1. Introduction to Matplotlib

- **Matplotlib** is the fundamental Python library for creating plots and visualizations.
- Useful for visualizing financial data, time series, distributions, etc.
- Two main approaches:
  1. **Pyplot interface** → simple and quick, MATLAB-like.
  2. **Object-oriented (OO) interface** → more flexible for complex figures.

Example:

```python
import matplotlib.pyplot as plt

plt.plot([1,2,3,4], [10,20,25,30])
plt.title("Simple Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output:

- A simple line plot with axes and title.

2. Plot Types

Type	Description	Example
Line	Time series or sequential data	`plt.plot()`
Scatter	Relationship between two variables	`plt.scatter()`
Bar	Compare categories	`plt.bar()`
Histogram	Distribution of values	`plt.hist()`
Boxplot	Summary statistics	`plt.boxplot()`

3. Customization

- **Colors, markers, line styles:**

```python
plt.plot(x, y, color="red", linestyle="--", marker="o")
```

- **Legend:**

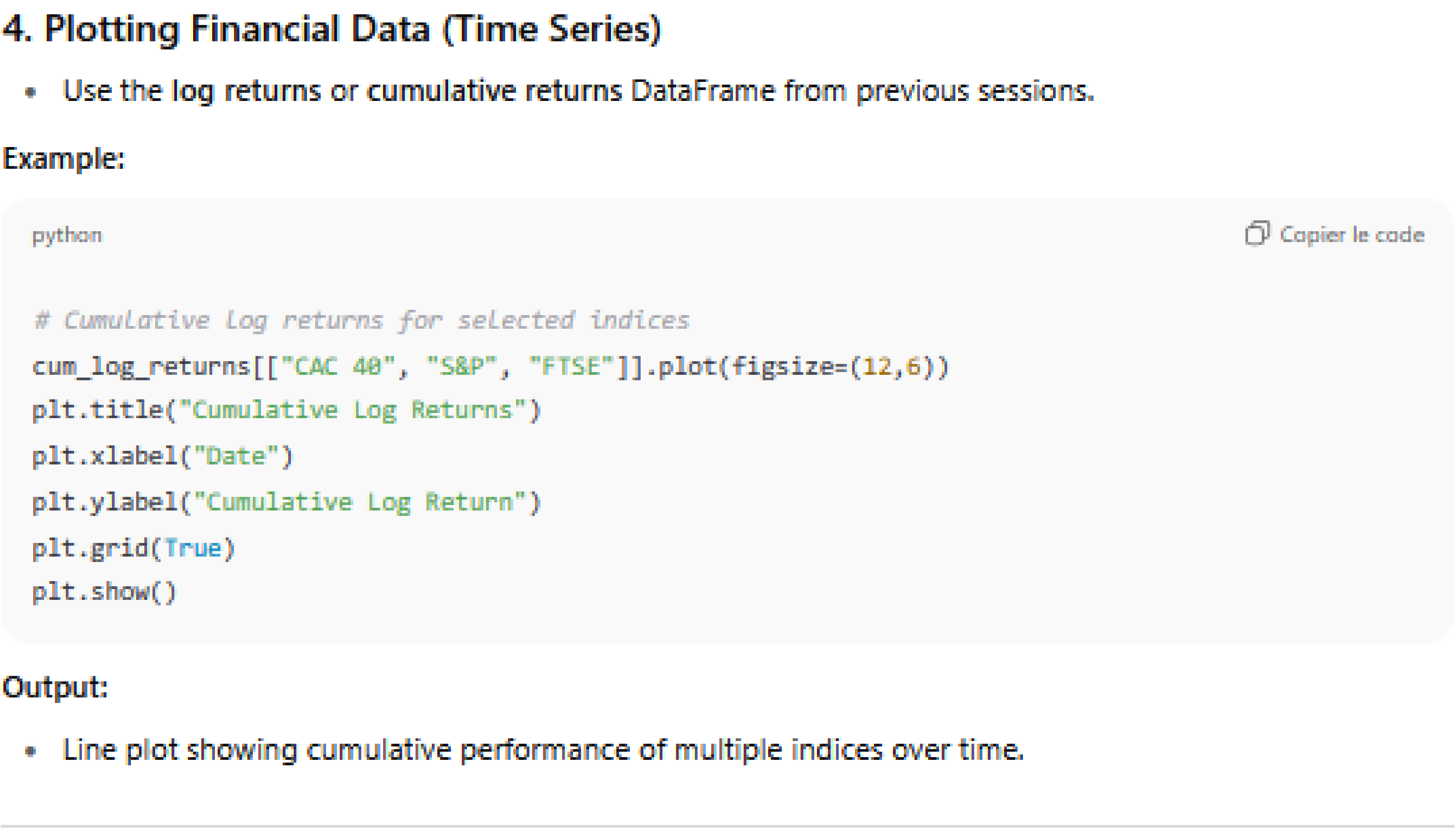
```python
plt.legend(["Asset 1"])
```

- **Grid:**

```python
plt.grid(True)
```

- **Figure size:**

```python
plt.figure(figsize=(10,6))
```



4. Plotting Financial Data (Time Series)

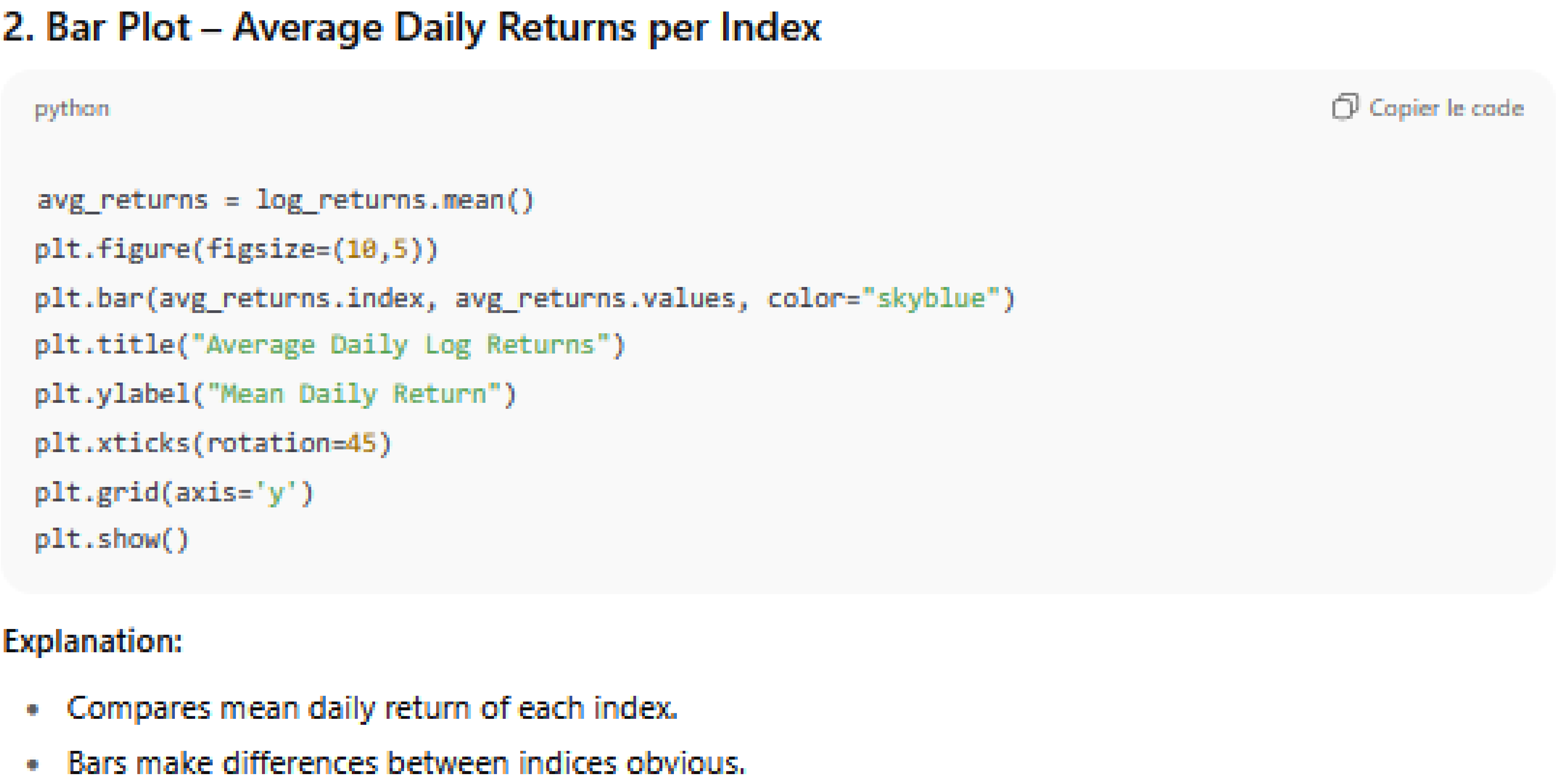
- Use the log returns or cumulative returns DataFrame from previous sessions.

Example:

```python
# Cumulative log returns for selected indices
cum_log_returns[["CAC 40", "S&P", "FTSE"]].plot(figsize=(12,6))
plt.title("Cumulative Log Returns")
plt.xlabel("Date")
plt.ylabel("Cumulative Log Return")
plt.grid(True)
plt.show()
```

Output:

- Line plot showing cumulative performance of multiple indices over time.

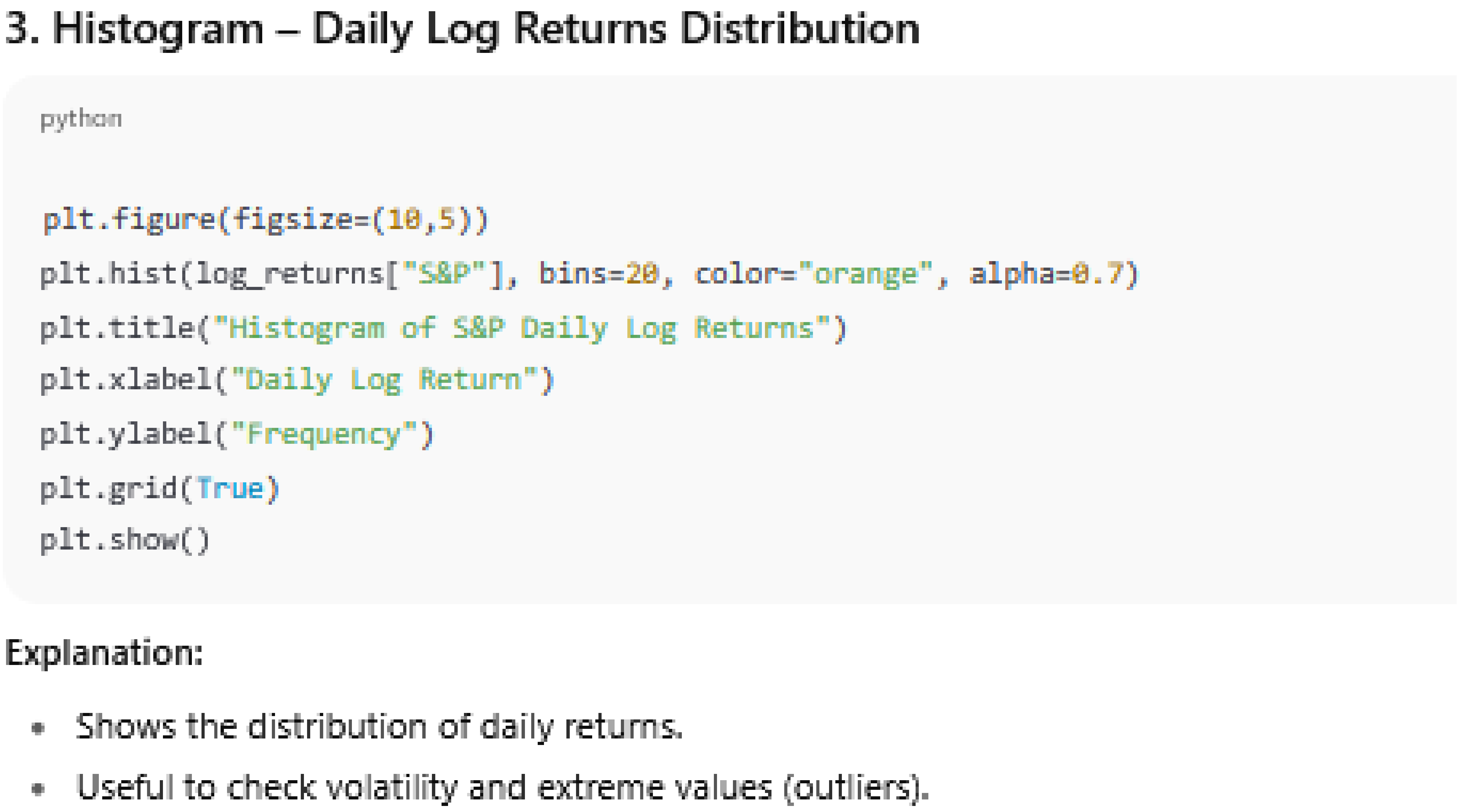


2. Bar Plot – Average Daily Returns per Index

```python
avg_returns = log_returns.mean()
plt.figure(figsize=(10,5))
plt.bar(avg_returns.index, avg_returns.values, color="skyblue")
plt.title("Average Daily Log Returns")
plt.ylabel("Mean Daily Return")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

Explanation:

- Compares mean daily return of each index.
- Bars make differences between indices obvious.

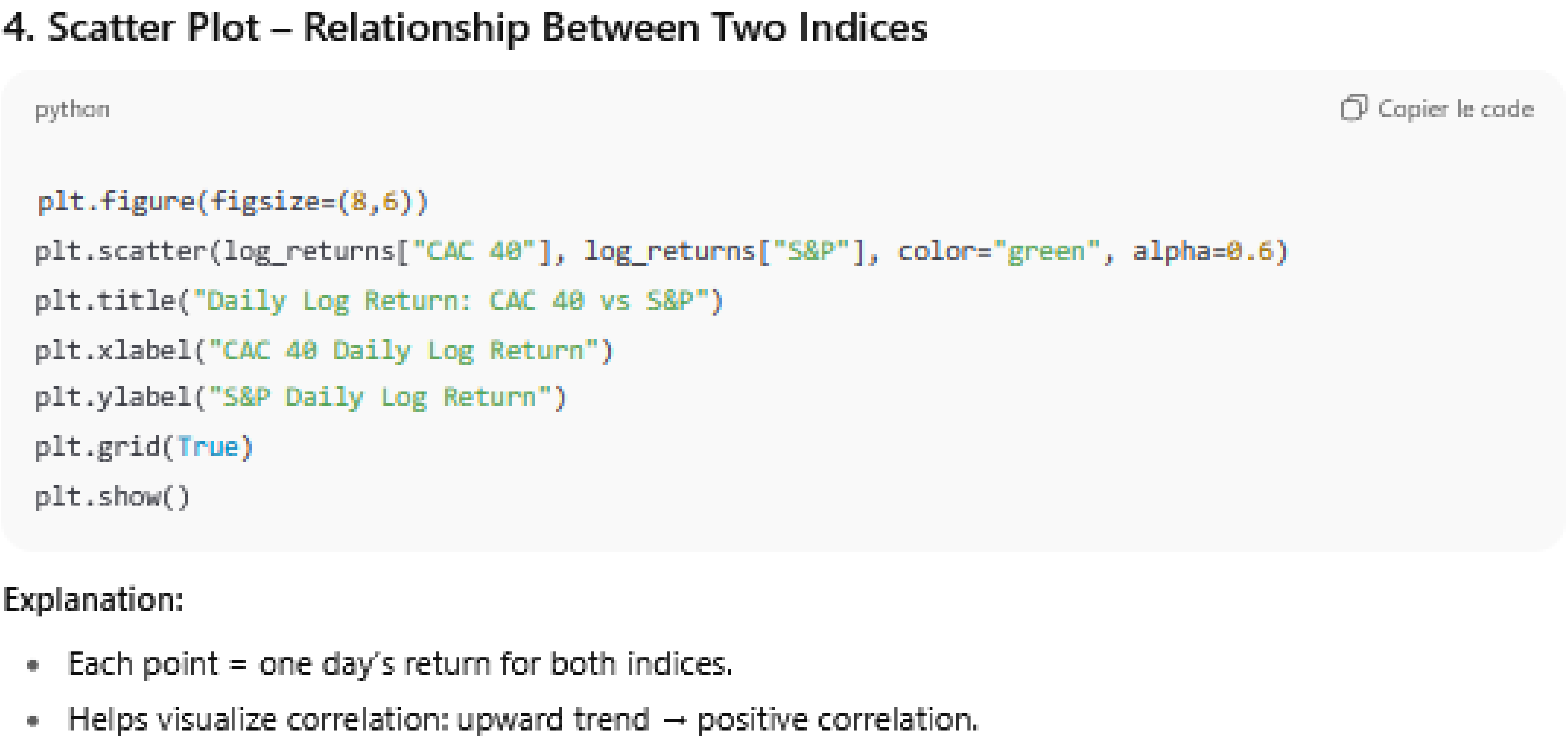


3. Histogram – Daily Log Returns Distribution

```python
plt.figure(figsize=(10,5))
plt.hist(log_returns["S&P"], bins=20, color="orange", alpha=0.7)
plt.title("Histogram of S&P Daily Log Returns")
plt.xlabel("Daily Log Return")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```

Explanation:

- Shows the distribution of daily returns.
- Useful to check volatility and extreme values (outliers).



4. Scatter Plot – Relationship Between Two Indices

```python
plt.figure(figsize=(8,6))
plt.scatter(log_returns["CAC 40"], log_returns["S&P"], color="green", alpha=0.6)
plt.title("Daily Log Return: CAC 40 vs S&P")
plt.xlabel("CAC 40 Daily Log Return")
plt.ylabel("S&P Daily Log Return")
plt.grid(True)
plt.show()
```

Explanation:

- Each point = one day's return for both indices.
- Helps visualize correlation: upward trend → positive correlation.

Exercise – Matplotlib Visualization of Financial Data

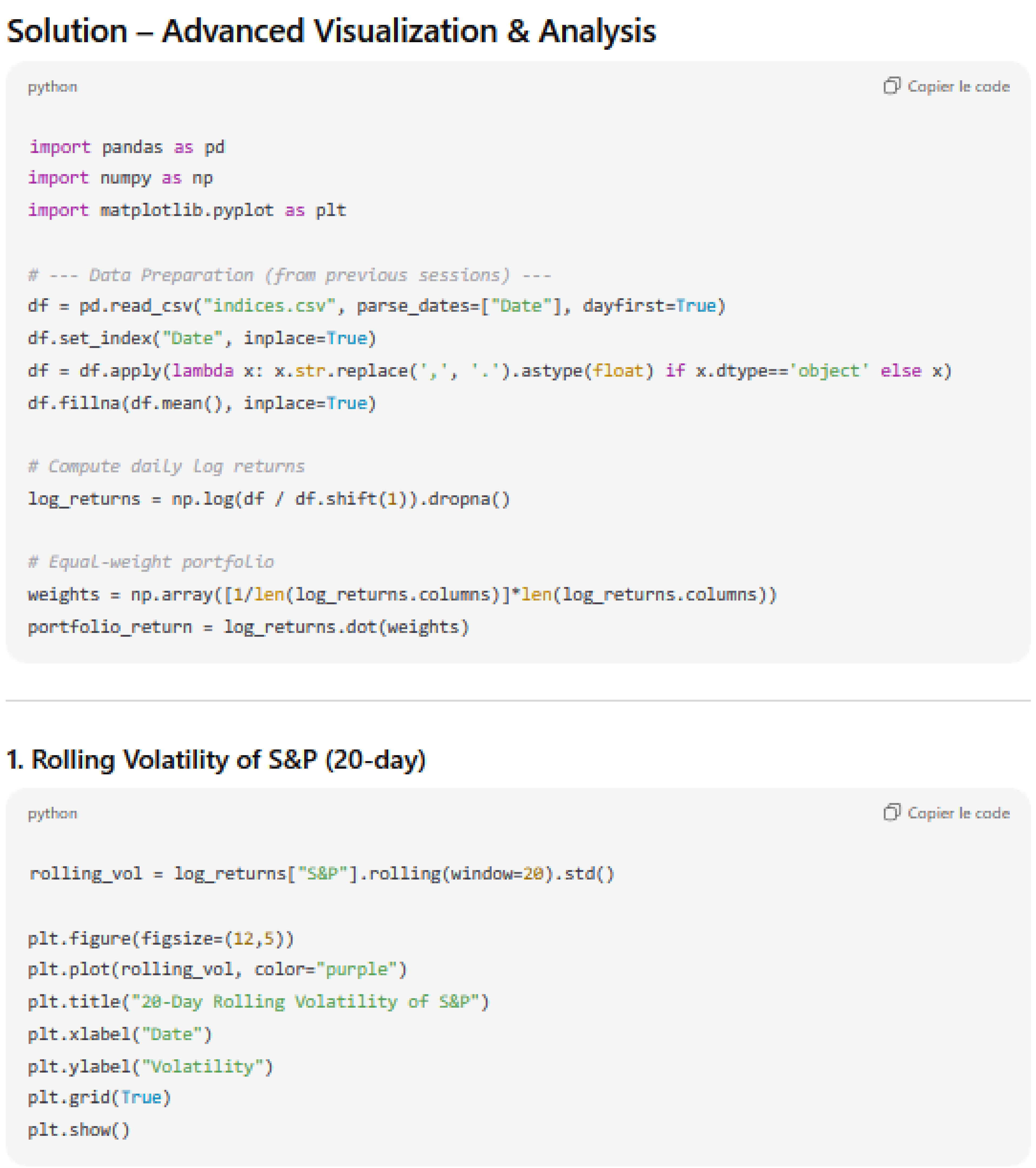
Context: Use the **log returns** and **cumulative returns** DataFrame from the previous sessions (`CAC 40`, `S&P`, `FTSE`, etc.).

Tasks:

1. **Line Plot:**
   - Plot **cumulative log returns** of `CAC 40` and `S&P` on the same figure.
   - **Add title, axis labels, legend, and grid.**
2. **Bar Plot:**
   - Compute **average daily log return** for all indices.
   - Plot a **bar chart** showing these mean returns.
3. **Histogram:**
   - Plot the **histogram of daily log returns** for `S&P`.
   - Use 20 bins and add gridlines.
4. **Scatter Plot:**
   - Plot a **scatter plot** of daily log returns between `CAC 40` (x-axis) and `S&P` (y-axis).
   - Add gridlines and axis labels.
5. **Optional Challenge:**
   - Plot the **portfolio cumulative returns** (equal-weight portfolio) alongside `CAC 40` and `S&P`.
   - Compare visually which line is smoother (diversification effect).

> 💡 **Hints:**

- Use `.plot()` for line plots, `plt.bar()` for bar charts, `plt.hist()` for histograms, and `plt.scatter()` for scatter plots.
- Customize with `title`, `xlabel`, `ylabel`, `legend()`, `grid()`, `figsize=(..)`



Solution – Advanced Visualization & Analysis

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# --- Data Preparation (from previous sessions) ---
df = pd.read_csv("indices.csv", parse_dates=["Date"], dayfirst=True)
df.set_index("Date", inplace=True)
df = df.apply(lambda x: x.str.replace(',', '.').astype(float) if x.dtype=='object' else x)
df.fillna(df.mean(), inplace=True)

# Compute daily log returns
log_returns = np.log(df / df.shift(1)).dropna()

# Equal-weight portfolio
weights = np.array([1/len(log_returns.columns)]*len(log_returns.columns))
portfolio_return = log_returns.dot(weights)
```

1. Rolling Volatility of S&P (20-day)

```python
rolling_vol = log_returns["S&P"].rolling(window=20).std()

plt.figure(figsize=(12,5))
plt.plot(rolling_vol, color="purple")
plt.title("20-Day Rolling Volatility of S&P")
plt.xlabel("Date")
plt.ylabel("Volatility")
plt.grid(True)
plt.show()
```

2. Top 3 Volatile Days – CAC 40

python Copier le code

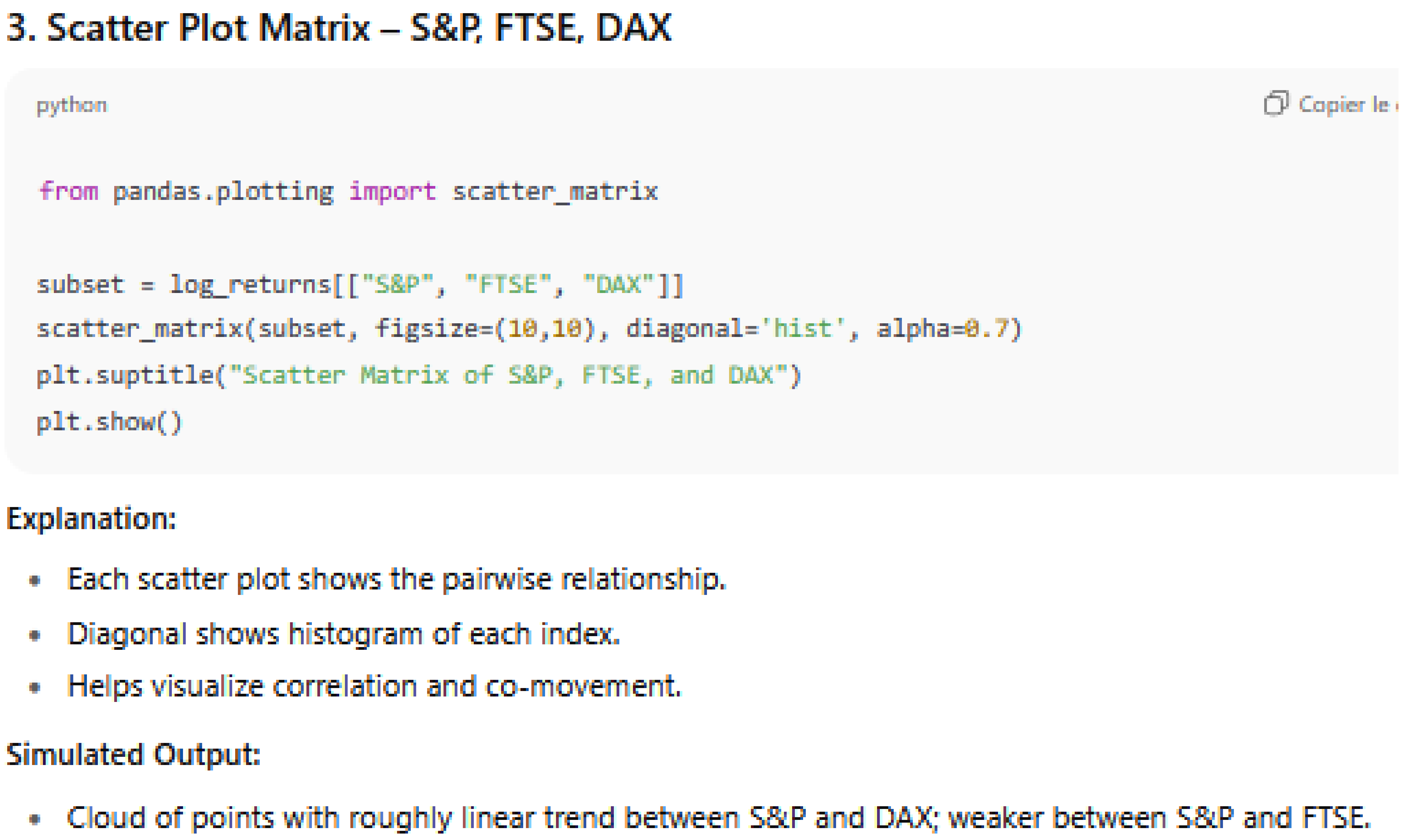
```python
top3_cac = log_returns["CAC 40"].abs().nlargest(3)

plt.figure(figsize=(8,5))
plt.bar(top3_cac.index.astype(str), top3_cac.values, color="red")
plt.title("Top 3 Most Volatile Days for CAC 40")
plt.ylabel("Absolute Daily Log Return")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

Simulated Output:

makefile Copier le code

```makefile
2016-11-02: 0.0345
2016-11-10: 0.0312
2016-11-01: 0.0289
```

- Bar chart with 3 tallest bars corresponding to these dates.



3. Scatter Plot Matrix – S&P, FTSE, DAX

```python
from pandas.plotting import scatter_matrix

subset = log_returns[["S&P", "FTSE", "DAX"]]
scatter_matrix(subset, figsize=(10,10), diagonal='hist', alpha=0.7)
plt.suptitle("Scatter Matrix of S&P, FTSE, and DAX")
plt.show()
```

Explanation:

- Each scatter plot shows the pairwise relationship.
- Diagonal shows histogram of each index.
- Helps visualize correlation and co-movement.

Simulated Output:

- Cloud of points with roughly linear trend between S&P and DAX; weaker between S&P and FTSE.

4. Portfolio vs. S&P Scatter

python

Copier le code

```python
plt.figure(figsize=(8,6))
plt.scatter(log_returns["S&P"], portfolio_return, color="green", alpha=0.6)
plt.title("Equal-Weight Portfolio vs. S&P Daily Log Returns")
plt.xlabel("S&P Daily Log Return")
plt.ylabel("Portfolio Daily Log Return")
plt.grid(True)
plt.show()
```

Explanation:

- Scatter shows portfolio return vs. S&P return.
- Points clustered along a diagonal → positive correlation.
- Smoother cloud than individual S&P → diversification effect.

Simulated Output:

- Scatter points forming a narrower upward-sloping cloud than S&P alone.