# MATPLOTLIB LIBRARY

### Matplotlib is an open-source drawing library that supports various drawing types. It is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the border SciPy stack and consists of several plots like line, bar, scatter, histogram, etc. It is often used in web application servers, shells, and Python scripts.

## Key Features of Matplotlib

1. Versatility: Matplotlib can generate a wide range of plots, including line plots, scatter plots, bar plots, histograms, pie charts, and more.
2. Customization: It offers extensive customization options to control every aspect of the plot, such as line styles, colours, markers, labels, and annotations.
3. Extensible: Matplotlib is highly extensible, with a large ecosystem of add-on toolkits and extensions like seaborn, pandas plotting functions and basemap for geographical plotting.
4. Interactive Plot: Matplotlib supports interactive plotting through the use of widgets and event handling, enabling users to explore data dynamically.

### Basic Components of Matplotlib Figure

### Figures in Matplotlib: The figure object is the top-level container for all elements of plot. It serves as the canvas on which the plot is drawn.

### Axes in Matplotlib: Axes are the rectangular areas within the figures where data is plotted. Axes provide the coordinate system and are where most of the plotting occurs.

### Matplotlib Title: The title is the text element that provides a descriptive title for the plot. It typically appears at the top of the figure and provide context or information about the data being visualization.

### Matplotlib Legend: Legends provide a key to the symbols or colours used in the plot to represent different data series or categories. They help users interpret the plot and understand the meaning of each element.

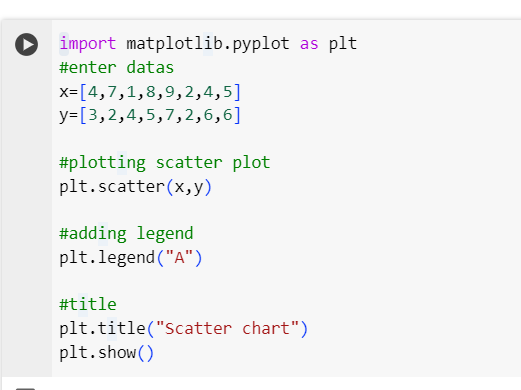
### Graph Types

### Scatter Plot: Scatter plots are utilized to see how different variables are related to each other. The dots on the plot shows how variables are related.

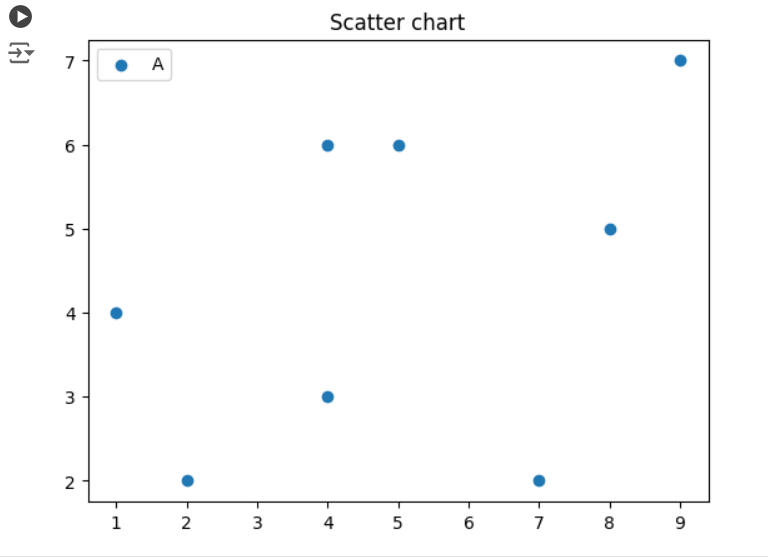
Use Cases:

* For a large set of data points given.
* Each set comprises a pair of values.
* The given data is in numeric form.

Code Snippet:



### Output:

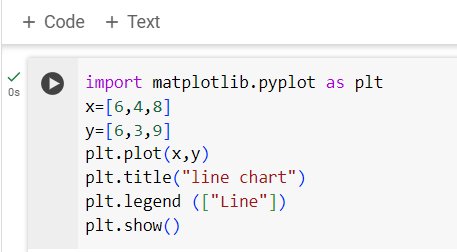


1. Line Plot: A line plot visualizes information as a series of data points. A simple line chart is generated using NumPy to define data values. A line plot is excellent at showcasing trends and fluctuations in data.

Use Cases:

* It is often use to represent time series.
* Supports monitoring behaviour in a set of data.
* When smaller changes exist line charts are used.

Code Snippet:



Output:

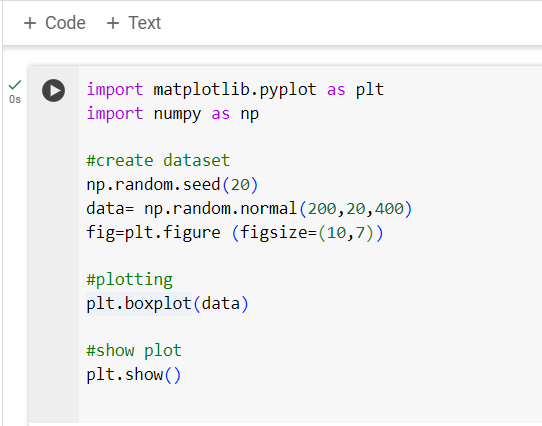
### 

1. Box Plot: It provides a graphical summary of a data set with features such as minimum, first quartile, median, third quartile, and maximum. It uses boxes and lines to depict the distributions of one or more groups of numeric data.

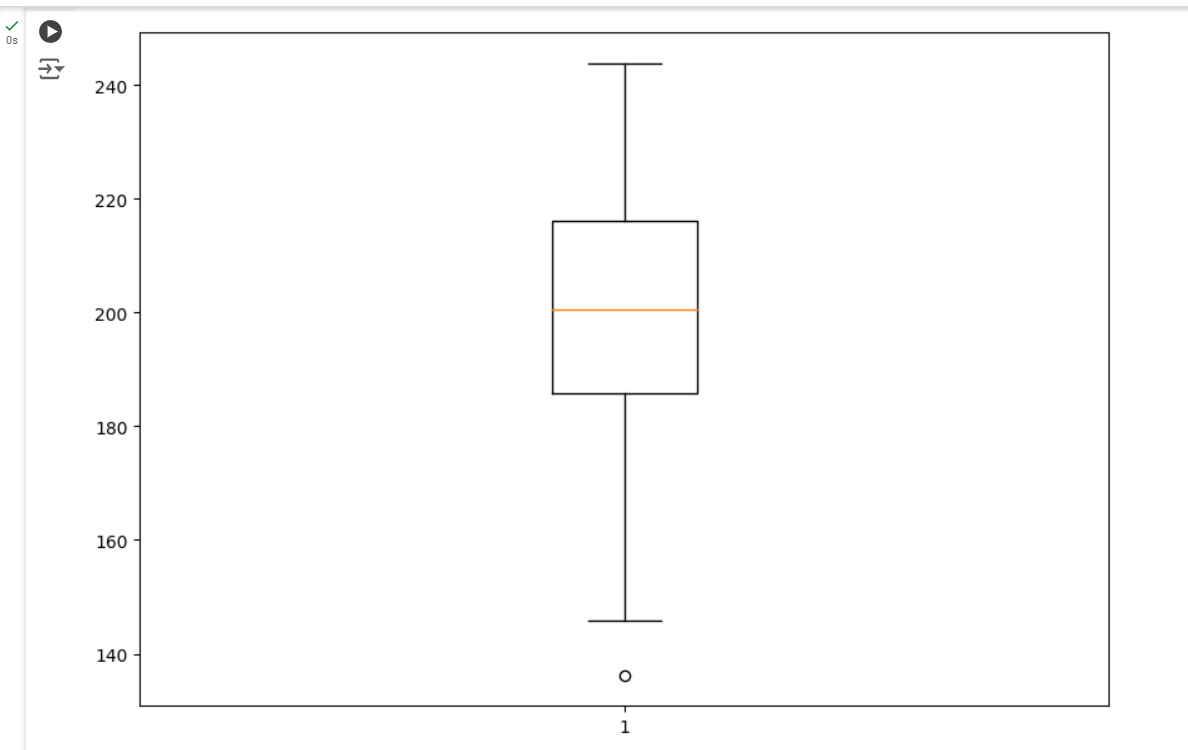
Use Cases:

* Comparing data across categories.
* Organising large amount of data.
* Getting a high-level view.

Code Snippet:



Output:

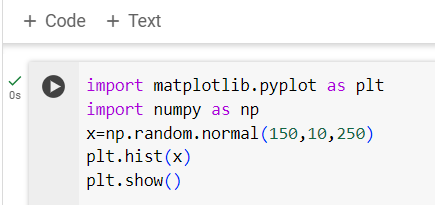


1. Histogram: A Histogram is used to represent data provided in the form of some groups. It is an accurate method for the graphical representation of numerical data distribution. The bar height represents the frequency or count of data points within each group.

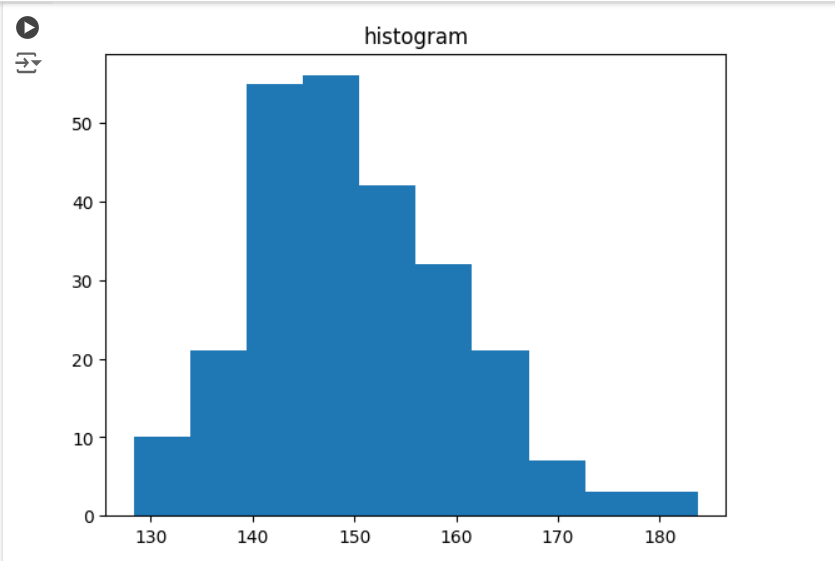
Use Cases:

* To analyse Distribution of Data.
* Compare the distribution of data across different ranges or intervals.
* Troubleshooting aids.

Code Snippets:



Output:

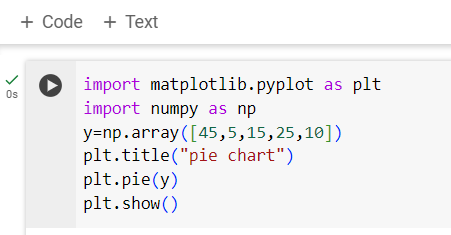


1. Pie Chart: A type of graph representing data in a circular form, with each slice of the circle representing a fraction or proportionate part of the whole. It shows the size of items in one data series, proportional to the sum of the items. The data points in a pie chart are shown as a percentage of the whole pie.

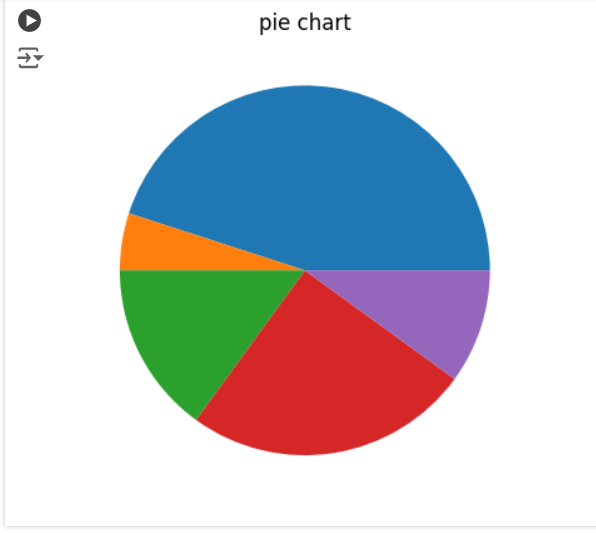
Use Cases:

* When total number can be split into a smaller-categories.
* To show how each part contributes to the whole.
* To provide a quick analysis or understanding of information.

Code Snippet:



Output:



# PANDAS LIBRARY

# Pandas is a python library used for working with data sets. It is a powerful and open-source library. It provides fast, flexible, and expensive data structures designed to make working with “relational” or “labelled” data both easy and intuitive.

## Key Features of Pandas

### Data Cleaning: Pandas can handle missing data, remove duplicate data, and fix formatting issues.

1. Data Visualization: Pandas has a plot library that allows users to create interactive visualization with a few lines of code.
2. Data Set Merging and Joining: Pandas has techniques for merging, joining, and concatenating, Data Frames to combine multiple datasets into one.
3. Data Aggregation: Pandas has functions for aggregating data.

## Basic Components of Pandas

1. Series: The Pandas series is a One-Dimensional data structure that can hold data of any type, like integers, floats, strings, Python objects, etc. It is the primary data structure of pandas and is built on top of the NumPy of Python. All of the axis labels are referred to as an index. In Python pandas.series() is the constructor that is used for creating series.
2. Data Frame: A Pandas Data Frame is a two- dimensional, size mutable data structure, consisting of labelled axes, i.e., rows and columns. Each column is a pandas-series. Data Frame can be created using pandas.DataFrame() constructor.
3. Panel: The panel is the essential container of three-dimensional data in Pandas. The panel can be created using pandas.Panel()

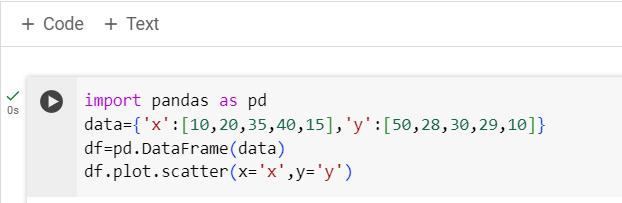
# Graph Types

1. Scatter Plot: A scatter plot is a type of data visualization technique that shows the relationship between two numerical variables. For plotting to scatter plot using pandas there is a Data Frame class and this class has a member called plot.

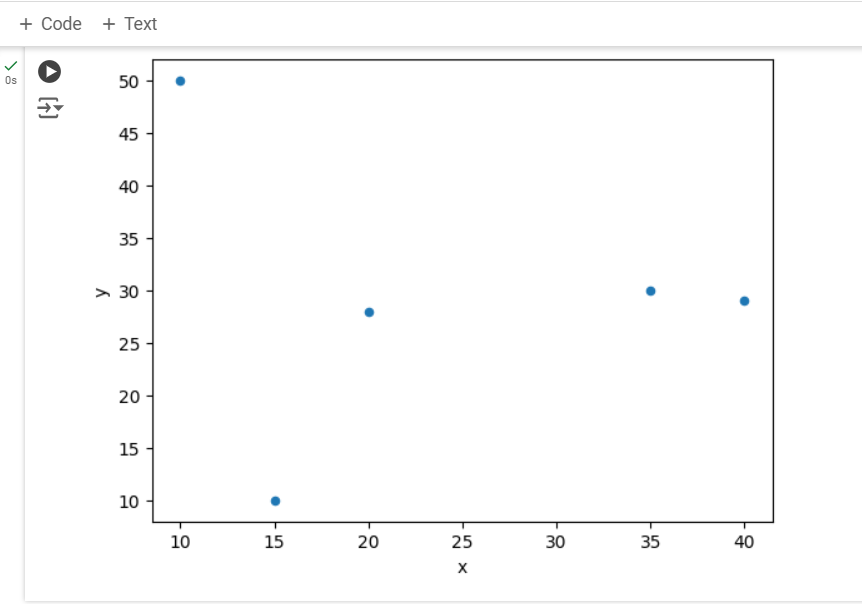
Use Cases:

* To see complex correlation between two variables.
* Visualizing data points in a two-dimensional space.

Code Snippet



Output:

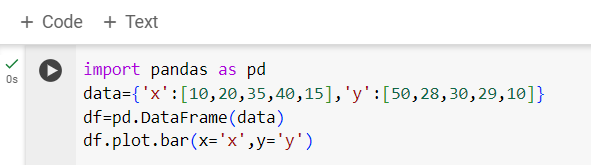


1. Bar Plot: A bar plot shows comparisons among discrete categories. One axis of the plot shows the specific categories being compared. And the other axis represents a measured value. To generate scatter plot DataFrame.plot.bar() is used.

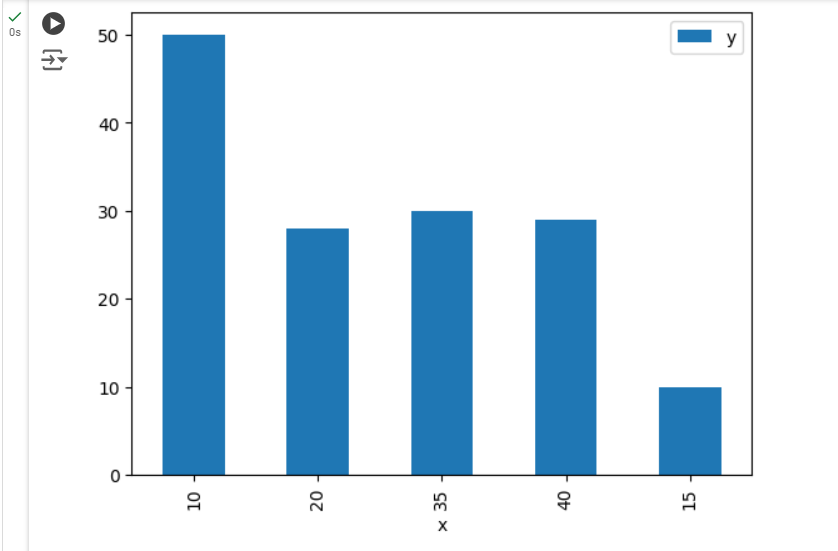
Use Cases:

* To compare the categories.
* To show distribution of data.
* pandas.DataFrame.plot.bar method for vertical bar plot.
* Pandas.DataFrame.plot.barh method for horizontal bar plot.

Code Snippet



Output

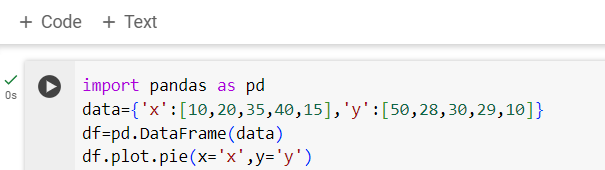


### Pie Chart: In research, engineering and business, it is frequently utilized. It is a statistical plot that can display only one series of data. The area of the plot is the total percentage of the data.

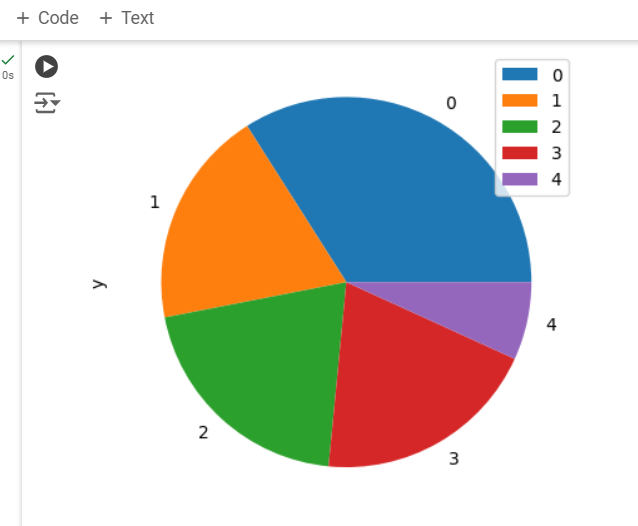
Use Cases:

* If we have a dimension with just a couple of categories to compare, then pie chart is used.
* To convey that one segment of the total is relatively small or large.
* Interpreting and representing the data more clearly.

Code Snippet



Output

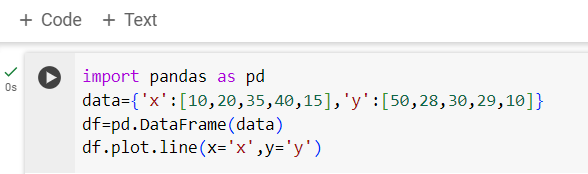


1. Line Plot: A line plot is a graphical display that visually represents the correlation between certain variables or changes in data over time using several points, usually ordered in their x-axis and connected by straight line segments. The syntax used is DataFrame.plot.line(x,y)

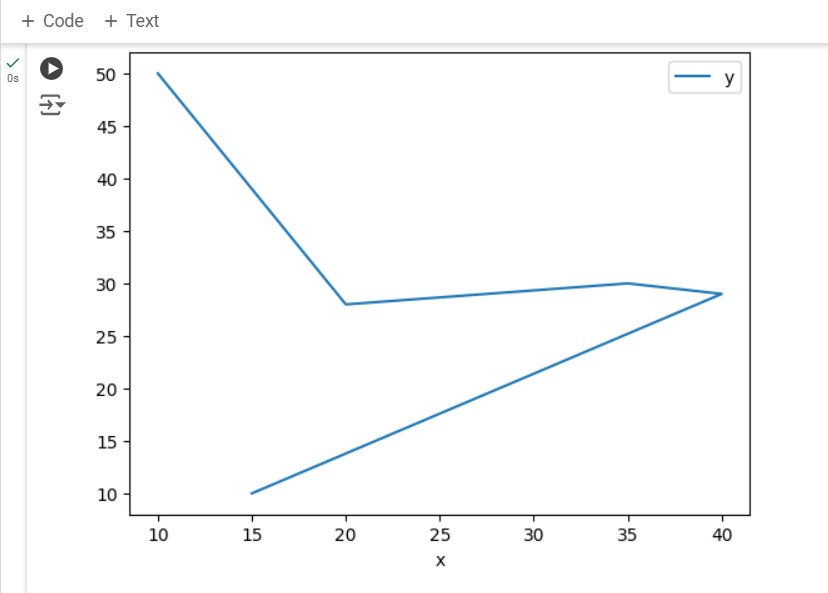
Use Cases:

* To track changes over short and long periods of time.
* To show trends clearly and accurately.
* To represent quantitative data collected over a specific subject and a specific time interval.

Code Snippet



Output

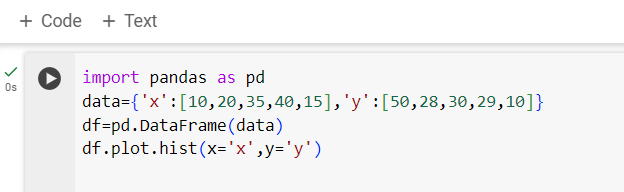


1. Histogram: To create a Histogram from a pandas.DataFrame, we first need to extract the data we want to plot. We can do this by selecting a column from the Data Frame using its name or index. Once we have the data, we can pass it to a histogram function from a visualization library to generate the plot.

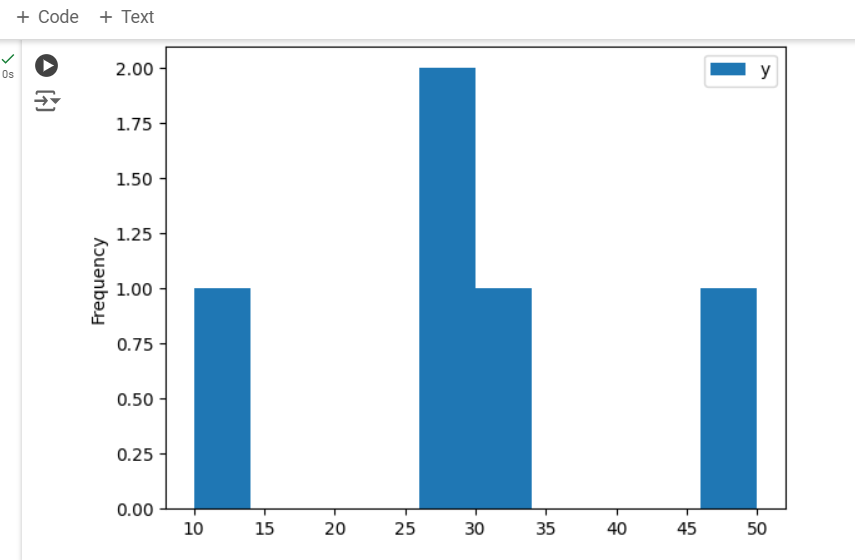
Use Cases:

* To get a quick understanding of the distribution of certain numerical variables within it.
* Pandas.DataFrame.hist() function plots Histogram of the given data frame.
* To do visualize a large data set and to complete exploratory data analysis.

Code Snippet:



Output



# COMPARISON BETWEEN MATPLOTLIB AND PANDAS

|  |  |  |
| --- | --- | --- |
|  | **MATPLOTLIB** | **PANDAS** |
| **STRENGTHS** | * Flexibility: Matplotlib offers fine-grained control over every aspect of a plot, allowing users to create highly customized visualizations. * Object-oriented: Matplotlib’s object-oriented approach allows for creating complex, multi-component visualizations. * Wide Range of Plot Types: From simple line plots to complex 3D plots, Matplotlib supports a vast array of plot types. | * Seamless Integration with Data Frames: Pandas plotting functions can be directly applied to Data Frame and series objects, eliminating the need for data manipulation. * Quick and easy: Pandas provide a high-level interface for generating common plot types with minimal code. * Autonomic Handling of Missing Data: Pandas autonomically handles missing data, making it suitable for working with real-world datasets. |
| **WEAKNESS** | * Default Aesthetic: The default plot aesthetic in Matplotlib often considered less visually appealing compared to pandas, requiring more effort to make plots visually attractive. * Limited Interactive: While Matplotlib does support interactive plotting to some extent, it does not offer as many interactive features and options as other libraries like Plot. * Less Modern Features: Matplotlib has been around for a long time, and some users find that it lacks some of the modern plotting features and interactive visualization capabilities found in newer libraries. | * Memory Usage: Pandas can be memory-intensive, especially when dealing with large datasets. * Not Optimized for Speed: Some operations in Pandas can be slower compared to low-level languages, as it priorities ease of use over raw performance. * Not dealing with Big Data: Pandas may not scale well for Bigdata processing. |