Pandas Introduction

Pandas is a powerful and open-source Python library. The Pandas library is used for data manipulation and analysis. Pandas consist of data structures and functions to perform efficient operations on data.

Pandas is well-suited for working with **tabular data**, such as **spreadsheets** or **SQL tables**.

What is Python Pandas used for?

The Pandas library is generally used for data science, but have you wondered why? This is because the Pandas library is used in conjunction with other libraries that are used for data science. It is built on top of the NumPy Library which means that a lot of the structures of NumPy are used or replicated in Pandas.

The data produced by Pandas is often used as input for plotting functions in Matplotlib statistical analysis in **SciPy** and machine learning in scikit-learn

You must be wondering, Why should we use the Pandas Library. Python's Pandas library is the best tool to analyze, clean, and manipulate data.

Here is a list of things that we can do using Pandas.

- Data set cleaning, merging, and joining.
- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data.
- Columns can be inserted and deleted from DataFrame and higher-dimensional objects.
- Powerful group by functionality for performing split-apply-combine operations on data sets.
- Data Visualization.

Getting Started with Pandas

Let's see how to start working with the Python Pandas library:

Installing Pandas

The first step in working with Pandas is to ensure whether it is installed in the system or not. If not, then we need to install it on our system using the **pip command**.

Importing Pandas

After the Pandas have been installed in the system, we need to import the library. This module is generally imported as follows:

import pandas as pd

Note: Here, pd is referred to as an alias for the Pandas. However, it is not necessary to import the library using the alias, it just helps in writing less code every time a method or property is called.

Data Structures in Pandas Library

Pandas generally provide two data structures for manipulating data. They are:

- **Series**
- **DataFrame**

Pandas Series

A Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, Python objects, etc.). The axis labels are collectively called **indexes**.

Creating a Series

Panda Series is created by loading the datasets from existing storage (which can be a SQL database, a CSV file, or an Excel file).

Pandas Series can be created from lists, dictionaries, scalar values, etc.

Example: Creating a series using the Pandas Library.

```
import pandas as pd
import numpy as np

# Creating empty series
ser = pd.Series()
print("Pandas Series: ", ser)

# simple array
data = np.array(['V', 'S', 'S', 'U', 'T'])

ser = pd.Series(data)
print("Pandas Series:\n", ser)
```

Output

```
Pandas Series: Series([], dtype: float64)
Pandas Series:
0    V
1    S
2    S
3    U
4    T
dtype: object
```

Pandas DataFrame

Panda dataframe is a two-dimensional data structure with labeled axes (rows and columns).

Creating DataFrame

Pandas DataFrame is created by loading the datasets from existing storage (which can be a SQL database, a CSV file, or an Excel file).

Pandas DataFrame can be created from lists, dictionaries, a list of dictionaries, etc.

Example: Creating a DataFrame Using the Pandas Library

```
import pandas as pd

# Calling DataFrame constructor
df = pd.DataFrame()
print(df)
```

```
# list of strings
lst = ['VEER', 'SURENDRA', 'SAI', 'UNIVERSITY', 'OF', 'TECHNOLOGY']

# Calling DataFrame constructor on list
df = pd.DataFrame(lst)
print(df)
```

Output:

```
Empty DataFrame
Columns: []
Index: []
         0
0    VEER
1      SURENDRA
2    SAI
3      UNIVERSITY
4   OF
5      TECHNOLOGY
```

Introduction to Matplotlib

Matplotlib is a powerful and versatile open-source plotting library for Python, designed to help users visualize data in a variety of formats. Developed by John D. Hunter in 2003, it enables users to graphically represent data, facilitating easier analysis and understanding. **If you want to convert your boring data into interactive plots and graphs, Matplotlib is the tool for you**.

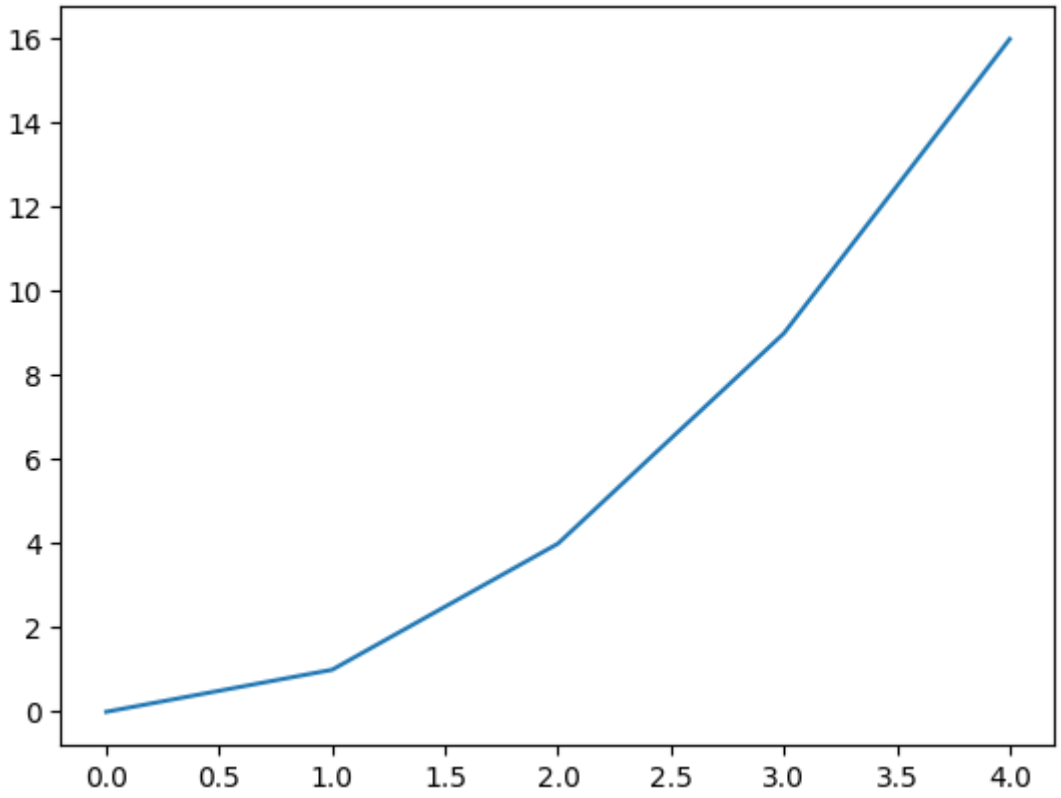
Example of a Plot in Matplotlib:

Let's create a simple line plot using Matplotlib, showcasing the ease with which you can visualize data.

```
import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4]
y = [0, 1, 4, 9, 16]
```

```
plt.plot(x, y)
plt.show()
```

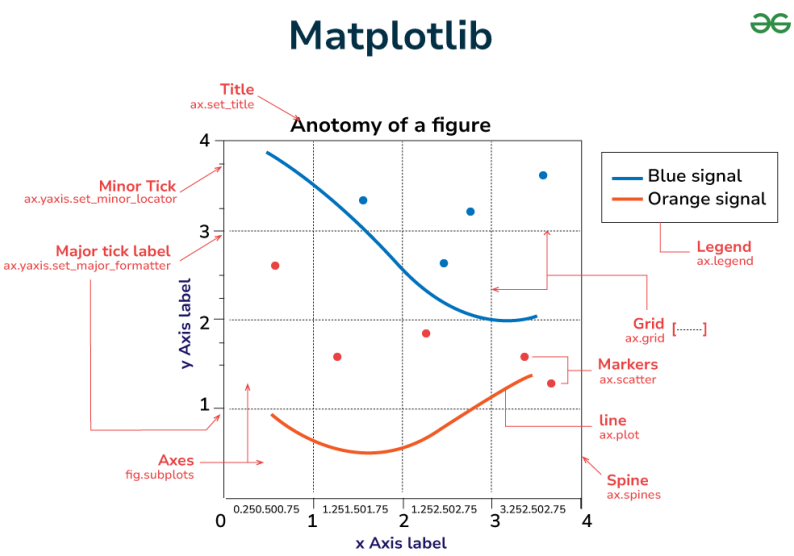
Output:



Si mplest plot in Matplotlib

Components or Parts of Matplotlib Figure

Anatomy of a Matplotlib Plot: This section dives into the key components of a Matplotlib plot, including figures, axes, titles, and legends, essential for effective data visualization.



The parts of a Matplotlib figure include (as shown in the figure above):

- **Figure**: The overarching container that holds all plot elements, acting as the canvas for visualizations.
- **Axes**: The areas within the figure where data is plotted; each figure can contain multiple axes.
- **Axis**: Represents the x-axis and y-axis, defining limits, tick locations, and labels for data interpretation.
- **Lines and Markers**: Lines connect data points to show trends, while markers denote individual data points in plots like scatter plots.
- **Title and Labels**: The title provides context for the plot, while axis labels describe what data is being represented on each axis.

Matplotlib Pyplot

Pyplot is a module within Matplotlib that provides a MATLAB-like interface for making plots. It simplifies the process of adding plot elements such as lines, images, and text to the axes of the current figure. **Steps to Use Pyplot:**

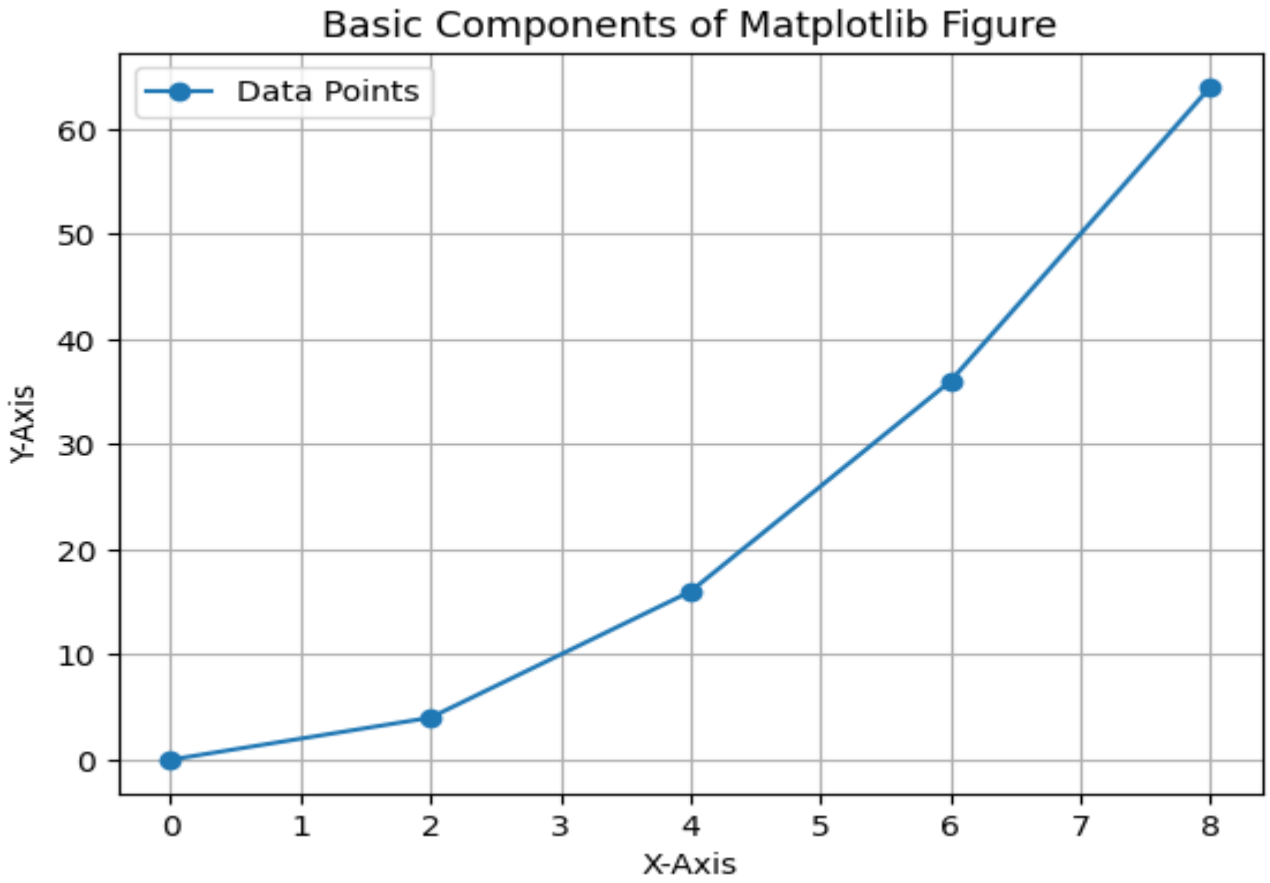
- **Import Matplotlib:** Start by importing matplotlib.pyplot as plt.
- **Create Data:** Prepare your data in the form of lists or arrays.
- **Plot Data**: Use plt.plot() to create the plot.

- **Customize Plot**: Add titles, labels, and other elements using methods like plt.title(), plt.xlabel(), and plt.ylabel().
- **Display Plot**: Use plt.show() to display the plot.

Let's visualize a basic plot, and understand basic components of matplotlib figure:

```
import matplotlib.pyplot as plt

x = [0, 2, 4, 6, 8]
y = [0, 4, 16, 36, 64]

fig, ax = plt.subplots()
ax.plot(x, y, marker='o', label="Data Points")

ax.set_title("Basic Components of Matplotlib Figure")
ax.set_xlabel("X-Axis")
ax.set_ylabel("Y-Axis")


plt.show()
```

Output:



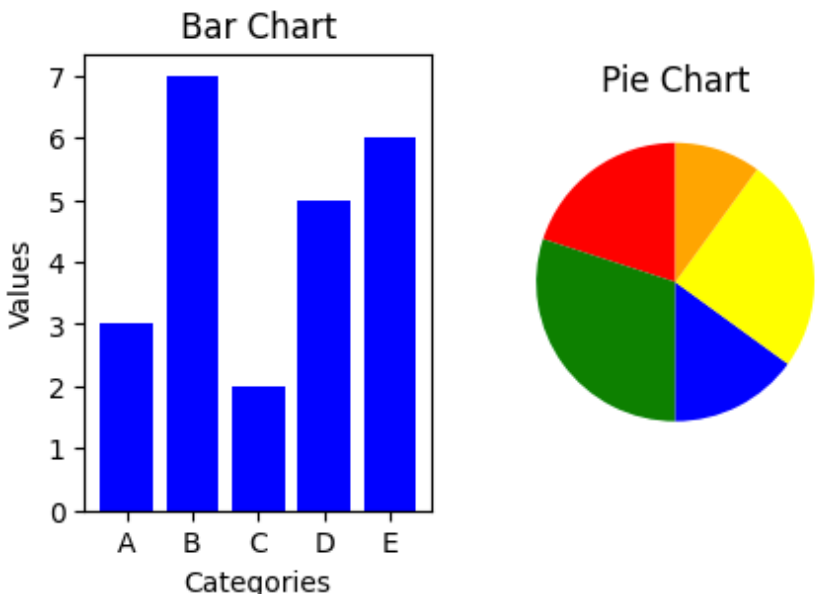
Basic Components of matplotlib figure

Different Types of Plots in Matplotlib

Matplotlib offers a wide range of plot types to suit various data visualization needs. Here are some of the most commonly used types of plots in Matplotlib:

- 1. Line Graph
- 2. Bar Chart
- 3. Histogram
- 4. Scatter Plot
- 5. Pie Chart
- 6. 3D Plot

and many more..



Bar chart and Pie chart

Key Features of Matplotlib

- **Versatile Plotting**: Create a wide variety of visualizations, including line plots, scatter plots, bar charts, and histograms.
- **Extensive Customization**: Control every aspect of your plots, from colors and markers to labels and annotations.
- **Seamless Integration with NumPy**: Effortlessly plot data arrays directly, enhancing data manipulation capabilities.
- **High-Quality Graphics**: Generate publication-ready plots with precise control over aesthetics.
- **Cross-Platform Compatibility**: Use Matplotlib on Windows, macOS, and Linux without issues.
- **Interactive Visualizations**: Engage with your data dynamically through interactive plotting features.

What is Matplotlib Used For?

Matplotlib is a Python library for data visualization, primarily used to create static, animated, and interactive plots. It provides a wide range of plotting functions to visualize data effectively.

Key Uses of Matplotlib:

- **Basic Plots:** Line plots, bar charts, histograms, scatter plots, etc.
- **Statistical Visualization:** Box plots, error bars, and density plots.
- **Customization:** Control over colors, labels, gridlines, and styles.
- **Subplots & Layouts:** Create multiple plots in a single figure.
- **3D Plotting:** Surface plots and 3D scatter plots using mpl_toolkits.mplot3d.
- **Animations & Interactive Plots:** Dynamic visualizations with FuncAnimation.
- **Integration:** Works well with Pandas, NumPy and Jupyter Notebooks.