**DOCUMENTATION.**

# **1.Pandas**:

Pandas is a fast, powerful, flexible and easy to use open source data analysis

and manipulation tool, built on top of the python programming language.

**Installation:**

The easiest way to install pandas is to install it as part of the Anaconda

distribution, a cross platform distribution for data analysis and scientific

computing. The Conda package manager is the recommended installation

method for most users.

Here are some pf the operations performed by pandas:

1. Data Loading.
2. Data Inspection.
3. Data Cleaning.
4. Data selection And filtering.
5. Data Transformation.
6. Grouping And Aggregation.
7. Data Export.

Here are some of the Data Structures used by pandas:

1. Series.
2. Data Frame.
3. Index.

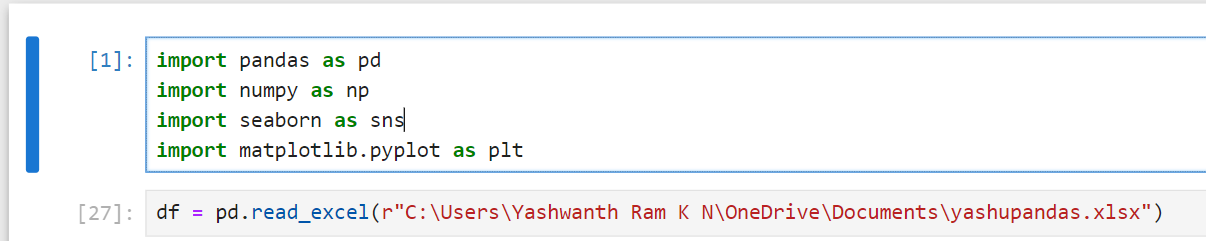
Here are some of the major functions performed by pandas:

1. **Data Loading:**
2. pd.read\_csv(): Reads data from a CSV file.
3. pd.read\_excel(): Reads data from an Excel file.
4. pd.read\_json(): Reads data from a JSON file.
5. pd.read\_sql(): Reads data from a SQL database.
6. **Data Inspection:**
7. df.head(): Displays the first few rows of the DataFrame.
8. df.tail(): Displays the last few rows.
9. df.info(): Provides a concise summary of the DataFrame (e.g., column names, data types, non-null counts).
10. df.describe(): Generates descriptive statistics of numerical columns (e.g., mean, median, quartiles).
11. **Data Selection and Indexing:**
12. df['column\_name']: Select a specific column.
13. df[['col1', 'col2']]: Select multiple columns.
14. df.loc[]: Label-based indexing for selecting rows/columns.
15. df.iloc[]: Integer-location-based indexing.
16. df[df['column'] > value]: Filters rows based on a condition.
17. **Data Cleaning:**
18. df.isnull(): Checks for missing values.
19. df.fillna(): Fills missing values with a specified value.
20. df.dropna(): Drops rows or columns with missing values.
21. df.replace(): Replaces specified values in the DataFrame.
22. df.duplicated(): Checks for duplicate rows.

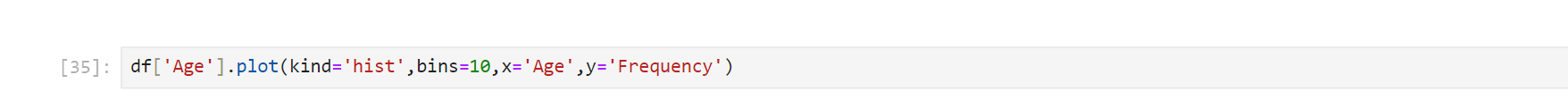
Some of the Visualization Techniques can also be Used:

1. Bar Graph.
2. Histogram.
3. Pie plot.
4. Box Plot.

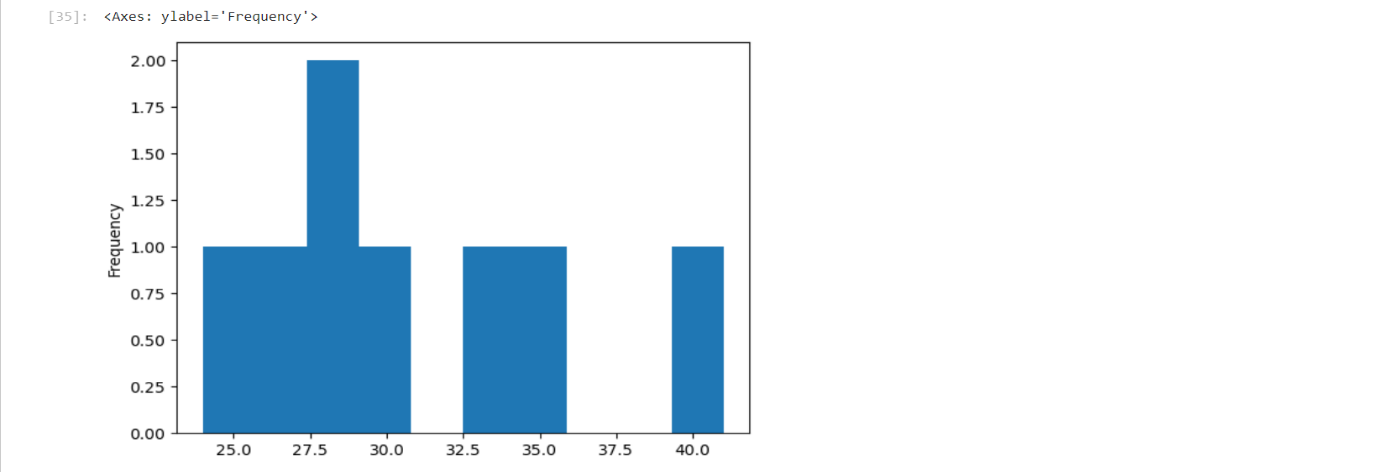
* **Importing a Excel File:**



**1.Histogram:**

****

**Output:**

****

A Histogram in Pandas is a type of plot that represents the distribution of a

numerical dataset by dividing it into “bins” (intervals) and displaying the

frequency of data points into each bin.

It provides a visual way to see how the values in a dataset are distributed across

a range, making it easier to identify patterns such as skewness, central tendency,

spread, and outliers.

A Histogram Plot in Pandas is Useful for visualizing the distribution of a

numerical variable.

Pandas provides the .plot() method with the kind=’hist’ option to create

histogram.

Let us take above given Dataset which “Age” Column, and we want to visualize

the distribution of Age.

**Kind=’hist’:** Specifies that we are creating a histogram.

**Bins=’10’:** Divides the data into 10 intervals or “bins”. You can adjust this to

increase or decrease the granularity of the histogram.

**y=’Frequency’:** Labels y-axis as “Frequency”.

**x=’Age’:** Labels x-axis as “Age”.

**2.Box Plot:**

A Box Plot in pandas is a useful visualization for summarizing the distribution

of a dataset. It shows the data’s minimum, first quartile(Q1), median(Q2), third

quartile(Q3), and maximum values, while also highlighting any outliers. The

box plot is practically helpful in identifying the central tendency, spread, and

skewness of the data.

Key Components of the Box Plot:

Box: Represents the Interquartile range (IQR), i.e., the rage between first

quartile (Q1) and third quartile (Q3).

Median: A line inside the box representing the median (Q2) of the dataset.

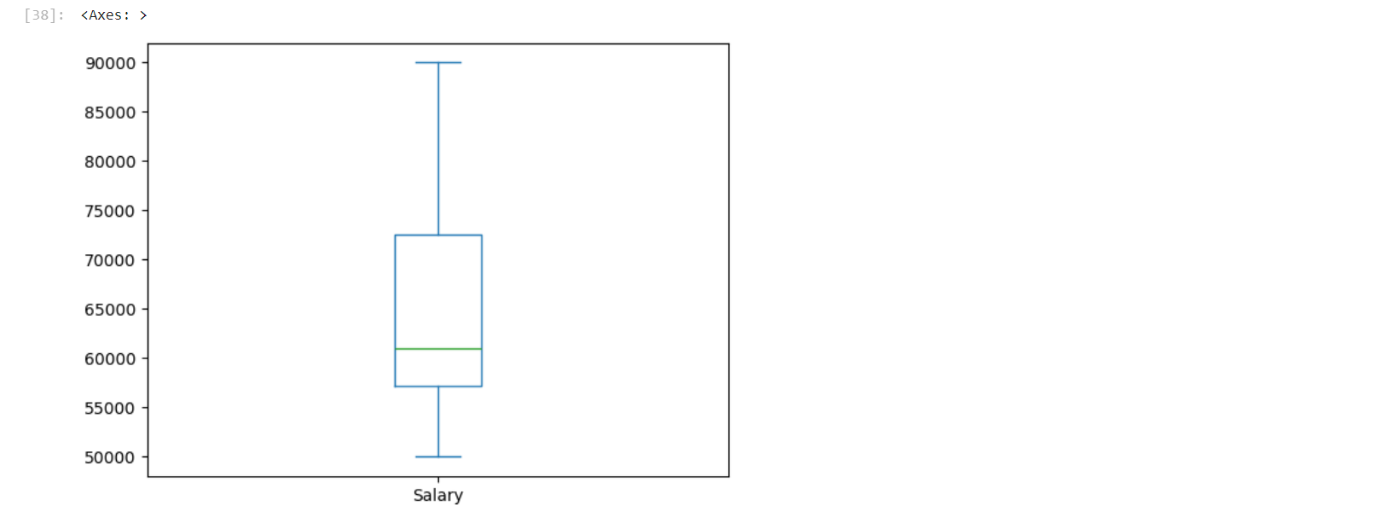
Whiskers: A line inside the box to the minimum and maximum values within

1.5 times the IQR. Anything beyond this is considered as an outlier.

Outliers: Data points that lie outside the whiskers.



**Output:**

****

Let us take the above employee dataset with “Salary” column and we want to

Visualize the distribution using a box plot.

The **kind=’box’** Specifies that we are creating a box plot.

Outliers: If there are any salary values that are too far from the rest of the

data, they will be plotted as individual points outside the “whiskers”.

Box: Shows the IQR where 50% of the salary data falls.

Whiskers: Extend to the minimum and maximum non-outlier values.

**3.Pie Chart:**

A Pie Chart in pandas is used to display the portion of different categories in a

Dataset as slices if a pie.

It’s an excellent way to represent the relative sizes or percentages of different

groups visually.

Pandas allows you to easily create pie charts by using the plot() function with

kind=’pie’. This method is typically applied to a series object, which contains

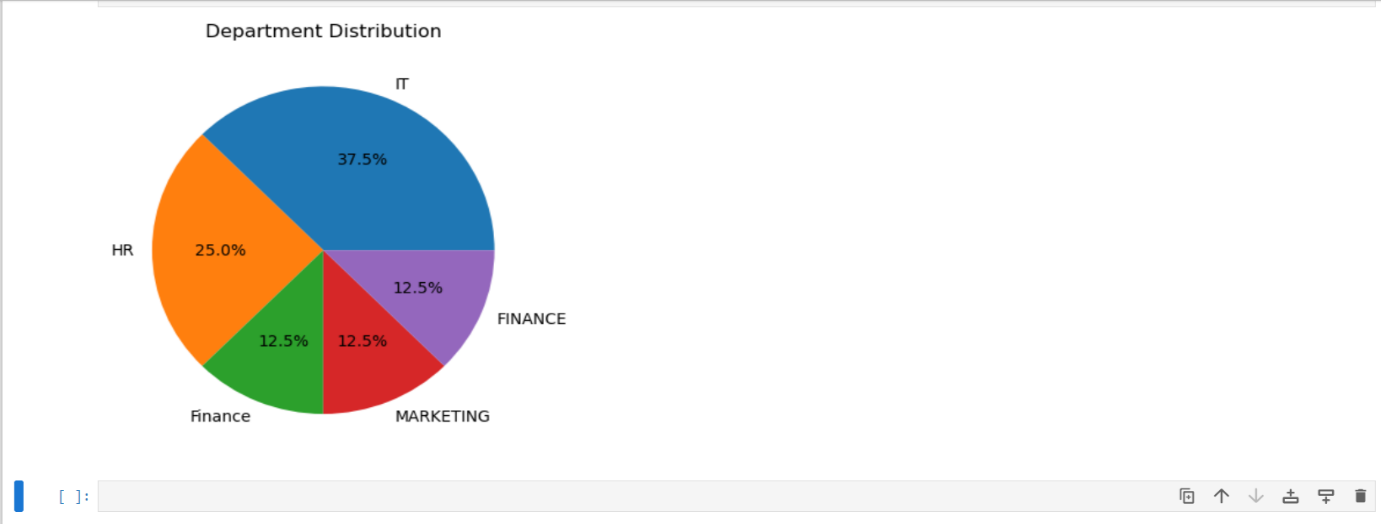
categorical data.

Assume we have a dataset of employees, and we want to show the distribution

of employees by department.



**Output:**



df[‘Department’].value\_counts(): Counts the number occurrences of each

department (HR,IT,Finance,etc).

plot(kind=’pie’): Specifies that we want to create a pie chart.

Autopct=’%1.1f%%’: Adds labels to the pie slices that show the percentage

of each category.

plt.ylabel(“): Hides the y-axis label, as it’s not in a pie chart.

# **2.MATPLOTLIB:**

Matplotlib is a popular plotting library that allows you to create static,

animated, and interactive visualizations in a variety of formats.

It provides a flexible and powerful way to generate a wide variety of plots and

charts.

Basic Components of Matplotlib are:

1. Figure: the entire window or image that contains the plot.
2. Axes: The area where the data is plotted. A figure can contain multiple axes.
3. Artist: Anything that is drawn on the figure, including lines, text, and markers.

**Common Plot Types are:**

1. Line Plot.
2. Box Plot.
3. Scatter Plot.
4. Histogram.
5. Pie Chart.

Some Functions of Matplotlib are:

1. **Basic Plotting Functions:**
2. **plot()**: Creates a line plot.
3. **scatter()**: Generates a scatter plot.
4. **bar()**: Creates a bar chart.
5. **barh()**: Creates a horizontal bar chart.
6. **hist()**: Generates a histogram.
7. **pie()**: Creates a pie chart.
8. **boxplot()**: Generates a box plot for visualizing distributions.
9. **Customization Functions:**
10. **title():** Sets the title of the plot.
11. **xlabel():** Sets the label for the x-axis.
12. **ylabel():** Sets the label for the y-axis.
13. **legend():** Adds a legend to the plot.
14. **grid():** Displays a grid in the plot area.
15. **xticks() and yticks():** Customize the ticks on the x and y axes.
16. **xlim() and ylim():** Set the limits for the x and y axes.
17. **Style Functions:**
18. **style.use():** Applies a style to the plot (e.g., 'ggplot', 'seaborn').
19. **set\_facecolor():** Sets the background color of the plot.
20. **set\_linewidth():** Changes the width of lines.
21. **set\_marker():** Changes the marker style.

**4. Interactivity:**

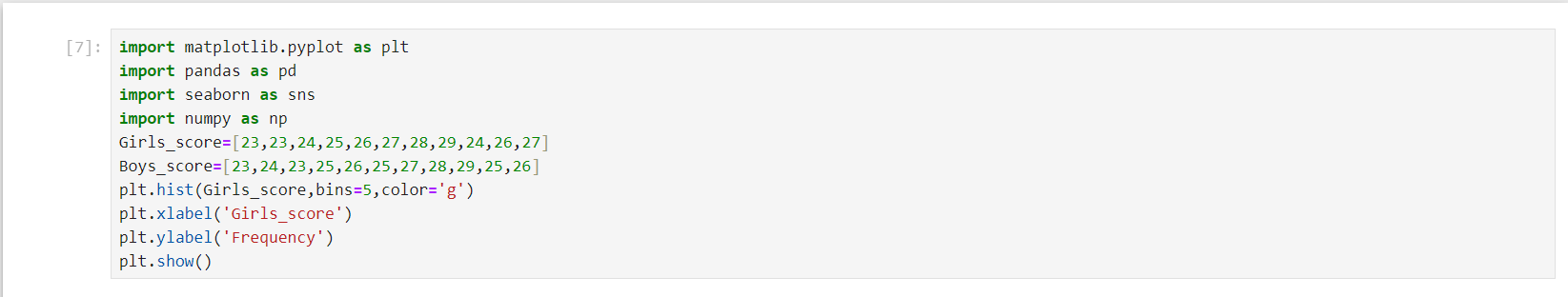
* **plt.show(): Displays the plot interactively.**
* **plt.pause(): Pauses the display (useful for animations or dynamic updating).**

**1.Histogram:**

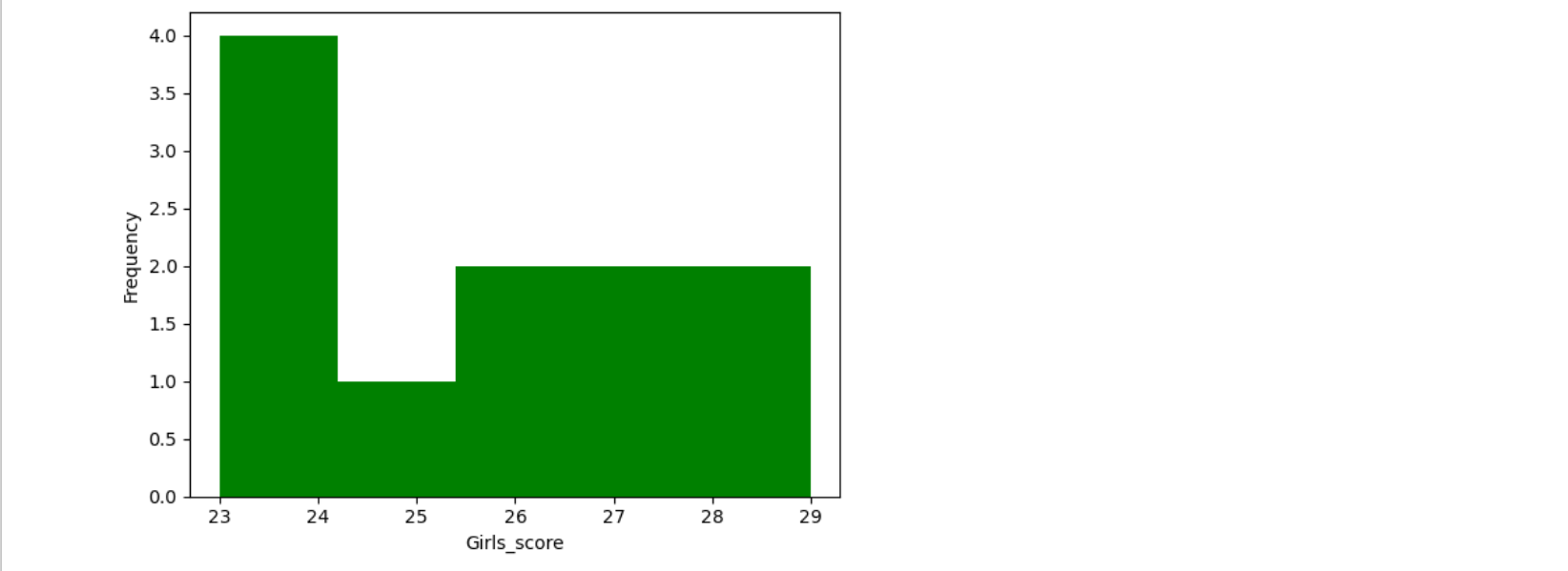
InMatplotlib, a histogram is a powerful tool to visualize the distribution of a

dataset It divides the data into bins and displays the number of data points in

each bin, giving an overview of data intensity, spread, and shape.



**Output:**

****

Let us take an example of Boys\_score and Girls\_score attributes.

Girls\_score are the data array you want to plot. The array represents the

scores of Girls.

bins=5: Defines the number of bins of grouping the data points along the

x-axis. A higher number of bins will create smaller bins, showing more

detail, while a smaller number of bins will group data into broader

intervals. In this example girls score are divided into 5 bins.

Color=’g’: Specifies the colours for the Girls\_score.

alpha=0.5: Controls the transparency of the bars. A lower alpha value

(e.g., 0.1) makes the bars more transparent, while value (up to 1.0) makes

them more opaque. Setting alpha=0.5 makes the bars partially transparent,

so the overlapping areas between the two histograms are visible.

label=’Girls’ and label=’Boys’: Provides a label for each histogram

dataset. These labels are used in legend, allowing viewers to identify the

data for girls and boys on the plot.

plt.xlabel(‘Score’): Sets the x-axis label to “Score”.

plt.ylabel(‘Frequency’): Sets the y-axis label to “Frequency”, representing

the count of scores within each bin.

plt.title(‘Histogram of Girls Score’): Adds a title to the histogram.

plt.legend(): Displays the legend, showing the labels (‘Girls’) for each

dataset, which are specified in the label parameter.

**2.** **Box Plot:**

In Matplotlib, a box plot is used to visualize the distribution, variability,

and Outliers of a dataset. It displays the summary of a dataset using five

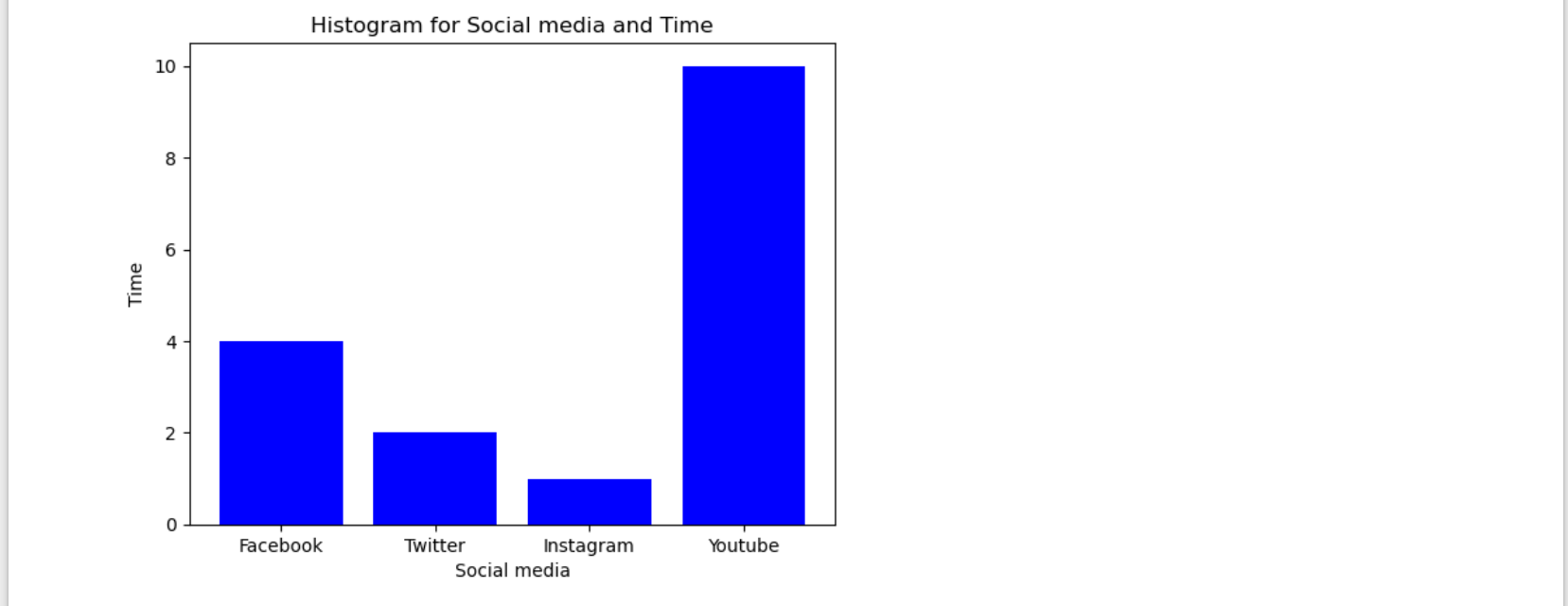
keys:

Summary statistics: minimum, first quartile (Q1), median, third quartile

(Q3), and maximum.



**Output:**

****

plt.bar(Social\_media, Time, color=’b’):

Social\_media: This list defines the categories for the x-axis reach social

media platform(Facebook, Twitter, Instagram, and YouTube).

Time: This list provides the values for the y-axis, representing time spent

on each platform.

Color=’b’: Specifies the color of the bars as blue (‘b’), so all bars will

appear in blue color.

plt.title(“Histogram for Social media and Time”):

Adds the title “Histogram for Social media and Time” to the plot for

content.

plt.xlabel(“Social media”): Labels the x-axis as “Social media” to clarify

what each category on the axis represents.

plt.ylabel(“Time”): Labels the y-axis as “Time”, indicating that the height

of each bar represents the amount of time spent on each platform.

plt.show(): Displays the plot, making the chart visible in the output

window.

**3.Scatter Plot:**

A scatter plot in Matplotlib is used to visualize the relationship between

two variables by plotting individual data points. It’s ideal for showing

correlations, clusters, and trends in datasets.

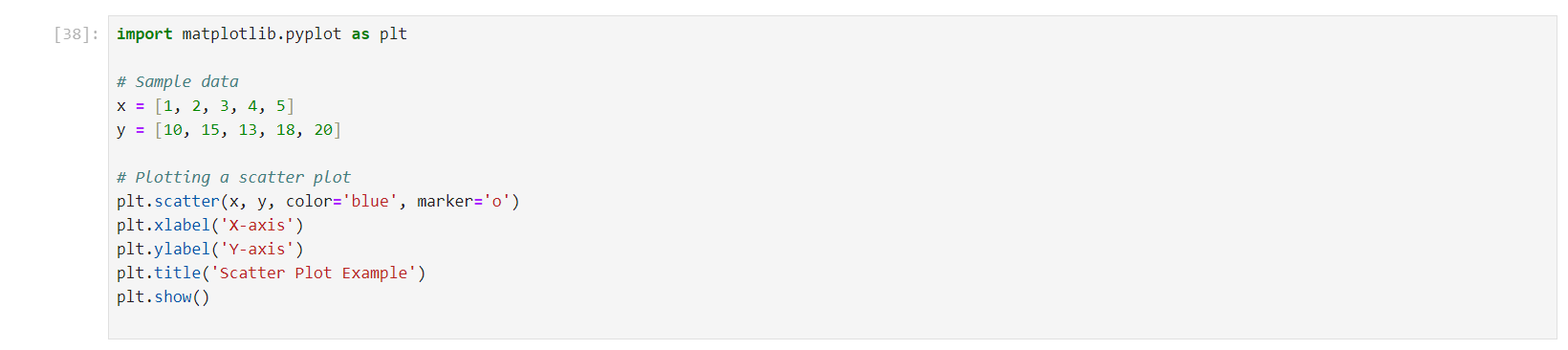
Matplotlib’s scatter() function offers customization options for colours,

sizes, shapes, and transparency of points, enabling detailed and visually

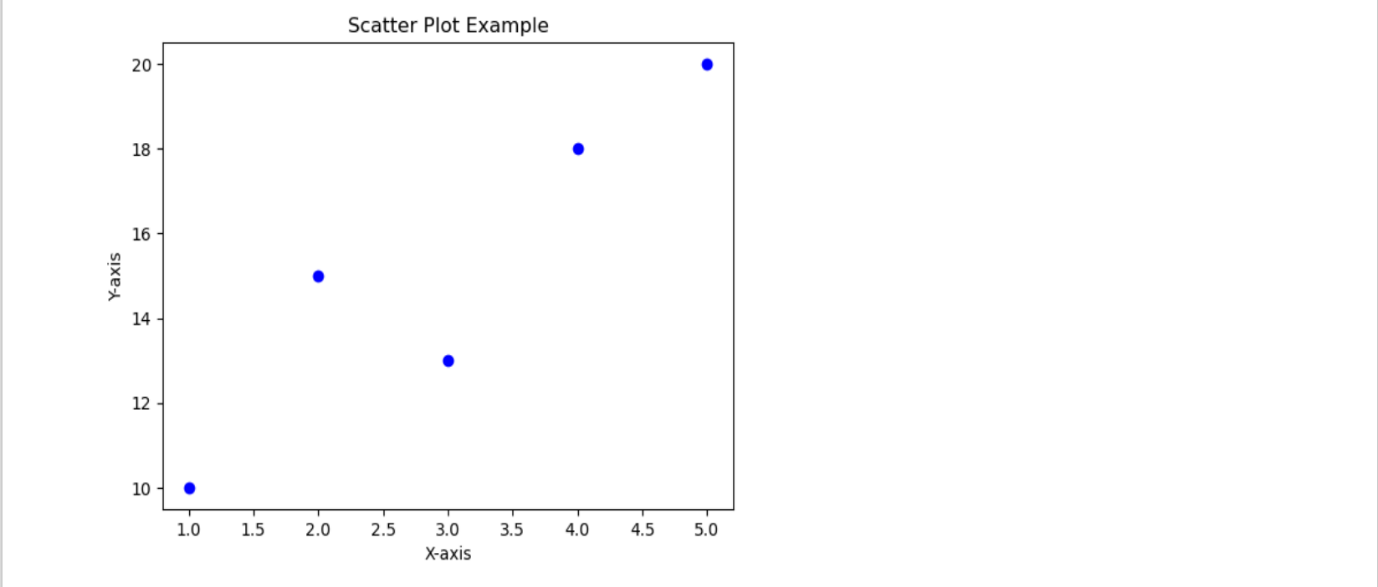
clear data representation. Scatter plots are also often enhanced by using

additional variables to colour-code or resize points, providing multi-

dimensional insights within a two-dimensional space.



**Output:**



**plt.scatter(x, y, color='blue', marker='o'):**

**x** and **y**: These lists represent the x and y coordinates for each point on the

plot. Here, x values are [1, 2, 3, 4, 5], and y values are [10, 15, 13, 18, 20].

Each pair creates a single point.

**color='blue'**: Sets the color of the points in the scatter plot. Here, all points

are set to blue.

**marker='o'**: Defines the shape of each point. 'o' specifies circular markers,

which are commonly used in scatter plots. Other marker types include 's' for

squares, '^' for triangles, and more.

**plt.xlabel('X-axis')**: Sets the label for the x-axis as "X-axis" to describe what

the x values represent.

**plt.ylabel('Y-axis')**: Sets the label for the y-axis as "Y-axis" to describe what

the y values represent.

**plt.title('Scatter Plot Example')**: Adds a title, "Scatter Plot Example," to the

plot for context.

**plt.show()**: Displays the plot in the output.