Data Visualization Using Matplotlib and Plotly

VISUALIZATION

- Visualization is the process of representing data or information in a graphical or pictorial form. It helps to understand large amounts of data easily by showing patterns, trends, and relationships through visuals like charts, graphs, and maps.
- Instead of looking at raw numbers, visualization allows us to see data in a clear and meaningful way. It helps in making decisions faster and communicating information effectively. Common tools used for visualization include Microsoft Excel, Tableau, Power BI, and programming libraries like Matplotlib and Seaborn in Python.
- In simple terms, visualization turns data into pictures that make it easier to analyze, compare, and share insights.

MATPLOTLIB

- Matplotlib is a Python library used for creating graphs and charts from data. It helps to visualize data in an easy and understandable way. With Matplotlib, we can draw different types of charts like line charts, bar charts, pie charts, histograms, and scatter plots.
- It is mostly used in data analysis and machine learning projects to show patterns and trends in data. Matplotlib works well with other Python libraries like NumPy and Pandas.
- A plot of matplotlib contains:

  * Figure
  * Axes
  * Axis
  * Artists

Figure

- It is the overall container or canvas that holds all plots and elements.
- A Figure can contain one or more Axes (plots) inside it.

Axes

- The area where data is actually plotted (contains the chart).
- Each Axes can have X and Y Axis and multiple Artists (lines, labels, etc.).

Axis

- Represents the scale and direction of the data (X-axis and Y-axis).
- Controls limits, ticks, and labels for the plot.

Artists

- All visual elements like lines, text, titles, labels, and legends.
- Everything that appears on the plot is considered an Artist in Matplotlib.

PYPLOT

Pyplot is a module of Matplotlib that provides all the basic plotting functions in one place. It makes creating charts and graphs simple, such as line plots, bar charts, scatter plots, pie charts, histograms, and area charts.

To use Pyplot, we import it from Matplotlib like this:

- import matplotlib.pyplot as plt

We also import **NumPy** to help with numerical calculations and data handling:

- import numpy as np

Some of the plots in Matplotlib:

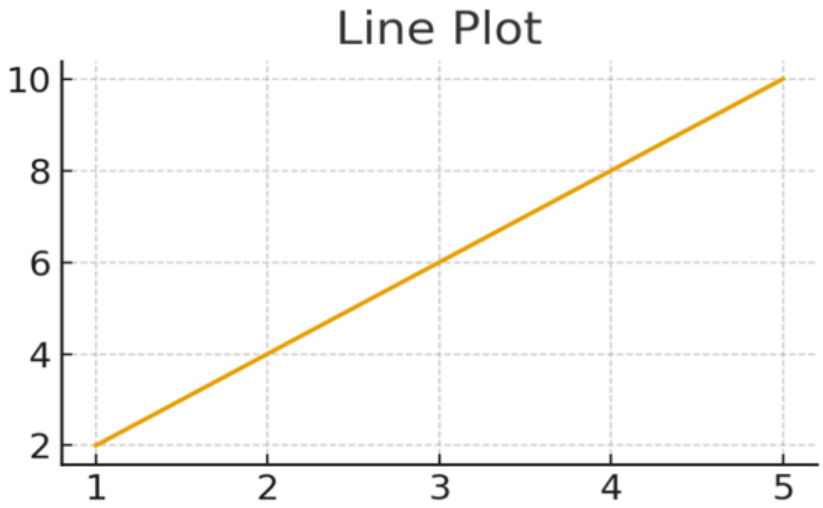
LINE PLOT

Explanation: Shows the relationship between two variables or trends over time.
Input: Lists or arrays for X and Y values.

Code Example:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

plt.plot(x, y)

plt.title("Line Plot Example")

plt.xlabel("X-Axis")

plt.ylabel("Y-Axis")

plt.show()
```

Output: A simple line connecting points (1,2), (2,4), … showing an increasing trend.



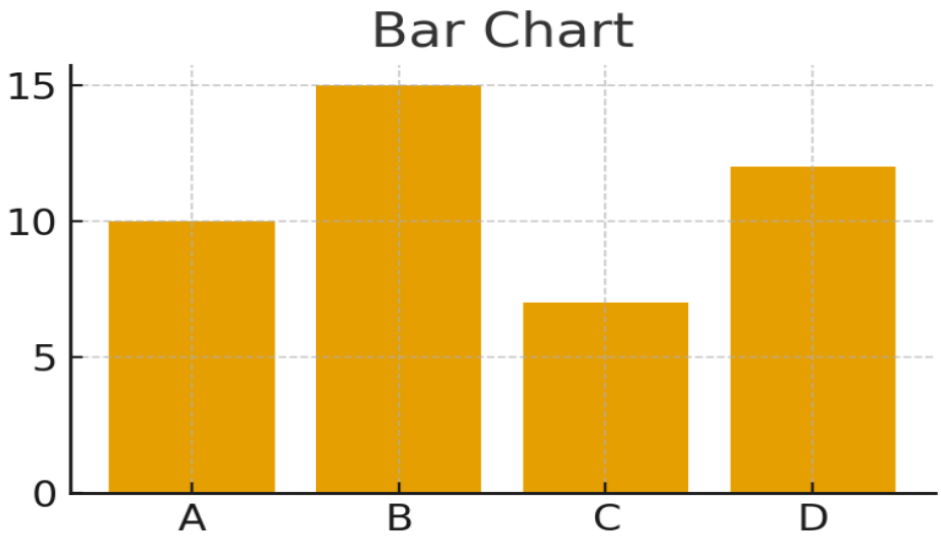
BAR CHART

Explanation: Compares values across different categories.
Input: Lists of categories and their corresponding values.

Code Example:

```
categories = ['A', 'B', 'C', 'D']

values = [10, 15, 7, 12]

plt.bar(categories, values)

plt.title("Bar Chart Example")

plt.show()
```

Output: Vertical bars representing the values of each category.



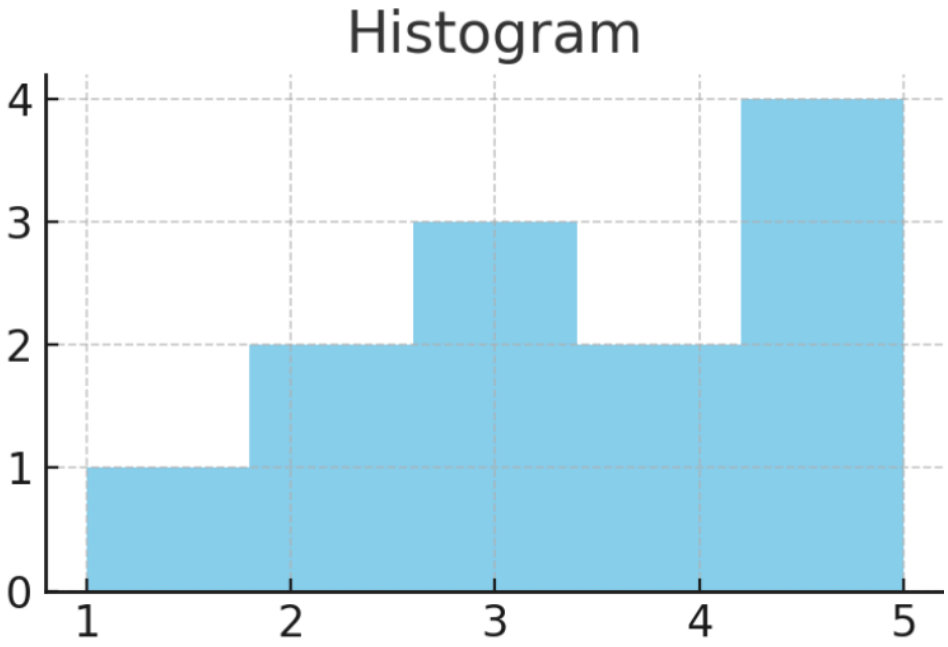
HISTOGRAM

Explanation: Shows frequency distribution of data in intervals (bins).
Input: List or array of numerical data.

Code Example:

```
data = [1,2,2,3,3,3,4,4,5,5,5,5]

plt.hist(data, bins=5, color='skyblue')

plt.title("Histogram Example")

plt.show()
```

Output: Bars showing how many times each value or range occurs.



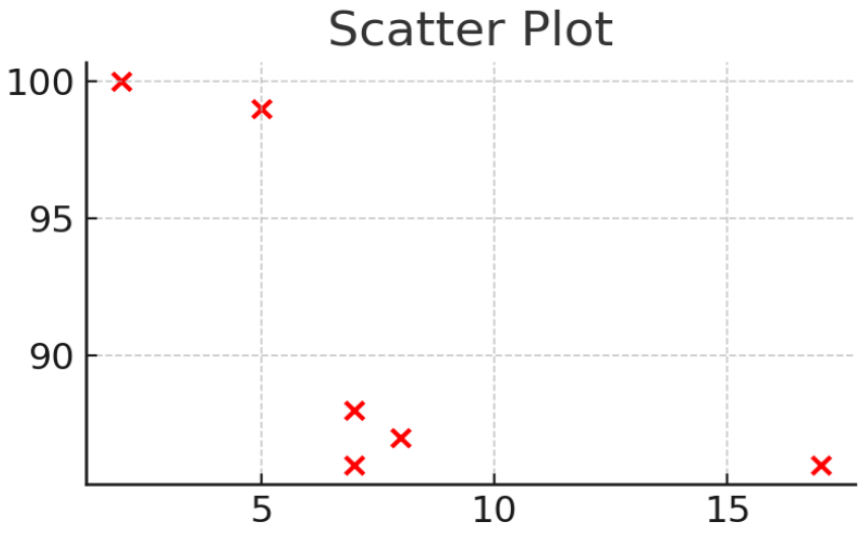
SCATTER PLOT

Explanation: Shows the relationship or correlation between two variables.
Input: Lists or arrays for X and Y values.

Code Example:

```
x = [5, 7, 8, 7, 2, 17]

y = [99, 86, 87, 88, 100, 86]

plt.scatter(x, y, color='red')

plt.title("Scatter Plot Example")

plt.xlabel("X-Axis")

plt.ylabel("Y-Axis")

plt.show()
```

Output: Individual points plotted on X-Y axes showing the data distribution.



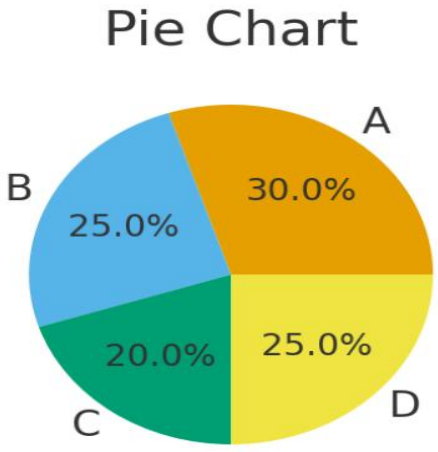
PIE CHART

Explanation: Shows proportions or percentages of a whole.
Input: List of values and labels.

Code Example:

```
sizes = [30, 25, 20, 25]

labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%')

plt.title("Pie Chart Example")

plt.show()
```

Output: Circular chart divided into slices proportional to the values.



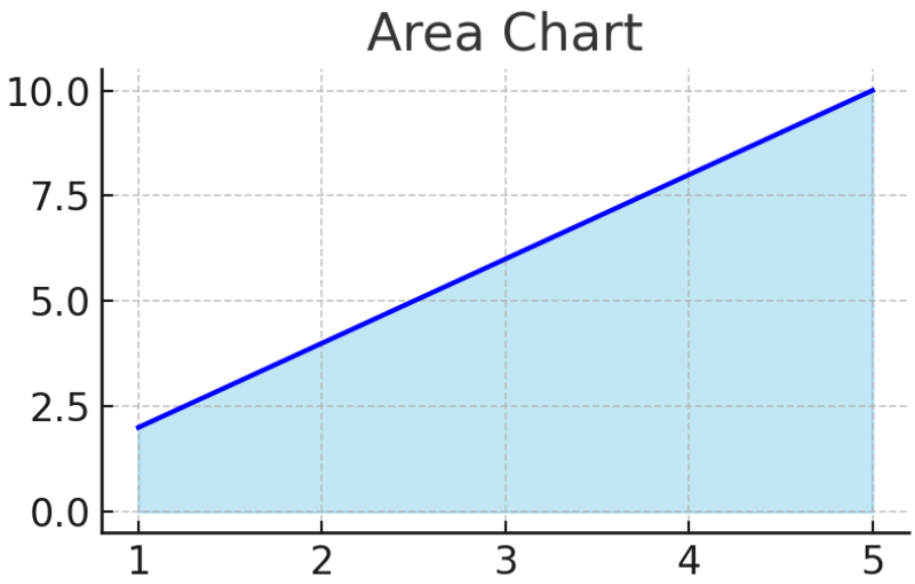
AREA CHART

Explanation: Similar to a line plot but highlights the area under the curve.
Input: Lists or arrays for X and Y values.

Code Example:

```
x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

plt.fill_between(x, y, color="skyblue", alpha=0.5)

plt.plot(x, y, color="blue")

plt.title("Area Chart Example")

plt.show()
```

Output: Line plot with the area under the line filled with color.



PLOTLY

- Plotly is a Python library used to create interactive and dynamic visualizations. Unlike Matplotlib, which produces static images, Plotly lets you zoom, hover, and explore data in real-time.
- It's widely used for data analysis, dashboards, and machine learning visualization because of its beautiful and customizable charts.
- Plotly works well with other Python libraries like Pandas and NumPy, and can even integrate with Dash to build full-fledged web applications.

PLOTLY EXPRESS

Plotly Express (px) is a high-level interface of Plotly that makes plotting super easy with just one line of code. It supports a wide range of charts such as line plots, bar charts, scatter plots, pie charts, histograms, area charts, and even 3D and map-based visualizations.

To use Pyplot, we import it from Matplotlib like this:

- import plotly.express as px

Some of the plots in Plotly:

LINE PLOT

Explanation: Shows the relationship between two variables or trends over time, just like in Matplotlib — but interactively (hover, zoom, and pan).

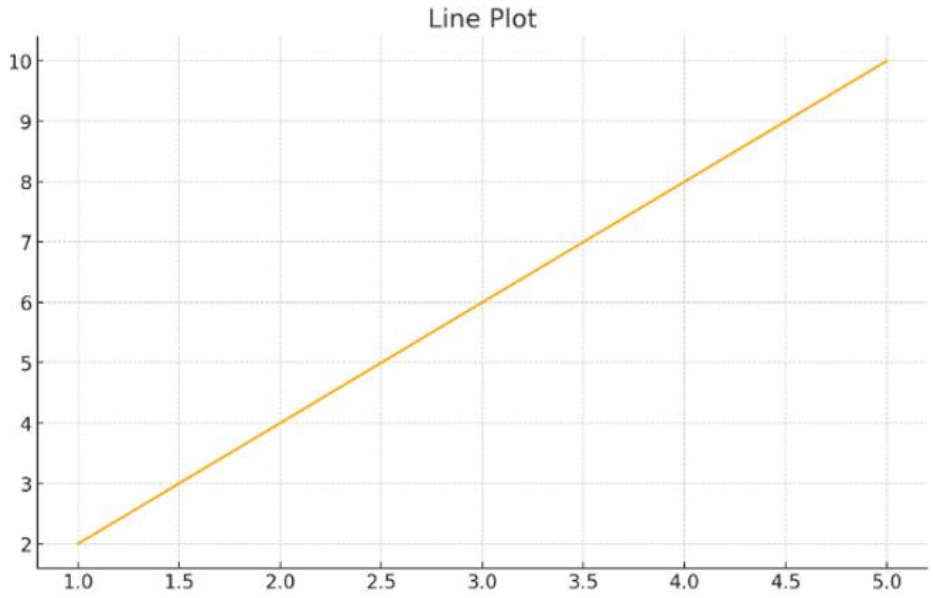
Input: Lists or arrays for X and Y values.

Code Example:

```
import plotly.express as px

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

fig = px.line(x=x, y=y, title="Line Plot Example", labels={'x':'X-Axis', 'y':'Y-Axis'})

fig.show()
```

Output:

An interactive line chart where you can hover over points to see exact values and zoom in/out easily.



BAR CHART

Explanation: Compares values across different categories using vertical or horizontal bars.

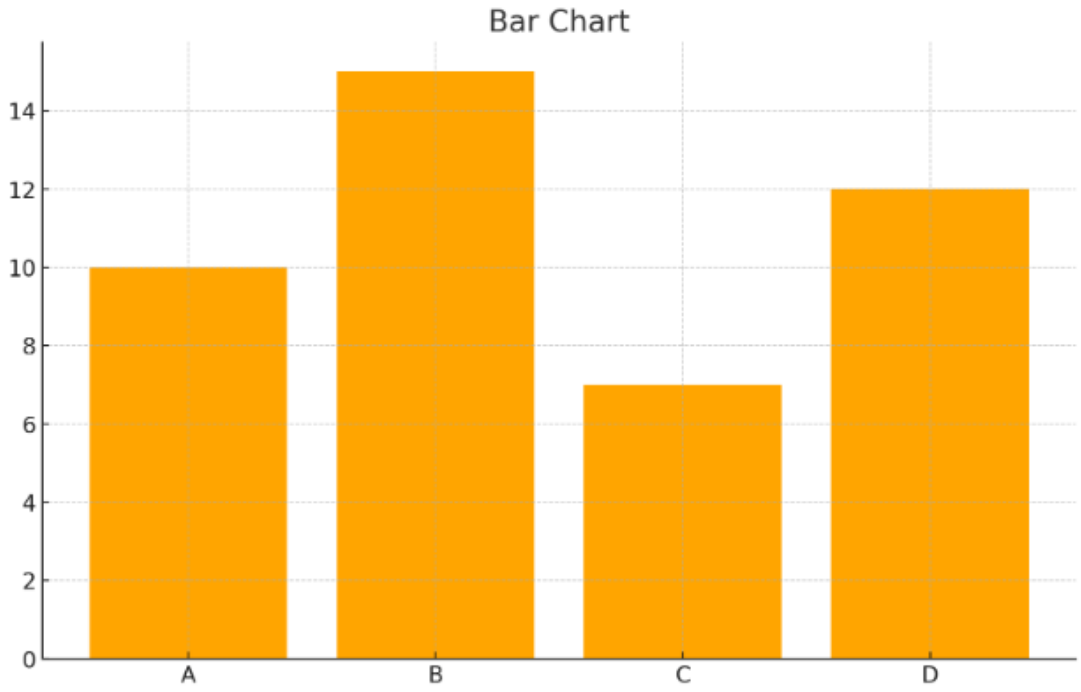
Input: Lists of categories and their corresponding values.

Code Example:

```
import plotly.express as px

categories = ['A', 'B', 'C', 'D']

values = [10, 15, 7, 12]

fig = px.bar(x=categories, y=values, title="Bar Chart Example", labels={'x':'Categories', 'y':'Values'})

fig.show()
```

Output:

Interactive bars that highlight and show tooltips when hovered over.



Statistical & Analytical Charts

Tree Map

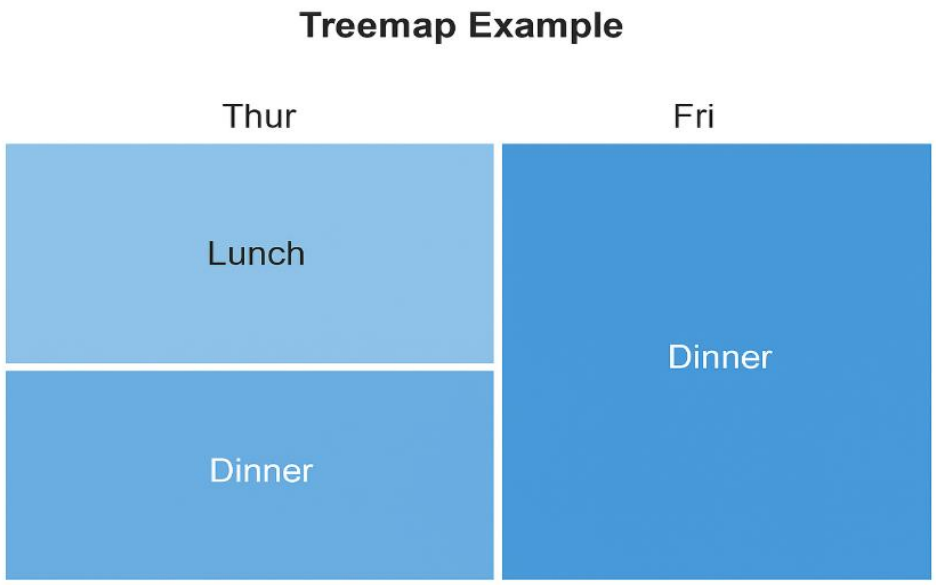
Explanation: Represents hierarchical data using nested rectangles — perfect for category proportions.

Code Example:

```
import plotly.express as px
df = px.data.tips()
fig = px.treemap(df, path=['day', 'time'], values='total_bill', title="Treemap Example")
fig.show()
```

Output:

Interactive boxes sized by total bill, grouped by day and time.



Geographical Charts

Choropleth Map

Explanation:

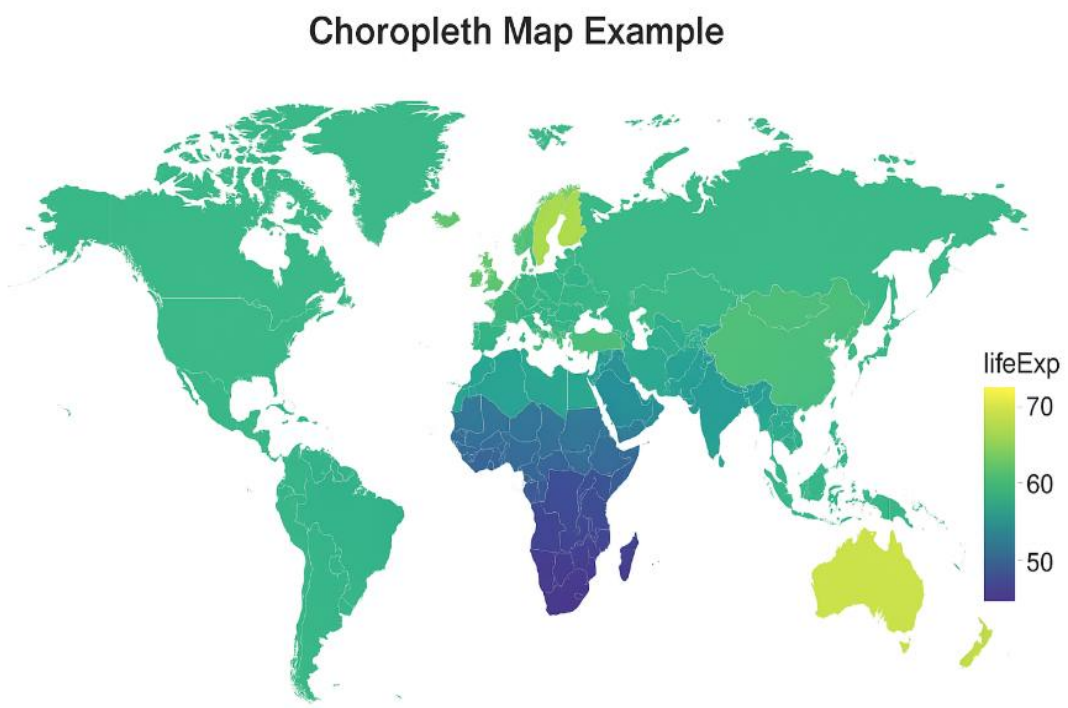
Shows how a metric varies across geographic regions (like countries).

Code Example:

```
import plotly.express as px
df = px.data.gapminder().query("year == 2007")
fig = px.choropleth(df, locations="iso_alpha", color="lifeExp", hover_name="country",
color_continuous_scale="Viridis", title="Choropleth Map Example")
fig.show()
```

Output:

Interactive world map where countries are shaded by life expectancy.



3D Visualizations

3D Scatter Plot

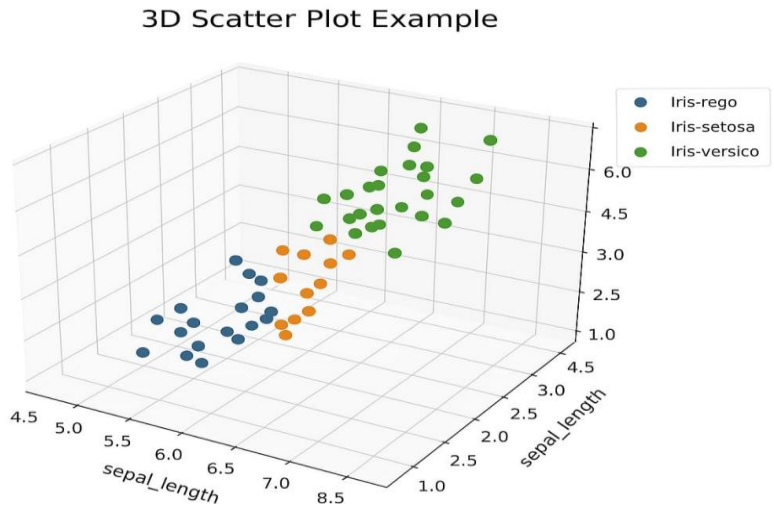
Explanation:

Shows relationships among three numeric variables with depth perception.

Code Example:

```
import plotly.express as px

df = px.data.iris()

fig = px.scatter_3d(df, x='sepal_length', y='sepal_width', z='petal_length'  color='species',
title="3D Scatter Plot Example")

fig.show()
```

Output: A fully rotatable 3D scatter plot — hover shows species and dimensions.



Advanced Interactive Charts

Funnel Chart

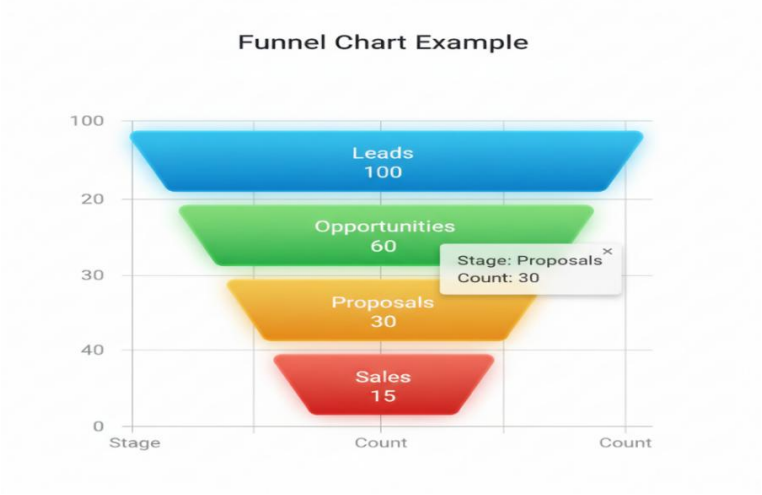
Explanation:

Shows stages in a process (like sales or conversion pipelines).

Code Example:

```
import plotly.express as px
data = {'Stage': ['Leads', 'Opportunities', 'Proposals', 'Sales'],
        'Count': [100, 60, 30, 15]}
fig = px.funnel(data, x='Count', y='Stage', title="Funnel Chart Example")
fig.show()
```

Output: An inverted funnel showing how values drop through stages.



Comparison Table

Feature / Aspect	Matplotlib	Plotly
Library Type	Static 2D plotting library	Interactive visualization library
Interactivity	Mostly static (non-interactive)	Fully interactive (hover, zoom, pan)
Ease of Use	Moderate – requires manual tuning	Very easy – automatic layout and interaction
Customization	Extremely flexible with manual control	Flexible but slightly abstracted
3D Visualization	Limited (via mpl_toolkits.mplot3d)	Excellent 3D support out-of-the-box
Animation Support	Supported using FuncAnimation	Supported natively with smooth transitions
Output Type	Static images (PNG, PDF, SVG)	Interactive HTML, web-based visuals
Integration	Great with NumPy, Pandas, Seaborn	Great with Pandas, Dash (for web apps)
Community & Usage	Very large academic/scientific community	Strong among web devs, dashboards, and data apps
Code Length	Usually longer; requires more setup	Shorter; auto-handles aesthetics and interactivity
Learning Curve	Steeper – more control, more config	Easier for beginners to get cool results fast
Aesthetics	Plain by default, customizable manually	Modern, visually polished by default
Best Use Case	Static reports, research papers, technical plots	Dashboards, web apps, and interactive data stories
Example Library Import	import matplotlib.pyplot as plt	import plotly.express as px
Example Visualization Output	PNG or inline static image	Interactive HTML chart (hover + zoom)