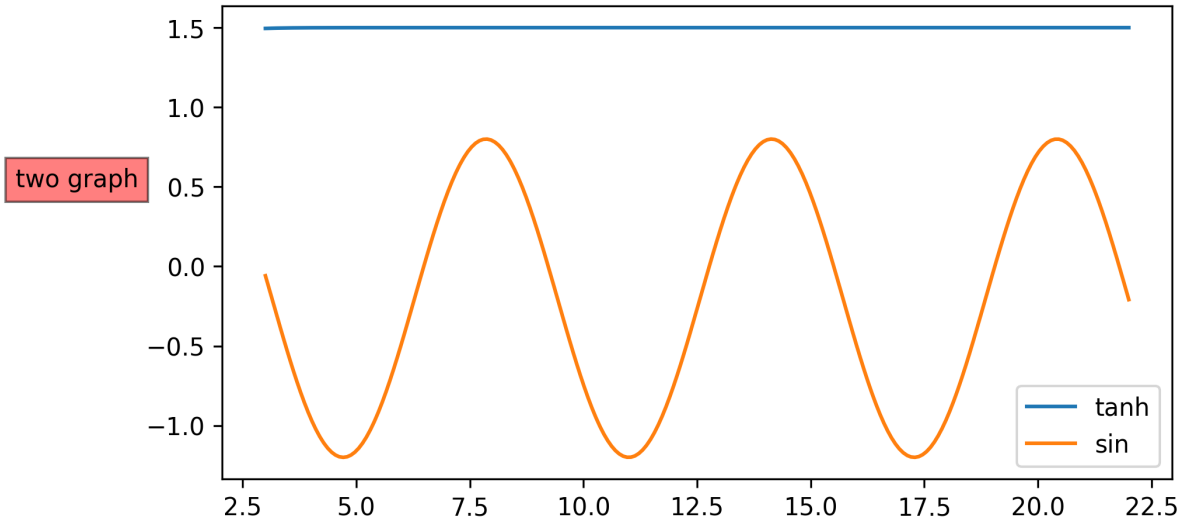


Text

Thay vì sử dụng hàm annotate() để chú thích các thành phần cụ thể, ta có thể thêm text trên đồ thị, vẫn có nhiều lựa chọn để tùy chỉnh phần text hiển thị

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt

points = np.linspace(3, 22, 223)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2

fig, axes = plt.subplots(figsize=(7, 3.5), dpi=300)
axes.plot(points, y1)
axes.plot(points, y2)
axes.legend(["tanh", "sin"])

# Thêm một đoạn text "two graph" với các thông số vị trí (-2.5, 0.5),
# đoạn text được đặt box bằng thuộc tính bbox(màu nền)
axes.text(-2.5, 0.5, "two graph", bbox=dict(facecolor='red', alpha=0.5))
```

Out[16]:
```
Text(-2.5, 0.5, 'two graph')
```

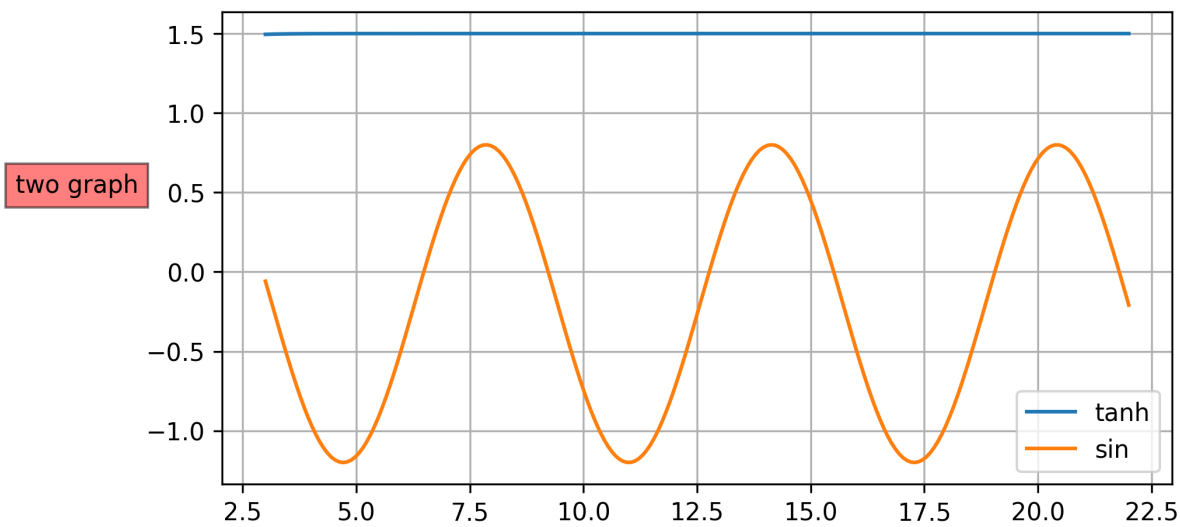


Ngoài ra text còn hỗ trợ nhiều chức năng khác như phông chữ, cỡ chữ, thêm các công thức toán (thông qua Latex - mình quên latex rồi nên không có ví dụ đoạn này)

Grid

Đối với các biểu đồ, đồ thị, lưới có thể được thêm vào để so sánh, định vị các điểm hoặc trong trường hợp nhúng một map cụ thể vào trong biểu đồ

In [17]:
```
import numpy as np
import matplotlib.pyplot as plt

points = np.linspace(3, 22, 223)
y1 = np.tanh(points) + 0.5
y2 = np.sin(points) - 0.2

fig, axes = plt.subplots(figsize=(7, 3.5), dpi=300)
axes.plot(points, y1)
axes.plot(points, y2)
axes.legend(["tanh", "sin"])

# Thêm một đoạn text "two graph" với các thông số vị trí (-2.5, 0.5),
# đoạn text được đặt box bằng thuộc tính bbox(màu nền)
axes.text(-2.5, 0.5, "two graph", bbox=dict(facecolor='red', alpha=0.5))
axes.grid()
```



In [ ]: