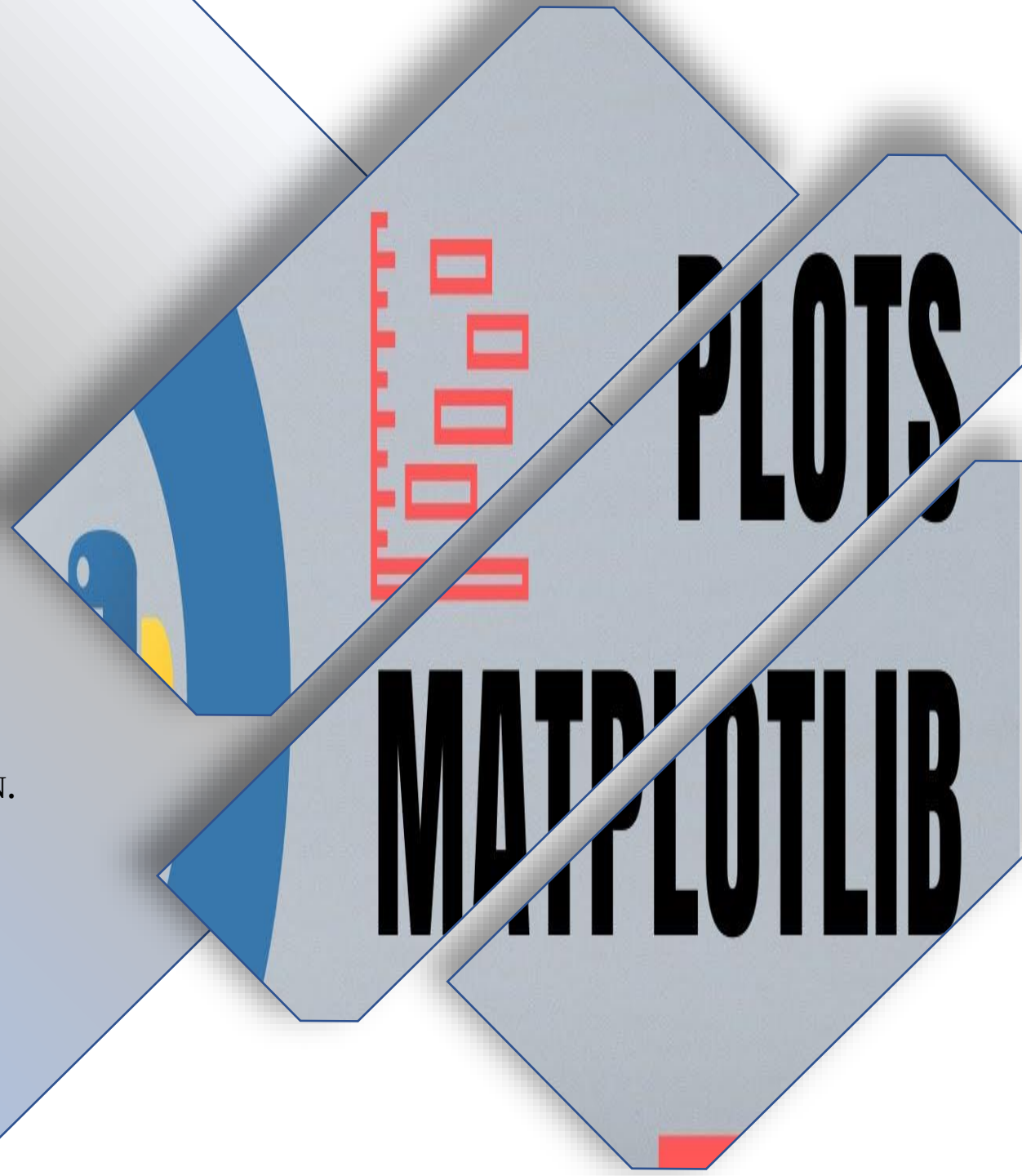
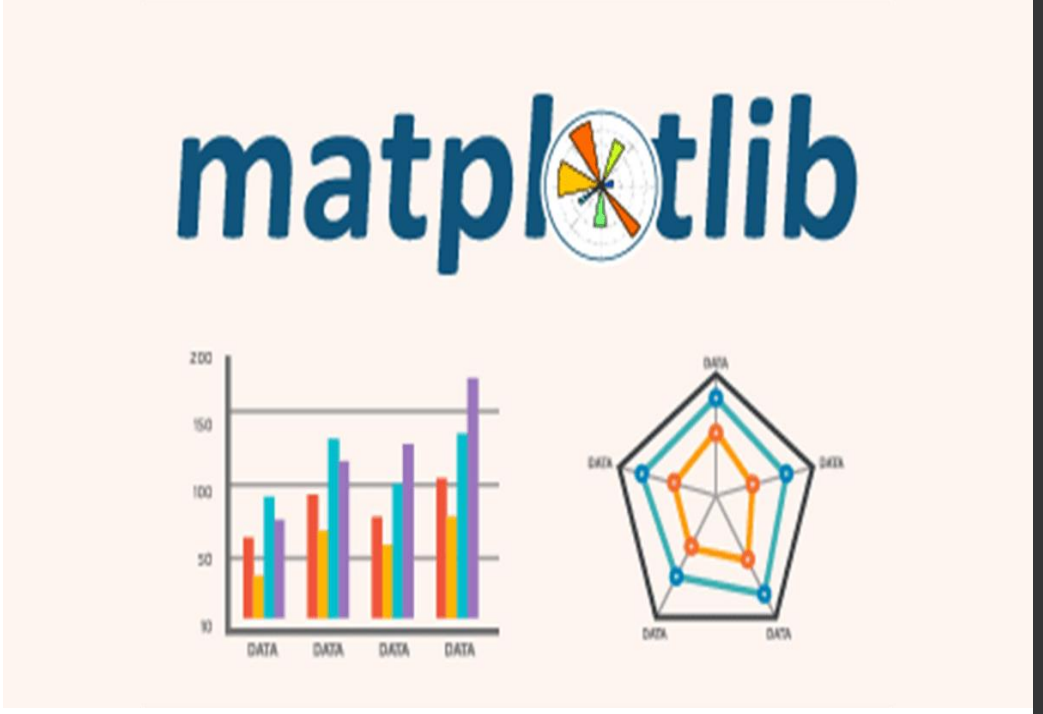
MATPLOTLIB

A Python Library for Data Visualization.



INTRODUCTION TO MATPLOTLIB

- Matplotlib is a plotting library for Python, widely used for creating static, animated, and interactive visualizations.
- The core plotting module is pyplot, often imported as plt.
- It is designed for creating publication-quality graphs and visualizations.
- Supports a variety of plot types like line, scatter, bar, histogram, pie, and more.
- Works seamlessly with NumPy, pandas, and other data manipulation libraries.
- Highly customizable for adjusting plot properties like color, style, labels, and ticks.
- Supports multiple backends, which allows for rendering on different platforms.
- Can be used in Jupyter notebooks for interactive plotting.
- Offers support for 3D plotting via the mplot3d toolkit.
- Can be extended with custom functions and libraries.
- Works on different platforms like Windows, macOS, and Linux.
- It's one of the most widely used libraries for data visualization in Python.



BASIC PLOTTING

- Simple Line Plot:

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

- Add labels;

```
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
```

- Title: plt.title('Title of the Plot') to give your plot a title.
- Grid: plt.grid(True) to display gridlines.
- Multiple Lines: Plot multiple lines by calling plt.plot(x, y) multiple times.
- Show Plot: plt.show() is used to display the plot.
- Line Styles: Use linestyle='--' for dashed, '-.' for dash-dot, etc.
- Line Colors: You can specify color using names or hex values, e.g., color='r' or color='#FF5733'.
- Markers: Add markers to lines using marker='o', marker='x', etc.
- Legend: Display legends with plt.legend().
- Save Plot: Use plt.savefig('filename.png') to save the plot to a file.
- Show Multiple Plots: You can plot multiple lines in a single plot by calling plt.plot() multiple times.

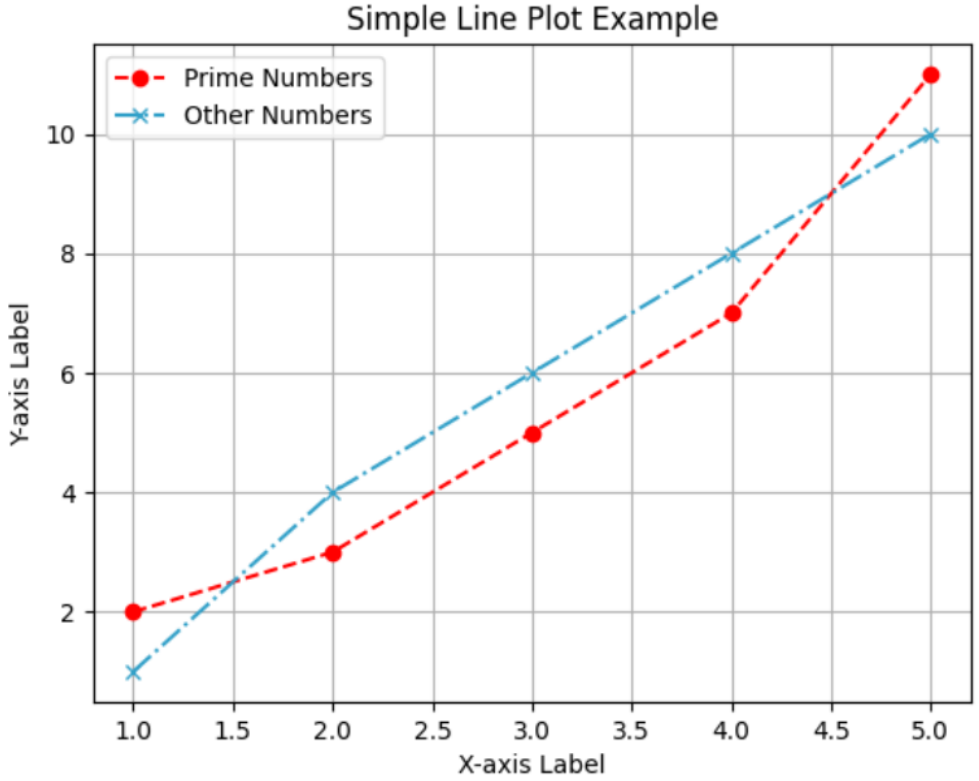
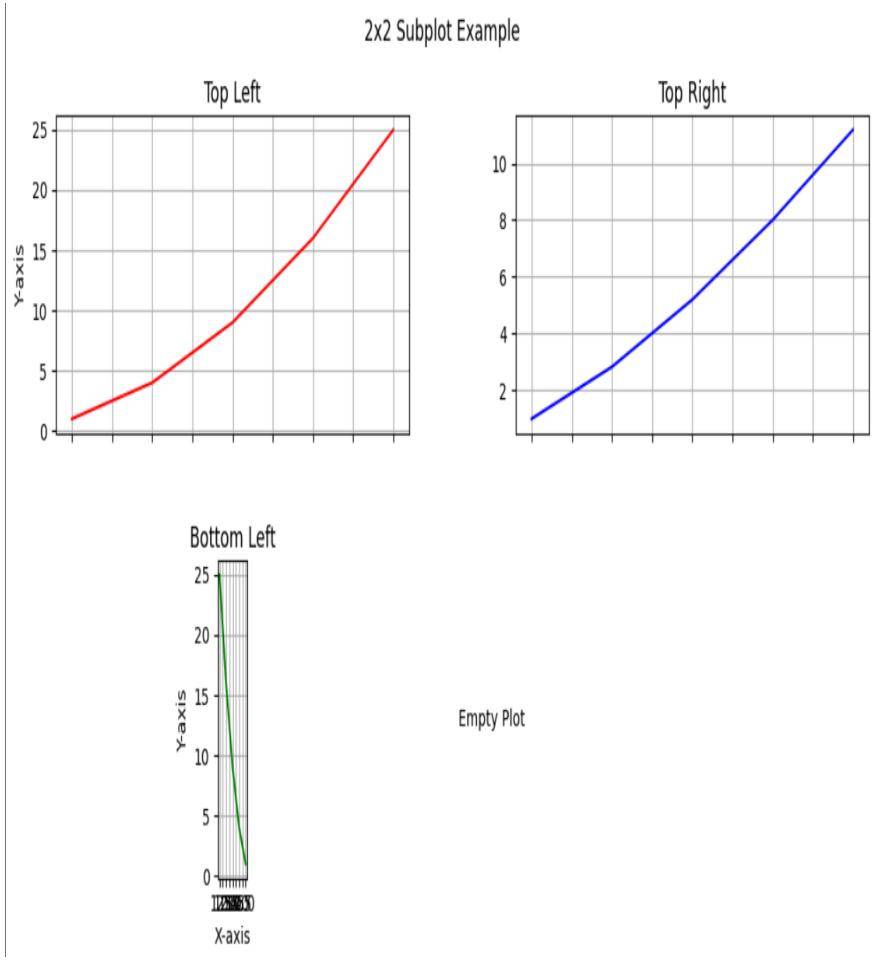


FIGURE AND AXES

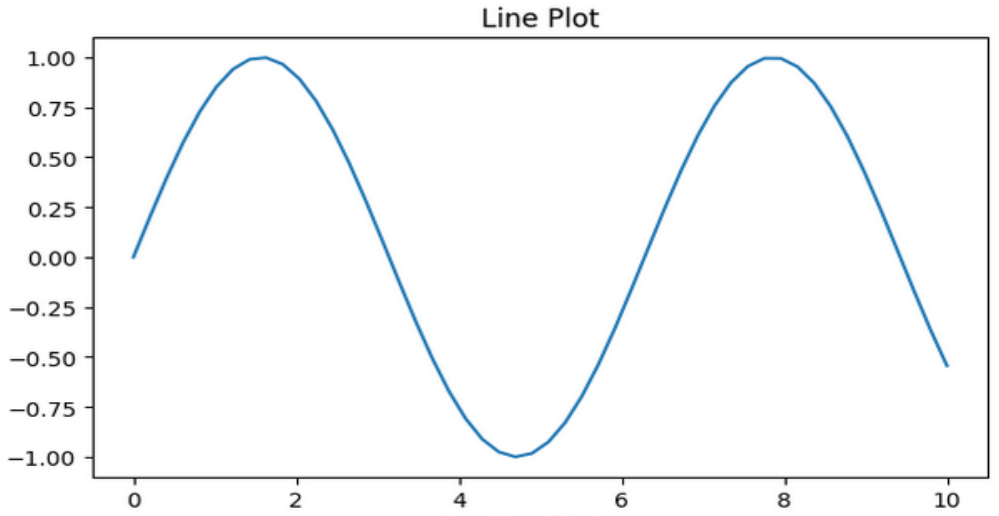
- Create Figure: fig = plt.figure() to create a new figure.
- Subplot: ax = fig.add_subplot(111) to add a subplot with grid specification (rows, columns, index).
- Multiple Subplots: Use plt.subplots(2, 2) to create a 2x2 grid of subplots.
- Access Axes: You can manipulate the axes with methods like ax.set_title(), ax.set_xlabel(), etc.
- Adjusting Size: Adjust the size of the figure using fig.set_size_inches().
- Share Axes: Share the x or y axis between subplots using plt.subplots(2, 2, sharex=True) or sharey=True.
- Grid Lines: ax.grid(True) to add grid lines to a subplot.
- Aspect Ratio: Control the aspect ratio of the plot using ax.set_aspect('equal').
- Tight Layout: Use plt.tight_layout() to automatically adjust subplot parameters for a better fit.
- Add Titles: Set titles for each subplot with ax.set_title('Subplot Title').
- Remove Axes: You can remove axes with ax.set_axis_off().
- Subplot Adjustment: plt.subplots_adjust() can be used to fine-tune the space between subplots.



PLOT TYPES

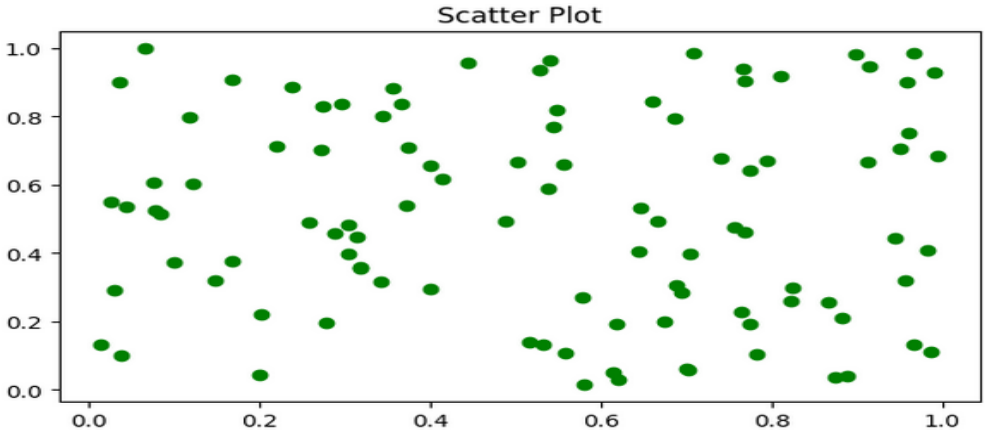
LINE PLOT

- Line plots connect individual data points with lines.
- They are ideal for showing trends over time or continuous data.
- Widely used in time series analysis and scientific data visualization.
- Helps in observing patterns, growth, or decline over intervals.



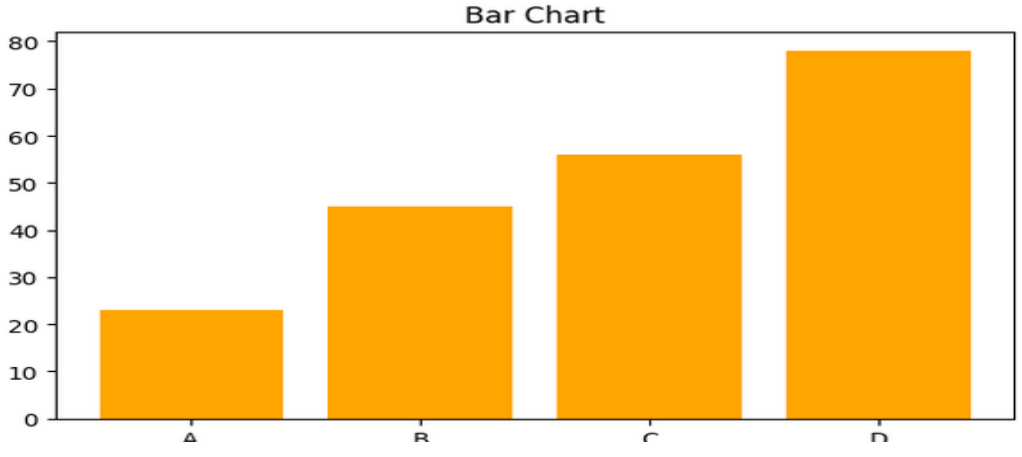
SCATTER PLOT

- Scatter plots display data as individual points on a plane.
- Useful for showing relationships or correlations between variables.
- Each point represents one observation in a dataset.
- Good for identifying clusters, outliers, and trends.



BAR CHART

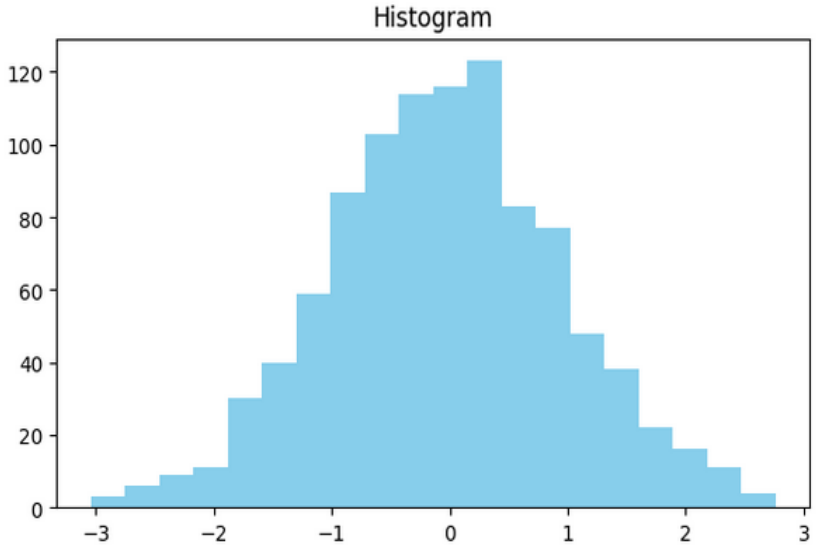
- Bar charts represent categorical data with rectangular bars.
- Heights of bars correspond to the values of each category.
- They make comparisons between groups easy and intuitive.
- Can be oriented vertically or horizontally.



PLOT TYPES

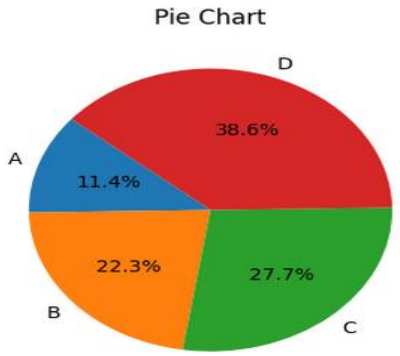
HISTOGRAM

- Histograms show the frequency distribution of numerical data.
- Data is grouped into bins, and bars represent the number of data points in each bin.
- They are useful for understanding the shape of the data distribution.
- Commonly used in statistical analysis and data exploration.



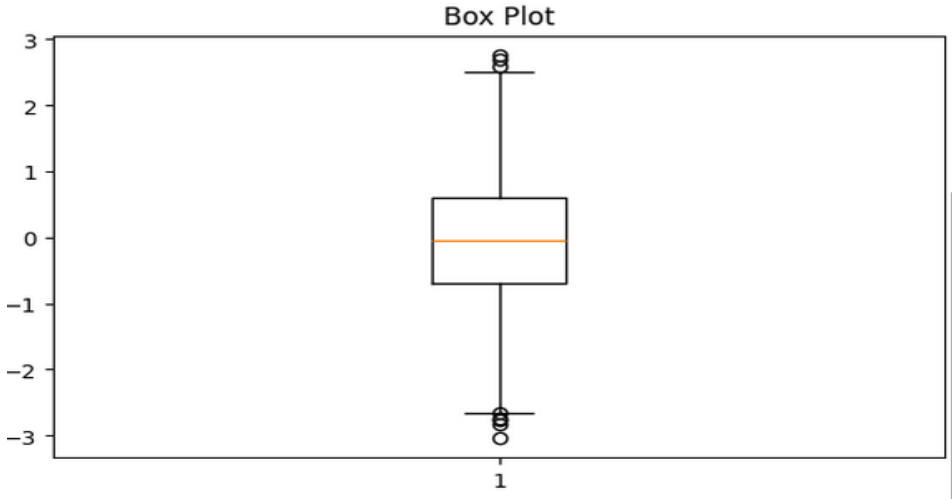
PIE CHART

- Pie charts show parts of a whole as slices of a circle.
- Each slice's size is proportional to its percentage of the total.
- Useful for visualizing simple proportions and percentages.
- Best used when there are limited categories for comparison.



BOX PLOT

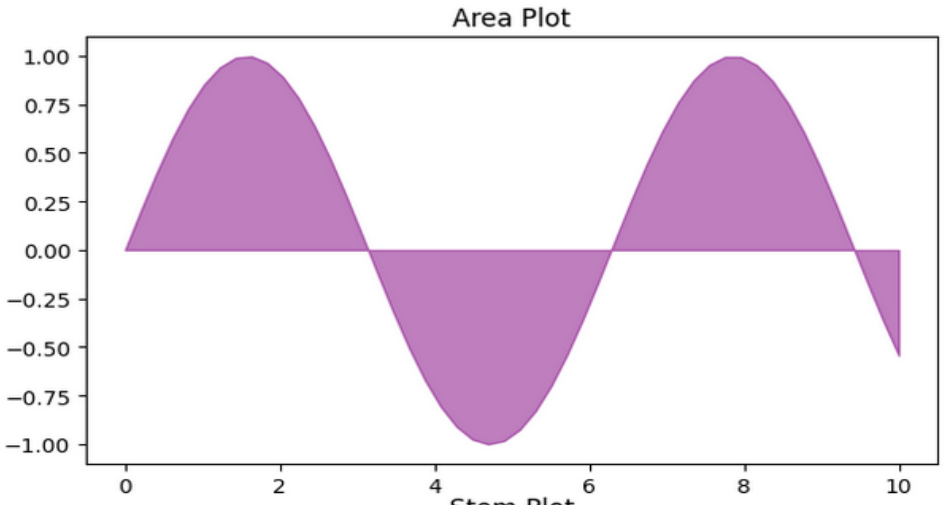
- Box plots display the distribution of data using five-number summary.
- These include minimum, Q1, median, Q3, and maximum.
- Outliers are shown as individual points outside the whiskers.
- Helpful for comparing distributions and detecting skewness.



PLOT TYPES

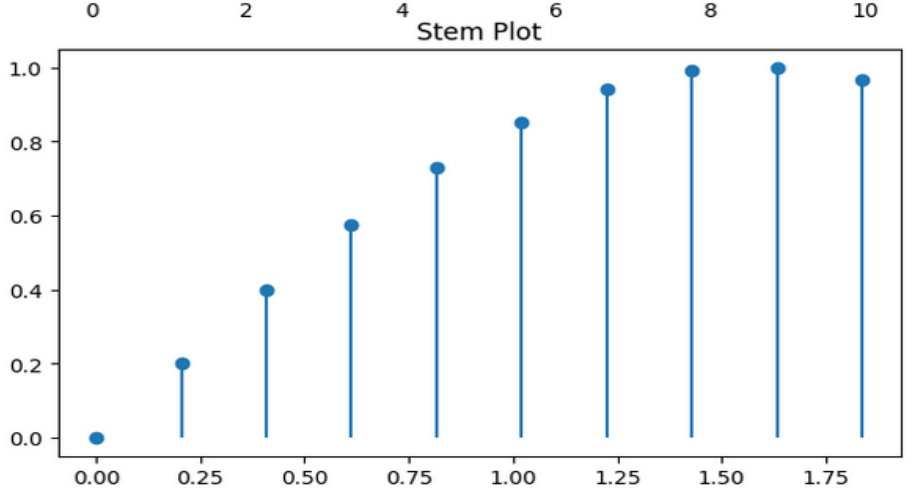
AREA PLOT

- Area plots are like line plots but with the area under the line filled.
- They emphasize the magnitude of change over time.
- Often used to show cumulative totals or stacked data.
- Effective for visualizing trends in quantities across categories.



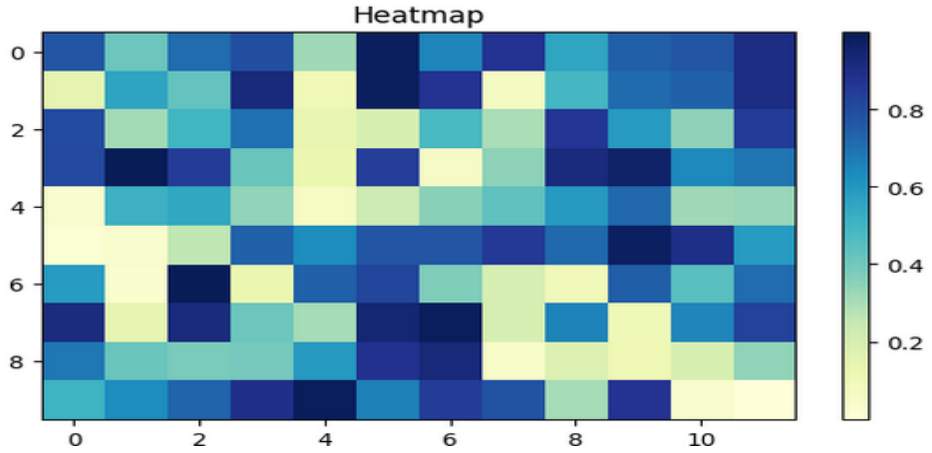
STEM PLOT

- Stem plots show data points as stems with markers at their ends.
- They are ideal for small, discrete datasets.
- Provide a clear view of individual values and their distribution.
- Less common but useful for educational or technical purposes.



Heatmap

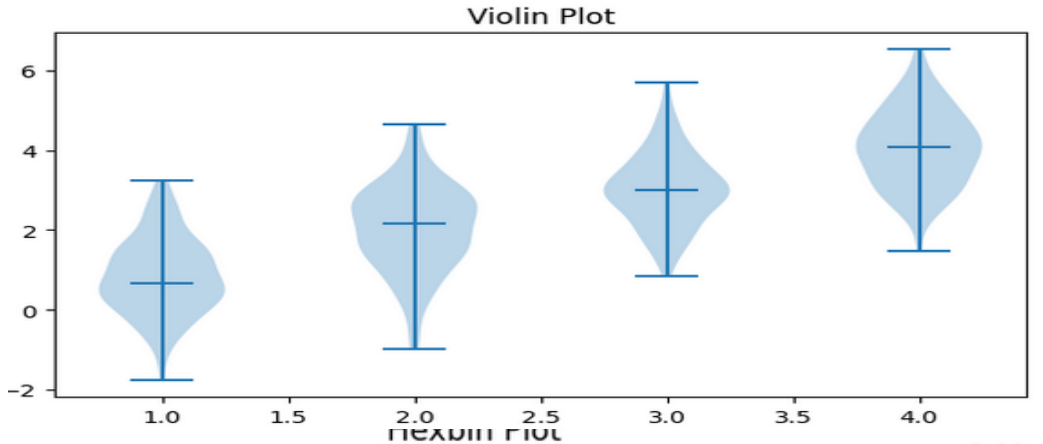
- Heatmaps use color to represent values in a matrix.
- Each cell's color intensity reflects the data magnitude.
- Common in correlation matrices and image data.
- Great for identifying patterns and anomalies visually.



PLOT TYPES

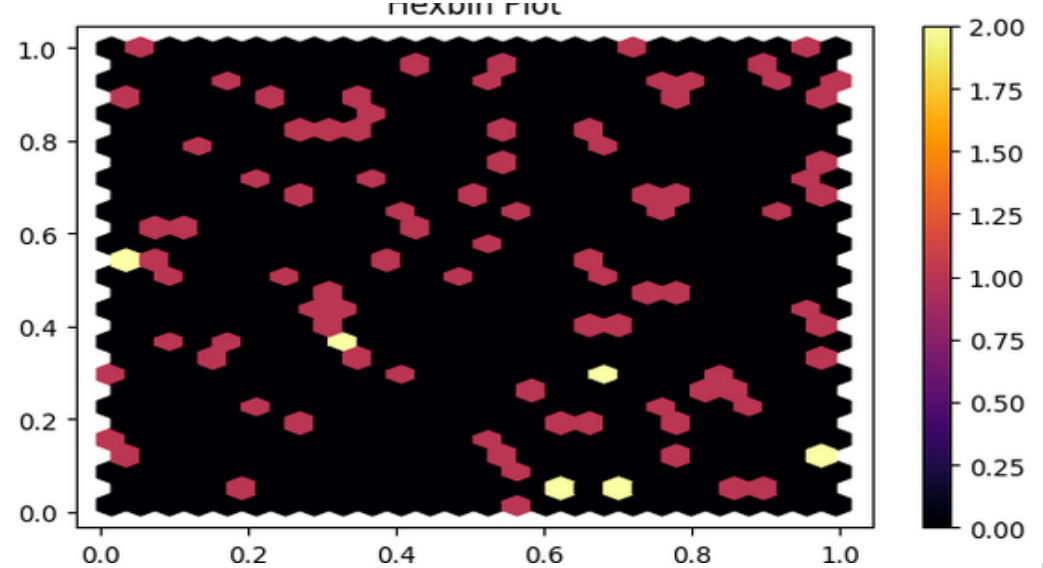
VIOLIN PLOT

Violin plots combine box plots and KDE (kernel density estimate).
They show the distribution, spread, and probability density.
Helpful in comparing multiple distributions simultaneously.
Useful for visualizing complex datasets with many observations.



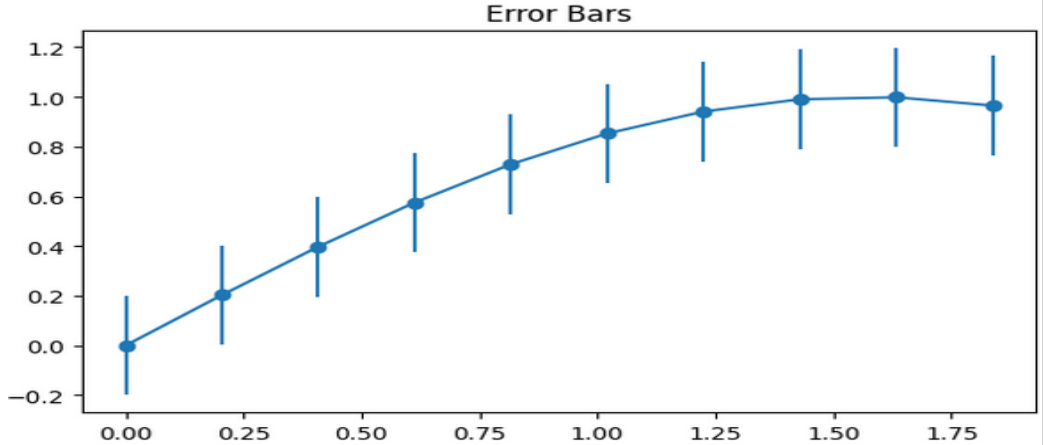
HEXBIN PLOT

Hexbin plots bin 2D data points into hexagonal areas.
Color intensity represents the number of points in each bin.
Useful when dealing with large datasets with overlapping points.
Great for density estimation and spatial data visualization.



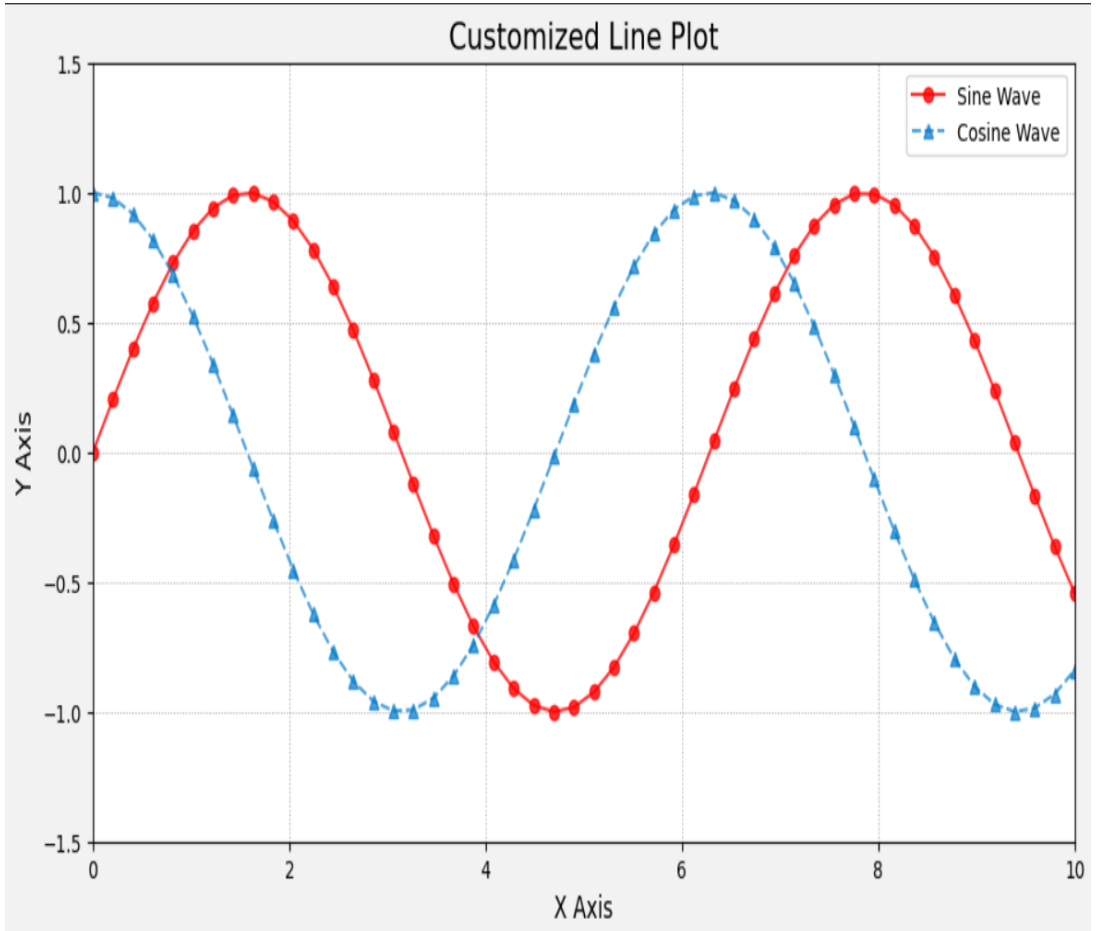
ERROR BARS

Error bars show uncertainty or variability in data points.
They can represent standard deviation, confidence intervals, or errors.
Useful in scientific and engineering plots to show reliability.
Can be applied in both x and y directions.



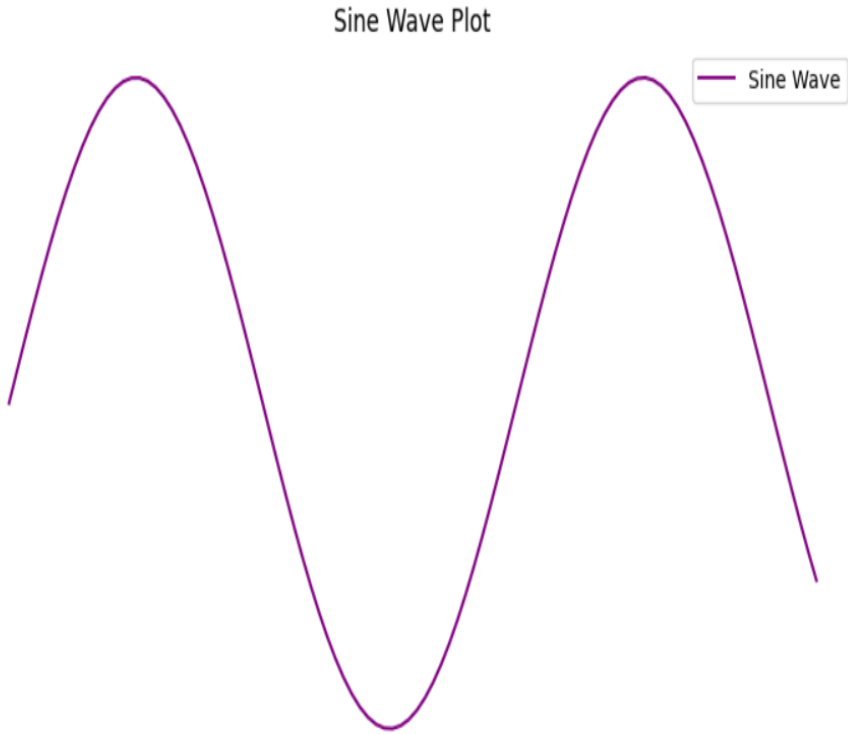
STYLING AND CUSTOMIZATION

- Line Styles: '-', '--', '-.', ':'.
- Markers: 'o', 'x', 's', '^', etc.
- Colors: Use named colors ('red', 'green'), hex codes (#FF5733), or RGB tuples.
- Transparency: Control transparency with alpha=0.5.
- Font Size: Adjust font size using plt.xlabel('Label', fontsize=12).
- Tick Marks: Customize ticks with plt.xticks(), plt.yticks().
- Axis Limits: Set limits with plt.xlim() and plt.ylim().
- Grid Lines: Customize grid lines using plt.grid().
- Legend: Use plt.legend() to add a legend, with options like loc='best'.
- Subplot Spacing: Adjust subplot spacing with plt.subplots_adjust().
- Background Color: Use fig.patch.set_facecolor() to change the figure background.



SAVING FIGURES

- Save to File: Use plt.savefig('filename.png') to save the current figure.
- File Formats: Supports .png, .jpg, .svg, .pdf, etc.
- DPI: Control the resolution using the dpi argument, e.g., plt.savefig('plot.png', dpi=300).
- Transparent Background: Save with a transparent background using transparent=True.
- Bounding Box: Use bbox_inches='tight' to adjust bounding box when saving.
- Saving with Legends: Legends are saved with plt.legend() in the saved file.
- Saving Multiple Figures: Each figure can be saved with different filenames.
- Tight Layout: Save the plot with a tight layout using plt.tight_layout() before saving.
- No Axes: Save a plot without axes using ax.set_axis_off() before saving.
- PDF Saving: Save plots to PDF for vector graphics output.
- High-Resolution Images: For high-quality images, set a higher dpi in savefig().
- Saving with Custom DPI: Specify DPI while saving: plt.savefig('plot.png', dpi=150).



THANK YOU