Matplotlib

Matplotlib is a Python library used for creating static and interactive visualizations.
pyplot – A module in Matplotlib that provides a MATLAB-like interface for simple plotting.

note:

1. Data Visualization:
   - Helps in visualizing large datasets
   - Easy to identify trends, patterns, and correlations.
   - Converts raw data into informative graphs.
2. Supports Multiple Plot Types:
   - Line plots, bar charts, histograms, scatter plots, pie charts, box plots etc.
   - Supports Customization like (xlabel,ylabel,legend,xticks,etc)
3. Integration with Other Libraries: NumPy, Pandas, and Seaborn
4. Syntax to import the library: **import matplotlib.pyplot as plt**

Matplotlib Plotting Functions: Definitions and Syntax

plot()

Used to create line plots by plotting `x` and `y` coordinates.

Syntax:

```
plt.plot(x, y, linestyle, marker, color, label)
```

Parameters:

- `x and y – Data points for the X and Y axes.`
- `linestyle – Type of line (e.g., '-', '--', ':').`
- `marker – Style of markers (e.g., 'o', '^', '*').`
- `color – Line color (e.g., 'red', 'blue').`
- `label – Name of the line (used with legend()).`

Matplotlib Linestyle, Marker, and Color Reference Table

1. Linestyle Options

Symbol	Description
'-'	Solid line
'--'	Dashed line
'-.'	Dash-dot line
:	Dotted line
'' or ' '	No line (invisible)

3. Color Options

Symbol	Color Name	Hex Code
b	Blue	#0000FF
g	Green	#008000
r	Red	#FF0000
c	Cyan	#00FFFF
m	Magenta	#FF00FF
y	Yellow	#FFFF00
k	Black	#000000
w	White	#FFFFFF

2. Marker Options

Symbol	Description
.	Point marker
,	Pixel marker
o	Circle marker
v	Triangle down marker
^	Triangle up marker
<	Triangle left marker
>	Triangle right marker
1	Tri-down marker
2	Tri-up marker
3	Tri-left marker
4	Tri-right marker
s	Square marker
p	Pentagon marker
*	Star marker
h	Hexagon 1 marker
H	Hexagon 2 marker
+	Plus marker
x	X marker
D	Diamond marker
d	Thin diamond marker
|	Vertical line marker
_	Horizontal line marker

figure()

Creates a new figure or activates an existing one to plot multiple plots.

Syntax:

```
plt.figure(figsize=(width, height))
```

Parameters:

`figsize` – Tuple defining width and height in inches.

title()

Adds a title to the plot.

Syntax:

```
plt.title('Your Title', fontsize=value, color='color')
```

xlabel()

Adds a label to the X-axis.

Syntax:

```
plt.xlabel('X-axis Label', fontsize=value, color='color')
```

ylabel()

Adds a label to the Y-axis.

Syntax:

```
plt.ylabel('Y-axis Label', fontsize=value, color='color')
```

xticks()

Sets the locations and labels of the X-axis ticks.

Syntax:

```
plt.xticks(tick_values, tick_labels, rotation=value)
```

yticks()

Sets the locations and labels of the Y-axis ticks.

Syntax:

```
plt.yticks(tick_values, tick_labels, rotation=value)
```

legend()

Adds a legend to the plot to show labels of plotted data.

Syntax:

```
plt.legend(loc='position')
```

grid()

Adds a grid to the plot for better readability.

Syntax:

```
plt.grid(True, linestyle='--', color='gray')
```

show()

Displays the plot.

Syntax:

```
plt.show()
```

subplot()

Creates multiple plots in a grid layout within a single figure.

Syntax:

```
plt.subplot(nrows, ncols, index)
```

subplots()

Creates multiple subplots and returns a figure and axes object.

Syntax:

```
fig, ax = plt.subplots(nrows, ncols, figsize=(width, height))
```

suptitle()

Adds a title for the entire figure when using subplots.

Syntax:

```
plt.suptitle('Main Title', fontsize=value, color='color')
```

set_xlabel()

Sets the label for the X-axis on a specific subplot.

Syntax:

```
ax.set_xlabel('X-axis Label')
```

set_ylabel()

Sets the label for the Y-axis on a specific subplot.

Syntax:

```
ax.set_ylabel('Y-axis Label')
```

tight_layout() automatically adjusts the spacing between subplots to prevent overlapping and make the layout look better.

Syntax:

```
plt.tight_layout(rect)
```

Seaborn

Seaborn is a Python data visualization library built on top of **Matplotlib**

Seaborn works well with **Pandas DataFrames** and provides built-in functions to handle datasets, making data visualization simpler and more intuitive

Syntax:

```
pip install seaborn

import seaborn as sns
```

Why use plots/graphs?

- Makes data easier to understand.
- Helps identify trends, patterns, and outliers.
- Simplifies complex data by presenting it visually.
- Supports better decision-making with clear insights.

In data visualization, plots are often classified into **Univariate**, **Bivariate**, and **Multivariate** based on the number of variables they represent.

1) **Univariate:** Focuses on one variable .

   - Count Plot
   - KDE Plot
   - Histplot
   - Pie Chart

2) **Bivariate:** Focuses on the relationship between two variables .

   - Scatter Plot
   - Line Plot
   - Bar Plot
   - Box Plot
   - Jointplot

3) **Multivariate:** Focuses on the relationship between three or more variables .

   - Pair Plot
   - Heatmap

1) Count Plot:

Displays the count of each category in a column.

Syntax: `sns.countplot(data=df , x='column_name' , hue)`

2) KDE Plot (Kernel Density Estimate):

Shows the smooth distribution of a numeric variable.

Syntax: `sns.kdeplot(x='column_name',data=df ,hue,fill=True)`

3) Histplot (Histogram Plot):

Shows the distribution of numeric data by dividing it into bins.

Syntax: `sns.histplot(x='column_name',data=df,kde=True,hue)`

4) Pie Chart:

Shows the proportion of each category as slices of a circle.

Syntax: plt.pie(values, labels=categories, autopct , explode)

5) Scatter Plot:

Shows the relationship between two numeric variables using points.

Syntax: sns.scatterplot(x='column1', y='column2', data=df,
Hue , style , size)

6) Line Plot:

Shows the relationship between two numeric variables using a continuous line.

Syntax: `sns.lineplot(x='column1', y='column2', data=df)`

7) Bar Plot:

Shows the relationship between a categorical variable and a numeric variable using bars.

Syntax: `sns.barplot(x='category_column', y='value_column',`
`data=df , estimator='mean', hue)`

`estimator can be mean , median , size(count) , max , min etc..`

8) Box Plot:

Displays the distribution of numeric data and identifies outliers.

Syntax: `sns.boxplot(x='category_column', y='value_column',`
`data=df , hue)`

10) Jointplot:

Combines scatter plot and distribution plots (kde / histplot) to show the relationship between two numeric variables.

Syntax: `sns.jointplot(x='column1', y='column2', data=df)`

11) Heatmap:

Displays data in a matrix form using color to represent values.

Syntax: `sns.heatmap(dataframe_corr, annot=True, cmap)`

12) Pairplot:

Plots pairwise relationships between numeric variables in a dataset.

Syntax: `sns.pairplot(df)`