MATPLOTLIB

Library Overview

Matplotlib is a foundational Python library for creating static, animated, and interactive visualizations. It offers extensive control over every element of a plot, making it suitable for publication-quality figures and complex custom visualizations. Matplotlib is widely used for its flexibility, broad range of supported plot types, and ability to embed plots into various interfaces and applications. It also serves as the backbone for many higher-level libraries like Seaborn and Pandas plotting functions

Typical Use Cases:

- Scientific and engineering plots
- Custom, highly detailed visualizations
- Embedding plots in GUIs or web apps

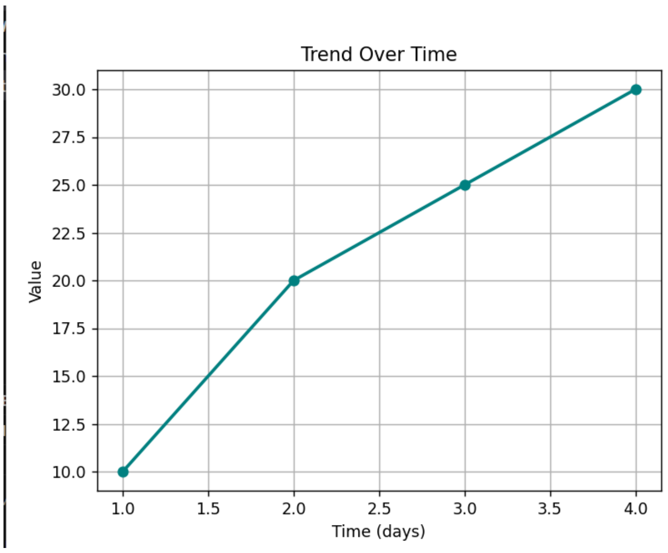
Things to be install before executing all this concept

```
pip install matplotlib

pip install numpy
```

Graph Types:

1. Line Graphs:
   Connects data points with straight lines, ideal for showing trends over time or ordered categories

   Use Case:
   Visualizing stock prices, temperature changes, or website visits over days



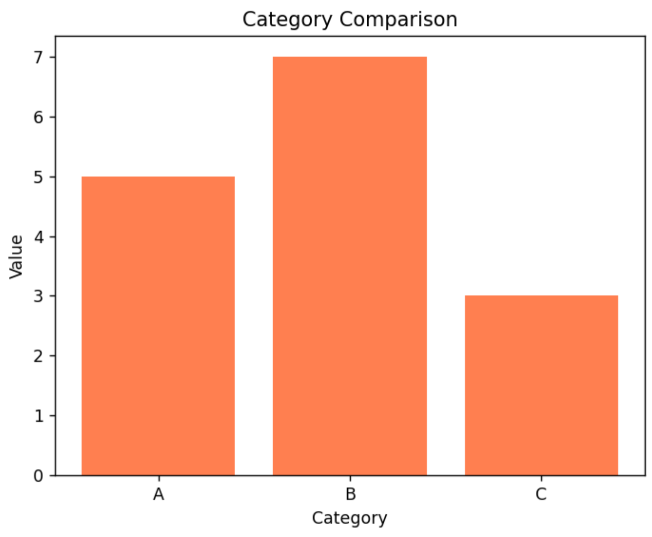
Code Snippets :

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y, marker='o', color='teal', linewidth=2)
plt.xlabel('Time (days)')
plt.ylabel('Value')
plt.title('Trend Over Time')
plt.grid(True)
plt.show()
```

2. Bar Chart:

Displays quantities for different categories using rectangular bars

Use Case:
Comparing sales across products, survey responses, or population by region.



Code Snippets:

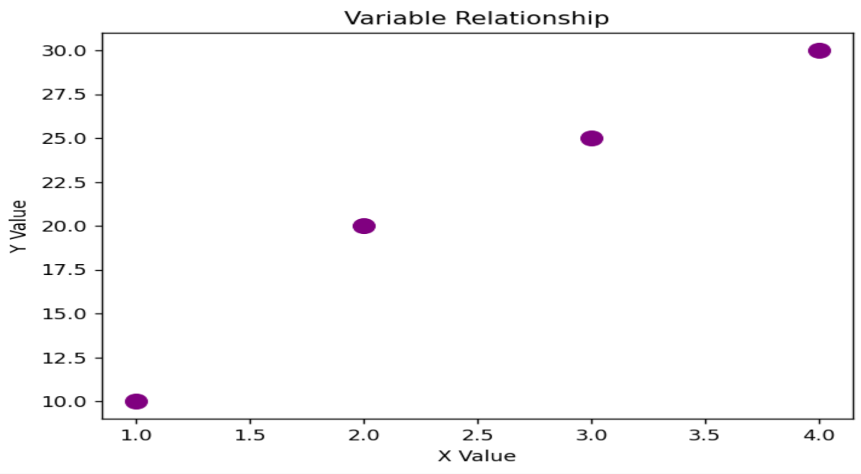
```
import matplotlib.pyplot as plt
categories = ['A', 'B', 'C']
values = [5, 7, 3]
plt.bar(categories, values, color='coral')
plt.title('Category Comparison')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```

3. Scatter Plot:

Plots individual data points to reveal relationships or patterns between two variables.

Use Case:

Examining the correlation between height and weight, or study hours and test scores.



Code Snippets:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.scatter(x, y, color='purple', s=100)
plt.title('Variable Relationship')
plt.xlabel('X Value')
plt.ylabel('Y Value')
plt.show()
```

4. Histogram

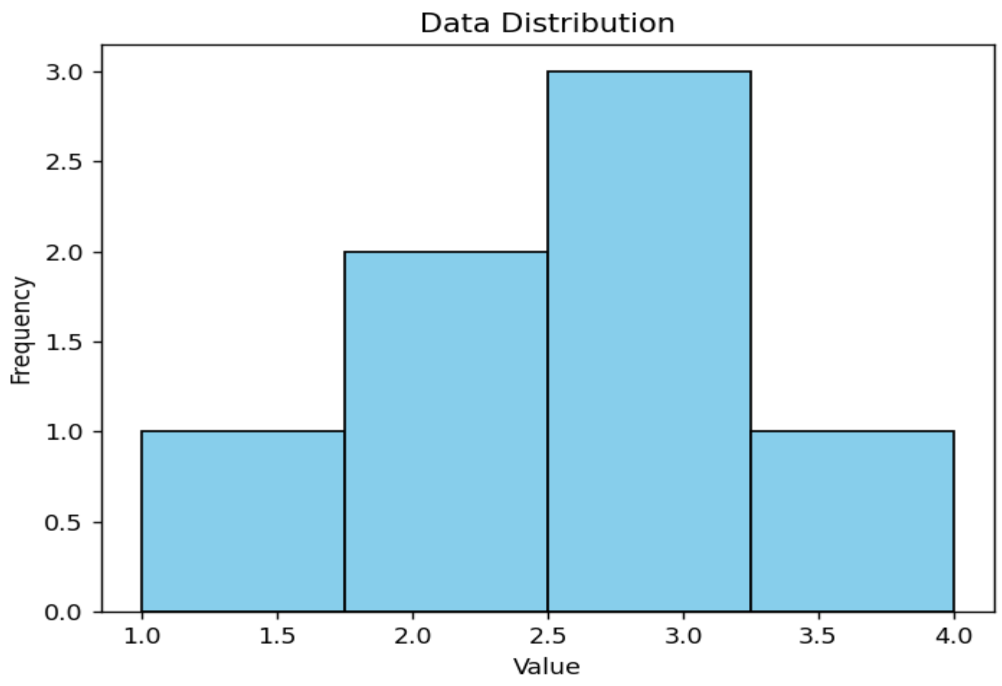
Shows the distribution of a dataset by grouping values into bins.

Use Case:

Analysing exam scores, age distribution, or frequency of measurements.

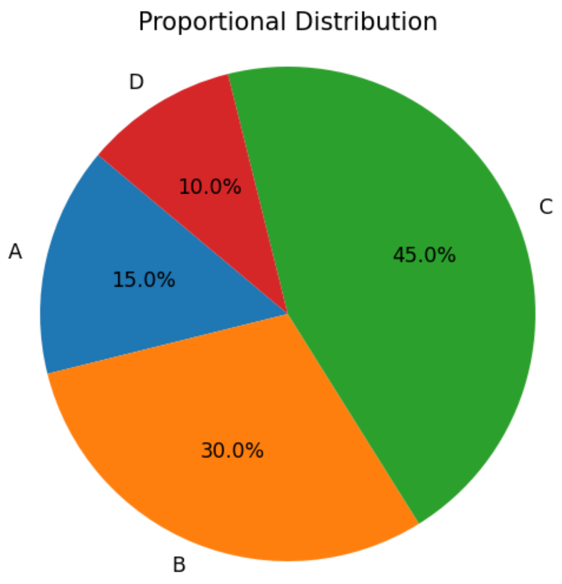
Code Snippets: .

```
import matplotlib.pyplot as plt

data = [1, 2, 2, 3, 3, 3, 4]

plt.hist(data, bins=4, color='skyblue', edgecolor='black')
plt.title('Data Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



5. **Pie Chart:**
Displays proportions of a whole as slices of a circle.

Use Case:
Visualizing market share, budget allocation, or survey results.



Code Snippets:

```
import matplotlib.pyplot as plt

sizes = [15, 30, 45, 10]
labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Proportional Distribution')
plt.axis('equal')
plt.show()
```

Pandas

Library Overview:

Pandas is primarily a data analysis library, but it provides built-in plotting capabilities that leverage Matplotlib under the hood. Its plotting API is designed for quick, convenient visualization of DataFrames and Series, making it ideal for exploratory data analysis. While less customizable than Matplotlib, Pandas plots are concise and integrate seamlessly with data manipulation workflows.

Typical Use Cases:

- Rapid data exploration and visualization
- Plotting directly from DataFrames/Series
- Visual summaries of tabular data

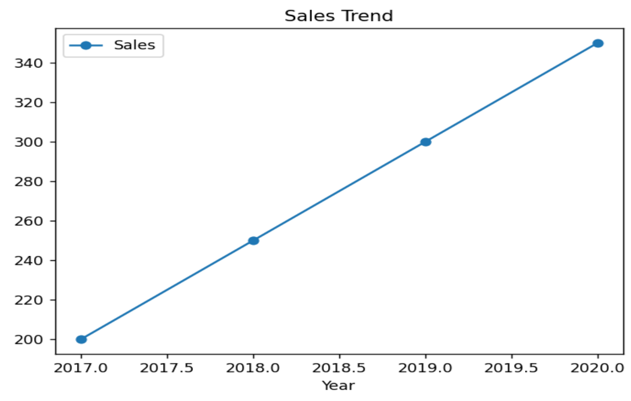
Note: Pandas plotting is built on top of Matplotlib, offering quick plotting methods directly from DataFrames and Series. Common plot types include:

1. **Line Plot:**

Automatically connects DataFrame or Series values with lines—perfect for trend analysis.

Use Case:

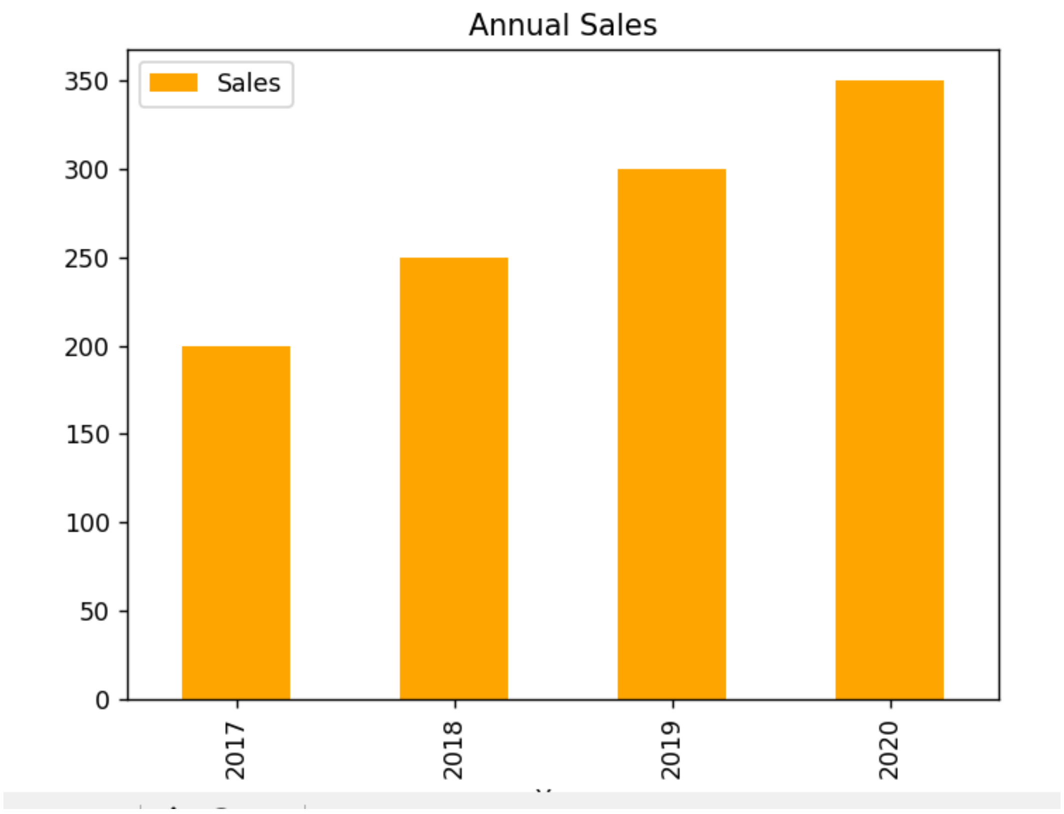
Tracking company revenue or temperature over time.



Code Snippets:

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

df = pd.DataFrame({
    'Year': [2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350]
})

df.plot(x='Year', y='Sales', kind='line', marker='o', title='Sales Trend')

plt.show()
```

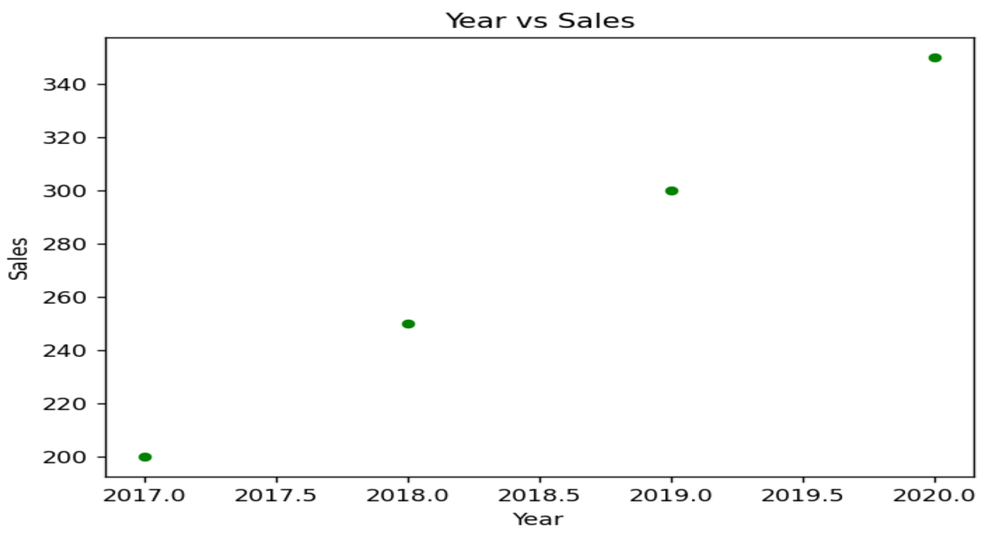
2. Bar Chart:



Code Snippets:

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

df = pd.DataFrame({
    'Year': [2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350]
})

df.plot(x='Year', y='Sales', kind='bar', color='orange', title='Annual Sales')


plt.show()
```

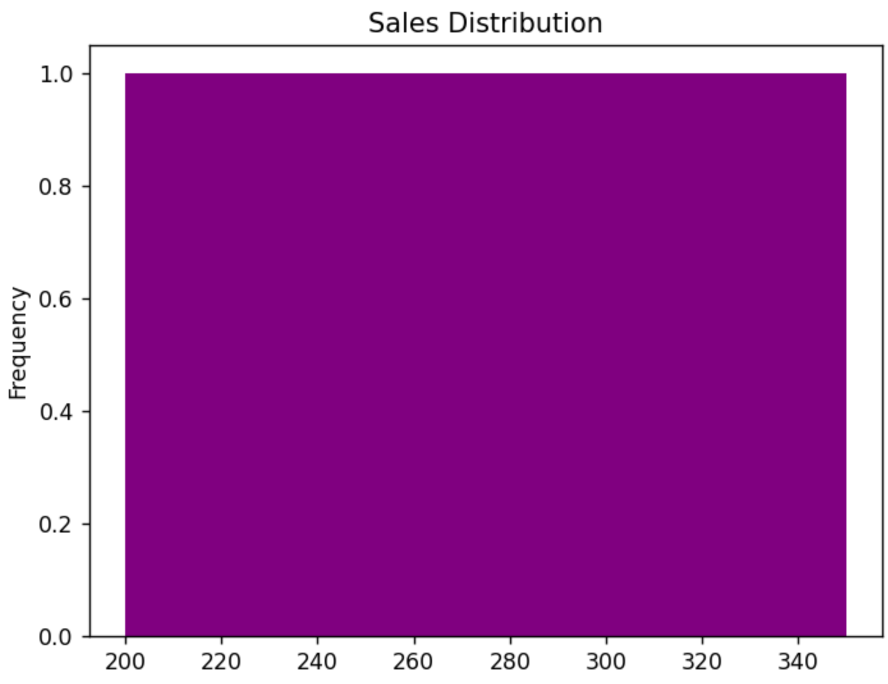
3. Scatter Plot



Code Snippets:

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

df = pd.DataFrame({
    'Year': [2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350]
})

df.plot.scatter(x='Year', y='Sales', color='green', title='Year vs Sales')

plt.show()
```

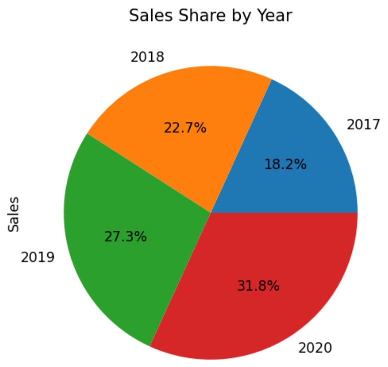
4. Histogram



Code Snippets:

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

df = pd.DataFrame({
    'Year': [2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350]
})

df['Sales'].plot(kind='hist', bins=4, color='purple', title='Sales Distribution')

plt.show()
```

5. Pie Chart:



Code Snippets:

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

df = pd.DataFrame({
    'Year': [2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350]
})

df['Sales'].plot(kind='pie', labels=df['Year'], autopct='%1.1f%%', title='Sales Share by Year')

plt.show()
```

Comparison:

Feature	Matplotlib	Pandas
Ease of Use	Requires more setup, but highly flexible	Extremely easy—one line from DataFrame/Series
Customization	Full control over every plot detail	Basic options; advanced tweaks use Matplotlib
Interactivity	Supports interactive and animated plots	Basic interactivity; relies on Matplotlib
Performance	Efficient for large and complex datasets	Good for moderate data; large data may be slower
Best For	Publication-quality, scientific, custom plots	Fast EDA, quick insights, inline in notebooks
Integration	Works with GUIs, web apps, and other libraries	Seamless with Pandas data manipulation

Summary:

- **Matplotlib** is best for highly customized, publication-ready, or complex visualizations and when fine control over every plot element is needed.
- **Pandas** is ideal for quick, convenient plotting directly from data structures during data analysis, with minimal code.

Both libraries are often used together: Pandas for rapid exploration and Matplotlib for final, detailed visualizations