**Beginners Guide To Python Visualisation Libraries:**

**Introduction:** In Python, visualization libraries are tools that let you turn data into charts, graphs, and other visual representations making it easier to analyse and communicate insights.

Python offers many visualization libraries to help turn raw data into meaningful visual insights.  
Let’s walk through two widely used ones — **Matplotlib** and **Pandas** — to understand how they help create different types of plots and charts for data analysis.

**MATPLOTLIB :**

Matplotlib is one of Python’s most popular and foundational visualization libraries, used for creating a wide variety of static, animated, and interactive plots.  
It provides fine-grained control over every aspect of a figure — from plot types and colors to labels and axes making it highly customizable.  
Because many other visualization libraries (like Seaborn and Pandas plotting) are built on top of it, Matplotlib is often considered the backbone of Python data visualization.

**Key Features of Matplotlib**

1. **Wide Range of Plots** – Supports line, bar, scatter, histogram, pie, and many more types of charts.
2. **High Customization** – Full control over colors, fonts, line styles, axes, and figure size.
3. **Multiple Output Formats** – Can save plots as PNG, JPG, PDF, SVG, etc.
4. **Interactive & Static Plots** – Works for both static figures and interactive environments like Jupyter Notebook.
5. **Integration with Libraries** – Works seamlessly with NumPy, Pandas, and other scientific libraries.
6. **Layered Approach** – Lets you build plots step-by-step for precision and flexibility.
7. **Multiple Backends** – Supports different rendering backends for various platforms and outputs.

**GRAPH TYPES**:

**1. Line Plot**

**Description:** Displays data points connected by straight lines, showing trends over time or continuous data.  
**Use Case:** Tracking stock prices, temperature changes, or sales trends.

**Sample Code:**

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

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

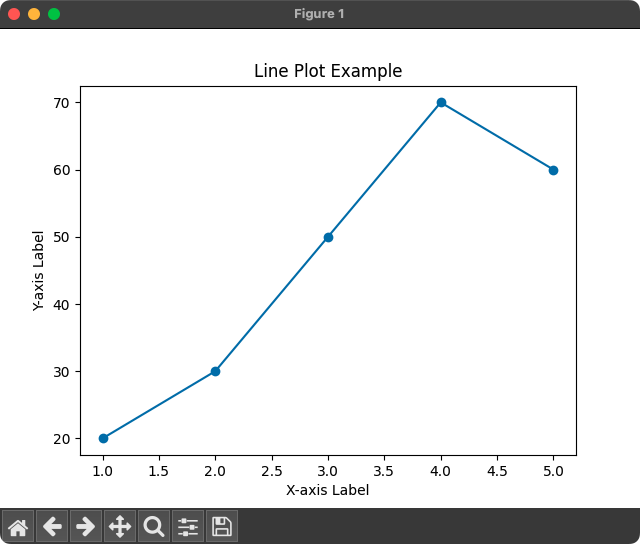
plt.plot(x, y, marker='o')

plt.title("Line Plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.show()



**2. Bar Chart**

**Description:** Represents categorical data with rectangular bars.  
**Use Case:** Comparing sales across different products.

**Sample Code**:

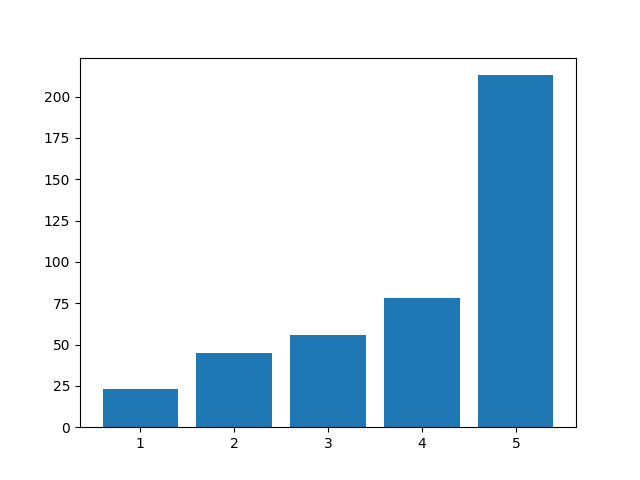
categories = ['A', 'B', 'C', 'D']

values = [4, 7, 2, 9]

plt.bar(categories, values, color='skyblue')

plt.title("Bar Chart")

plt.show()



**3. Histogram**

**Description:** Groups data into bins to show frequency distribution.  
**Use Case:** Analyzing test score distribution or income ranges.

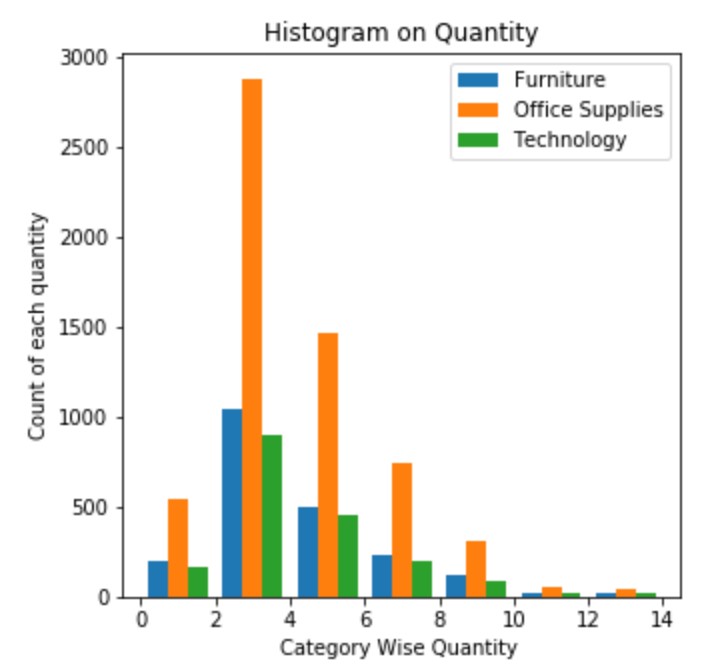
**Sample Code:**

data = [5, 7, 7, 8, 9, 9, 10, 10, 10, 11, 12, 13]

plt.hist(data, bins=5, color='orange', edgecolor='black')

plt.title("Histogram")

plt.show()



**4. Scatter Plot**

**Description:** Displays points to show relationships between two variables.  
**Use Case:** Checking correlation between height and weight.

**Sample Code:**

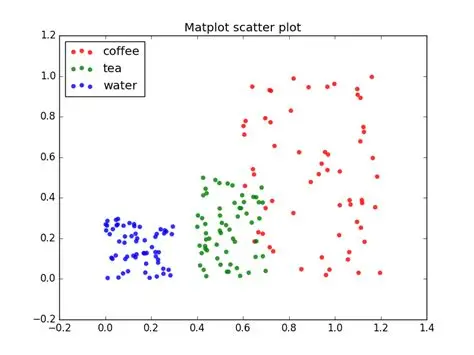
x = [5, 7, 8, 7, 6, 9, 5, 6]

y = [99, 86, 87, 88, 100, 86, 103, 87]

plt.scatter(x, y, color='red')

plt.title("Scatter Plot")

plt.show()



**5. Pie Chart**

**Description:** Shows proportions of a whole as slices of a circle.  
**Use Case:** Visualizing market share or budget allocation.

**Sample Code:**

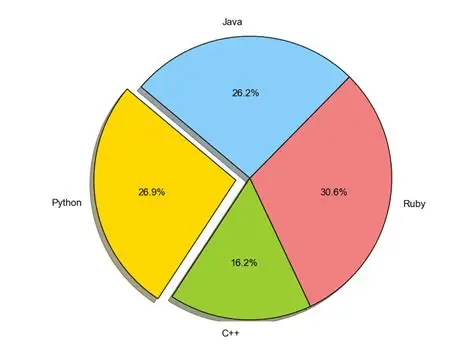
sizes = [30, 20, 25, 25]

labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)

plt.title("Pie Chart")

plt.show()



**PANDAS:**

Pandas is a powerful Python library designed for data manipulation, analysis, and visualization. It provides two main data structures -Series (1D) and DataFrame (2D) - which make it easy to work with structured data like tables.

While Pandas is primarily used for data cleaning, transformation, and analysis, it also has built-in visualization capabilities that allow quick creation ofplots directly from Series or DataFrames, making it ideal for exploratory data analysis.

**Key Features of Pandas**

1. **Powerful Data Structures** – Offers **Series** (1D) and **DataFrame** (2D) for handling structured data.
2. **Easy Data Manipulation** – Supports filtering, grouping, merging, joining, and reshaping datasets.
3. **Handling Missing Data** – Provides tools to detect, remove, or fill in missing values.
4. **Data Import & Export** – Reads and writes data from formats like CSV, Excel, JSON, SQL, etc.
5. **Built-in Visualization** – Creates quick plots directly from Series/DataFrames using .plot().
6. **Integration with Other Libraries** – Works seamlessly with NumPy, Matplotlib, and other Python data tools.
7. **High Performance** – Optimized for large datasets through efficient indexing and operations.

**GRAPH TYPES**:

Top of Form

**1. Line Plot**

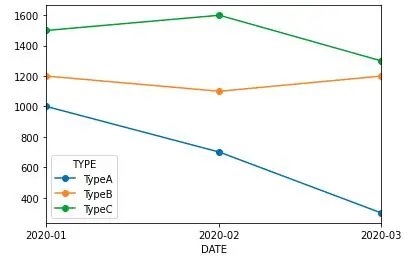
**Description:** Plots data points connected by lines, showing trends over continuous data.  
**Use Case:** Sales growth over months.

**Sample Code:**

import pandas as pd

data = pd.Series([10, 20, 15, 25, 30])

data.plot(title="Line Plot", marker='o')



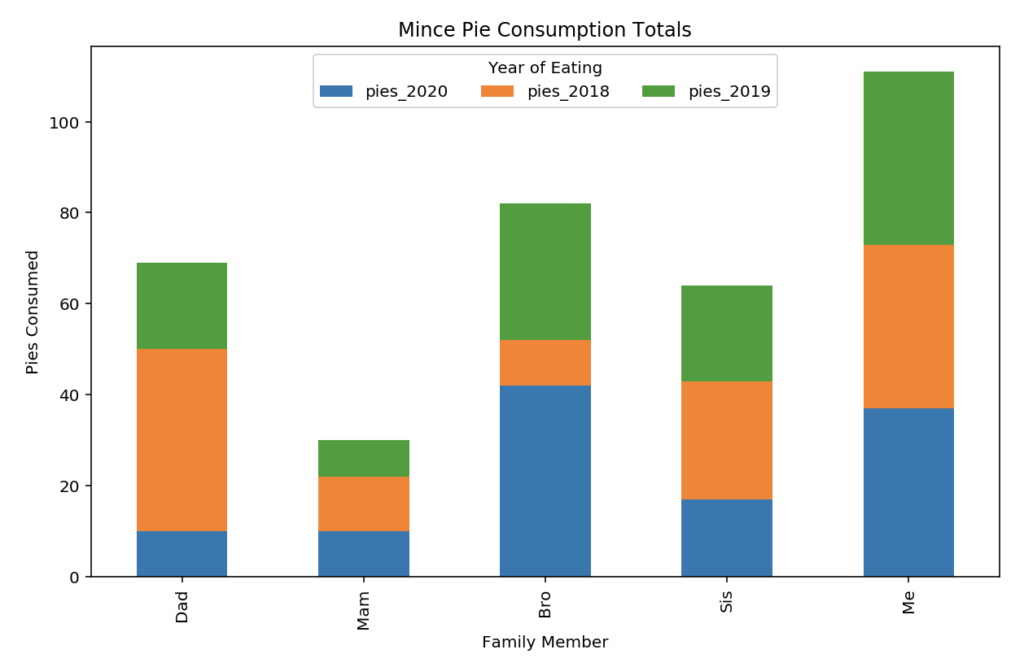
**2. Bar Chart**

**Description:** Displays data as rectangular bars for categorical comparison.  
**Use Case:** Comparing product sales.

**Sample Code:**

df = pd.Series([5, 8, 12], index=['A', 'B', 'C'])

df.plot(kind='bar', title="Bar Chart", color='skyblue')

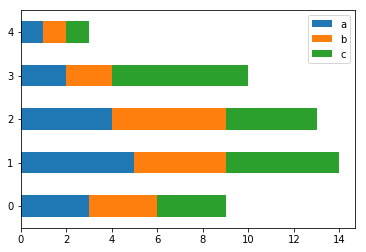


**3. Horizontal Bar Chart**

**Description:** Same as a bar chart but horizontal.  
**Use Case:** Comparing values when category labels are long.

**Sample Code:**

df.plot(kind='barh', title="Horizontal Bar Chart", color='lightgreen')



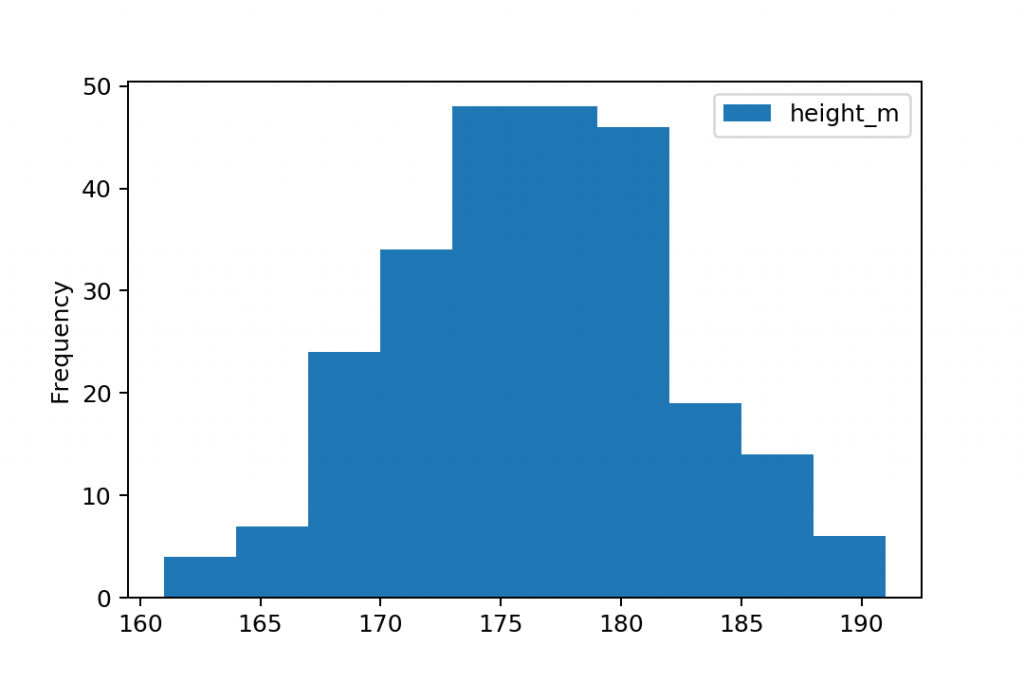
**4. Histogram**

**Description:** Groups numeric data into bins to show frequency distribution.  
**Use Case:** Analyzing exam score distributions

**Sample Code:**

data = pd.Series([3, 5, 5, 6, 7, 8, 8, 9, 10])

data.plot(kind='hist', bins=5, title="Histogram", color='orange', edgecolor='black')



**5. Area Plot**

**Description:** Like a line plot but the area under the line is filled.  
**Use Case:** Showing cumulative trends.

**Sample Code:**

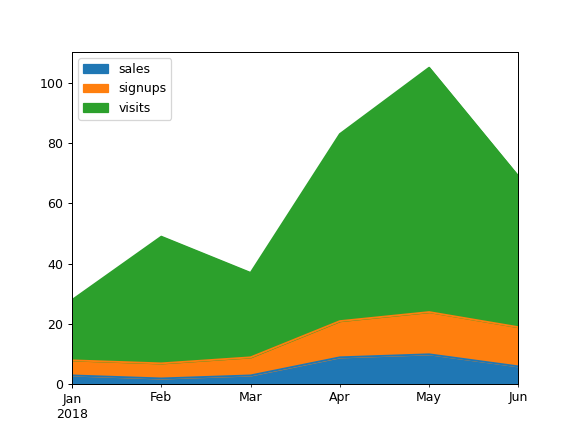
df = pd.DataFrame({

'A': [1, 3, 4],

'B': [2, 4, 6]

})

df.plot(kind='area', alpha=0.5, title="Area Plot")

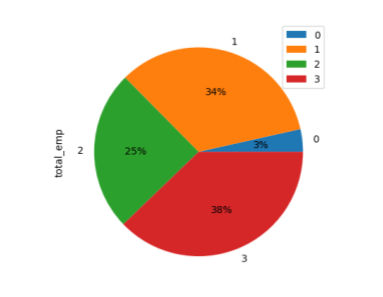


**6. Pie Chart**

**Description:** Shows proportions of a whole as slices of a circle.  
**Use Case:** Visualizing budget distribution.

**Sample Code:**

df = pd.Series([30, 20, 50], index=['A', 'B', 'C'])

df.plot(kind='pie', autopct='%1.1f%%', title="Pie Chart")

**COMPARISON:**

| **Feature** | **Matplotlib** | **Pandas** |
| --- | --- | --- |
| Level | Low-level | High-level |
| Ease | More code | Less code |
| Customization | High | Limited |
| Speed | Slower to set up | Quick plotting |
| Use Case | Detailed, advanced | Fast exploratory |

**RESOURCES:**

* [**Matplotlib — Visualization with Python**](https://matplotlib.org/)
* [**Matplotlib Tutorial - GeeksforGeeks**](https://www.geeksforgeeks.org/python/matplotlib-tutorial/)
* [**Pandas Tutorial**](https://www.w3schools.com/python/pandas/default.asp)
* [**A Beginner’s Guide to Pandas Library – Pythonista Planet**](https://pythonistaplanet.com/pandas/)

**CONCLUSION:**

* Both Matplotlib and Pandas are essential tools in Python’s data visualization ecosystem, each serving different needs.  
  For **in-depth, highly customized visuals** → use **Matplotlib**.For **fast, simple visualizations during analysis** → use **Pandas**.

Bottom of Form