

Unit-5: Data Visualization using data frame**Que) What is Data Visualization?**

Ans: Data visualization is the process of converting complex data into graphical formats such as charts, graphs, and maps. It allows users to understand patterns, trends, and outliers in large datasets quickly and clearly.

- Representation of data in graphical form.
- ummarizes data and helps to give outcomes in graphical form.
- Graphical representation makes data easier to understand.
- Types: line graph, bar graph, pie chart, scatter plot.
- Python library: Matplotlib.

Key Features for Data Visualization with Pandas:

- **Variety of Plot Types:** Pandas supports various plot types including line plots, bar plots, histograms, box plots, and scatter plots.
- **Customization:** Users can customize plots by adding titles, labels, and styling enhancing the readability of the visualizations.
- **Handling of Missing Data:** Pandas efficiently handles missing data ensuring that visualizations accurately represent the dataset without errors.
- **Integration with Matplotlib:** Pandas integrates with Matplotlib that allow users to create a wide range of static, animated, and interactive plots.

Que) What is matplotlib?

Ans:

- Matplotlib is a Python library for creating static, interactive and animated visualizations from data.
- It provides flexible and customizable plotting functions that help in understanding data patterns, trends and relationships effectively.
- It Works with Pandas DataFrame.
- It use Module: pyplot.
- To import matplotlib in library in your page use import matplotlib.pyplot as plt.

Que) How to Importing matplotlib.pyplot?

Ans: To install matplotlib use below command

```
py -m pip install matplotlib
python -m pip install matplotlib
```

- To install numpy

```
py -m pip install numpy
python -m pip install numpy
```

Example:

```
import matplotlib.pyplot as plt
import numpy as np
xpoints=np.array([0,6])
ypoints=np.array([0,250])
plt.plot(xpoints,ypoints)
plt.show()
```

Plotting Graphs (2D)

1. plot() function

Syntax: plot(x_values, y_values, optional_parameters)

Options:

- color='code' (b=blue, g=green, r=red, c=cyan, m=magenta, y=yellow, k=black, w=white)
- marker='symbol' (circle, triangle, square, star, etc.)
- linestyle='style' (solid, dashed, dotted, dash-dot)
- [marker][linestyle][color] shorthand
- linewidth, label

Example:

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = [1,2,3,4,5]
ypoints=[23,43,45,66,76]
plt.plot(xpoints, ypoints, 'r--', linewidth=2, label='Testing')
plt.show()
```

Que) What is Matplotlib.pyplot.legend()?

Ans: A legend is an area describing the elements of the graph. In the matplotlib library, there's a function called legend() which is used to Place a legend on the axes.

Example:

```
import numpy as np
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
plt.plot(x, y)
plt.legend(['single element'])
plt.show()
```

Que) Explain Concepts of Subplot()

Ans: With the subplot() function you can draw multiple plots.

Example

```
import matplotlib.pyplot as plt
import numpy as np
#plot
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
#plot
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```

Que) Explain Concepts of Range()

Ans: Matplotlib sets the default range of the axis by finding extreme values (i.e. minimum and maximum) on that axis.

Example:

```
import matplotlib.pyplot as plt
x_values = range(1, 11)
y_values = [2*x for x in x_values]
plt.plot(x_values, y_values)
plt.title('Sample Line Chart')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

Que) Explain Concepts of show()

Ans: Displays the figure in an interactive window.

- **Syntax:** plt.show()
- **Example:**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
plt.show()
```

Que) Explain Concepts of xlabel() / ylabel()

Ans: Sets labels for the x-axis and y-axis.

- **Syntax:**
 - plt.xlabel('label')
 - plt.ylabel('label')
- **Example:**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
plt.xlabel('X-axis values')
plt.ylabel('Y-axis values')
plt.show()
```

Que) Explain Concepts of title()

Ans: Purpose: Sets the chart title.

- **Syntax:** plt.title('label')
- **Example:**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
plt.title('Simple Line Graph')
plt.show()
```

Que) Explain Concepts columns

Ans: Lists the fields (column names) in a DataFrame.

- **Syntax:** df.columns
- **Example:**

```
import pandas as pd
data = {'Name': ['A', 'B', 'C'], 'Marks': [85, 90, 78]}
df = pd.DataFrame(data)
print(df.columns)
```

Output: Index(['Name', 'Marks'], dtype='object')

Que) Explain Concepts len()

Ans: Returns the number of elements in an object.

- **Syntax:** len(object)
- **Example:**

```
numbers = [10, 20, 30, 40]
print(len(numbers))
```

- **Output:** 4

Que) Explain Concepts grid()

Ans: Adds gridlines to the plot.

- **Syntax:** plt.grid()
- **Example:**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
plt.grid()
plt.show()
```

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 Chart

Line chart is one of the basic plots and can be created using plot() function. It is used to represent a relationship between two data X and Y on a different axis.

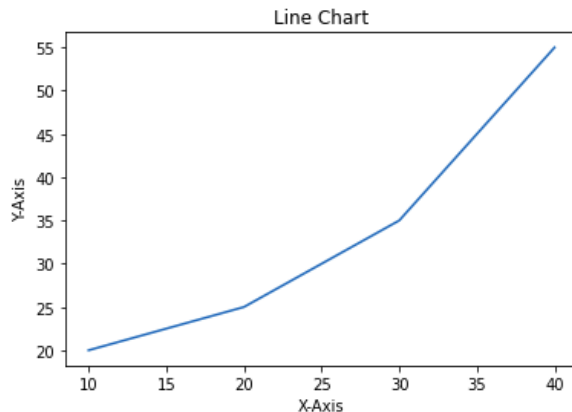
Syntax: matplotlib.pyplot.plot(x, y)

Parameter: x, y Coordinates for data points.

Example: This code plots a simple line chart with labeled axes and a title using Matplotlib.

```
import matplotlib.pyplot as plt
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]
plt.plot(x, y)
plt.title("Line Chart")
```

```
plt.ylabel('Y-Axis')  
plt.xlabel('X-Axis')  
plt.show()
```

Output**2. Bar Chart**

Bar chart displays categorical data using rectangular bars whose lengths are proportional to the values they represent. It can be plotted vertically or horizontally to compare different categories.

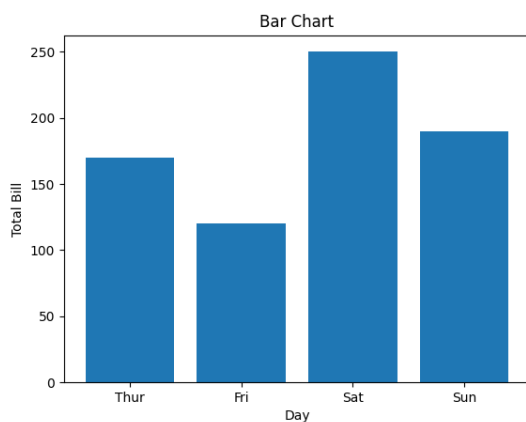
Syntax: `matplotlib.pyplot.bar(x, height)`

Parameter:

- **x:** Categories or positions on x-axis.
- **height:** Heights of the bars (y-axis values).

Example: This code creates a simple bar chart to show total bills for different days. X-axis represents the days and Y-axis shows total bill amount.

```
import matplotlib.pyplot as plt  
x = ['Thur', 'Fri', 'Sat', 'Sun']  
y = [170, 120, 250, 190]  
plt.bar(x, y)  
plt.title("Bar Chart")  
plt.xlabel("Day")  
plt.ylabel("Total Bill")  
plt.show()
```

Output

3. Histogram

Histogram shows the distribution of data by grouping values into bins. The `hist()` function is used to create it, with X-axis showing bins and Y-axis showing frequencies.

Syntax: `matplotlib.pyplot.hist(x, bins=None)`

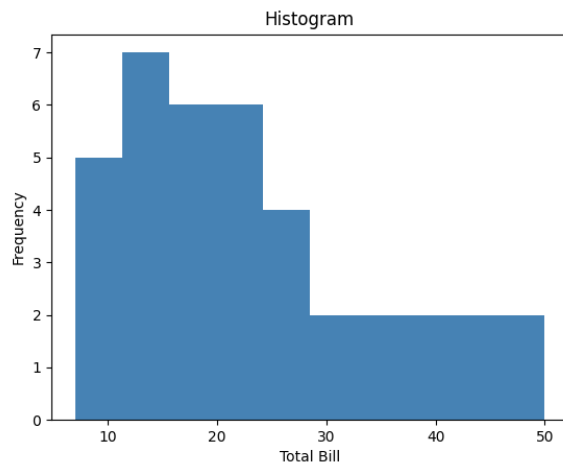
Parameter:

- **x:** Input data.
- **bins:** Number of bins (intervals) to group data.

Example: This code plots a histogram to show frequency distribution of total bill values from the list `x`. It uses 10 bins and adds axis labels and a title for clarity.

```
import matplotlib.pyplot as plt
x = [7, 8, 9, 10, 10, 12, 12, 12, 13, 14, 14, 15, 16, 16, 17, 18, 18, 19, 20, 20, 21, 22, 23, 24, 25, 25, 26, 28, 30, 32, 35, 36, 38, 40, 42, 44, 48, 50]
plt.hist(x, bins=10, color='steelblue')
plt.title("Histogram")
plt.xlabel("Total Bill")
plt.ylabel("Frequency")
plt.show()
```

Output



4. Scatter Plot

Scatter plots are used to observe relationships between variables. The `scatter()` method in the `matplotlib` library is used to draw a scatter plot.

Syntax: `matplotlib.pyplot.scatter(x, y)`

Parameter: `x, y` Coordinates of the points.

Example: This code creates a scatter plot to visualize the relationship between days and total bill amounts using `scatter()`.

```
import matplotlib.pyplot as plt
x = [10, 15, 20, 25, 30]
```

```
y = [12, 18, 25, 28, 35]  
plt.scatter(x, y)  
plt.title("Scatter Plot")  
plt.xlabel("Study Hours")  
plt.ylabel("Marks")  
plt.show()
```

Output

