**VISUALIZATION**

**Visualization** means turning data or concepts into visual forms like charts, graphs, maps, or diagrams to make them easier to understand.

It helps to:

* **Explore** data by revealing patterns, trends, and outliers.
* **Communicate** findings clearly to both technical and non-technical audiences.
* **Support decision-making** by making complex data more understandable and accessible.

In short, **data visualization is a bridge between raw data and actionable insight**, making it a crucial component of the data science process

In python, data visualization has multiple libraries like matplotlib, seaborn, plotly etc.

Let us dive deeper into matplotlib and pandas.

**MATPLOTLIB**

**Matplotlib** is a comprehensive **2D plotting library** for Python. It is one of the most foundational libraries in the Python data science ecosystem. It was created by **John D. Hunter** in 2003 and is now maintained by a large community.

Here are the **key components** of Matplotlib:

**1. Figure**

* The **top-level container** for all plot elements.
* It acts as the overall window or canvas on which plots are drawn.
* You can have multiple subplots (axes) within one figure.

**2. Axes**

* The **actual plot area** (or subplot) within the figure where data is visualized.
* A figure can contain multiple axes (e.g., in a grid).
* One Axes = one plot.

**3. Axis (Xaxis & Yaxis)**

* Represents the scale and ticks along x and y dimensions.
* Handles:
  + Ticks (location and labels)
  + Limits
  + Scaling

**4. Artist**

* Everything you see on a figure is an **Artist**: lines, text, ticks, labels, etc.
* There are two types:
  + **Primitive artists** (e.g., Line2D, Rectangle, Text)
  + **Composite artists** (e.g., Axes, Figure)

**PYPLOT**

Pyplot is a **module** in the matplotlib library that provides a **state-machine interface** to Matplotlib. It allows users to create and manage plots **step-by-step**, much like working in an interactive environment such as MATLAB or a Jupyter notebook



Some of the plots in matplotlib: -

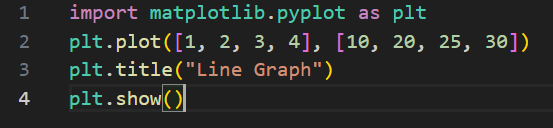
**1. Line Graph: -** A **line graph** shows data points connected by lines. It's ideal for showing **trends over time**.

* Function used is plt.plot().

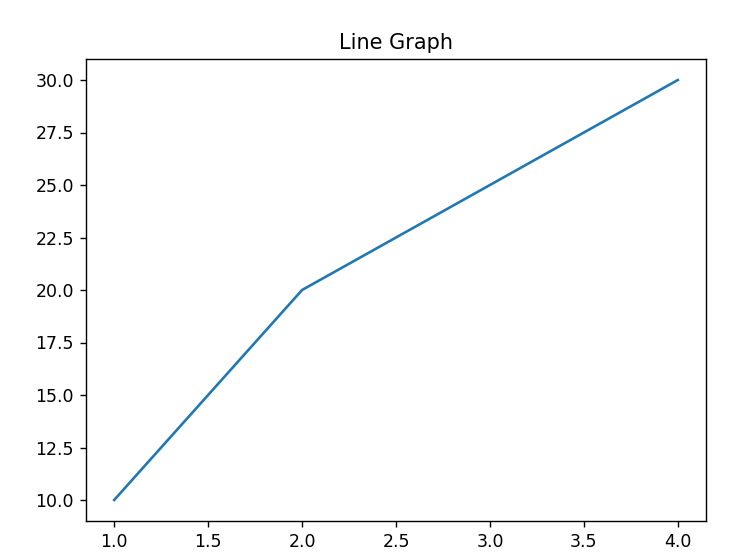
**Description**

* Shows **trends** or **progress over time**
* Lines connect data points

Code snippet:



Output:



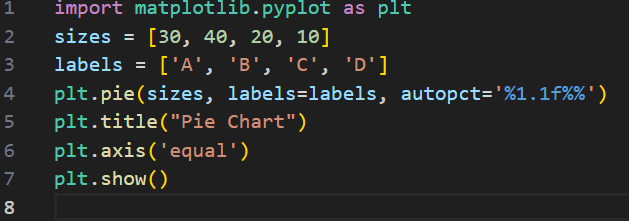
## **2. Pie Chart: -** A **circular chart** that shows proportions as slices of a pie.

* Function used is plt.pie()

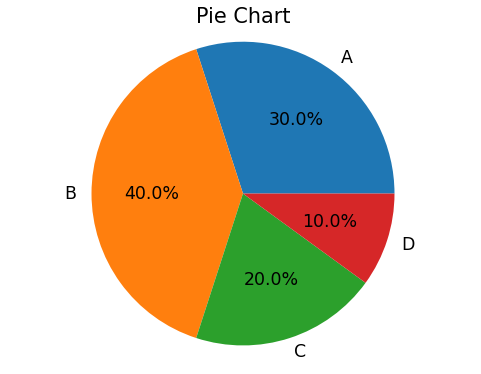
**Description**

* Displays **parts of a whole.**
* Each slice represents a percentage.

Code Snippet:



Output:



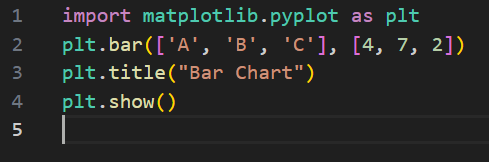
**3. Bar Chart: -**A **bar chart** displays data using rectangular bars. It's used for **categorical comparisons**

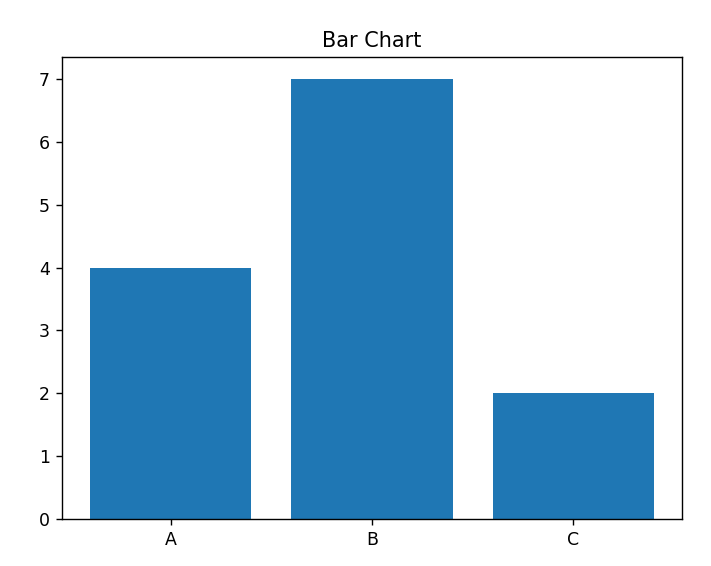
* Function used is plt.bar().

**Description**

* Uses **vertical bars** to show values.
* Compares **different categories.**

Code Snippet:



****Output:

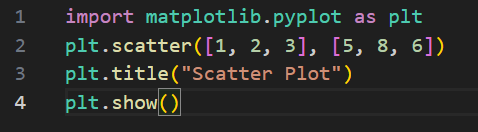
**4. Scatter Plot: -** Shows **individual data points**. Used to find **relationships** or **correlations**.

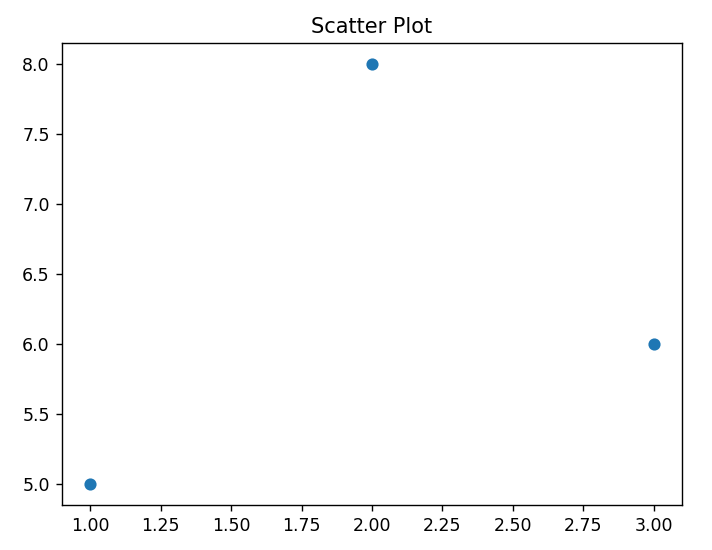
* Function used is **plt.scatter().**

**Description**

* Shows individual data **points.**
* Helps identify **correlation.**

Code Snippet:



Output:

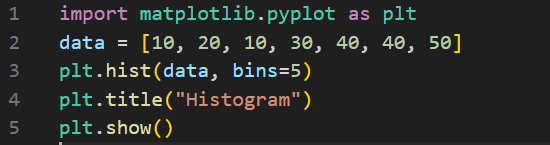
**5. Histogram: -**A **histogram** shows the **frequency distribution** of a dataset using bins.

* Function used is **plt.hist().**

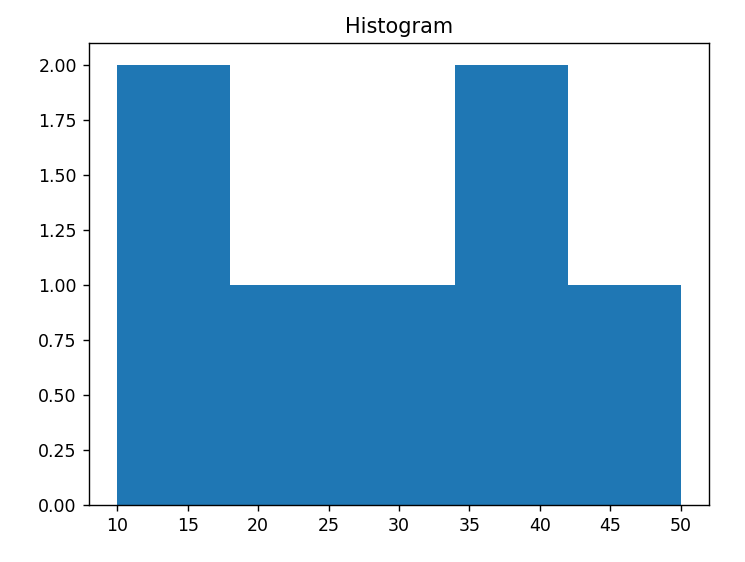
**Description**

* Groups data into **ranges** (bins).
* Shows **distribution.**

Code Snippet:



Output:



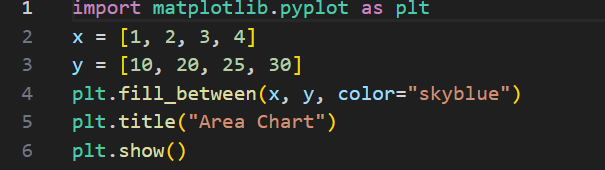
## **6.** **Area Chart: -** An **area chart** fills the space under a line, showing cumulative value over time.

* Function used is **plt.fill\_between().**

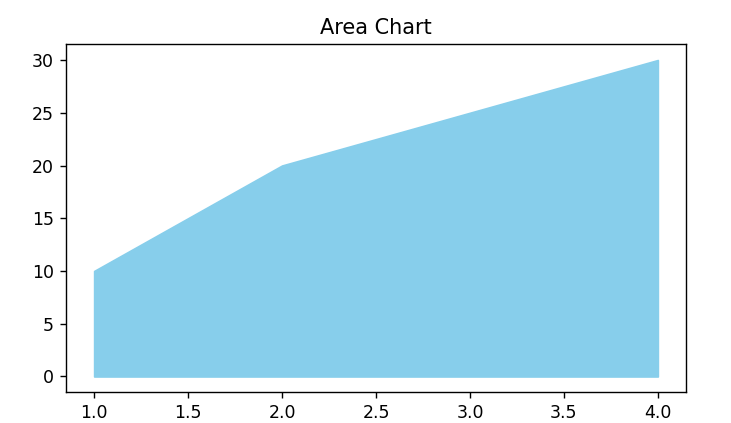
**Description**

* Like a line chart, but the **area under the line is filled.**

## Code Snippet:



Output:



**PANDAS**

**Pandas** is a **Python library** used for **data analysis, manipulation, and cleaning**. It provides **fast, flexible**, and **expressive data structures**, especially useful for working with **structured data** like tables, spreadsheets, and CSV files.



Here are the **key components** of pandas:

**1. Core Data Structures**

These are the **foundation** of pandas — everything revolves around these two:

**a. Series**

* A **one-dimensional** labeled array.

**b. DataFrame**

* A **two-dimensional** labeled data structure.

**2. Missing Data Handling**

* Handle NaN (Not a Number) and null values:

**3. Input/Output (I/O) Tools**

* Read and write files from many formats:

**4. Index Objects**

* Used to **label** the axes of Series and DataFrames (rows/columns).

**5. Data Manipulation Functions**

Pandas includes many functions for:

* Adding/removing columns and rows
* Sorting (sort\_values, sort\_index)
* Renaming (rename)
* Merging (merge, join, concat)
* Grouping (groupby)

**6. Plotting Functions**

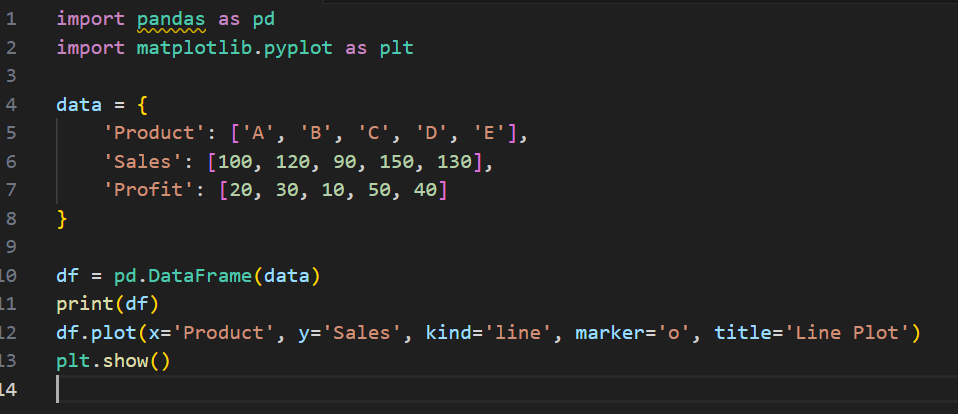
* Lightweight wrapper for **Matplotlib**
* Plot directly from Series/DataFrame using .plot()

Some of the plots in pandas: -

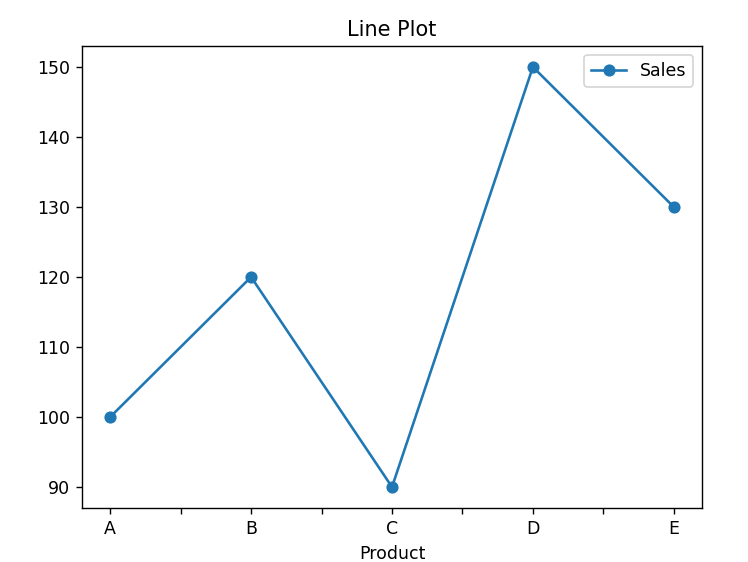
1. line plot: -

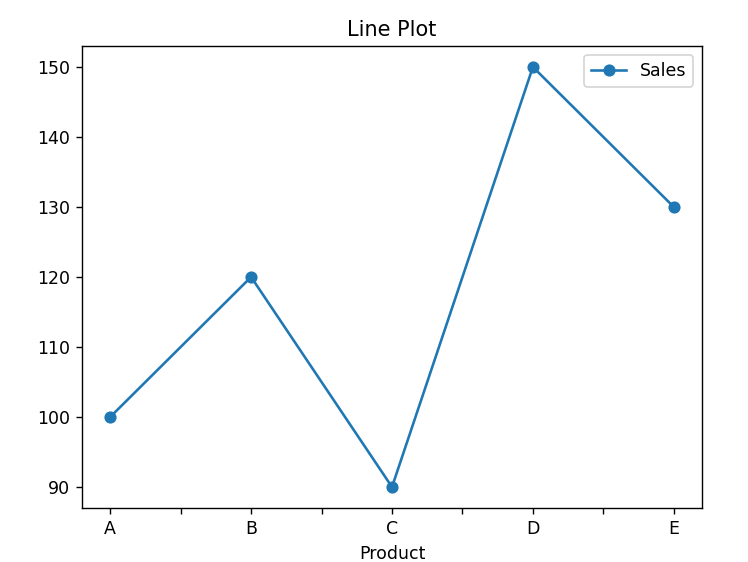
* **kind='line'**
* **Description**: Connects data points with lines.
* **Best for**: Showing trends over time.

Code snippet:



Output:

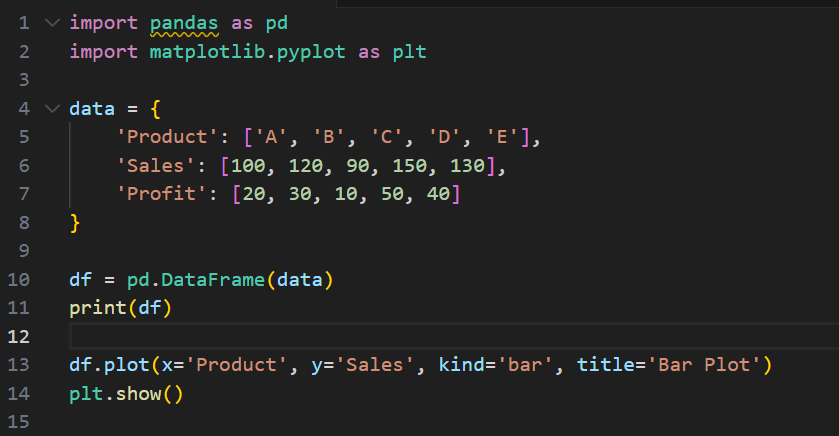




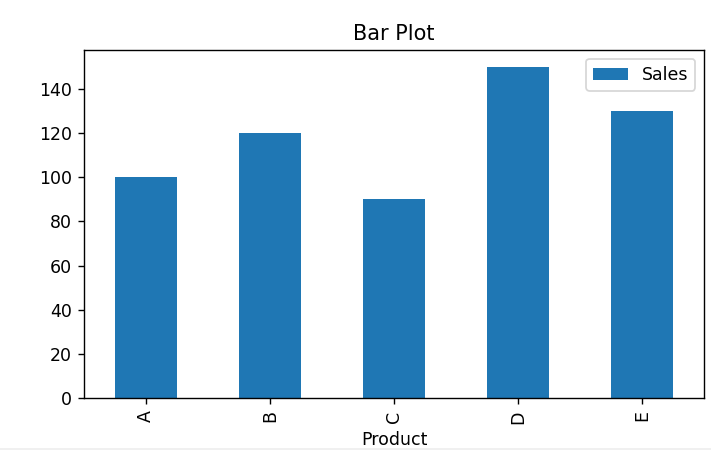
**2. Bar Chart**

* **kind='bar'**
* **Description**: Vertical bars for each category.
* **Best for**: Comparing categories side-by-side.

Code snippet:



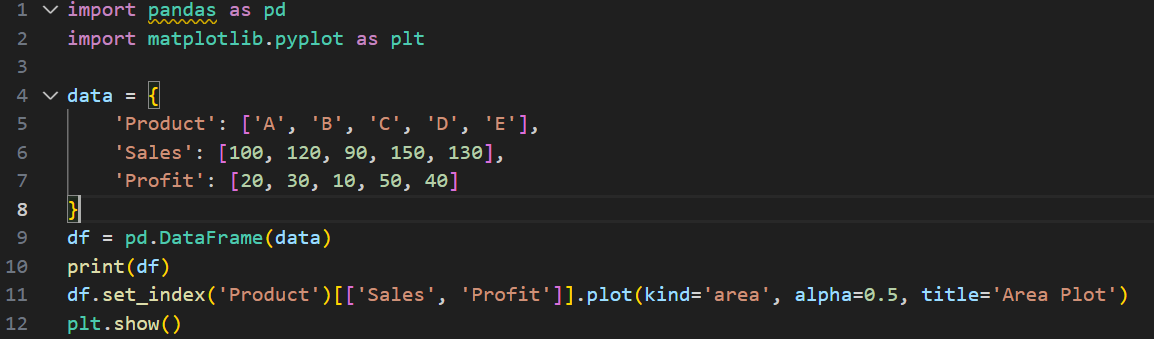
Output:



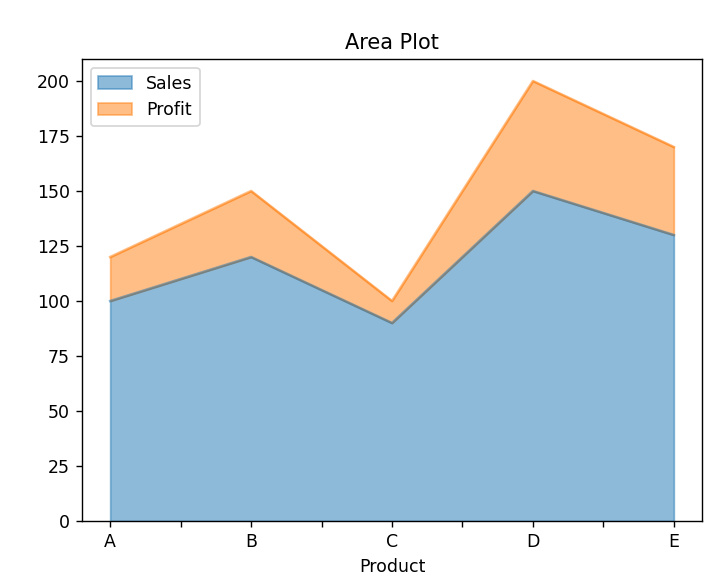
### 3. **Area Chart**

* **kind='area'**
* **Description**: Fills area under lines.
* **Best for**: Cumulative values over time.

Code snippet:



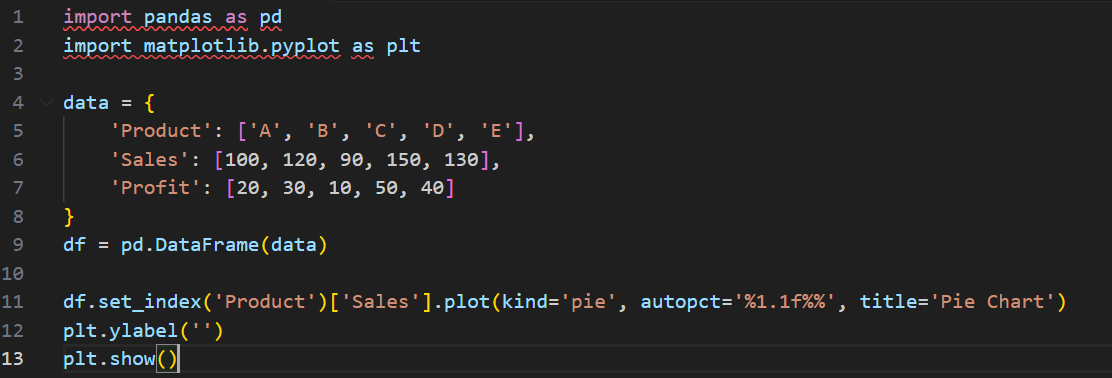
Output:

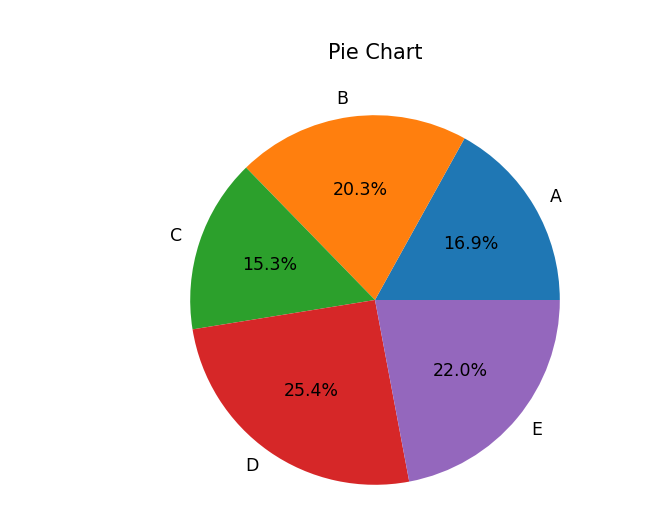


**4. Pie Chart (Series Only)**

* **kind='pie'**
* **Description**: Circular chart showing proportions.
* **Best for**: Percentage share.

Code Snippet:

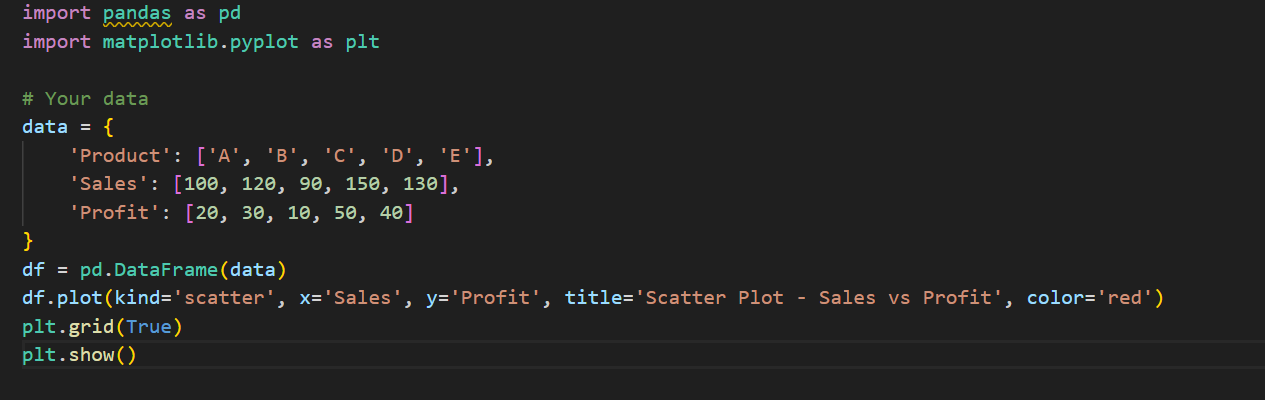


Output:

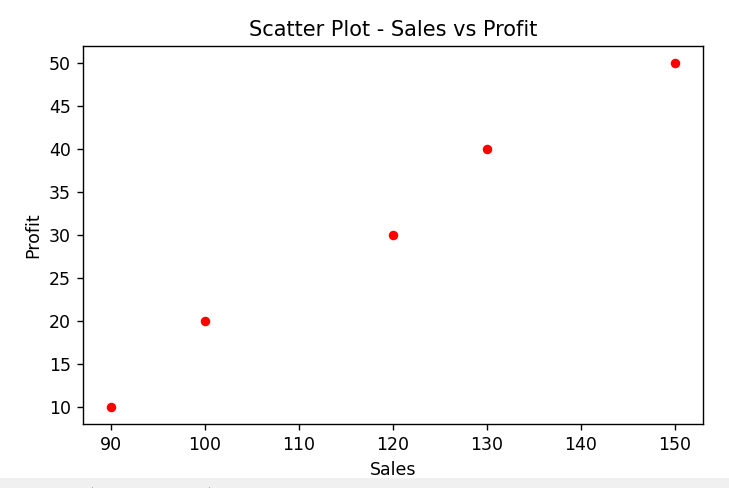
**5. Scatter Plot**

* **kind='scatter'**
* **Description**: Dots showing correlation between two variables.
* **Best for**: Detecting relationships.

Code snippet:



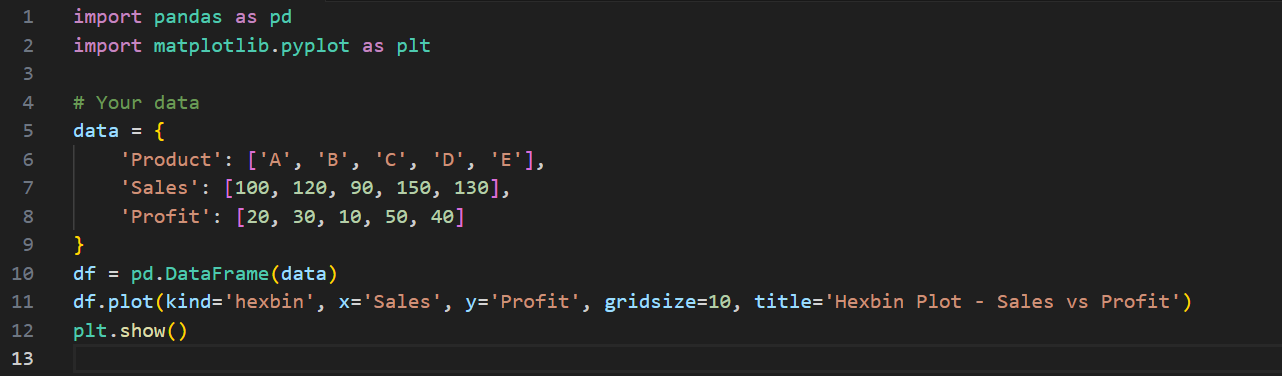
Output:



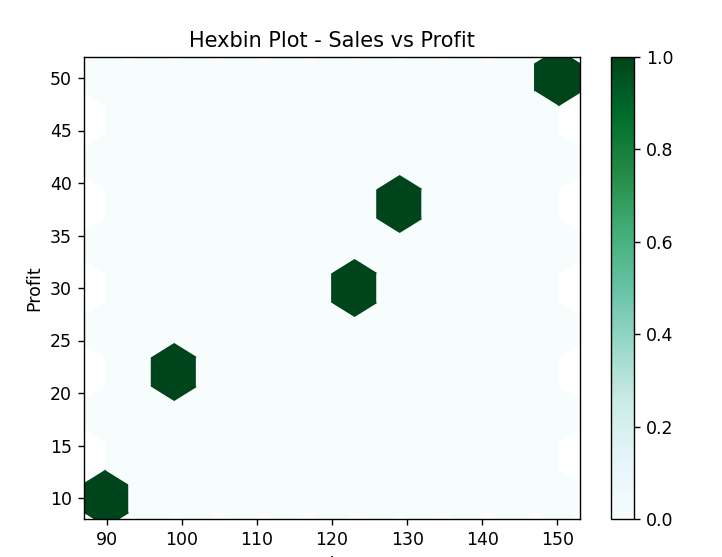
**6. Hexbin Plot**

* **kind='hexbin'**
* **Description**: 2D histogram with hexagons.
* **Best for**: Dense scatter plots.

Code snippet:



Output:



# Detailed Comparison Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Library | Ease of Use | Customization | Interactivity | Performance with Large Datasets |
| Matplotlib | ✅ Good for learning basics ❌ Requires verbose syntax for simple plots | ✅ Fully customizable (axes, labels, ticks, fonts) ❌ Complexity increases with customization | ❌ Static by default ⚠️ Requires additional tools (e.g., mpld3) | ⚠️ Moderate performance ❌ Slows down with very large datasets |
| Seaborn | ✅ Easy to use with concise syntax ✅ Built-in support for statistical plots | ⚠️ Limited fine-grained control ✅ Attractive default themes | ❌ Mostly static ⚠️ Depends on Matplotlib (inherits limitations) | ⚠️ Good for small–medium datasets ❌ Not optimized for very large datasets |
| Plotly | ✅ User-friendly API ✅ Interactive by default | ✅ High level of layout/styling control ⚠️ Slight learning curve | ✅ Highly interactive (zoom, hover, filter) ✅ Ideal for dashboards | ✅ Handles large data better than static libraries ⚠️ May lag in complex visuals |
| Bokeh | ✅ Relatively easy for interactive plots ⚠️ More structured code required | ✅ Highly customizable ✅ Built-in widgets for interaction | ✅ Built for interactivity (hover, zoom, sliders) ✅ Web integration ready | ✅ Efficient with large data ✅ Supports streaming and server-based apps |
| Pandas | ✅ Intuitive for data manipulation ✅ Clean syntax for analysis and plotting | ⚠️ Basic plotting options ❌ Relies on Matplotlib for visuals | ❌ Limited interactivity ⚠️ Can be integrated with Plotly for more features | ⚠️ Handles medium datasets well ❌ Not ideal for very large datasets without optimization |