VISUALIZATION

1.INTRODUCTION

Data visualization is an important component of data science. Data visualization enables us to understand data in a visual presentation. Though there are many other visualization libraries available in Python, the first two most popular and easiest to learn are Pandas Visualization and Matplotlib. Both libraries make creating many Different types of charts and graphs easy

2.PANDAS

Overview

Pandas is one of the most popular Python libraries for data handling and analysis. It provides powerful tools to organize, clean, and process data efficiently using data structures like **Series** (1D) and **DataFrame** (2D). Pandas also has built-in visualization features that allow users to create simple plots directly from data, making it quick and easy to understand datasets without switching to another library.

Key Features:

1. **Easy Plotting:** Create line, bar, histogram, pie, box, and area plots directly using .plot() function.
2. **Built-in with Matplotlib:** Uses Matplotlib in the background for visualization, so plots are smooth and flexible.
3. **Works with DataFrames:** You can visualize data directly from rows and columns — no extra setup required.
4. **Quick Insights:** Helps beginners quickly understand data trends and distributions.
5. **Customizable:** Supports styling, colors, labels, and legends to make plots more informative.

2.1 Line Plot

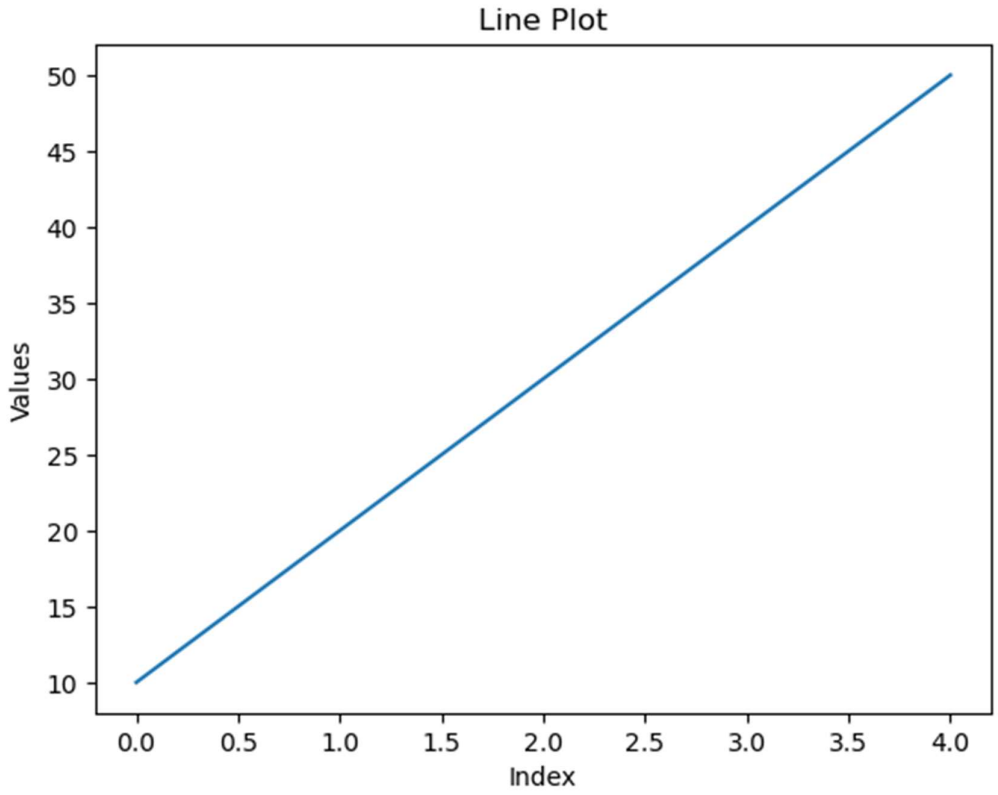
A line plot shows trends over time or continuous data

Code:

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

data = pd.Series([10, 20, 30, 40, 50])
data.plot(kind='line')
plt.title("Line Plot")
plt.xlabel("Index")
plt.ylabel("Values")
plt.show()
```

Output:



2.2 Bar Plot

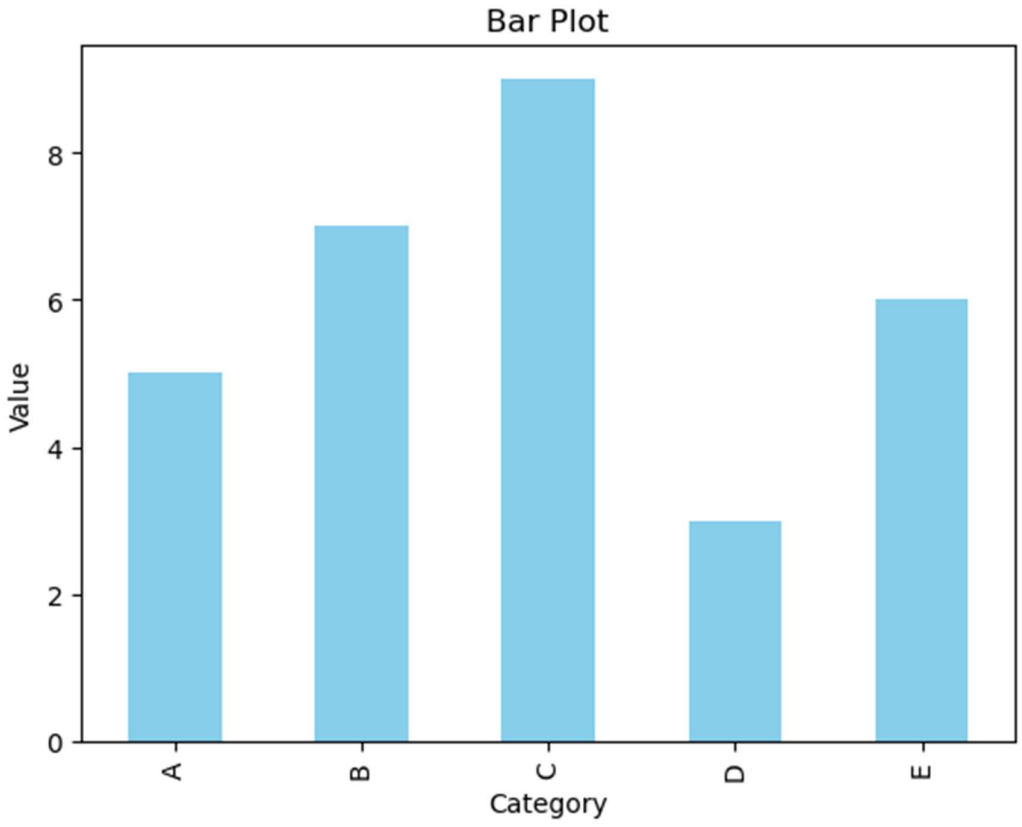
Bar plots represent categorical data with rectangular bars.

Code:

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

data = pd.Series([5, 7, 9, 3, 6], index=['A', 'B', 'C', 'D', 'E'])
data.plot(kind='bar', color='skyblue')
plt.title("Bar Plot")
plt.xlabel("Category")
plt.ylabel("Value")
plt.show()
```

Output:



2.3 Histogram

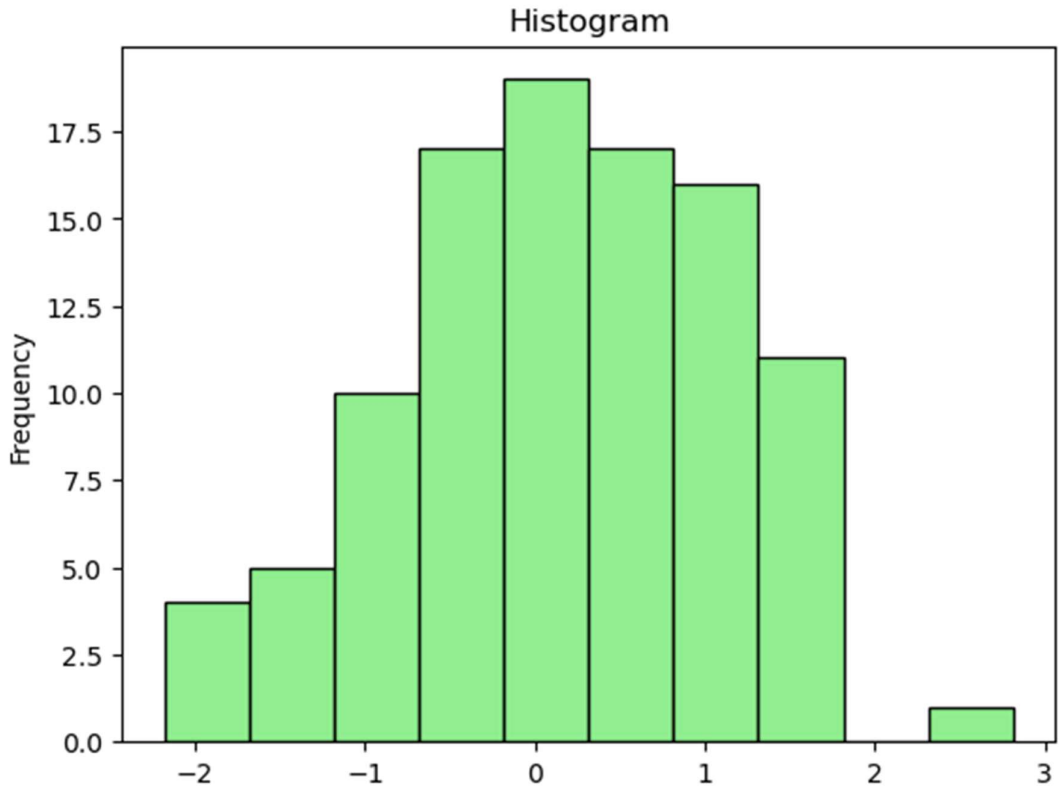
A histogram displays the frequency distribution of numeric data.

Code:

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

data = pd.Series(np.random.randn(100))
data.plot(kind='hist', bins=10, color='lightgreen', edgecolor='black')
plt.title("Histogram")
plt.show()
```

Output :



2.4 Pie Chart

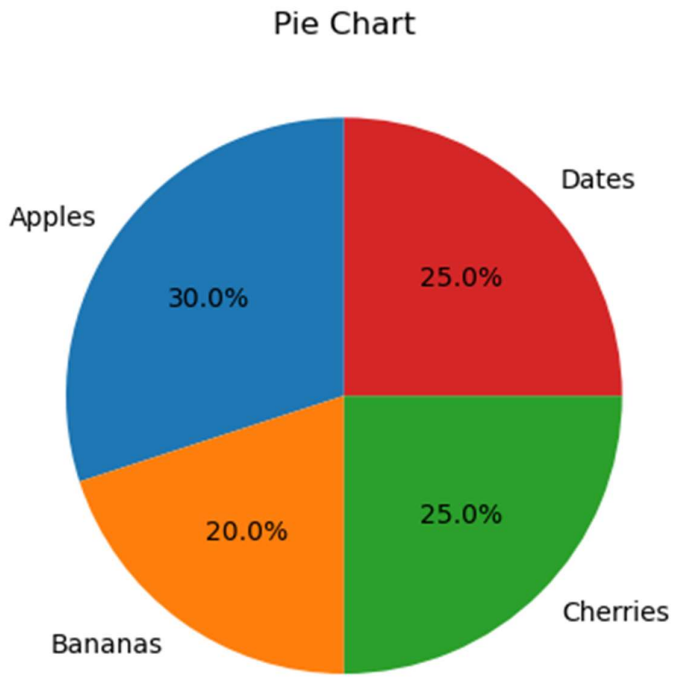
A pie chart shows proportions of different categories.

Code:

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

data = pd.Series([30, 20, 25, 25], index=['Apples', 'Bananas', 'Cherries', 'Dates'])
data.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart")
plt.ylabel("")  # hides the y-label
plt.show()
```

Output:



2.5 Box Plot

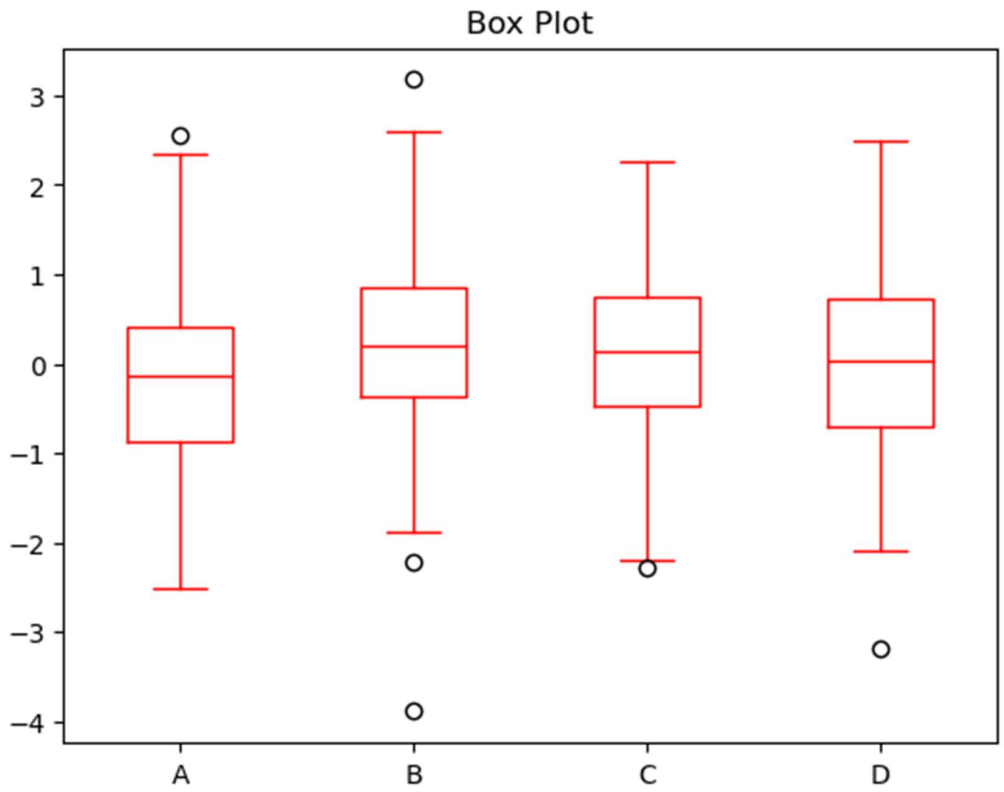
A box plot shows the spread and outliers in the data.

Code:

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

data = pd.DataFrame(np.random.randn(100, 4), columns=['A', 'B', 'C', 'D'])
data.plot(kind='box', color='red')
plt.title("Box Plot")
plt.show()
```

Output :



2.6 Area Plot

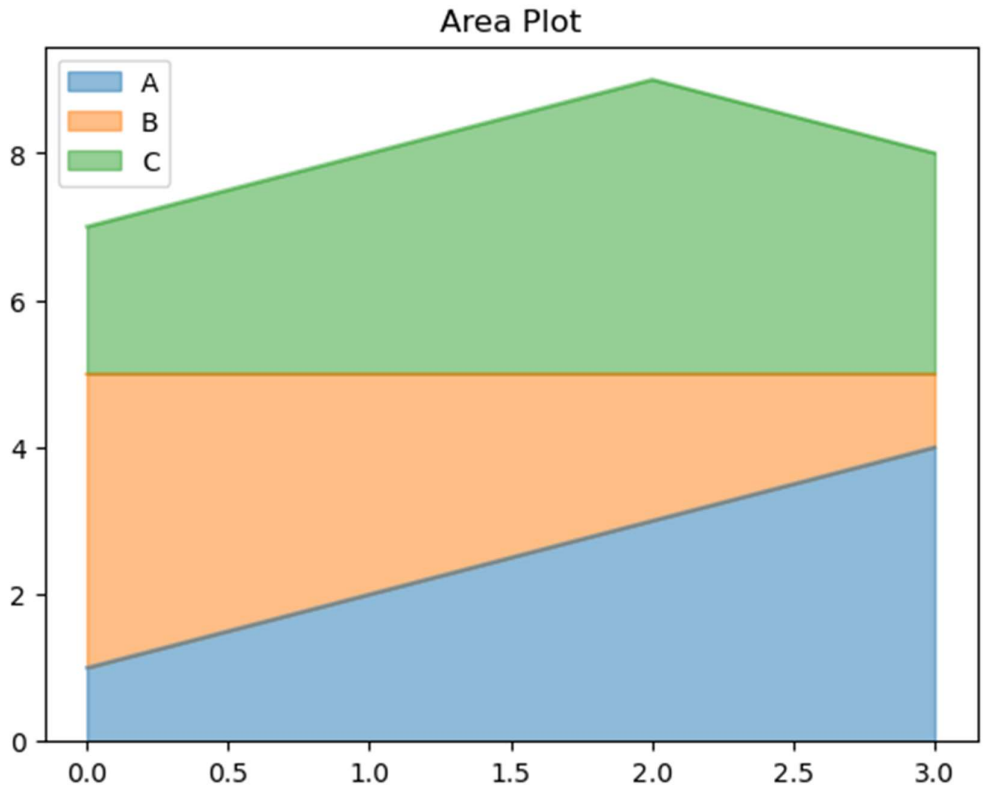
An area plot shows the cumulative total over time.

Code:

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

data = pd.DataFrame({
    'A': [1, 2, 3, 4],
    'B': [4, 3, 2, 1],
    'C': [2, 3, 4, 3]
})
data.plot(kind='area', alpha=0.5)
plt.title("Area Plot")
plt.show()
```

Output :



3.MATPLOTLIB

Overview:

Matplotlib is a Python library used to make different types of charts and graphs. It helps to show data in a visual way so we can understand it easily. We can draw line graphs, bar charts, pie charts, and many more using simple commands. It is one of the oldest and most popular libraries for data visualization in Python.

Key Features:

1. **Many Chart Types: You can make line, bar, scatter, pie, and area charts easily**
2. **Custom Design: You can change colors, labels, and styles of your graphs as you like.**
3. **Works with Other Libraries: It works well with Pandas and NumPy for better plotting.**
4. **High Quality: It helps to create neat and clear graphs for reports or presentations.**
5. **Easy to Learn: It is simple to start with, even for beginners learning Python visualization.**

To Import Matplotlib from Python :

```
import matplotlib.pyplot as plt
```

3.1 Line Chart

Displays trends or continuous data over time using plt.plot().

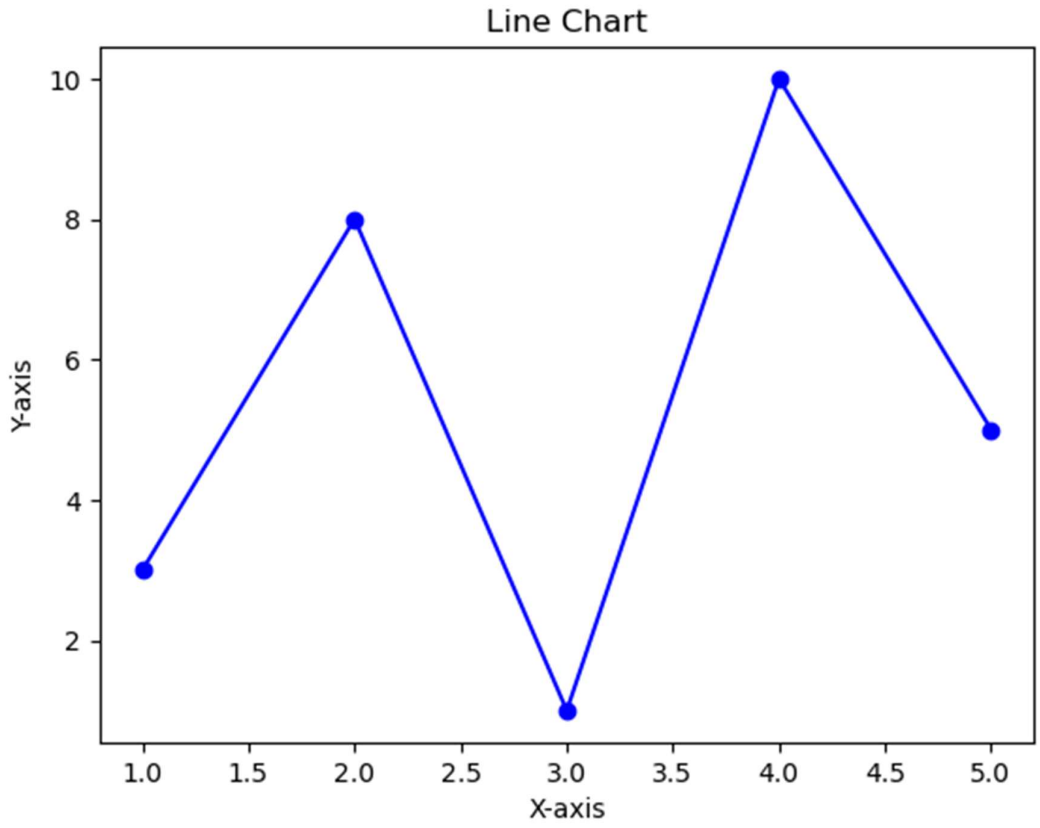
Code:

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

x = np.arange(1, 6)
y = np.array([3, 8, 1, 10, 5])

plt.plot(x, y, marker='o', color='blue')
plt.title("Line Chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output :



3.2 Bar Chart

Represents categorical data with rectangular bars using plt.bar().

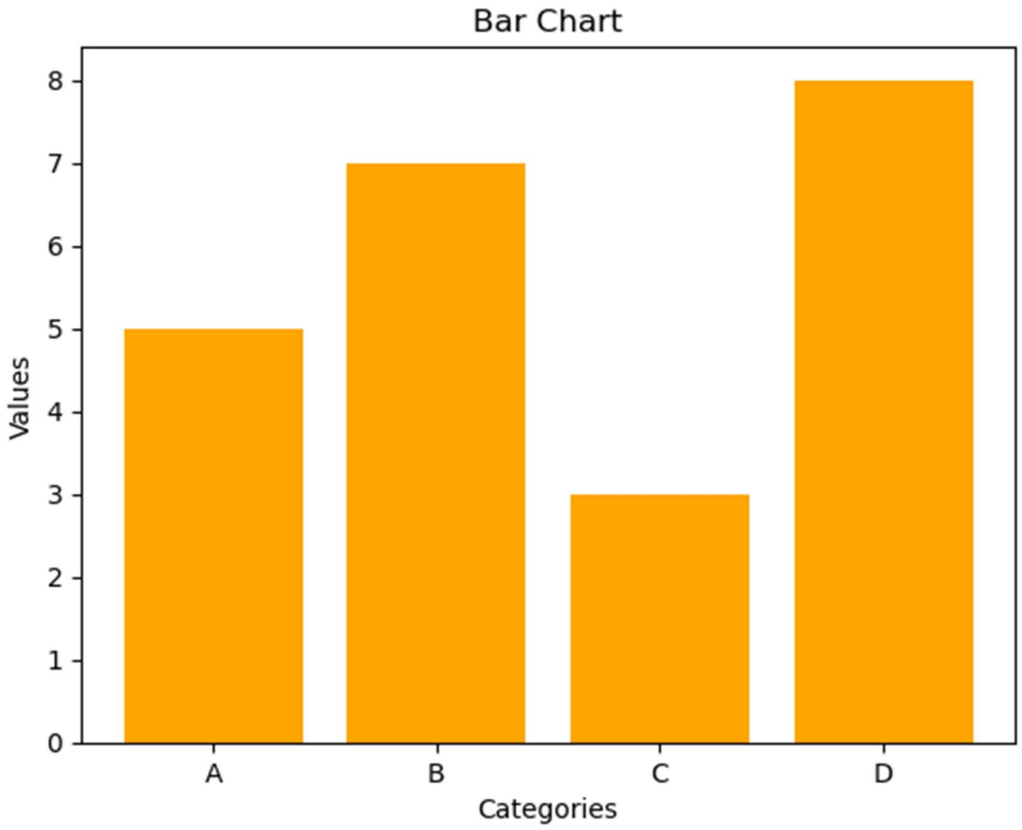
Code:

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

x = ['A', 'B', 'C', 'D']
y = [5, 7, 3, 8]

plt.bar(x, y, color='orange')
plt.title("Bar Chart")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```

Output:



3.3 Scatter Plot

Shows relationships between two variables using plt.scatter().

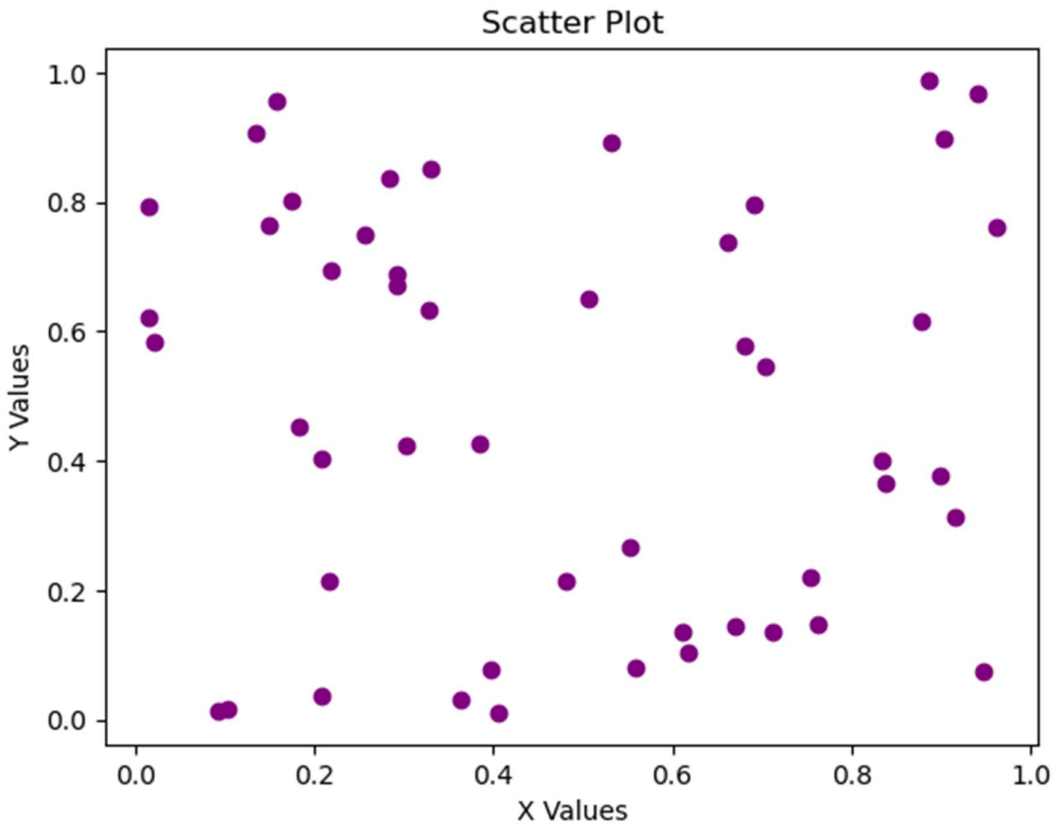
Code:

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

x = np.random.rand(50)
y = np.random.rand(50)

plt.scatter(x, y, color='purple')
plt.title("Scatter Plot")
plt.xlabel("X Values")
plt.ylabel("Y Values")
plt.show()
```

Output:



3.4 Histogram

Displays frequency distribution of data using plt.hist().

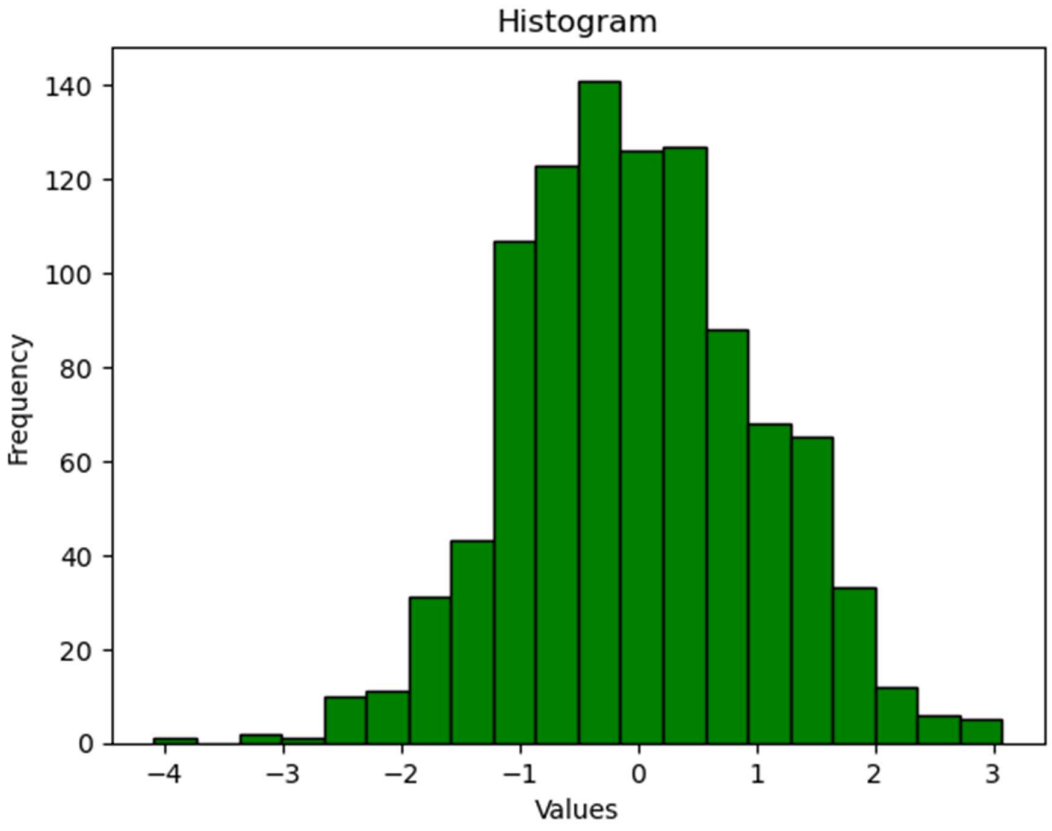
Code:

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

data = np.random.randn(1000)

plt.hist(data, bins=20, color='green', edgecolor='black')
plt.title("Histogram")
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.show()
```

Output :



3.5 Pie Chart

Represents proportions of categories using plt.pie().

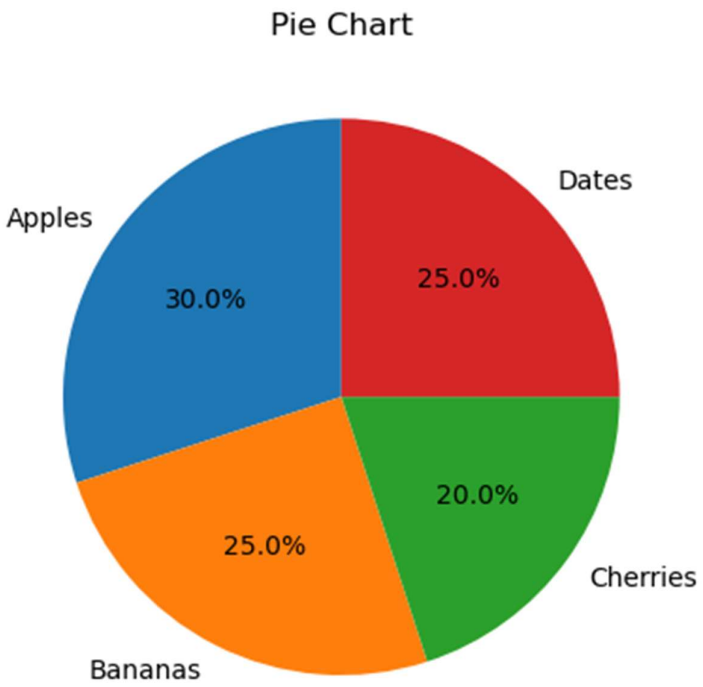
Code:

```
import matplotlib.pyplot as plt

sizes = [30, 25, 20, 25]
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart")
plt.show()
```

Output :



3.6 Area Chart

Fills the area below a line to visualize cumulative values using plt.fill_between().

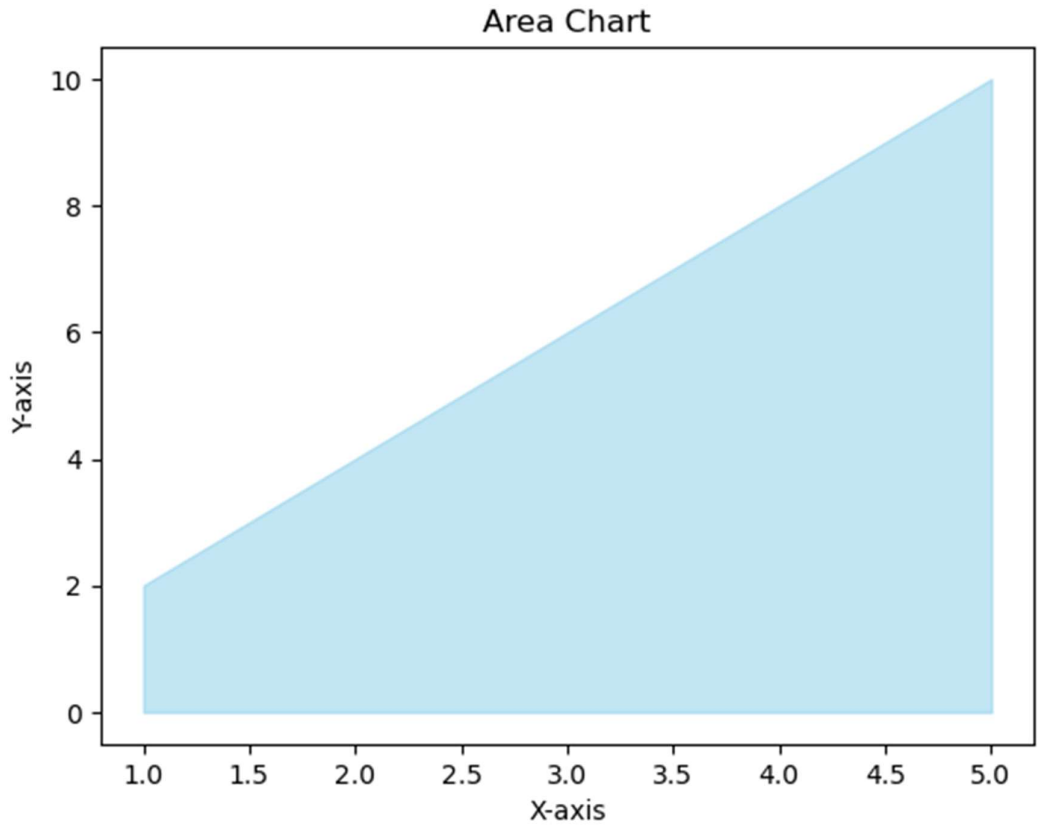
Code:

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

x = np.arange(1, 6)
y = np.array([2, 4, 6, 8, 10])

plt.fill_between(x, y, color='skyblue', alpha=0.5)
plt.title("Area Chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output:



4.Differences Between Pandas and Matplotlib

Aspect	Pandas	Matplotlib
Purpose	Mainly used for handling and analyzing data, but also supports simple plotting.	Specifically designed for creating a wide variety of detailed and professional visualizations.
Ease of Use	Very easy to use — allows quick plotting directly from DataFrames with one line of code.	Requires a bit more code but gives full control over every element of the graph.
Customization	Offers limited styling and customization options.	Highly customizable — you can adjust colors, labels, sizes, and much more.
Integration	Works directly with Pandas DataFrames, making it ideal for quick data checks.	Works well with Pandas, NumPy, and other libraries for advanced analysis.
Best Use	Great for beginners or when you need a fast, simple plot.	Best for creating polished, detailed, and presentation-quality charts.

5.Conclusion

Both Pandas and Matplotlib play an important role in data visualization.
Pandas makes it easy to create quick and simple graphs for a first look at the data.
Matplotlib allows you to design detailed and customized charts for reports or presentations.
Using them together gives the best of both worlds — simplicity and flexibility in one workflow.