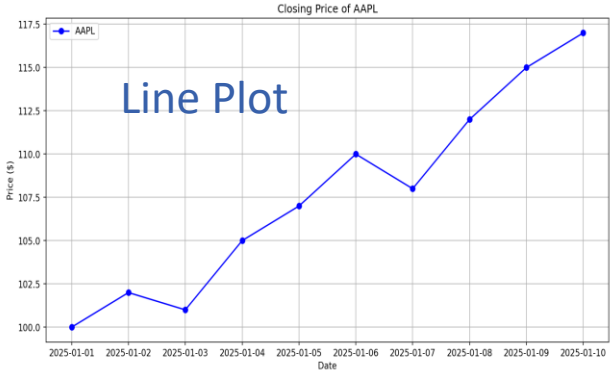
Chapter 7
Data Visualization with Matplotlib

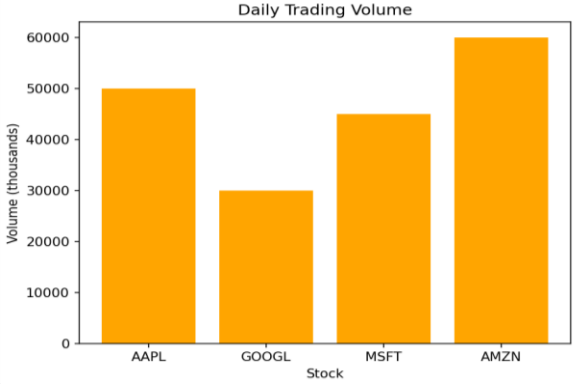
Chapter Content

- 1. Introduction
- 2. Install
- 3. Types of plot in matplotlib
- 4. exercises and practical examples

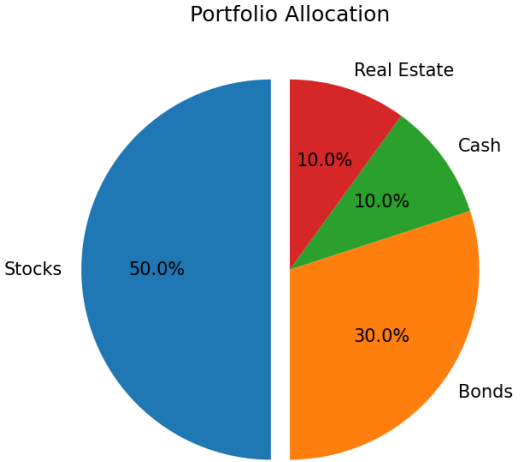
- Data Visualization:
  - is the process of presenting data in the form of graphs or charts.
  - is also used in high-level data analysis for Machine Learning and Exploratory Data Analysis (EDA)
- Matplotlib:
  - is a low-level library of Python which is used for data visualization.
  - is easy to use and emulates MATLAB like graphs and visualization.



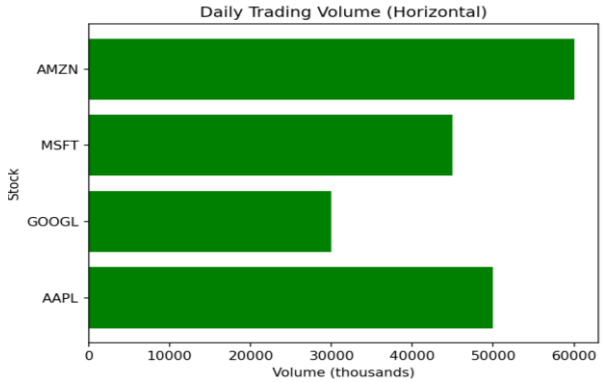
Line Plot

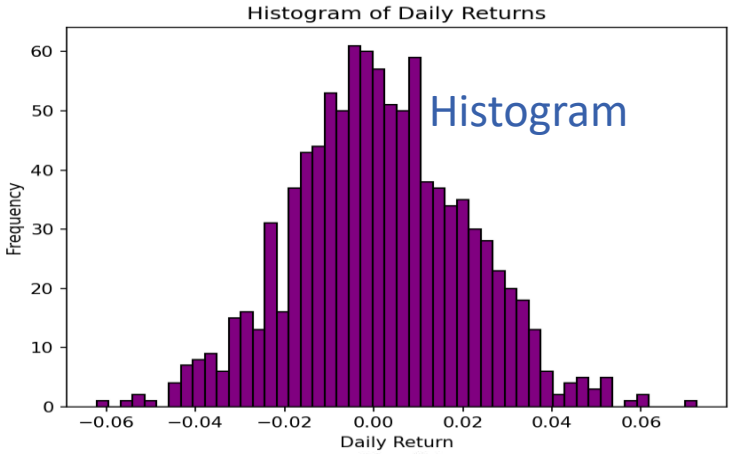


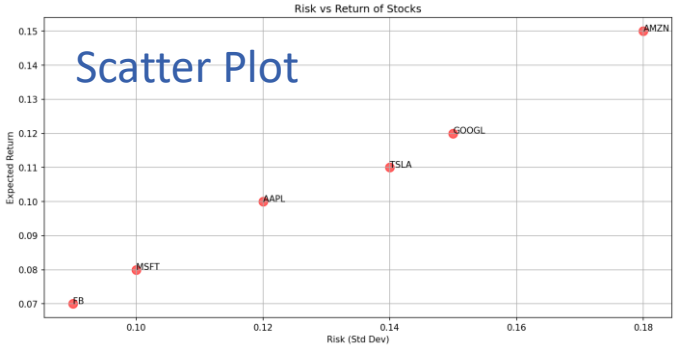
Bar Plot

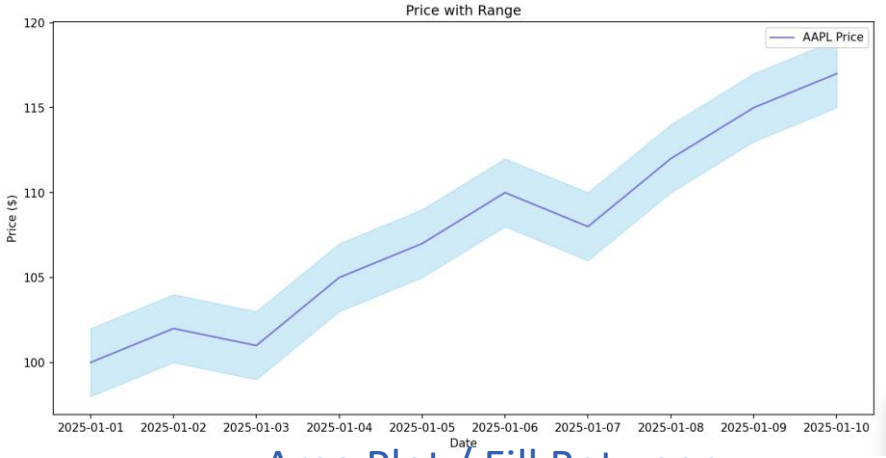


Pie Chart

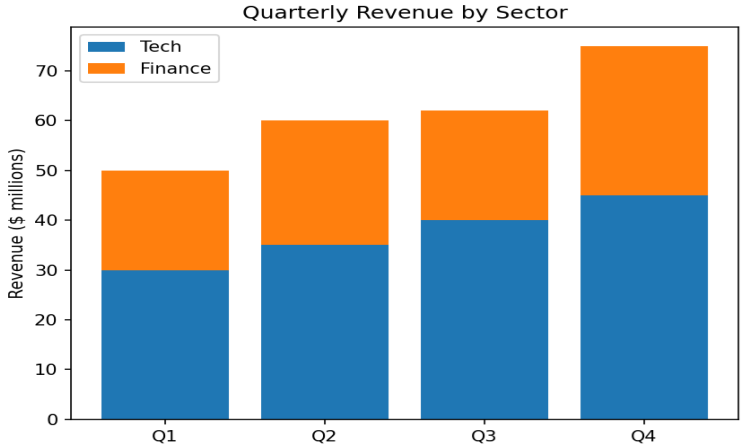




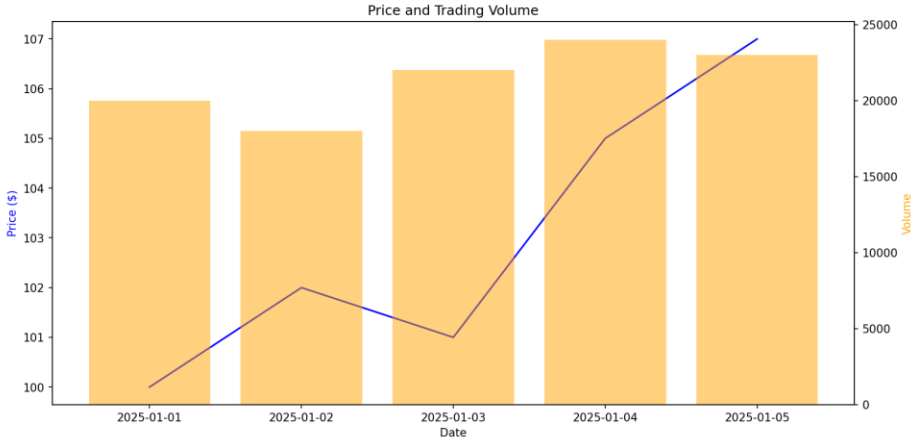




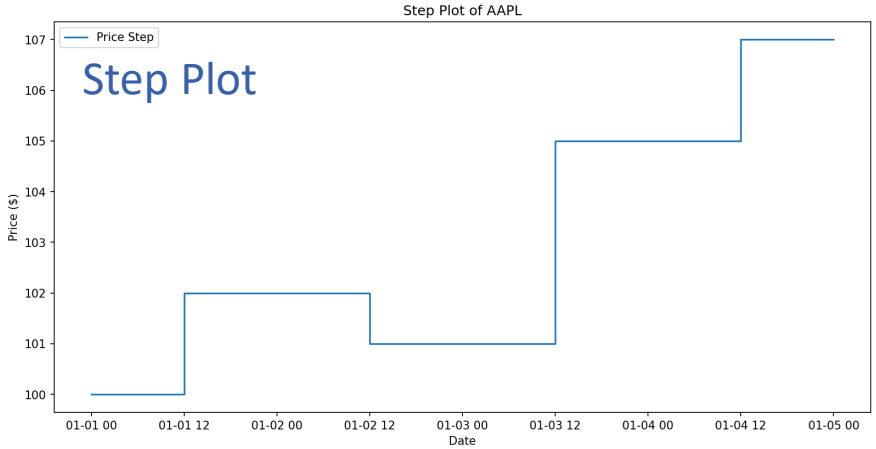
Area Plot / Fill Between



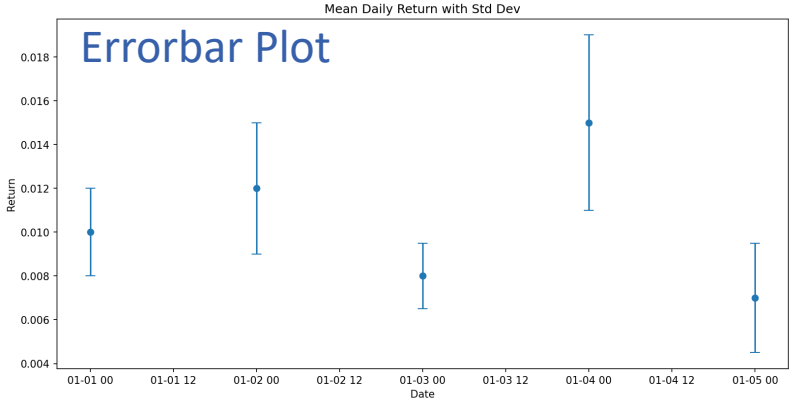
Stacked Bar Chart

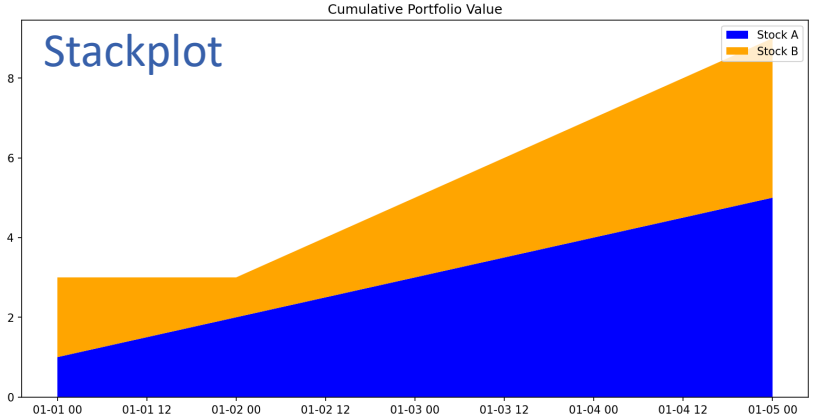


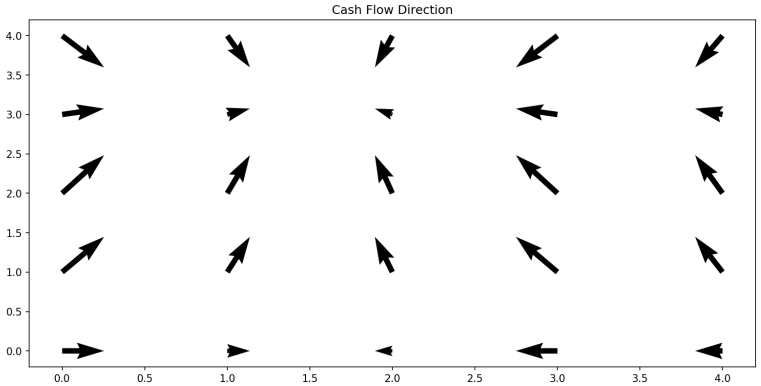
Combined Plot



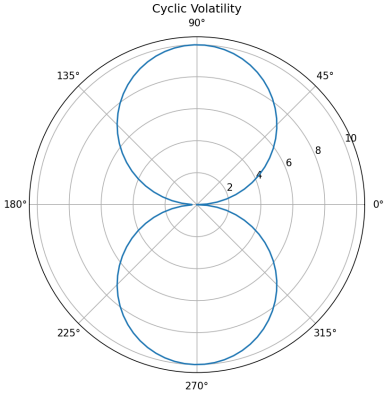
Step Plot







Quiver Plot



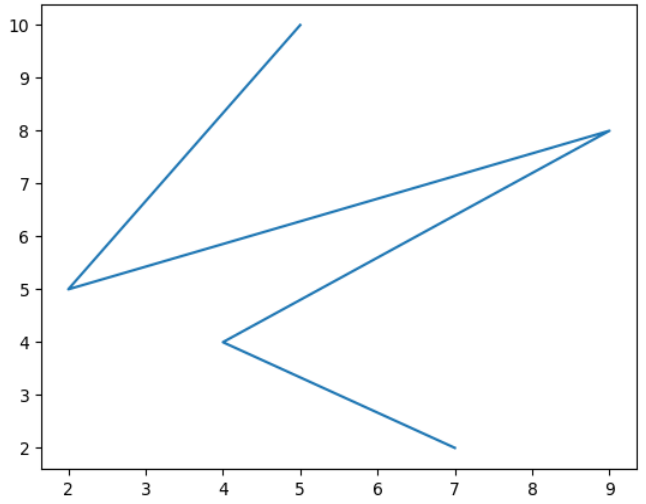
Polar Plot

2. Install

- Step 1 - Make sure Python and pip is preinstalled on your system
  - Check Python : python –version
  - Check pip : pip –V
- Step 2 - Install Matplotlib
  - Command : pip install matplotlib
- Step 3 - Check if it is installed successfully
  - Command : import matplotlib
  - Check version : Matplotlib.__version__

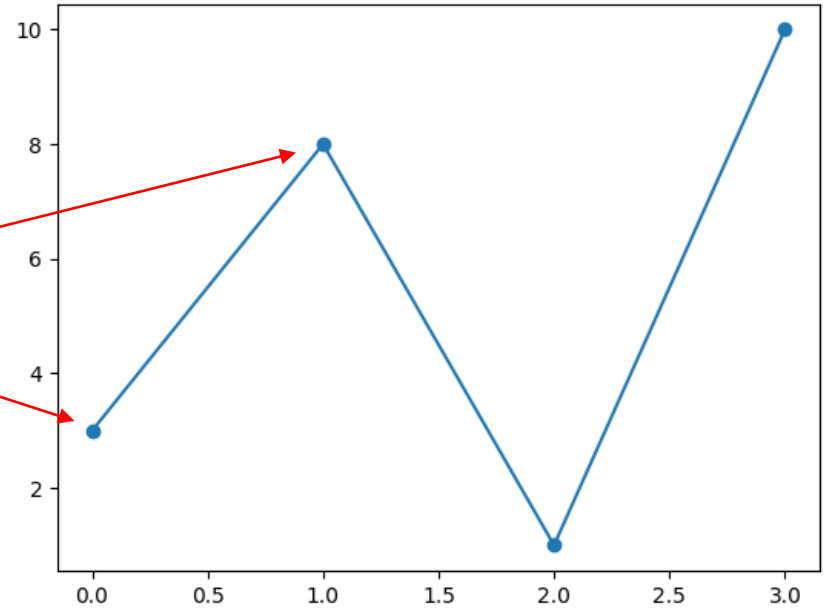
- Line Plot
  - A line plot displays information as a series of data points connected by straight line segments.
  - It is commonly used to visualize data trends or data changes over time
  - **Applications:** Daily stock prices, cumulative returns, market indices…
- Example 1

```
from matplotlib import
                pyplot as plt

x = [5, 2, 9, 4, 7]
y = [10,5, 8,4, 2]

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



- Example 2

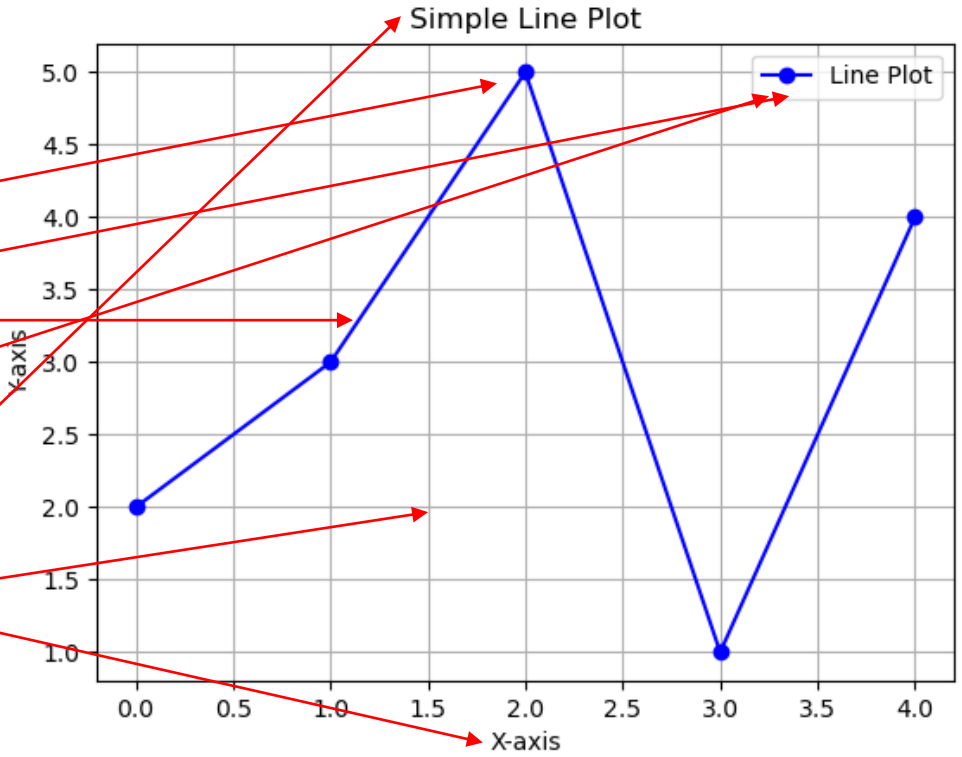
```
import matplotlib.pyplot as plt
import numpy as np

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

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



When only y values are provided in a plot, the x-axis automatically takes the index positions of the array as its values.

- Example 3

```
import matplotlib.pyplot as plt
import numpy as np

y = [2, 3, 5, 1, 4]
plt.plot(y, marker='o',
color='blue', label='Line
Plot')

legend = plt.legend()
plt.title("Simple Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid(True)
plt.show()
```



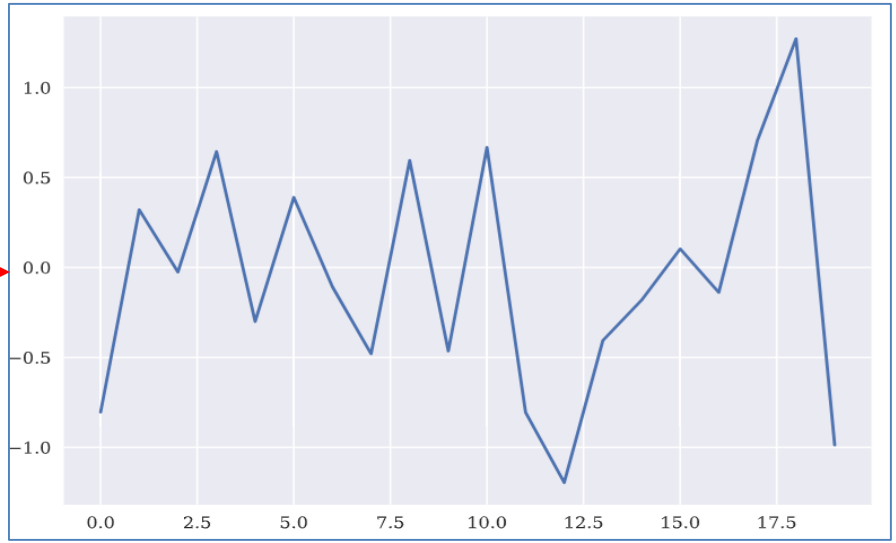
- Example 4

```
import numpy as np
np.random.seed(1000)
y = np.random.standard_normal(20)
plt.plot(y)
```

➔ Fixes the seed for the random number generator

➔ Draws the random numbers (y values).

➔ Calls the plt.plot() function with y objects




```
[-0.8044583   0.32093155
 -0.02548288  0.64432383
 -0.30079667  0.38947455
 -0.1074373  -0.47998308
  0.5950355  -0.46466753
  0.66728131 -0.80611561
 -1.19606983 -0.40596016
 -0.18237734  0.10319289
 -0.13842199  0.70569237
  1.27179528 -0.98674733]
```

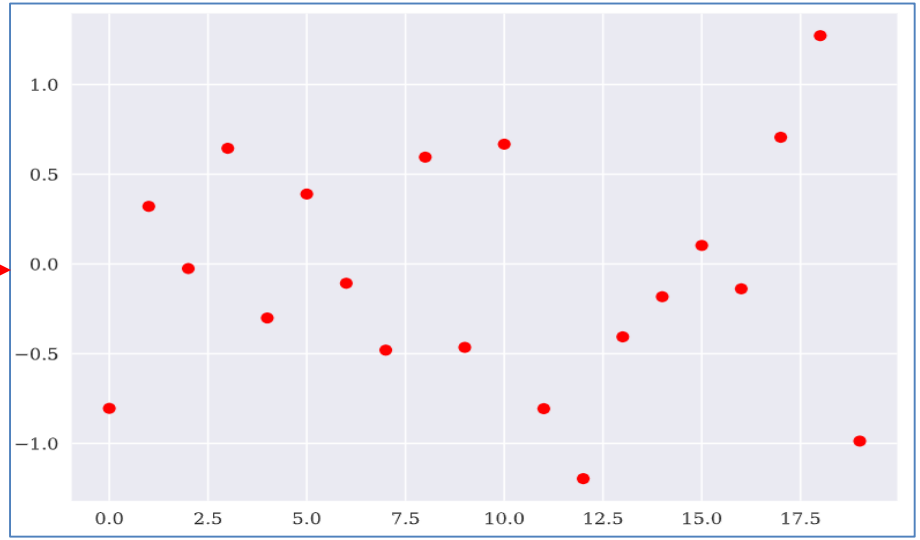
- Example 5

```
import numpy as np
np.random.seed(1000)
y = np.random.standard_normal(20)
plt.plot(y,'ro')
```

➔ Fixes the seed for the random number generator

➔ Draws the random numbers (y values).

➔ Plots the data as red (thick) dots, Circle marker



3. Types of plot in matplotlib - Line Plot

```
plt.figure(figsize=(10, 6))
```
➔ Increases the size of the figure

```
plt.plot(y, 'b', lw=1.5)
```
➔ Plots the data as a line in blue with line width of 1.5 points.

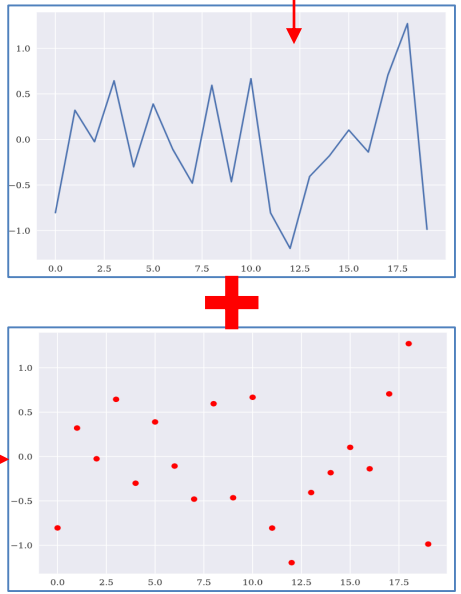
```
plt.plot(y, 'ro')
```
➔ Plots the data as red (thick) dots, Circle marker.

```
plt.xlabel('index')
```
➔ Places a label on the x-axis.

```
plt.ylabel('value')
```
➔ Places a label on the y-axis.

```
plt.title('A Simple Plot')
```
➔ Places a title.

=



3. Types of plot in matplotlib - Line Plot

Table. Standard color abbreviations

Character	Color
b	Blue
g	Green
r	Red
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Table. Standard style characters

Character	Symbol
-	Solid line style
--	Dashed line style
-.	Dash-dot line style
:	Dotted line style
.	Point marker
,	Pixel marker
o	Circle marker
v	Triangle_down marker
^	Triangle_up marker
<	Triangle_left marker
>	Triangle_right marker

3. Types of plot in matplotlib - Line Plot

```
plt.figure(figsize=(10, 6))
plt.plot(y, 'b', lw=1.5)
plt.plot(y, 'r^')
plt.xlabel('index')
plt.ylabel('value')
plt.title('A Simple Plot')
```

➔ Increases the size of the figure

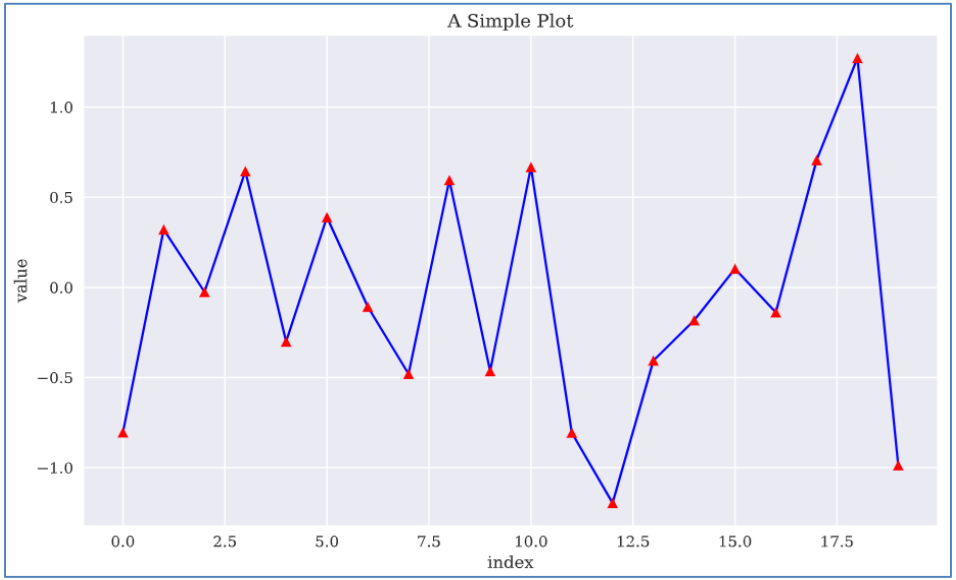
➔ Plots the data as a line in blue with line width of 1.5 points.

➔ Plots the data as red (thick) dots, Triangle_up marker .

➔ Places a label on the x-axis.

➔ Places a label on the y-axis.

➔ Places a title.

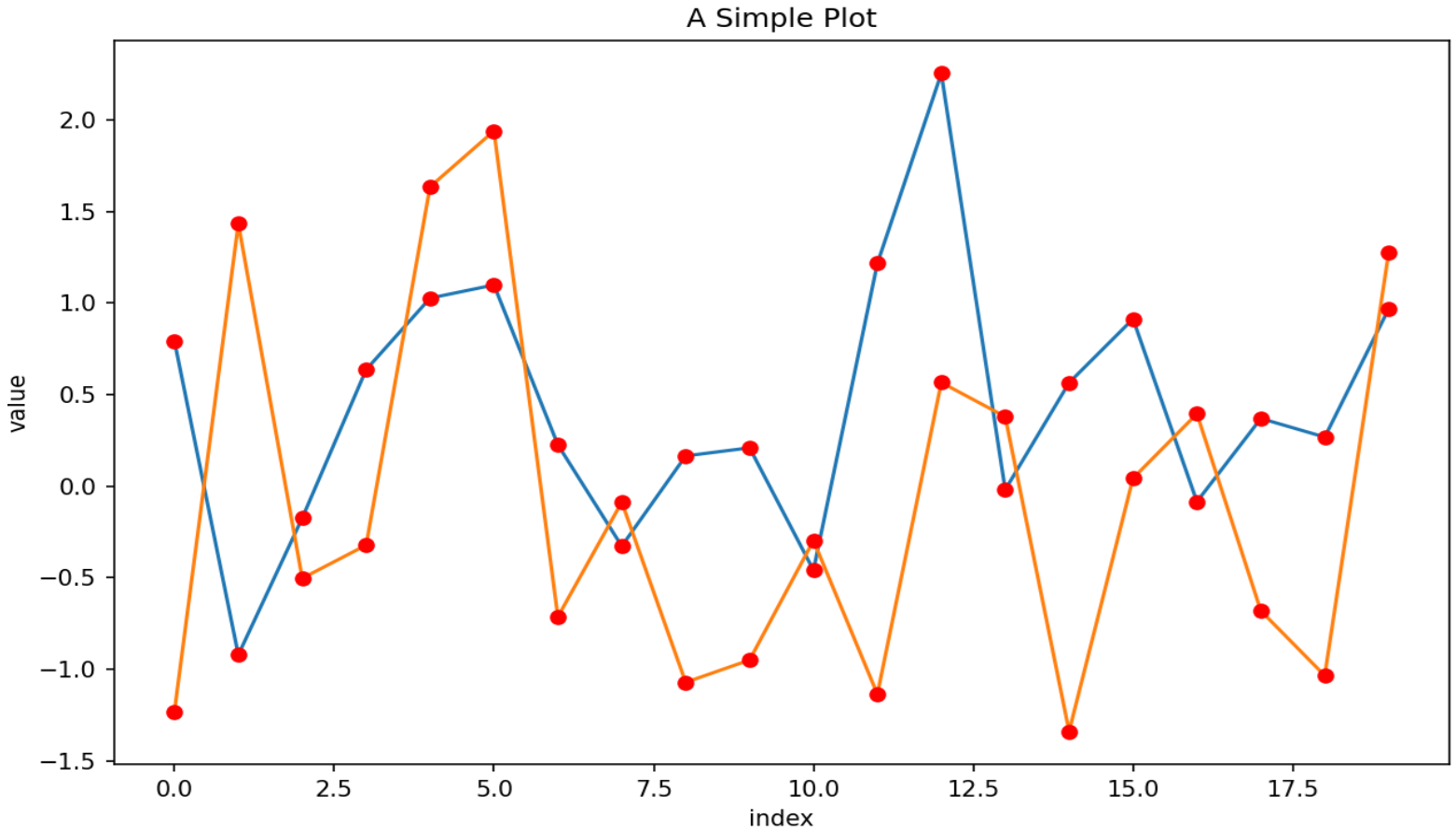


- Example 8

```
import matplotlib.pyplot as plt
import numpy as np
y = np.random.standard_normal((20, 2))
plt.figure(figsize=(10, 6))
plt.plot(y, lw=1.5)
plt.plot(y, 'ro')
plt.xlabel('index')
plt.ylabel('value')
plt.title('A Simple Plot')
```

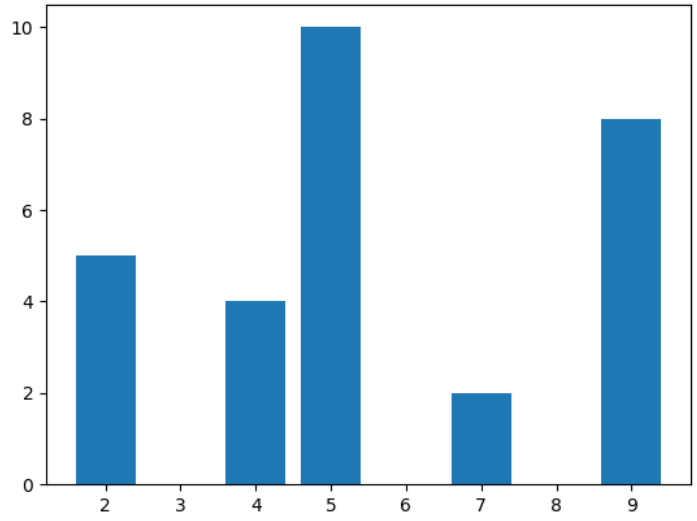
```
[[ 1.50167009  0.31034411]
 [ 1.01414783  0.48093473]
 [ 0.73800307  0.94685925]
 [ 0.01738271 -0.47428203]
 [-0.05815247 -0.1547446 ]
 [-0.98434795  0.22024653]
 [ 1.00150591 -0.82065426]
 [ 0.43496138 -0.36284299]
 [ 0.3893111  -0.19778511]
 [ 1.26038463  0.67003373]
 [-0.12788812 -1.40356686]
 [ 0.43526308 -1.1694499 ]
 [-0.0918581   0.06148139]
 [ 0.87209119 -0.35016075]
 [ 0.30374422  0.31238625]
 [ 0.3475887  -0.40458492]
 [-0.31216896  2.32419881]
 [ 0.90934476 -2.16542749]
 [-1.29162237  0.23839742]
 [-0.84816247 -0.69662249]]
```

3. Types of plot in matplotlib - Line Plot



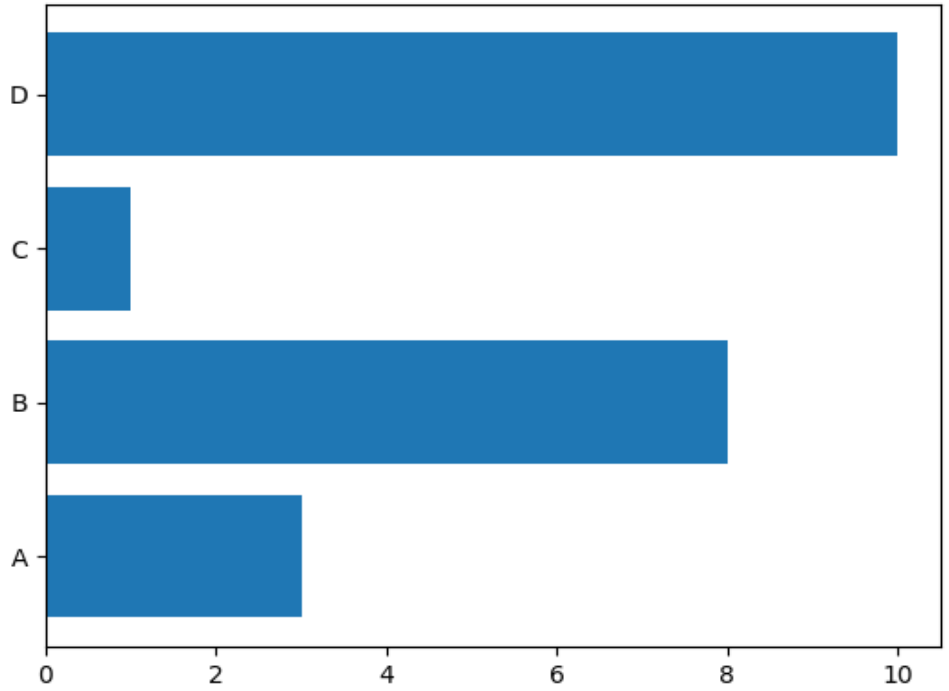
- Bar Plot:
  - A bar plot represents categorical data with rectangular bars.
  - Bar heights (or lengths) correspond to the values they represent.
  - Useful for comparing different discrete values groups or categories.
  - **Applications:** Trading volume, industry revenues….
- Example 9

```
from matplotlib import
                pyplot as plt

x = [5, 2, 9, 4, 7]
y = [10,5, 8,4, 2]

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



- Example 10
- Horizontal Bar Chart: Represents discrete data horizontally, convenient when labels are long

```
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()
```

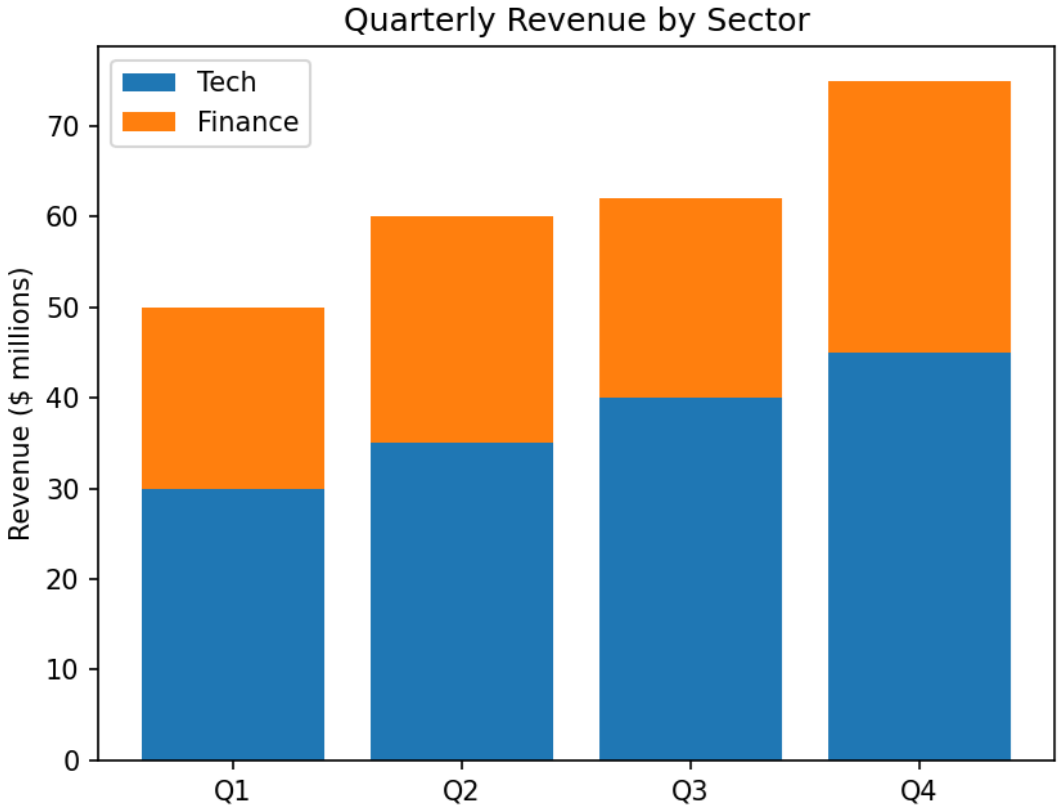


3. Types of plot in matplotlib - Stacked Bar Chart

- Stacked Bar Chart
  - Shows the proportion of each group within the total value.
  - Applications: Industry revenue, profit allocation…
- Example 11

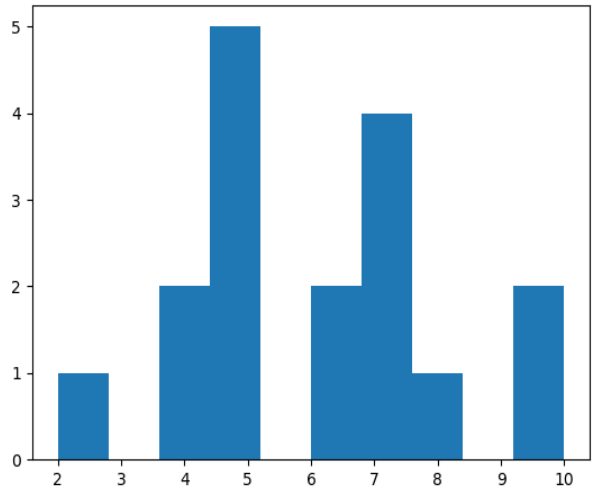
```
import matplotlib.pyplot as plt

quarters = ['Q1', 'Q2', 'Q3', 'Q4']
tech_revenue = [30, 35, 40, 45]
finance_revenue = [20, 25, 22, 30]

plt.bar(quarters, tech_revenue, label='Tech')
plt.bar(quarters, finance_revenue, bottom=tech_revenue, label='Finance')
plt.title("Quarterly Revenue by Sector")
plt.ylabel("Revenue ($ millions)")
plt.legend()
plt.show()
```

3. Types of plot in matplotlib - Stacked Bar Chart



- Histogram Plot
  - A histogram shows the distribution of a dataset.
  - It groups data into ranges (called bins) and counts how many values fall into each bin.
  - **Applications:** Useful for visualizing the frequency of data over intervals: Daily return distributions, testing for normality…
- Example 12

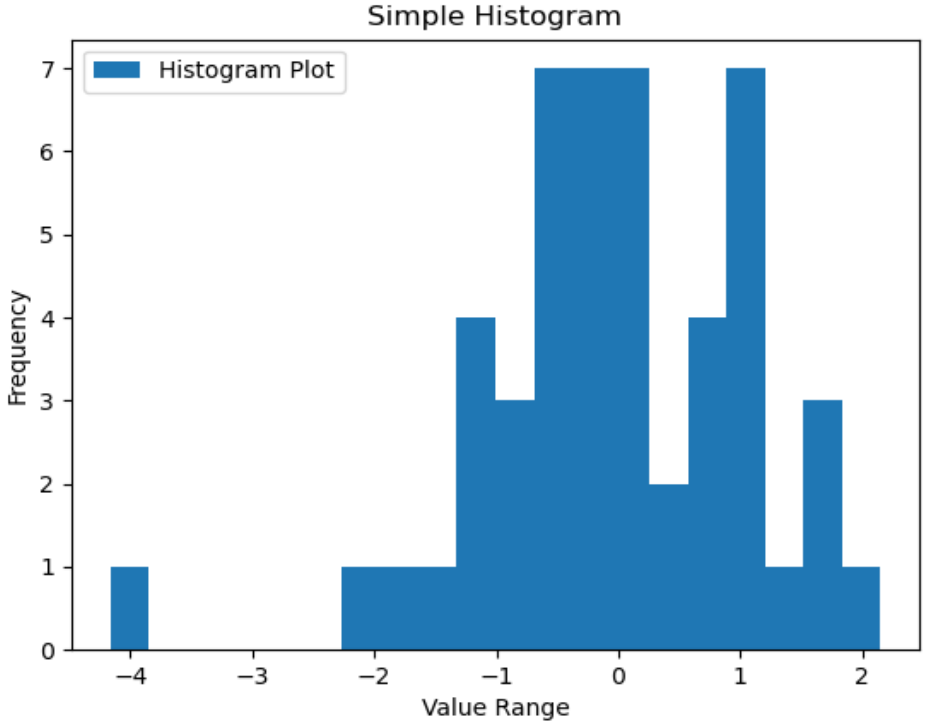
```
from matplotlib import
            pyplot as plt

y =[10,5,8,5,5,5,5,6,6,7,7,7,7,4,10,4,2]

plt.hist(y)
plt.show()
```



3. Types of plot in matplotlib - Histogram Plot

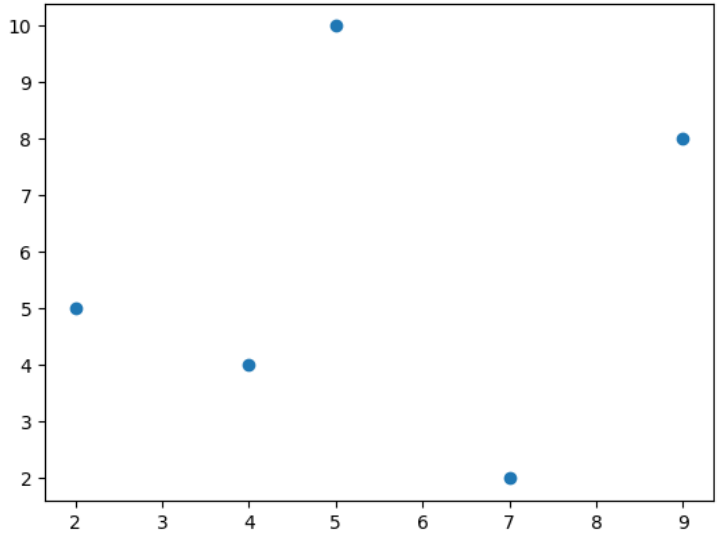
- Example 13

```
import matplotlib.pyplot as plt
import numpy as np
y = np.random.standard_normal((50))
plt.hist(y, bins=20,
label='Histogram Plot')
legend = plt.legend()
plt.title("Simple Histogram")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.show()
```



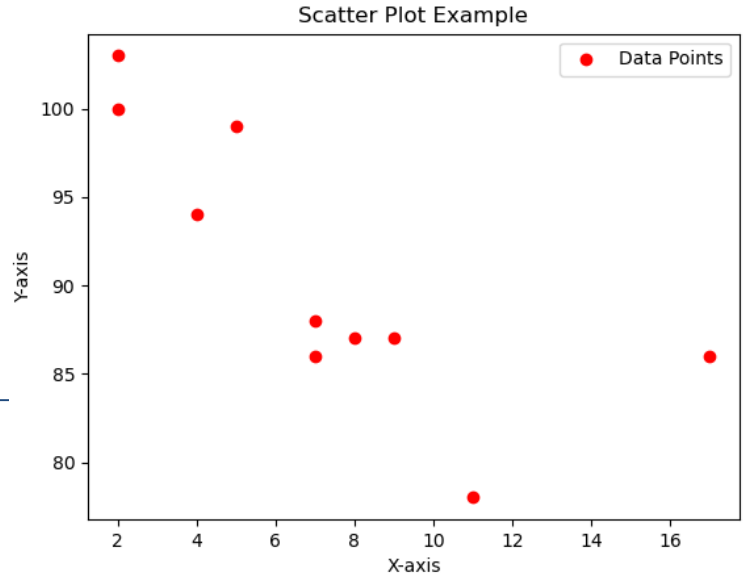
- Scatter Plot:
  - A scatter plot displays individual data points on a two-dimensional coordinate system (x, y). It helps to visualize the relationship or distribution between two variables.
  - Applications: Risk vs. return, price vs. volume…
- Example 14

```
from matplotlib import
                pyplot as plt

x = [5, 2, 9, 4, 7]
y = [10,5, 8,4, 2]
plt.scatter(x,y)
plt.show()
```

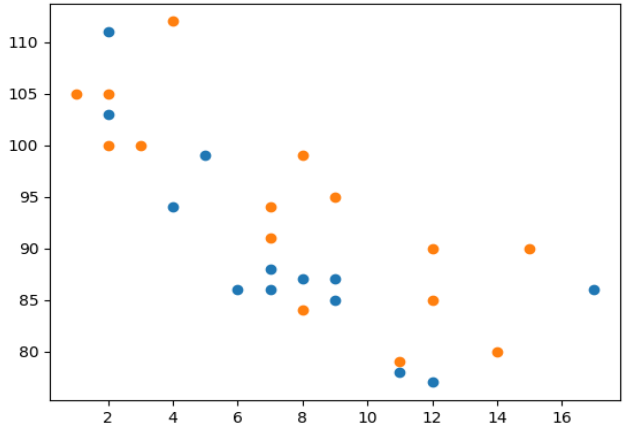


- Example 15

```
import matplotlib.pyplot as plt
import numpy as np
x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11]
y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]

plt.scatter(x, y, color='red',
label='Data Points')
plt.title("Scatter Plot Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.show()
```



- Scatter Plot: Draw two plots on the same figure
- Example 16



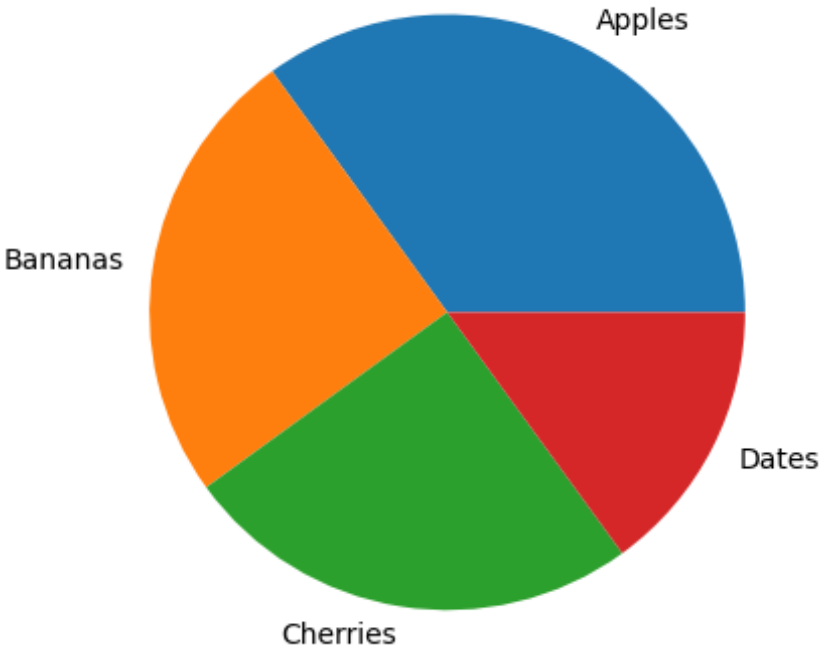

```
import matplotlib.pyplot as plt
import numpy as np

#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()
```

- Pie Chart:
  - A pie chart is a circular statistical graphic that shows how a whole is divided into parts. Each slice represents a proportion of the total
  - Applications: Portfolio allocation, industry market share…
- Example 17

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

- Add labels to the pie chart with the labels parameter.
- Example 18

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas",
            "Cherries", "Dates"]
plt.pie(y, labels = mylabels)
plt.show()
```

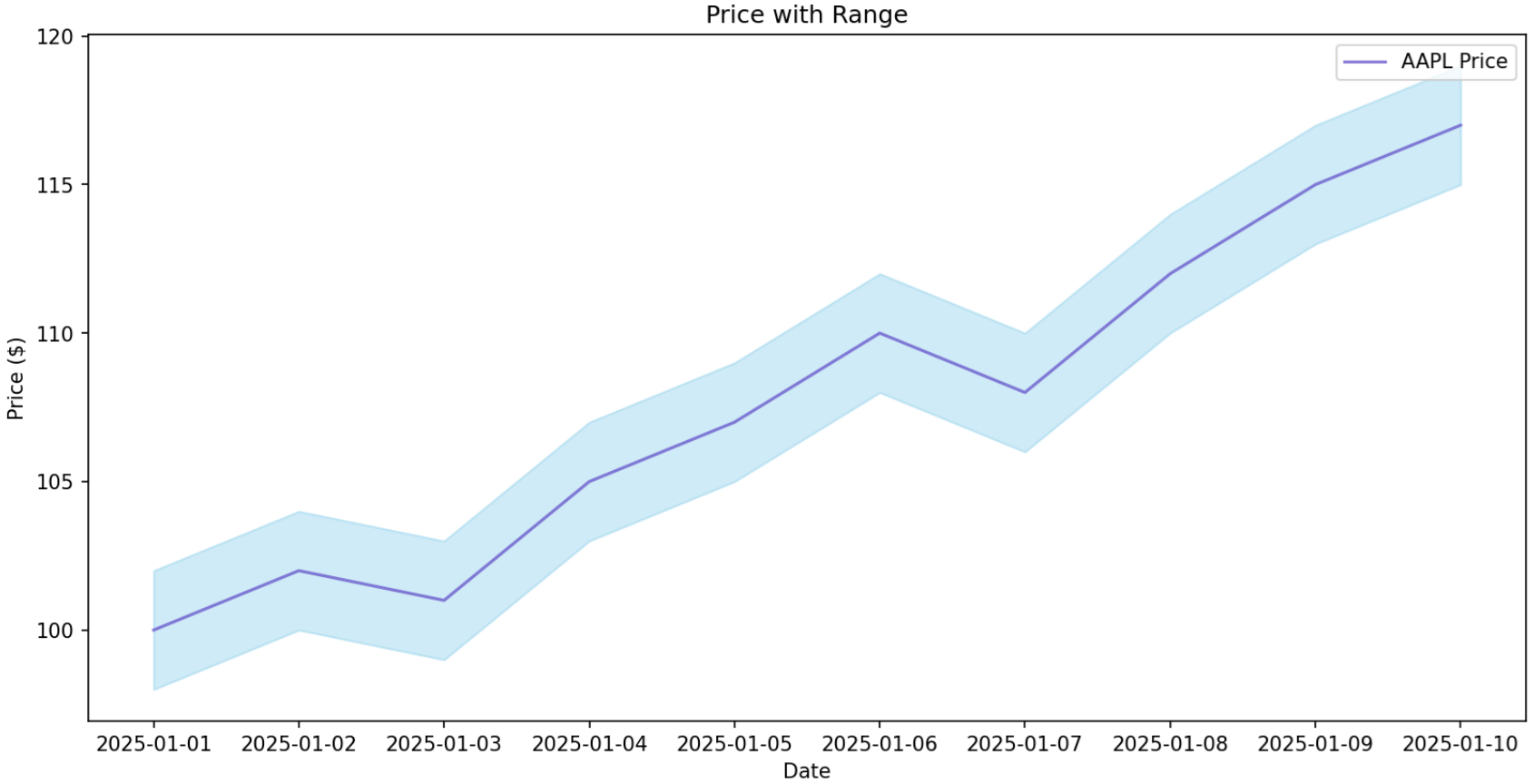


3. Types of plot in matplotlib - Area Plot / Fill Between

- Area Plot / Fill Between
  - Shows cumulative changes and ranges of variation.
  - Applications: Cumulative prices, high-low price ranges...
- Example 19

```
import matplotlib.pyplot as plt
import pandas as pd

dates = pd.date_range("2025-01-01", periods=10)
price = [100, 102, 101, 105, 107, 110, 108, 112, 115, 117]
lower = [p - 2 for p in price]
upper = [p + 2 for p in price]
plt.fill_between(dates, lower, upper, color="skyblue", alpha=0.4)
plt.plot(dates, price, color="Slateblue", alpha=0.8, label='AAPL Price')
plt.title("Price with Range")
plt.xlabel("Date")
plt.ylabel("Price ($)")
plt.legend()
plt.show()
```

3. Types of plot in matplotlib - Area Plot / Fill Between

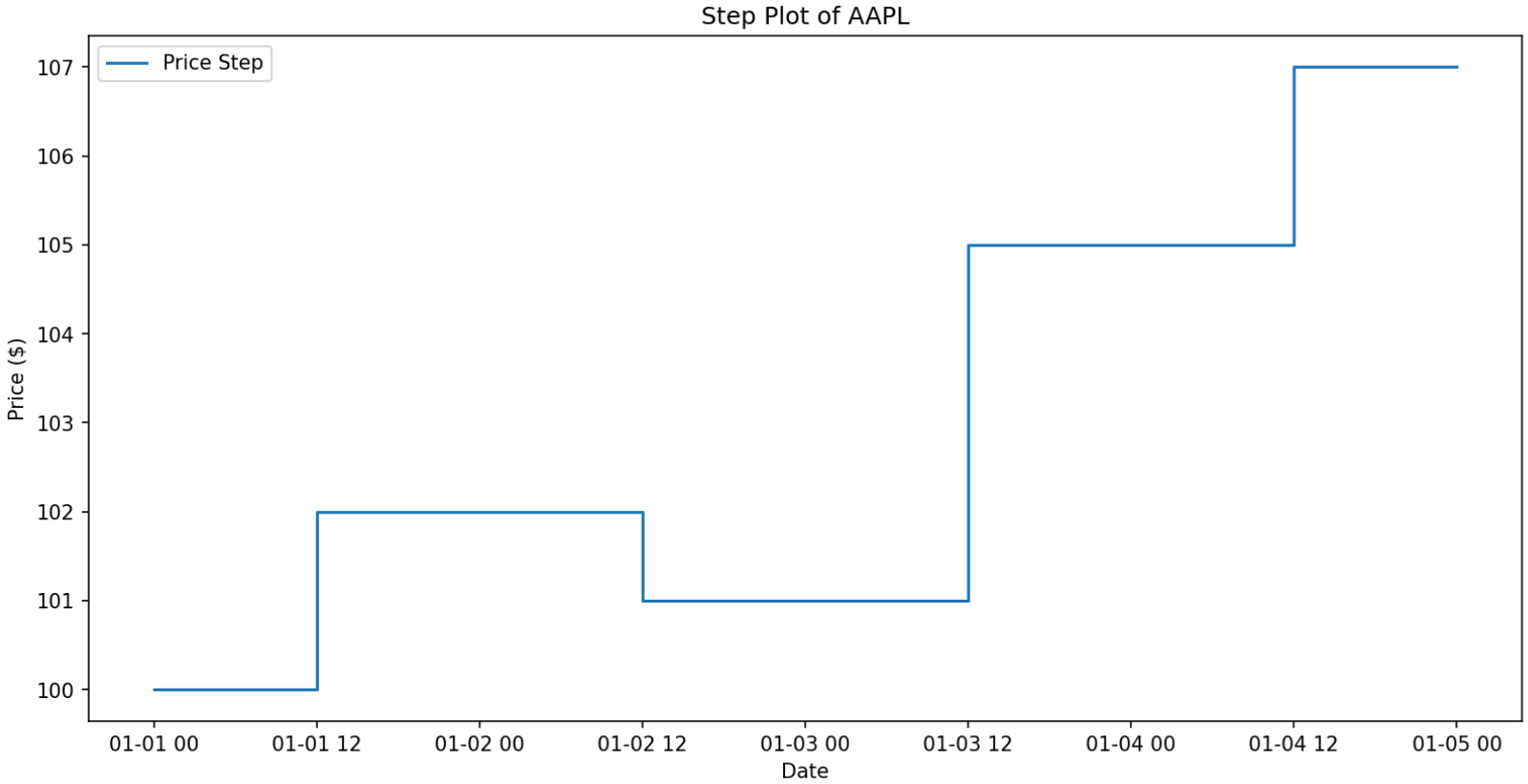


3. Types of plot in matplotlib - Step Plot

- Step Plot
  - Commonly used for closing prices and bid/ask prices.
  - Applications: Stock prices per trade, buy/sell orders...
- Example 20

```
import matplotlib.pyplot as plt
import pandas as pd

dates = pd.date_range("2025-01-01", periods=5)
price = [100, 102, 101, 105, 107]
plt.step(dates, price, where='mid', label='Price Step')
plt.title("Step Plot of AAPL")
plt.xlabel("Date")
plt.ylabel("Price ($)")
plt.legend()
plt.show()
```

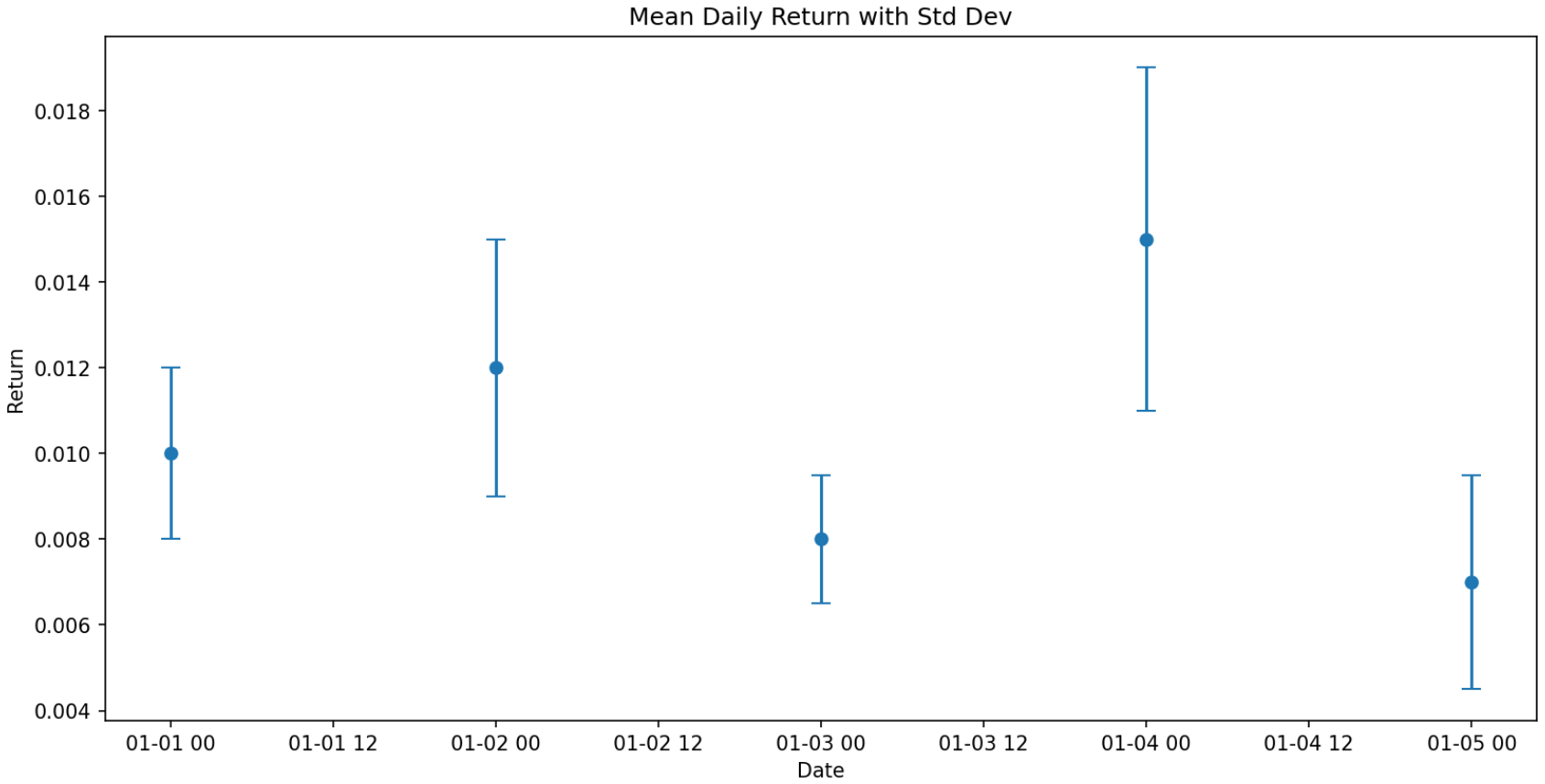
3. Types of plot in matplotlib - Step Plot



- Errorbar Plot
  - Shows variability or error.
  - Applications: Mean value ± standard deviation of returns...
- Example 21

```
import matplotlib.pyplot as plt
import pandas as pd
dates = pd.date_range("2025-01-01", periods=5)
price = [100, 102, 101, 105, 107]
mean_return = [0.01, 0.012, 0.008, 0.015, 0.007]
std_dev = [0.002, 0.003, 0.0015, 0.004, 0.0025]

plt.errorbar(dates, mean_return, yerr=std_dev, fmt='o', capsize=5)
plt.title("Mean Daily Return with Std Dev")
plt.xlabel("Date")
plt.ylabel("Return")
plt.show()
```

3. Types of plot in matplotlib - Errorbar Plot

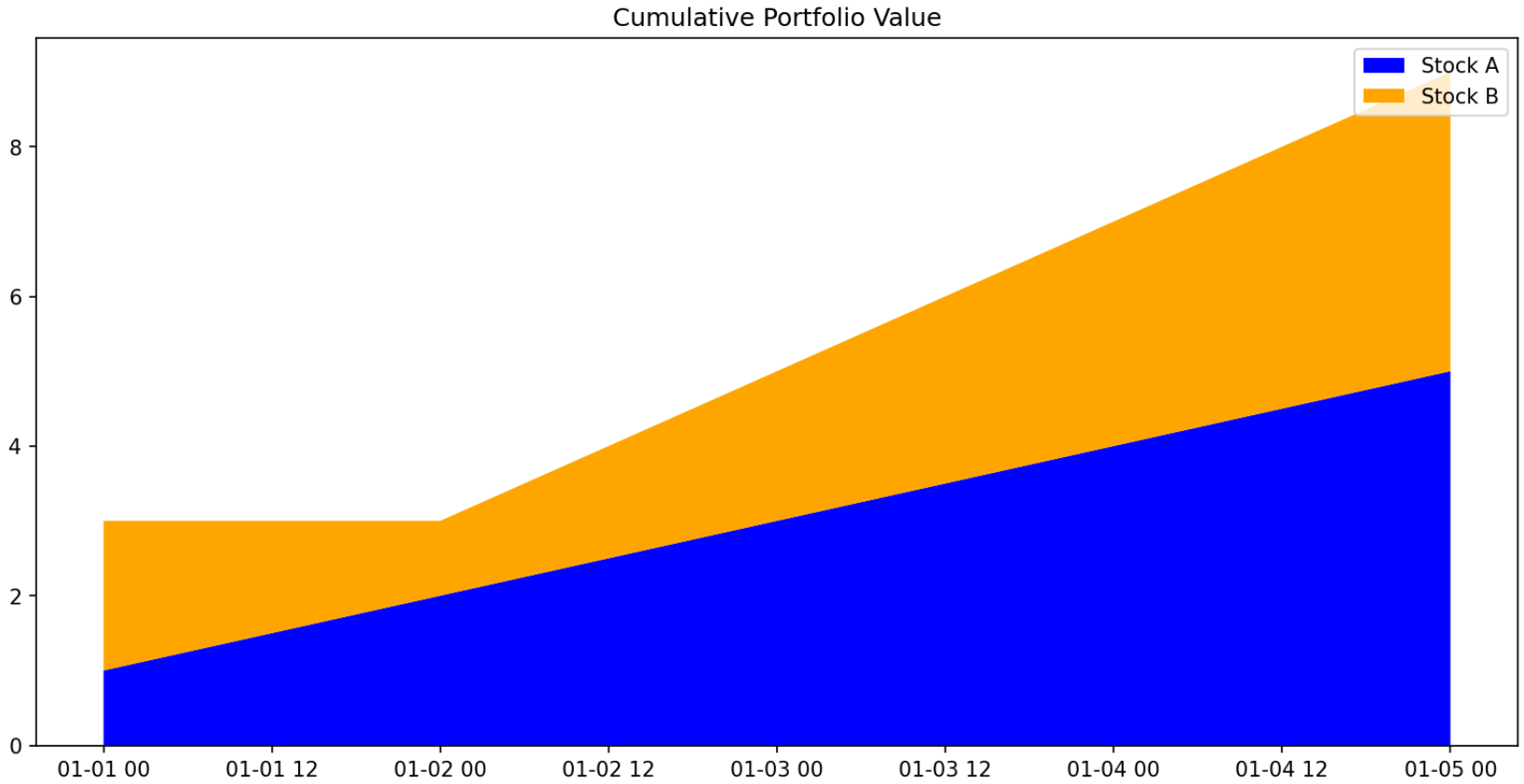


3. Types of plot in matplotlib – Stack Plot

- Stackplot
  - Compares multiple cumulative series simultaneously.
  - Applications: Cumulative asset values across multiple portfolio types…
- Example 22

```
import matplotlib.pyplot as plt
import pandas as pd

dates = pd.date_range("2025-01-01", periods=5)
stocks_a = [1, 2, 3, 4, 5]
stocks_b = [2, 1, 2, 3, 4]

plt.stackplot(dates, stocks_a, stocks_b, labels=['Stock A', 'Stock B'],
                                                colors=['blue','orange'])
plt.title("Cumulative Portfolio Value")
plt.legend()
plt.show()
```

3. Types of plot in matplotlib – Stack Plot

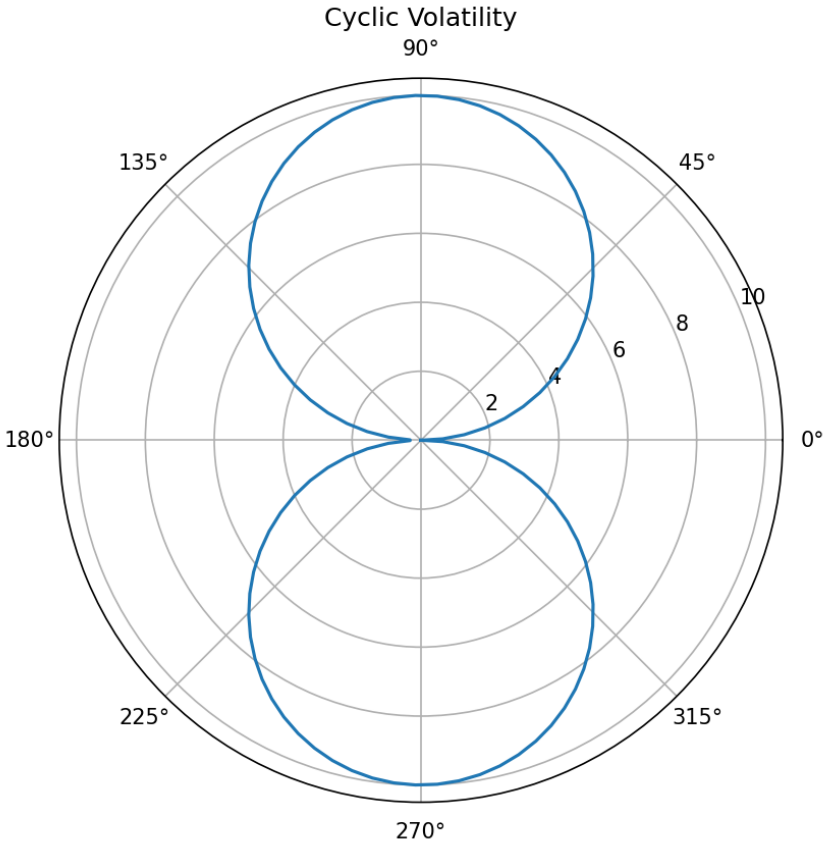


3. Types of plot in matplotlib – Polar Plot

- Polar Plot
  - Meaning: Represents data in a circular or angular format, useful for cyclical patterns.
  - Applications: Displaying risk or time-related data over yearly cycles…
- Example 23

```
import matplotlib.pyplot as plt
import numpy as np

theta = np.linspace(0, 2*np.pi, 100) # Create angle data from 0 to 2π
r = np.abs(np.sin(theta)*10)  # Cyclical volatility (absolute value of
                                sin(theta) multiplied by 10)
plt.polar(theta, r)
plt.title("Cyclic Volatility")
plt.show()
```

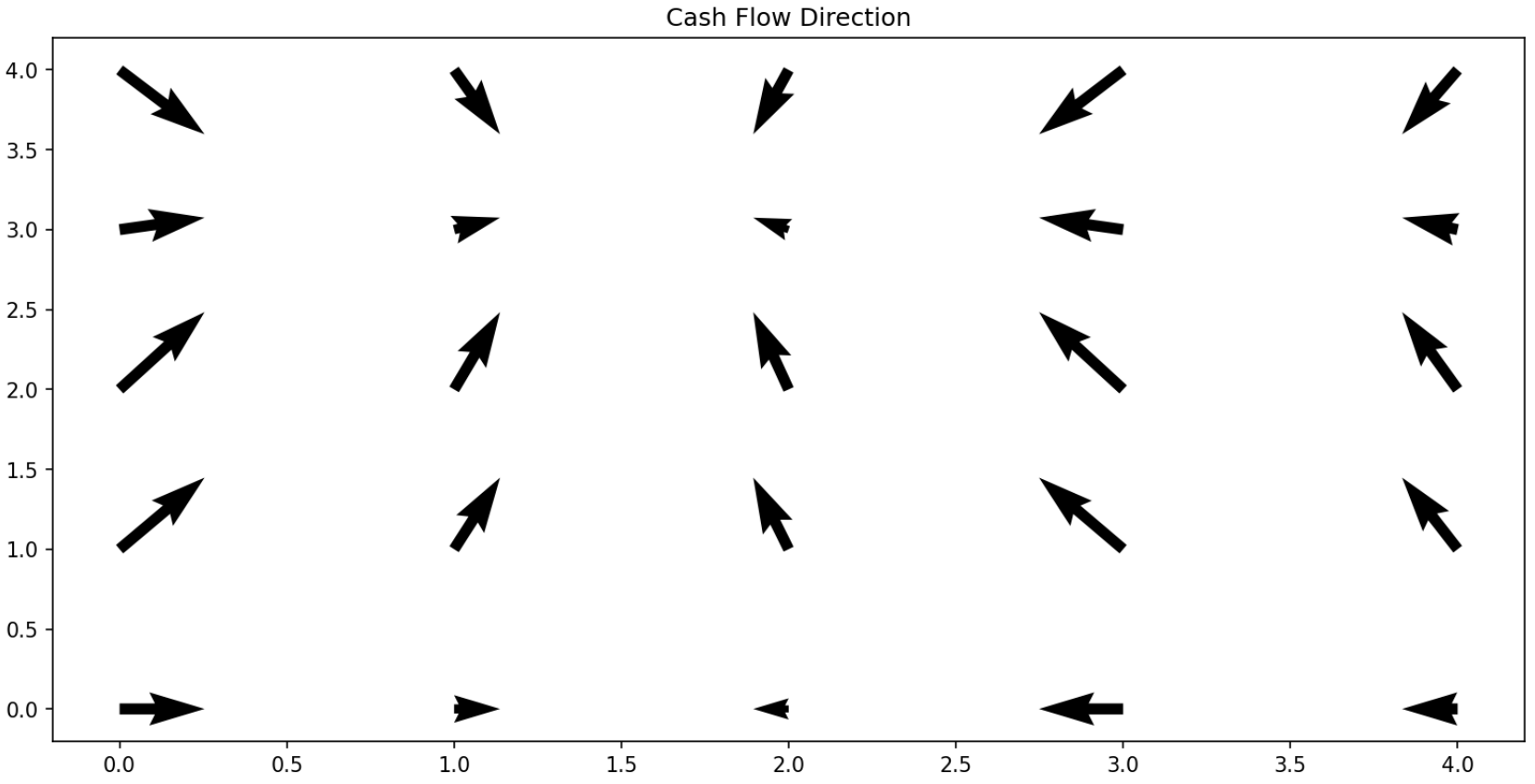
3. Types of plot in matplotlib – Polar Plot



- Quiver Plot
  - Represents direction and magnitude, such as cash flow.
  - Applications: Visualizing capital flow direction or risk-return gradients…
- Example 24

```
import matplotlib.pyplot as plt
import numpy as np

X, Y = np.meshgrid(np.arange(0,5,1), np.arange(0,5,1))
U = np.cos(X)
V = np.sin(Y)

plt.quiver(X, Y, U, V)
plt.title("Cash Flow Direction")
plt.show()
```

3. Types of plot in matplotlib – Quiver Plot

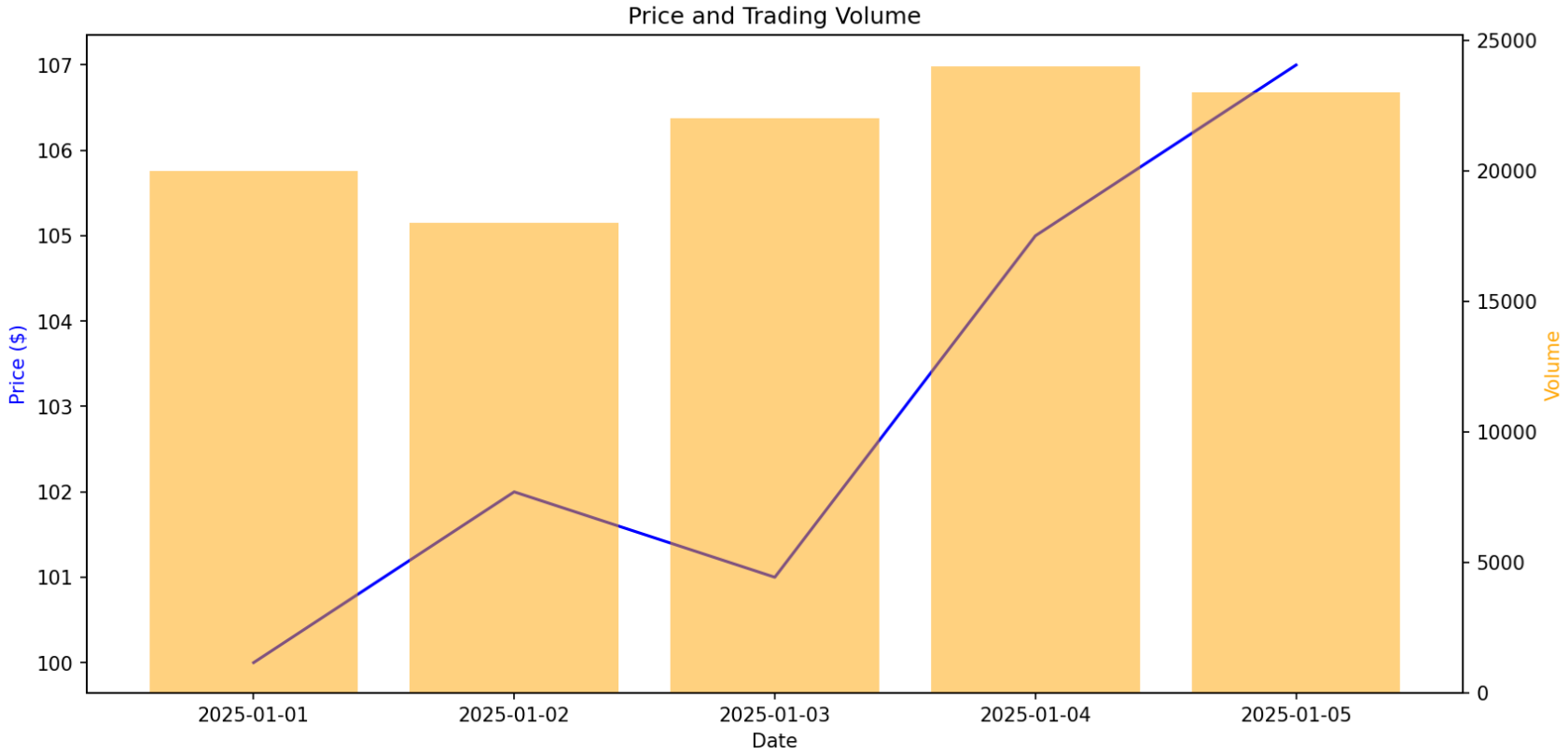


3. Types of plot in matplotlib - Combined Plot

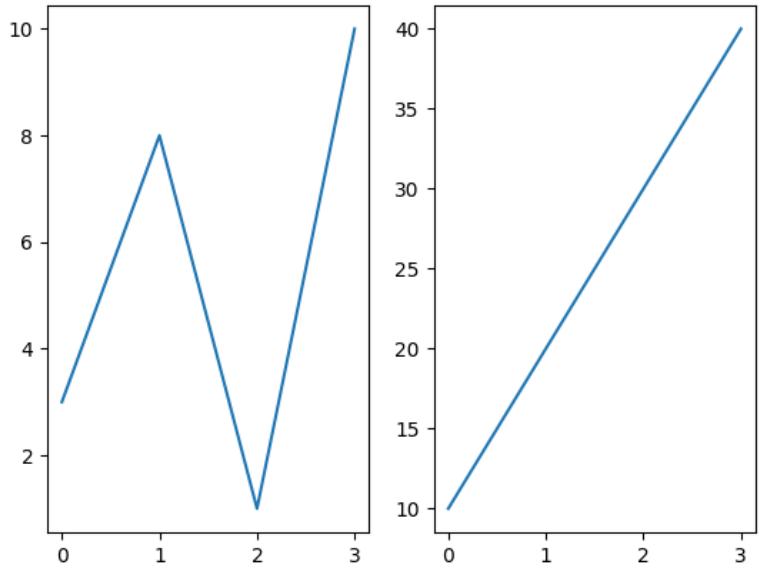
- Combined Plot
  - Meaning: Displays multiple types of data simultaneously.
  - Applications: Stock price with trading volume, return with risk...
- Example 25

```
import matplotlib.pyplot as plt
import pandas as pd
dates = pd.date_range("2025-01-01", periods=5)
price = [100, 102, 101, 105, 107]
volume = [20000, 18000, 22000, 24000, 23000]
fig, ax1 = plt.subplots()
ax1.plot(dates, price, color='blue', label='Price')
ax2 = ax1.twinx()
ax2.bar(dates, volume, color='orange', alpha=0.5, label='Volume')
ax1.set_xlabel("Date")
ax1.set_ylabel("Price ($)", color='blue')
ax2.set_ylabel("Volume", color='orange')
plt.title("Price and Trading Volume")
plt.show()
```

3. Types of plot in matplotlib - Combined Plot



- Multiple Graphs: by repeating the show() method or use a function called subplot() in order to print them horizontally as well.
- Example 26




```
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()
```

4. Thực hành - Bài 1. Vẽ biểu đồ tài chính

```
import matplotlib.pyplot as plt
import pandas as pd
import cufflinks as cf
import plotly.offline as plyo
raw = pd.read_csv('fxcm_eur_usd_eod_data.csv',
                  index_col=0, parse_dates=True)
quotes = raw[['AskOpen', 'AskHigh', 'AskLow', 'AskClose']]
quotes = quotes.iloc[-60:]

quotes.tail()
```

➔ Kết nối pandas với plotly, cho phép trực quan hóa nhanh chóng trực tiếp từ các DataFrame, cài đặt **!pip install cufflinks**

➔ Để hiển thị (render) biểu đồ Plotly mà không cần kết nối Internet

➔

➔ Lọc DataFrame để chỉ giữ lại các cột cần thiết cho biểu đồ nến (candlestick charts)

➔ Chỉ giữ lại 60 dòng cuối cùng, thường để tập trung vào dữ liệu thị trường gần đây

- Hiển thị 5 dòng cuối cùng của dataframe quotes
- Chỉ để kiểm tra, Không ảnh hưởng đến việc vẽ biểu đồ vì kết quả không được lưu

- Nạp dữ liệu từ file 'fxcm_eur_usd_eod_data.csv'.
- Cột đầu tiên là cột chỉ mục (index_col=0), kiểu date.
- parse_dates=True: Tự động chuyển đổi cột chỉ mục thành đối tượng datetime

4. Thực hành - Bài 1. Vẽ biểu đồ tài chính

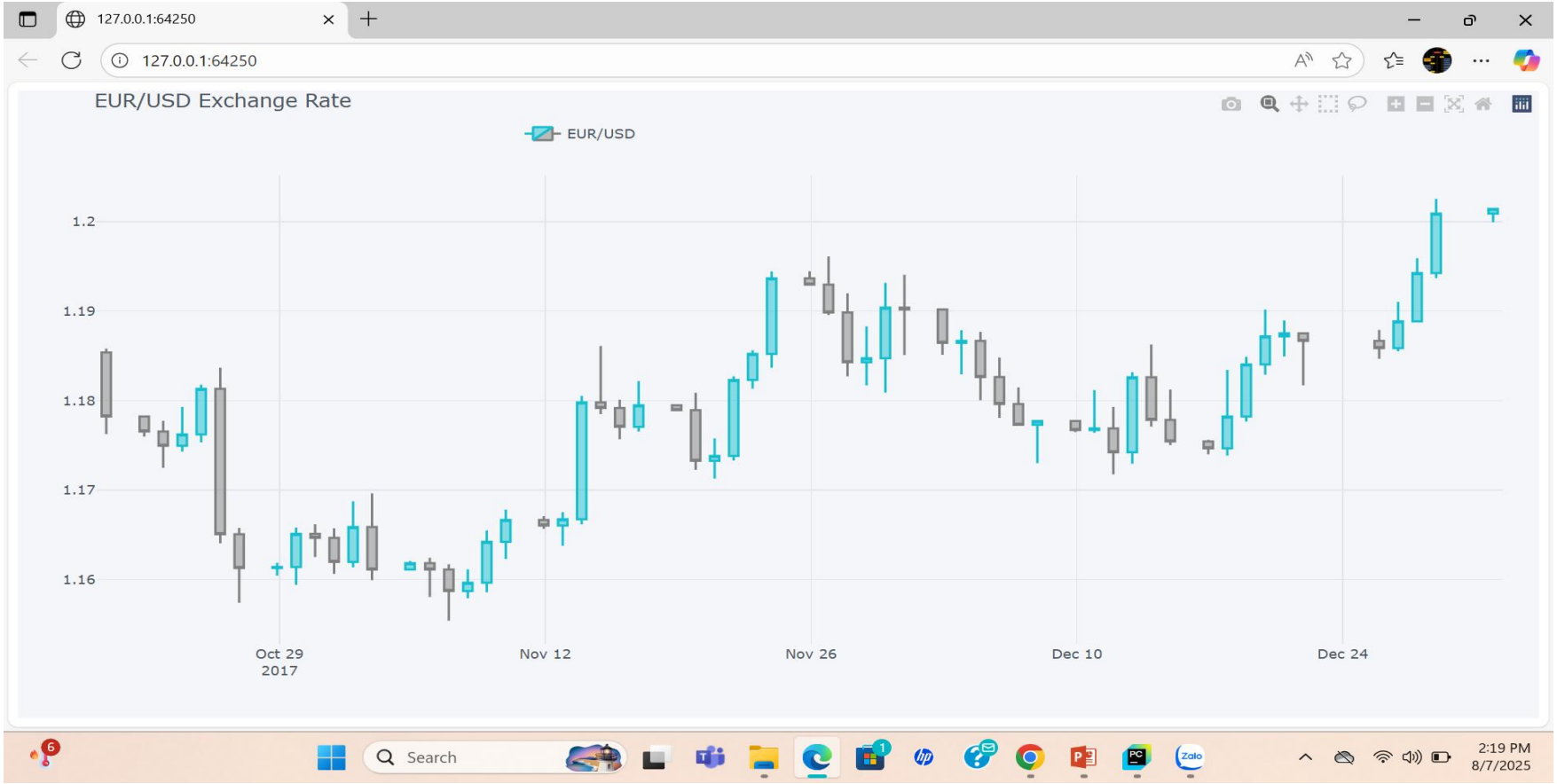
```
qf = cf.QuantFig(quotes,
    title='EUR/USD Exchange Rate',
    legend='top',
    name='EUR/USD')
```

- Tạo Quant Figure (Biểu đồ Nến):
  - Khởi tạo một biểu đồ tài chính bằng cufflinks.QuantFig.
  - quotes: dữ liệu OHLC đầu vào.
  - title: Tiêu đề biểu đồ.
  - legend='top': Đặt chú giải (legend) ở phía trên cùng.
  - name: tên hiển thị trong legend (chú giải).

```
plyo.iplot(
    qf.iplot(asFigure=True),
    image='png',
    filename='qf_01')
plt.show()
```

- Tạo ra một biểu đồ Plotly tương tác :
  - qf.iplot(asFigure=True) Tạo một đối tượng figure của Plotly.
  - plyo.iplot(...) Hiển thị nó ở chế độ offline.
  - image='png': Tùy chọn, cho phép xuất hình ảnh tĩnh.
  - filename='qf_01': Tên được sử dụng khi lưu biểu đồ.

4. Thực hành - Bài 1. Vẽ biểu đồ tài chính

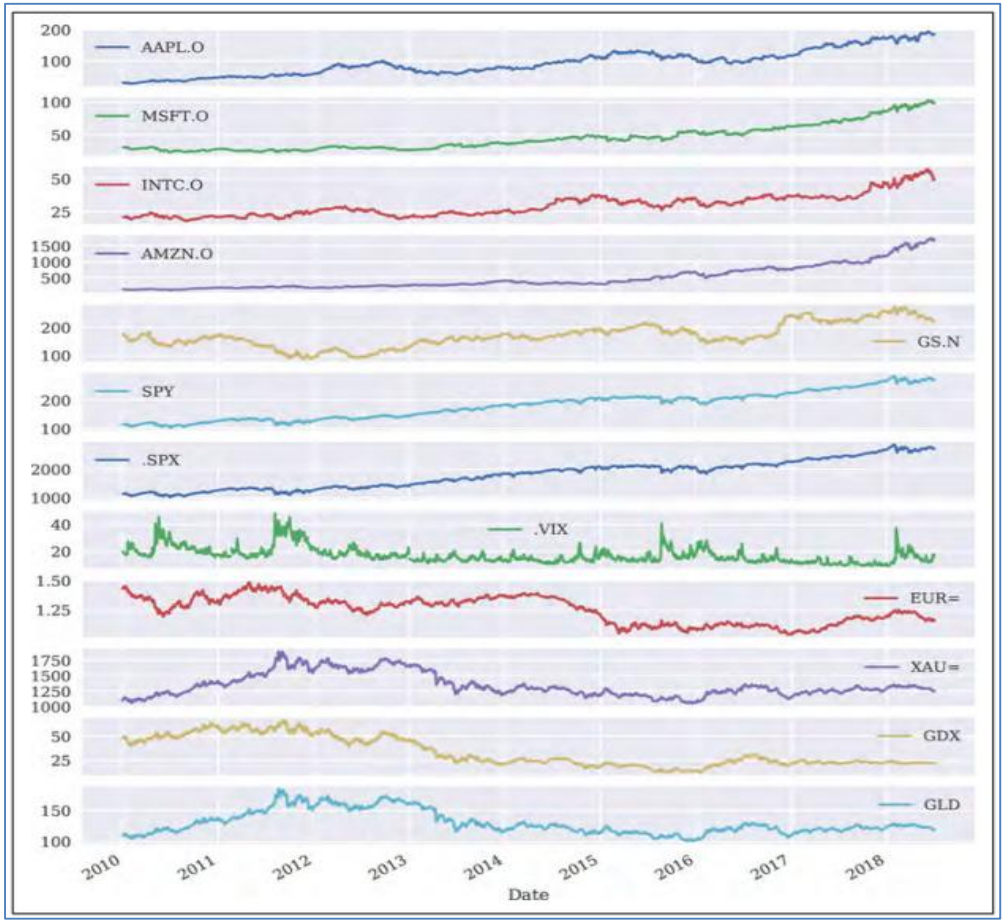


4. Thực hành - Bài 2. Vẽ biểu đồ phân tích tài chính

```
import numpy as np
import pandas as pd
from pylab import mpl, plt

filename = 'source/tr_eikon_eod_data.csv'
data = pd.read_csv(filename,index_col=0, parse_dates=True)

data.plot(figsize=(10, 12), subplots=True)#
```

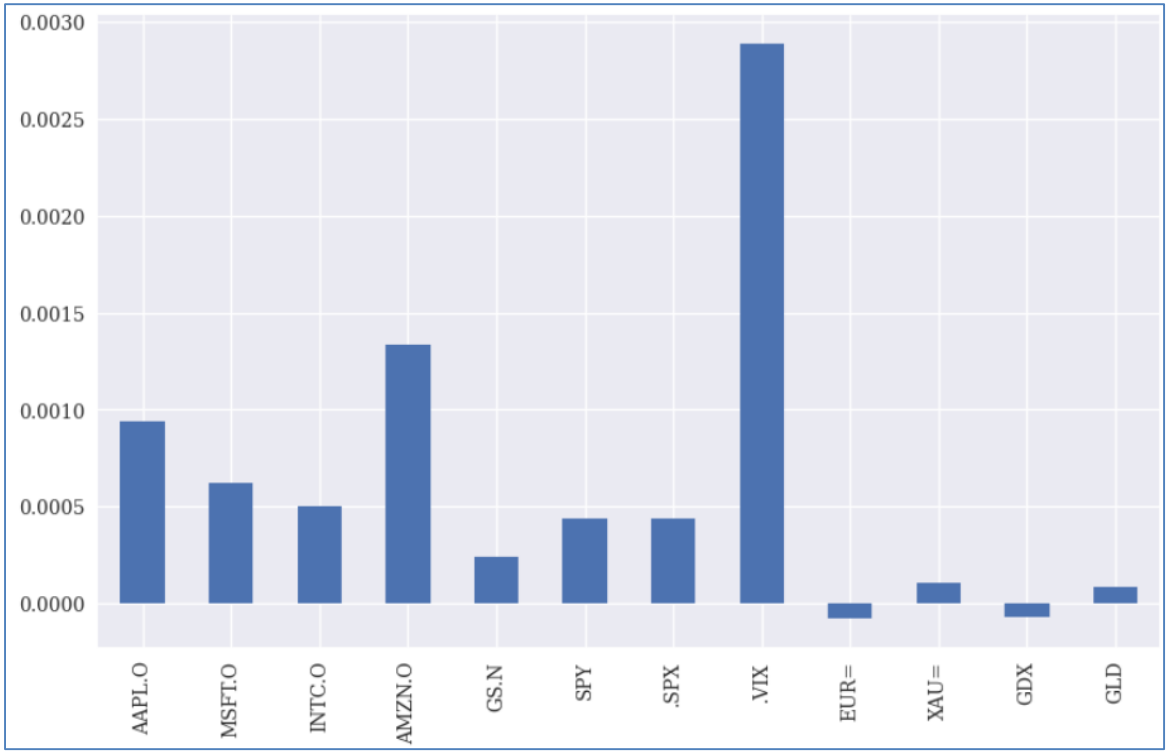
➔trực quan hóa toàn bộ tập dữ liệu thông qua nhiều biểu đồ con (subplots)

4. Thực hành - Bài 2. Vẽ biểu đồ phân tích tài chính



```
data.pct_change().mean().plot(kind='bar', figsize=(10, 6))
```

#➔ Vẽ biểu đồ Giá trị trung bình của các kết quả được trực quan hóa dưới dạng biểu đồ cột (bar plot)



```
rets = np.log(data / data.shift(1))#➔ Tính lợi nhuận theo hàm log
rets.head().round(3)    #➔ Tạo tập dữ liệu con.
```

Date												
2010-01-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2010-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.006	0.021	NaN	NaN
2010-01-05	0.002	0.000	-0.000	0.006	0.018	0.003	0.003	-0.035	-0.003	-0.001	0.010	-0.001
2010-01-06	-0.016	-0.006	-0.003	-0.018	-0.011	0.001	0.001	-0.010	0.003	0.018	0.024	0.016
2010-01-07	-0.002	-0.010	-0.010	-0.017	0.019	0.004	0.004	-0.005	-0.007	-0.006	-0.005	-0.006

```
rets.cumsum().apply(np.exp).plot(figsize=(10, 6))
```

➔ Vẽ biểu đồ lợi nhuận log tích lũy theo thời gian; trước tiên gọi phương thức **`cumsum()`**, sau đó áp dụng **`np.exp()`** lên kết quả.

- Thống kê dạng cuộn (rolling statistics)

```
sym = 'AAPL.O'
data = pd.DataFrame(data[sym]).dropna()
data.tail()
```

Date	AAPL.O
2018-06-25	182.17
2018-06-26	184.43
2018-06-27	184.16
2018-06-28	185.50
2018-06-29	185.11

4. Thực hành - Bài 2. Vẽ biểu đồ phân tích tài chính

```
window = 20 #➔ Xác định cửa sổ; tức là số lượng giá trị chỉ mục cần bao gồm
data['min'] = data[sym].rolling(window=window).min()#➔ Tính giá trị nhỏ nhất trong cửa sổ trượt
data['mean'] = data[sym].rolling(window=window).mean()#➔ Tính giá trị trung bình trong cửa sổ trượt.
data['max'] = data[sym].rolling(window=window).max()#➔ Tính giá trị lớn nhất trong cửa sổ trượt.
data.dropna().head()
```

Date	AAPL.O	min	mean	std	median	max	ewma
2010-02-01	27.818544	27.437544	29.580892	0.933650	29.821542	30.719969	27.805432
2010-02-02	27.979972	27.437544	29.451249	0.968048	29.711113	30.719969	27.936337
2010-02-03	28.461400	27.437544	29.343035	0.950665	29.685970	30.719969	28.330134
2010-02-04	27.435687	27.435687	29.207892	1.021129	29.547113	30.719969	27.659299
2010-02-05	27.922829	27.435687	29.099892	1.037811	29.419256	30.719969	27.856947

```
ax = data[['min', 'mean', 'max']].iloc[-200:].plot(figsize=(10, 6),
                                                   style=['g--', 'r--', 'g--'],
                                                   lw=0.8)
```

#➔ Vẽ biểu đồ 3 thống kê dạng cuộn (rolling statistics) cho 200 dòng dữ liệu cuối cùng.

```
data[sym].iloc[-200:].plot(ax=ax, lw=2.0)
```

#➔ Thêm dữ liệu chuỗi thời gian gốc vào biểu đồ.

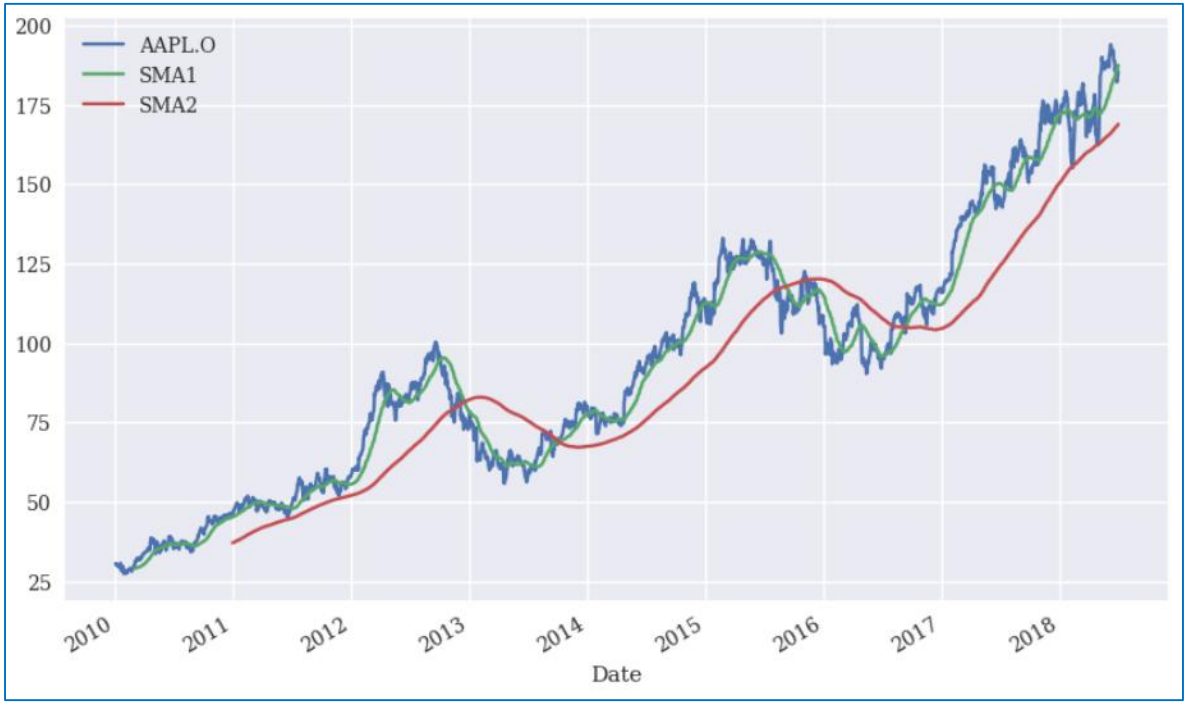
```
data['SMA1'] = data[sym].rolling(window=42).mean()
#➔ Tính toán các giá trị cho Đường Trung bình Động đơn giản (SMA) ngắn hạn
data['SMA2'] = data[sym].rolling(window=252).mean()
#➔ Tính toán các giá trị cho Đường Trung bình Động đơn giản (SMA) dài hạn
data[[sym, 'SMA1', 'SMA2']].tail()
```

Date	AAPL.O	SMA1	SMA2
2018-06-25	182.17	185.606190	168.265556
2018-06-26	184.43	186.087381	168.418770
2018-06-27	184.16	186.607381	168.579206
2018-06-28	185.50	187.089286	168.736627
2018-06-29	185.11	187.470476	168.901032

```
data[[sym, 'SMA1', 'SMA2']].plot(figsize=(10, 6))
```

#➔ Hiển thị dữ liệu giá cổ phiếu cùng với hai chuỗi thời gian SMA

#➔ Vẽ biểu đồ Giá cổ phiếu Apple và hai đường trung bình động đơn giản (SMA).



4. Thực hành - Bài 2. Vẽ biểu đồ phân tích tài chính

```
data.dropna(inplace=True)  #➔ Chỉ giữ lại các hàng dữ liệu đầy đủ.
data['positions'] = np.where(data['SMA1'] > data['SMA2'],#➔ Nếu giá trị
```

SMA ngắn hạn lớn hơn SMA dài hạn …

```
                                    1,  #➔ … mua vào cổ phiếu (put 1)
                                   -1)  #➔ Ngược lại, bán khống cổ phiếu (put a -1).

ax = data[[sym, 'SMA1', 'SMA2', 'positions']].plot(figsize=(10, 6),
                                                   secondary_y='positions')
ax.get_legend().set_bbox_to_anchor((0.25, 0.85));
```

#➔ Vẽ biểu đồ Giá cổ phiếu Apple, hai đường trung bình động đơn giản và vị thế giao dịch.

4. Thực hành - Bài 2. Vẽ biểu đồ phân tích tài chính

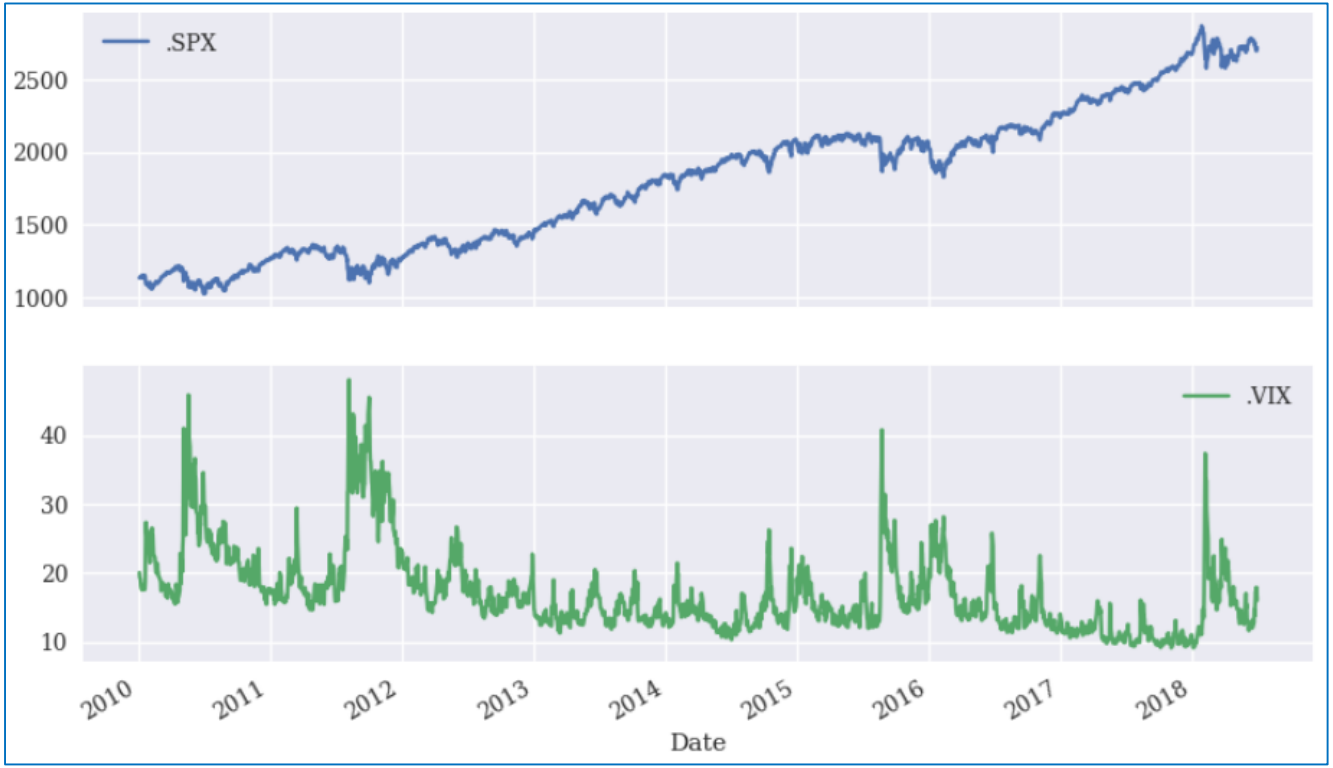
- Bộ dữ liệu hiện bao gồm hai chuỗi thời gian tài chính, cả hai đều được trực quan hóa trong Hình.

```
raw = pd.read_csv('source/tr_eikon_eod_data.csv',
                  index_col=0, parse_dates=True)

data = raw[['.SPX', '.VIX']].dropna()
data.tail()
```

Date	.SPX	.VIX
2018-06-25	2717.07	17.33
2018-06-26	2723.06	15.92
2018-06-27	2699.63	17.91
2018-06-28	2716.31	16.85
2018-06-29	2718.37	16.09

```
data.plot(subplots=True, figsize=(10, 6))
```

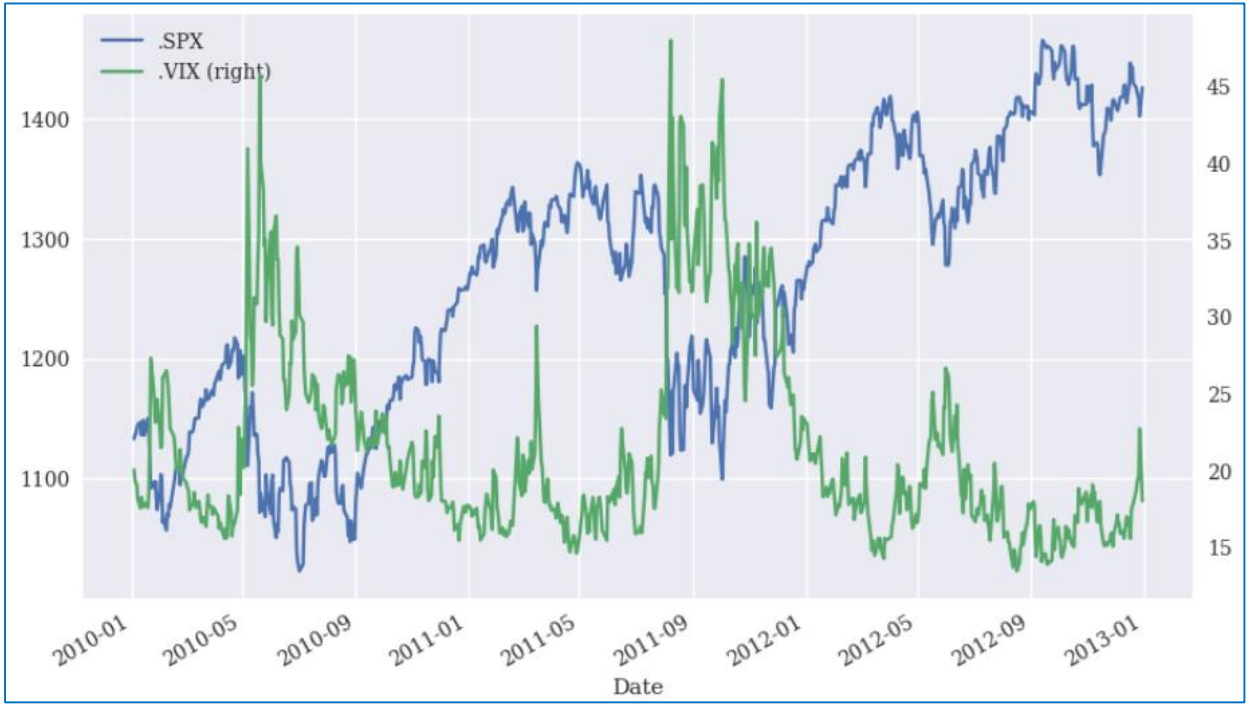
#➔ Vẽ biểu đồ dữ liệu chuỗi thời gian S&P 500 và VIX (trên các biểu đồ con khác nhau)



```
data.loc[:'2012-12-31'].plot(secondary_y='.VIX', figsize=(10, 6))
```

#➔chọn dữ liệu đến giá trị DATE được chỉ định

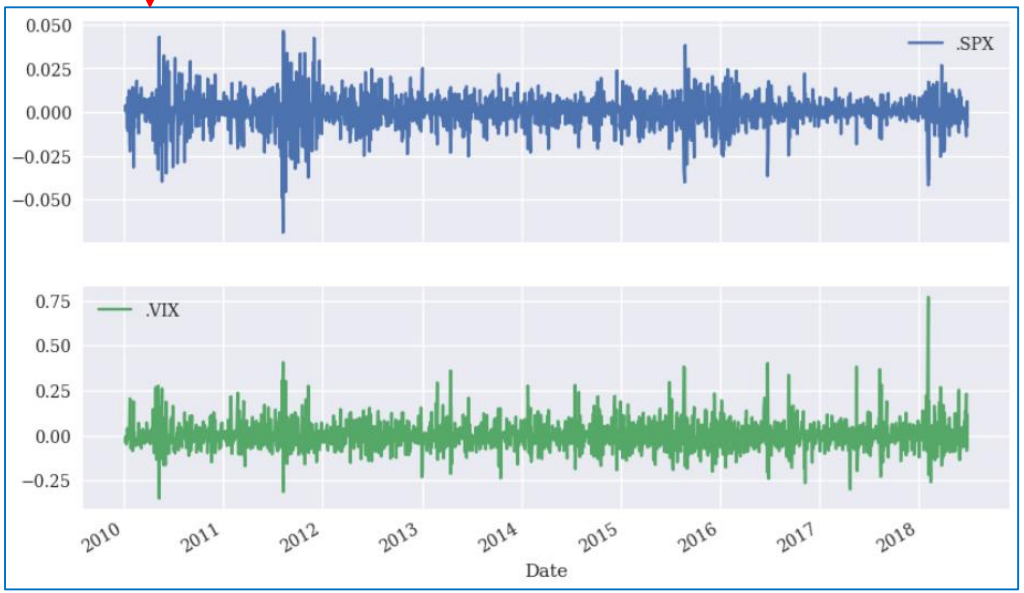
#➔ Vẽ biểu đồ dữ liệu chuỗi thời gian S&P 500 và VIX (trên cùng một biểu đồ).



4. Thực hành - Bài 3. Vẽ biểu đồ phân tích sự tương quan

```
rets = np.log(data / data.shift(1))
rets.head()
rets.dropna(inplace=True)
rets.plot(subplots=True, figsize=(10, 6))
#➔ Vẽ biểu đồ Lợi suất logarit của S&P 500 và VIX theo thời gian
```

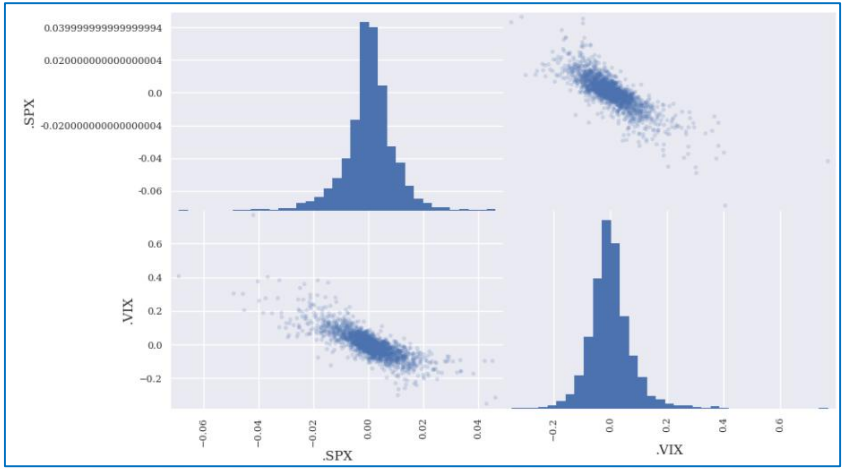
Date	.SPX	.VIX
2010-01-04	NaN	NaN
2010-01-05	0.003111	-0.035038
2010-01-06	0.000545	-0.009868
2010-01-07	0.003993	-0.005233
2010-01-08	0.002878	-0.050024



4. Thực hành - Bài 3. Vẽ biểu đồ phân tích sự tương quan

```
pd.plotting.scatter_matrix(rets,  #➔Dữ liệu để vẽ biểu đồ.
                           alpha=0.2,#➔ Tham số alpha để điều chỉnh độ mờ của các điểm.
                           diagonal='hist',#➔  Một biểu đồ tần suất (histogram)
                                               của dữ liệu trong cột.
                           hist_kwds={'bins': 35},#➔  Các từ khóa (keywords) được
                                                      truyền vào hàm vẽ
                                                      biểu đồ histogram
                           figsize=(10, 6))
```

#➔ Vẽ biểu đồ lợi suất logarit của S&P 500 và VIX dưới dạng ma trận phân tán (scatter matrix).

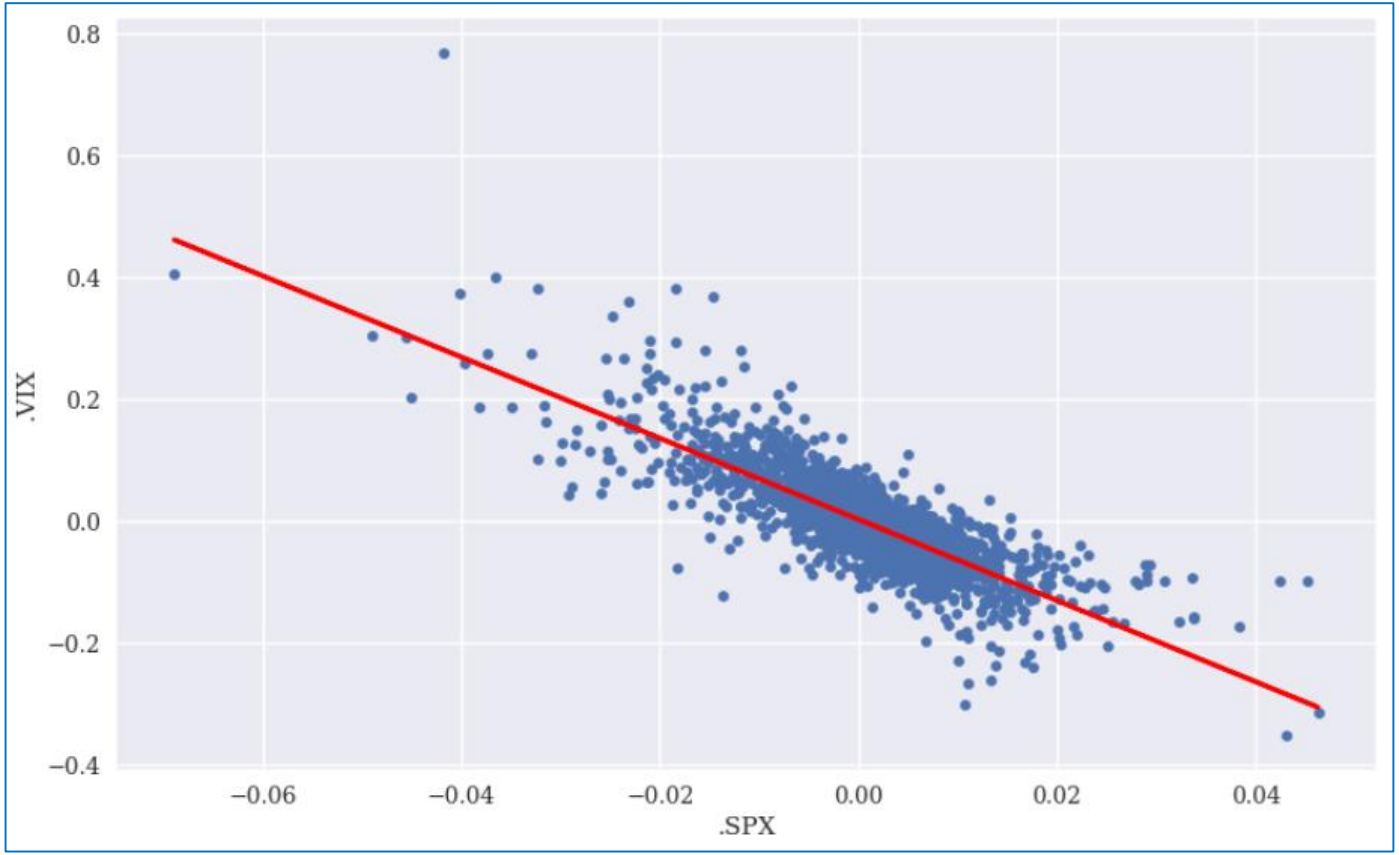


4. Thực hành - Bài 3. Vẽ biểu đồ phân tích sự tương quan

- Thực hiện hồi quy OLS (Ordinary Least Squares).
- Biểu đồ phân tán của lợi suất logarit kèm đường hồi quy cho thấy độ dốc âm rõ ràng, chứng tỏ hai chỉ số này có mối tương quan âm.

```
reg = np.polyfit(rets['.SPX'], rets['.VIX'], deg=1)
#➔ thực hiện hồi quy tuyến tính OLS
ax = rets.plot(kind='scatter', x='.SPX', y='.VIX', figsize=(10, 6))
#➔ vẽ lợi suất logarit dưới dạng biểu đồ phân tán
ax.plot(rets['.SPX'], np.polyval(reg, rets['.SPX']), 'r', lw=2)
#➔Đường hồi quy tuyến tính được thêm vào
#➔Vẽ biểu đồ lợi suất logarit của S&P 500 và VIX dưới dạng ma trận phân tán
```

4. Thực hành - Bài 3. Vẽ biểu đồ phân tích sự tương quan



- Tương quan
  - Văn bản giải thích rằng tương quan được đo theo hai cách: phép đo tĩnh sử dụng toàn bộ tập dữ liệu và phép đo trượt trong một khoảng thời gian cố định.
  - Tương quan trượt thay đổi theo thời gian nhưng vẫn duy trì giá trị âm, xác nhận mối quan hệ nghịch mạnh mẽ giữa hai chỉ số S&P 500 và VIX.

`rets.corr()` # ➔ Ma trận tương quan cho toàn bộ DataFrame.

	.SPX	.VIX
.SPX	1.000000	-0.804382
.VIX	-0.804382	1.000000

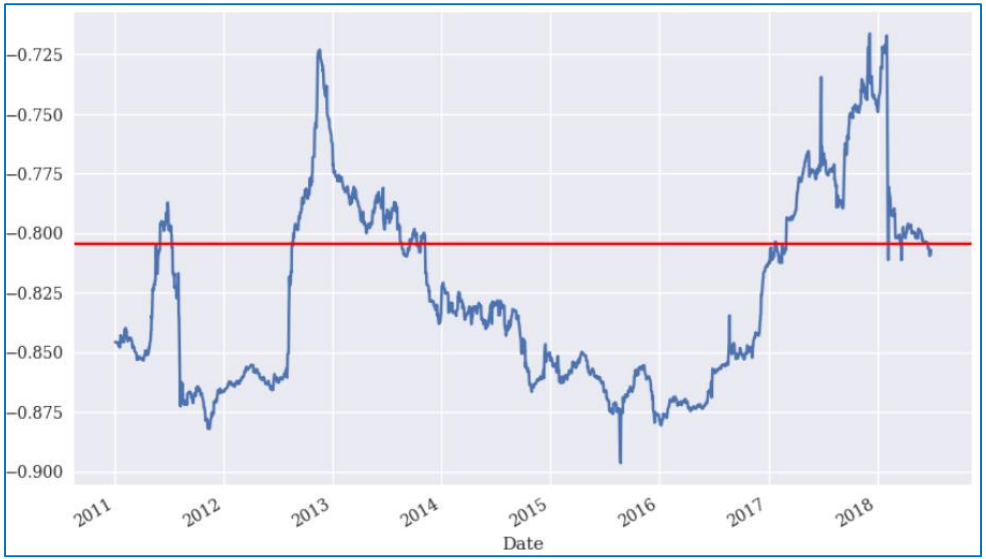
```
ax = rets['.SPX'].rolling(window=252).corr(
                                    rets['.VIX']).plot(figsize=(10, 6))
```

#➔ Vẽ biểu đồ tương quan theo thời gian

```
ax.axhline(rets.corr().iloc[0, 1], c='r')
```

#➔ và thêm giá trị tĩnh vào biểu đồ dưới dạng đường ngang.

#➔Vẽ biểu đồ tương quan giữa S&P 500 và VIX (cố định và theo cuộn thời gian)



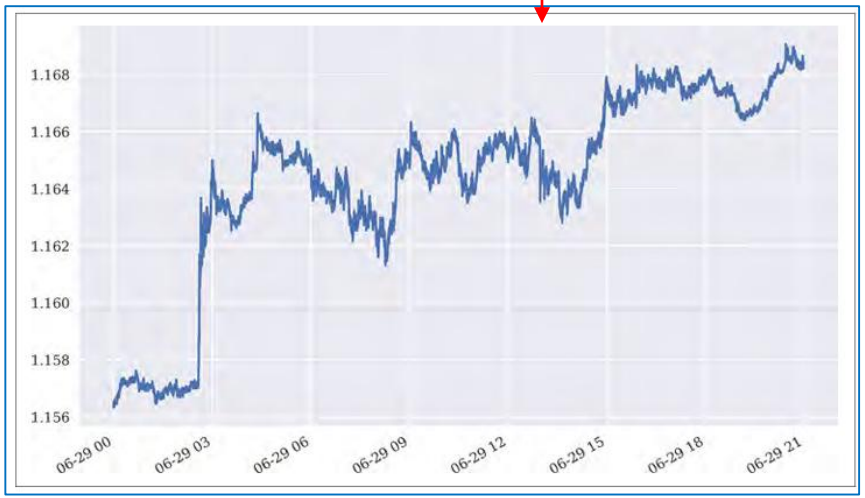
```
tick = pd.read_csv('source/fxcm_eur_usd_tick_data.csv',
                   index_col=0, parse_dates=True)
tick.info()
tick['Mid'] = tick.mean(axis=1)
```

➔ Tính giá trị trung bình của mỗi hàng.

```
tick['Mid'].plot(figsize=(10, 6))
```

➔ Vẽ biểu đồ dữ liệu tick cho tỷ giá hối đoái EUR/USD

```
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Bid     461357 non-null  float64
 1   Ask     461357 non-null  float64
dtypes: float64(2)
memory usage: 10.6 MB
```



```
tick_resam = tick.resample(rule='5min',
                           label='right').last()
tick_resam.head()
tick_resam['Mid'].plot(figsize=(10, 6))
```

➔ Vẽ biểu đồ dữ liệu nến 5 phút cho tỷ giá hối đoái EUR/USD.

	Bid	Ask	Mid
2018-06-29 00:05:00	1.15649	1.15651	1.156500
2018-06-29 00:10:00	1.15671	1.15672	1.156715
2018-06-29 00:15:00	1.15725	1.15727	1.157260
2018-06-29 00:20:00	1.15720	1.15722	1.157210
2018-06-29 00:25:00	1.15711	1.15712	1.157115

The end of Chapter