**DSA0512-QUERY PROCESSING FOR DATASCIENCE WITH FUZZY MATCHING**

21.Write a Pandas program to swap the cases of a specified character column in a given DataFrame.

**AIM:**

To develop a Pandas program to swap the cases of a specified character column in a given DataFrame.

**ALGORITHM:**

1. Import the `pandas` library.

2. Create a dictionary with columns for names and addresses.

3. Convert the dictionary into a DataFrame using `pd.DataFrame`.

4. Print the original DataFrame.

5. Specify the column to swap cases (`'name'`).

6. Swap the cases in the specified column using `str.swapcase()`.

7. Print the DataFrame after swapping cases in the specified column.

**PROGRAM:**

import pandas as pd

data = {

'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'David Parkes'],

'address': ['street1', 'street2', 'street3', 'street1', 'street4']

}

df = pd.DataFrame(data)

print("Original DataFrame:")

print(df)

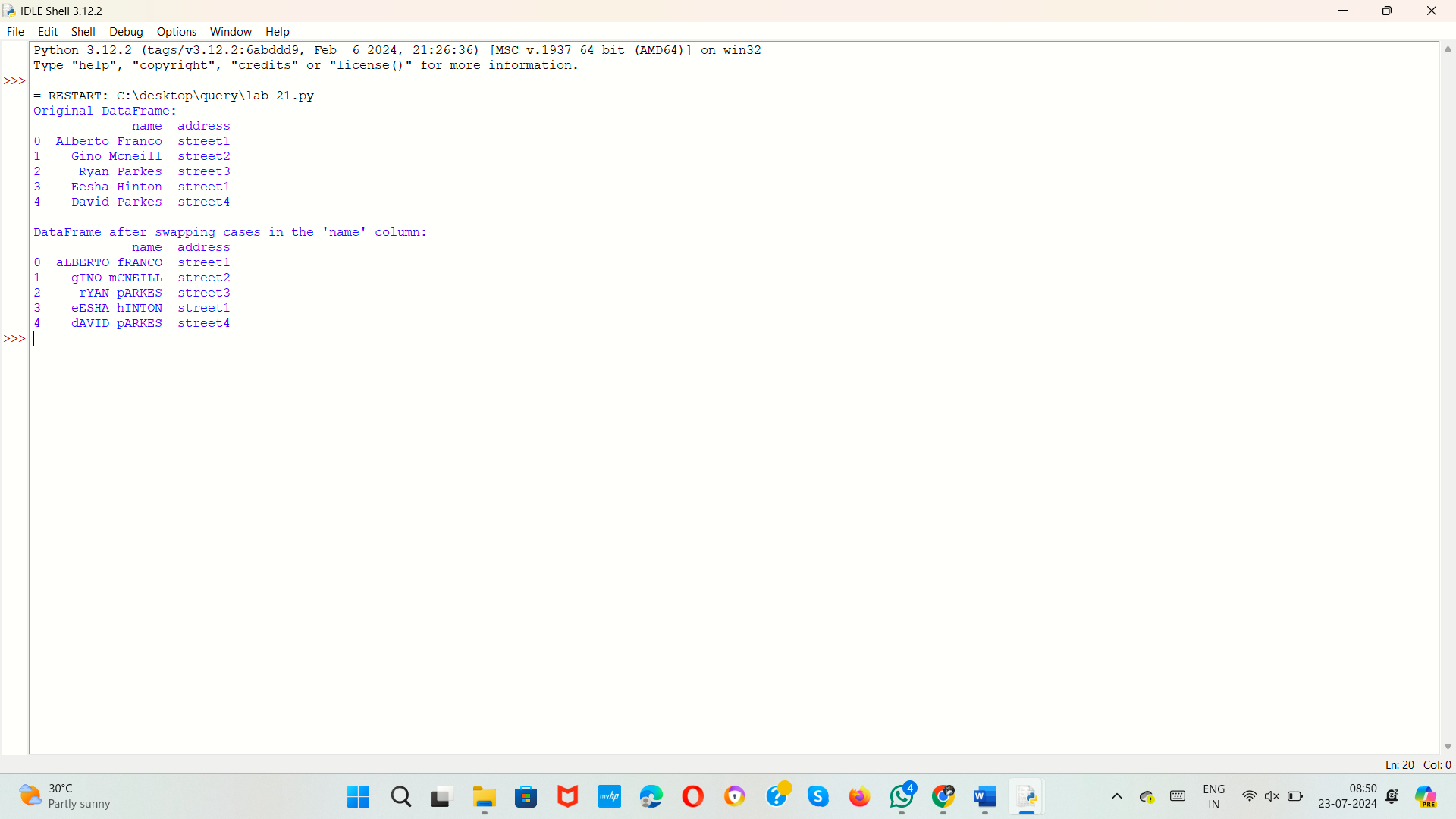
column\_to\_swap = 'name'

df[column\_to\_swap] = df[column\_to\_swap].str.swapcase()

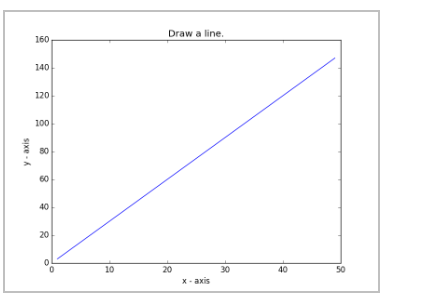
print("\nDataFrame after swapping cases in the '{}' column:".format(column\_to\_swap))

print(df)

**SAMPLE OUTPUT:**



22.Write a Python program to draw a line with suitable label in the x axis, y axis and a title.



**AIM:**

TO develop a Python program to draw a line with suitable label in the x axis, y axis and a title.

**ALGORITHM:**

1. Import the `matplotlib.pyplot` library as `plt`.

2. Define the data for the x-axis (`x`) and y-axis (`y`).

3. Plot the data using `plt.plot`, with a label for the line.

4. Set the label for the x-axis using `plt.xlabel`.

5. Set the label for the y-axis using `plt.ylabel`.

6. Set the title of the plot using `plt.title`.

7. Display the legend using `plt.legend`.

8. Show the plot using `plt.show`.

**PROGRAM:**

import matplotlib.pyplot as plt

x=[0,1,2,3,4,5]

y=[0,1,4,9,16,25]

plt.plot(x,y,label='y=x^2')

plt.xlabel("x-axis ")

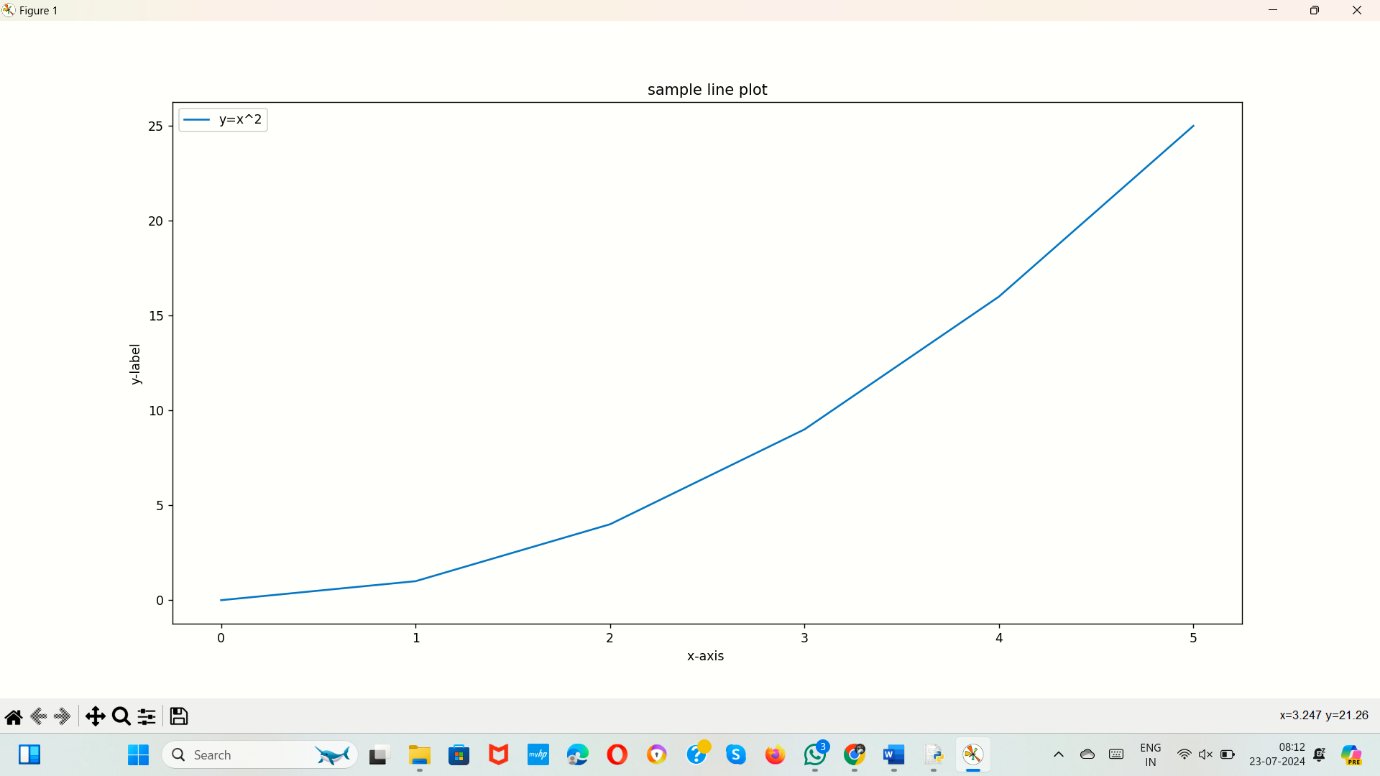
plt.ylabel("y-label")

plt.title("sample line plot")

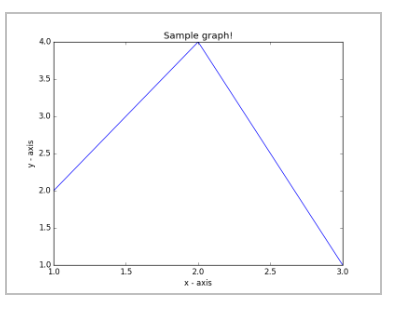
plt.legend()

plt.show()

**SAMPLE OUTPUT:**



23.Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.  
Test Data:  
test.txt  
1 2  
2 4  
3 1



**AIM:**

To develop a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

**ALGORITHM:**

1. Import `matplotlib.pyplot` as `plt`.

2. Define a function to create a text file and write user input to it.

3. Define a function to read x and y values from the text file.

4. Call the function to create and populate the text file, then read the data from it.

5. Plot the data with labels and display the plot.

**PROGRAM:**

import matplotlib.pyplot as plt

def create\_text\_file(file\_path):

with open(file\_path, 'w') as file:

while True:

user\_input = input("Enter x and y values separated by space (or type 'done' to finish): ")

if user\_input.lower() == 'done':

break

file.write(user\_input + '\n')

def read\_data(file\_path):

x\_values = []

y\_values = []

with open(file\_path, 'r') as file:

for line in file:

x, y = map(int, line.split())

x\_values.append(x)

y\_values.append(y)

return x\_values, y\_values

file\_path = 'user\_data.txt'

create\_text\_file(file\_path)

x, y = read\_data(file\_path)

plt.plot(x, y, marker='o', label='Line Plot')

plt.xlabel('X-axis Label')

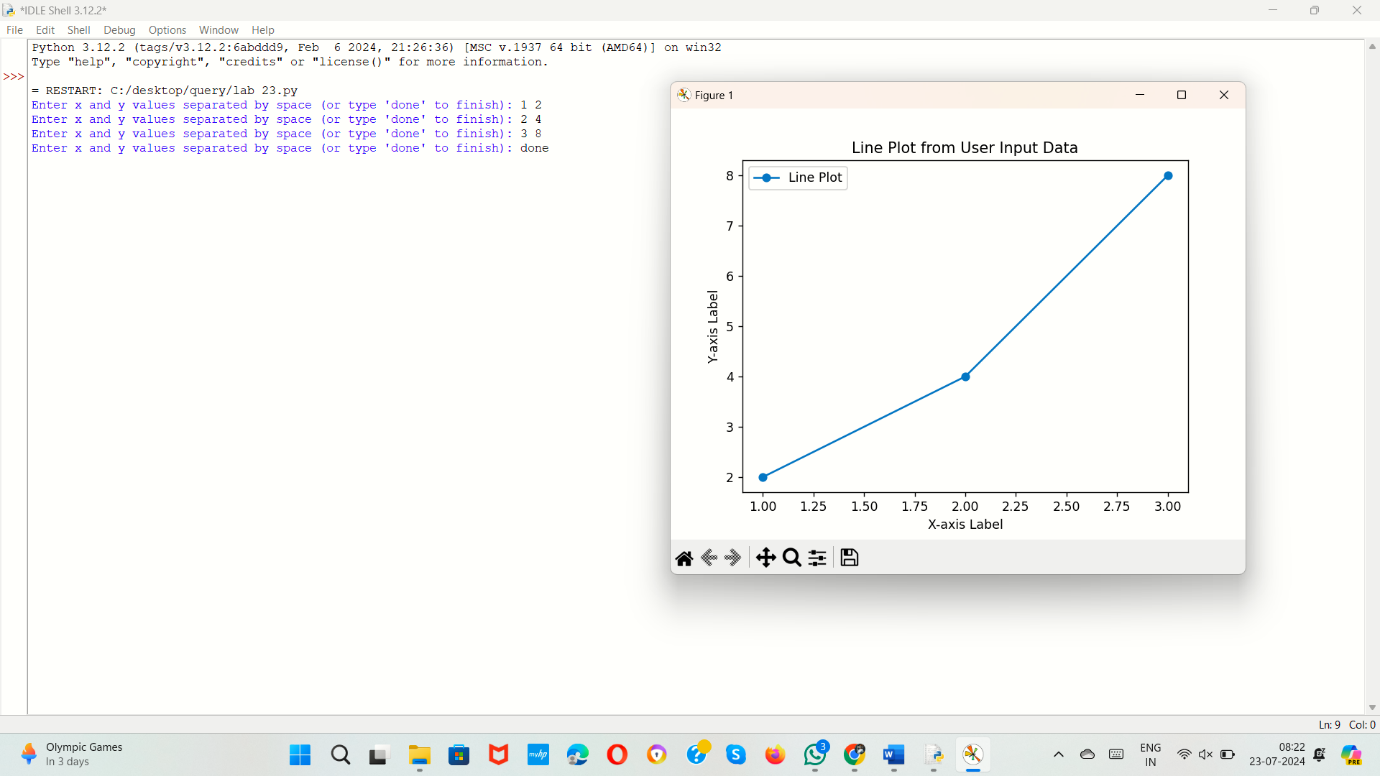
plt.ylabel('Y-axis Label')

plt.title('Line Plot from User Input Data')

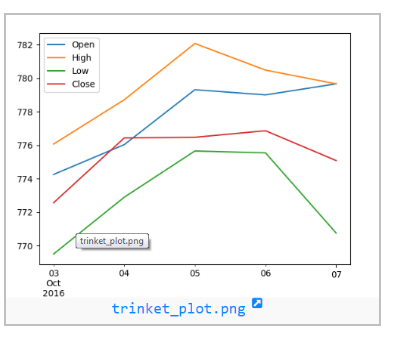
plt.legend()

plt.show()

**SAMPLE OUTPUT:**



24.Write a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.  
Sample Financial data (fdata.csv):  
Date,Open,High,Low,Close  
10-03-16,774.25,776.065002,769.5,772.559998  
10-04-16,776.030029,778.710022,772.890015,776.429993  
10-05-16,779.309998,782.070007,775.650024,776.469971  
10-06-16,779,780.47998,775.539978,776.859985  
10-07-16,779.659973,779.659973,770.75,775.080017



**AIM:**

To develop a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.

**ALGORITHM:**

1. Import the necessary libraries.

2. Create a dictionary with date and stock price data.

3. Convert the dictionary to a Pandas DataFrame.

4. Convert the 'Date' column to datetime format.

5. Create a figure for plotting.

6. Plot the 'Open' prices with labels and markers.

7. Plot the 'High' prices with labels and markers.

8. Plot the 'Low' prices with labels and markers.

9. Plot the 'Close' prices with labels and markers.

10. Set the x-axis label to 'Date'.

11. Set the y-axis label to 'Price'.

12. Set the title of the plot.

13. Add a legend to the plot.

14. Enable the grid on the plot.

15. Display the plot.

**PROGRAM:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

data = {

'Date': ['10-03-16', '10-04-16', '10-05-16', '10-06-16', '10-07-16'],

'Open': [774.25, 776.030029, 779.309998, 779, 779.659973],

'High': [776.065002, 778.710022, 782.070007, 780.47998, 779.659973],

'Low': [769.5, 772.890015, 775.650024, 775.539978, 770.75],

'Close': [772.559998, 776.429993, 776.469971, 776.859985, 775.080017]

}

df=pd.DataFrame(data)

df['Date']=pd.to\_datetime(df['Date'],format='%m-%d-%y')

plt.figure(figsize=(10,6))

plt.plot(df['Date'], df['Open'], label='Open', marker='o')

plt.plot(df['Date'], df['High'], label='High', marker='o')

plt.plot(df['Date'], df['Low'], label='Low', marker='o')

plt.plot(df['Date'], df['Close'], label='Close', marker='o')

plt.xlabel('Date')

plt.ylabel('Price')

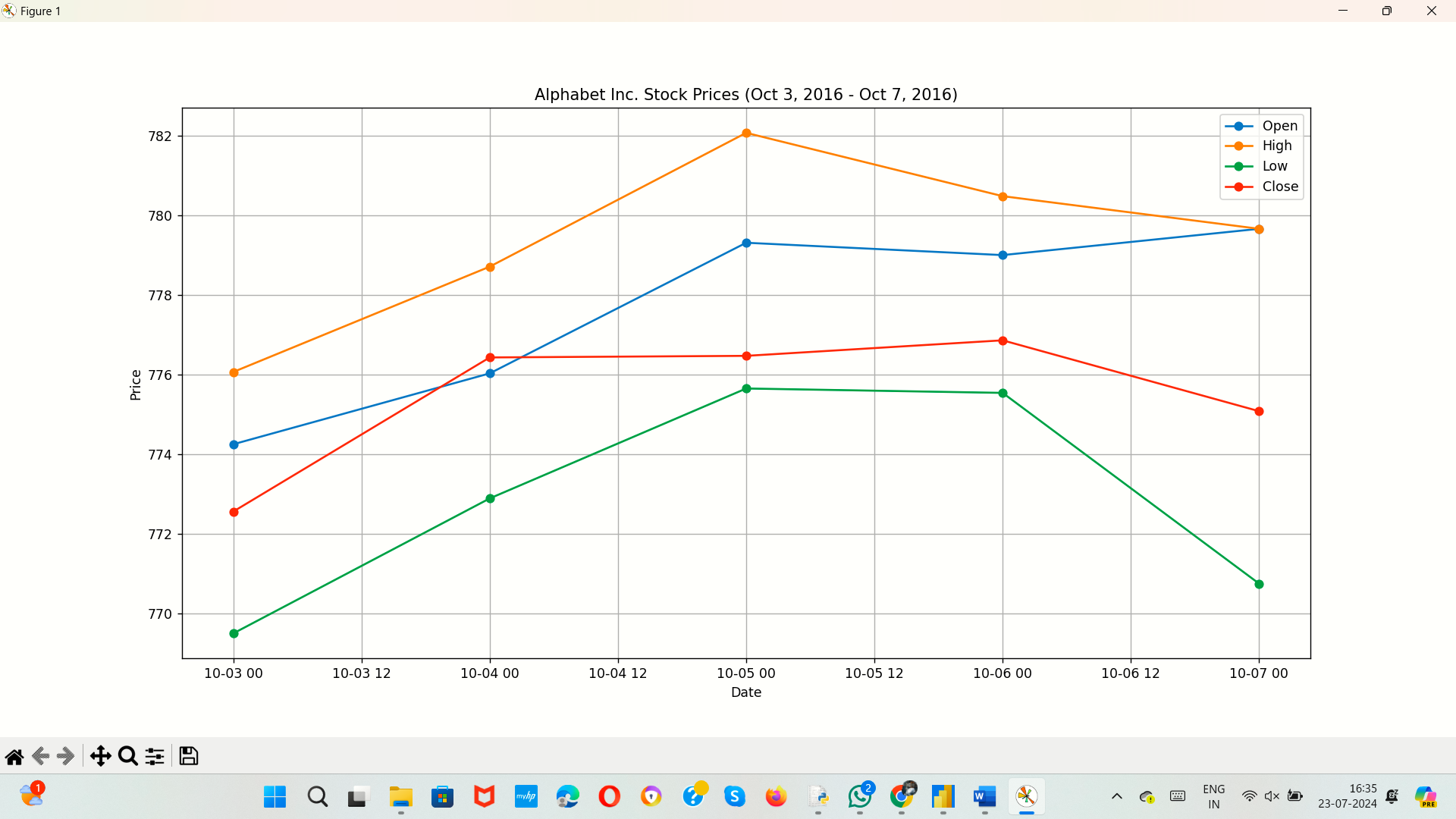
plt.title('Alphabet Inc. Stock Prices (Oct 3, 2016 - Oct 7, 2016)')

plt.legend()

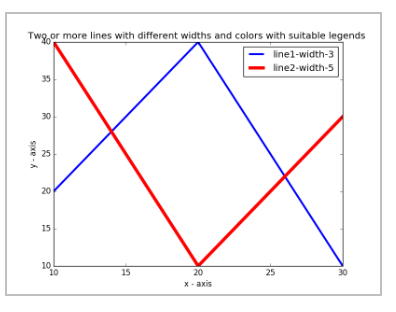
plt.grid(True)

plt.show()

**SAMPLE OUTPUT:**



25.Write a Python program to plot two or more lines with legends, different widths and colors.



**AIM:**

To develop a python program to plot two or more lines with legends, different widths and colors.

**ALGORITHM:**

1. Import the `matplotlib.pyplot` library as `plt`.

2. Define data for x and three y-values (`y1`, `y2`, `y3`).

3. Plot each set of y-values against x using `plt.plot`, specifying different colors and line widths for each.

4. Set the title, x-axis label, and y-axis label for the plot.

5. Display the legend and show the plot using `plt.show()`.

**PROGRAM:**

import matplotlib.pyplot as plt

x = [0, 1, 2, 3, 4, 5]

y1 = [0, 1, 4, 9, 16, 25]

y2 = [0, 1, 8, 27, 64, 125]

y3 = [0, -1, -4, -9, -16, -25]

plt.plot(x, y1, label='y = x^2', color='blue', linewidth=2)

plt.plot(x, y2, label='y = x^3', color='red', linewidth=4)

plt.plot(x, y3, label='y = -x^2', color='green', linewidth=1)

plt.title('Multiple Lines with Legends, Different Widths and Colors')

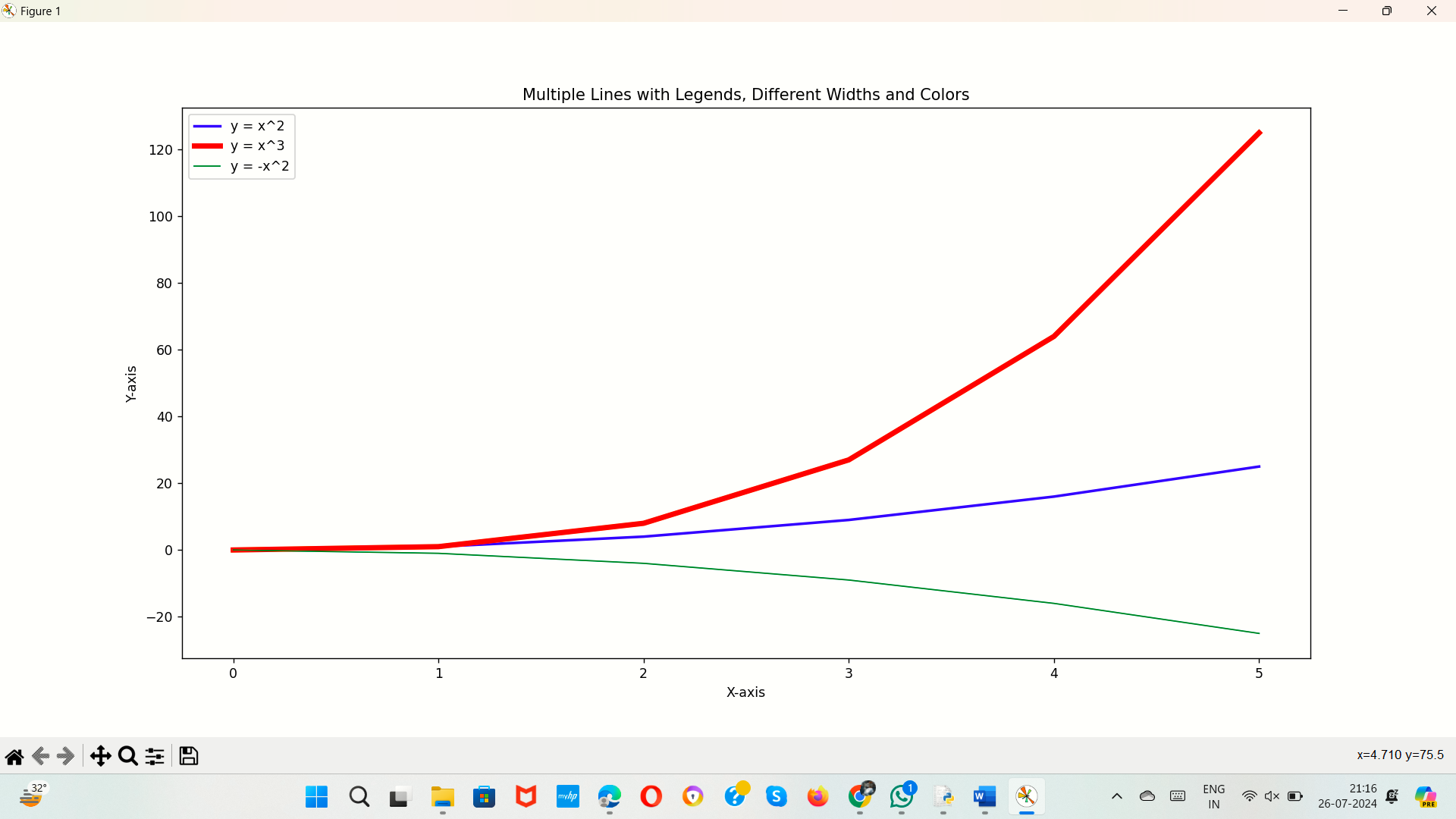
plt.xlabel('X-axis')

plt.ylabel('Y-axis')

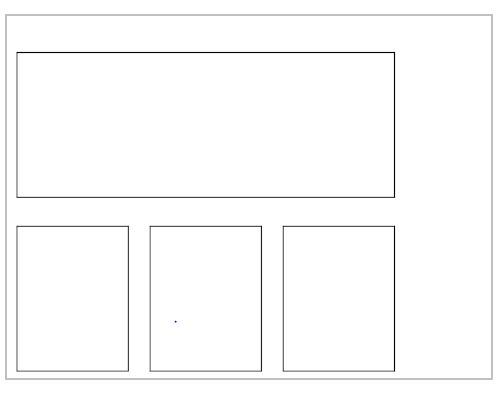
plt.legend()

plt.show()

**SAMPLE OUTPUT:**



26.Write a Python program to create multiple plots.



**AIM:**

To develop a Python program to create multiple plots.

**ALGORITHM:**

1. Import the `matplotlib.pyplot` and `numpy` libraries.

2. Generate an array of x-values from 0 to 10 with 100 points using `np.linspace`.

3. Compute y-values for sine, cosine, tangent, and exponential functions.

4. Create a 2x2 subplot layout with `plt.subplot` for each function and plot the respective y-values.

5. Set titles, x-axis labels, y-axis labels, and adjust y-limits where needed, then display the plot with `plt.tight\_layout()` and `plt.show()`.

**PROGRAM:**

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 10, 100)

y1 = np.sin(x)

y2 = np.cos(x)

y3 = np.tan(x)

y4 = np.exp(x / 3)

plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1)

plt.plot(x, y1, 'b')

plt.title('Sine Function')

plt.xlabel('x')

plt.ylabel('sin(x)')

plt.subplot(2, 2, 2)

plt.plot(x, y2, 'r')

plt.title('Cosine Function')

plt.xlabel('x')

plt.ylabel('cos(x)')

plt.subplot(2, 2, 3)

plt.plot(x, y3, 'g')

plt.title('Tangent Function')

plt.xlabel('x')

plt.ylabel('tan(x)')

plt.ylim(-10, 10)

plt.subplot(2, 2, 4)

plt.plot(x, y4, 'm')

plt.title('Exponential Function')

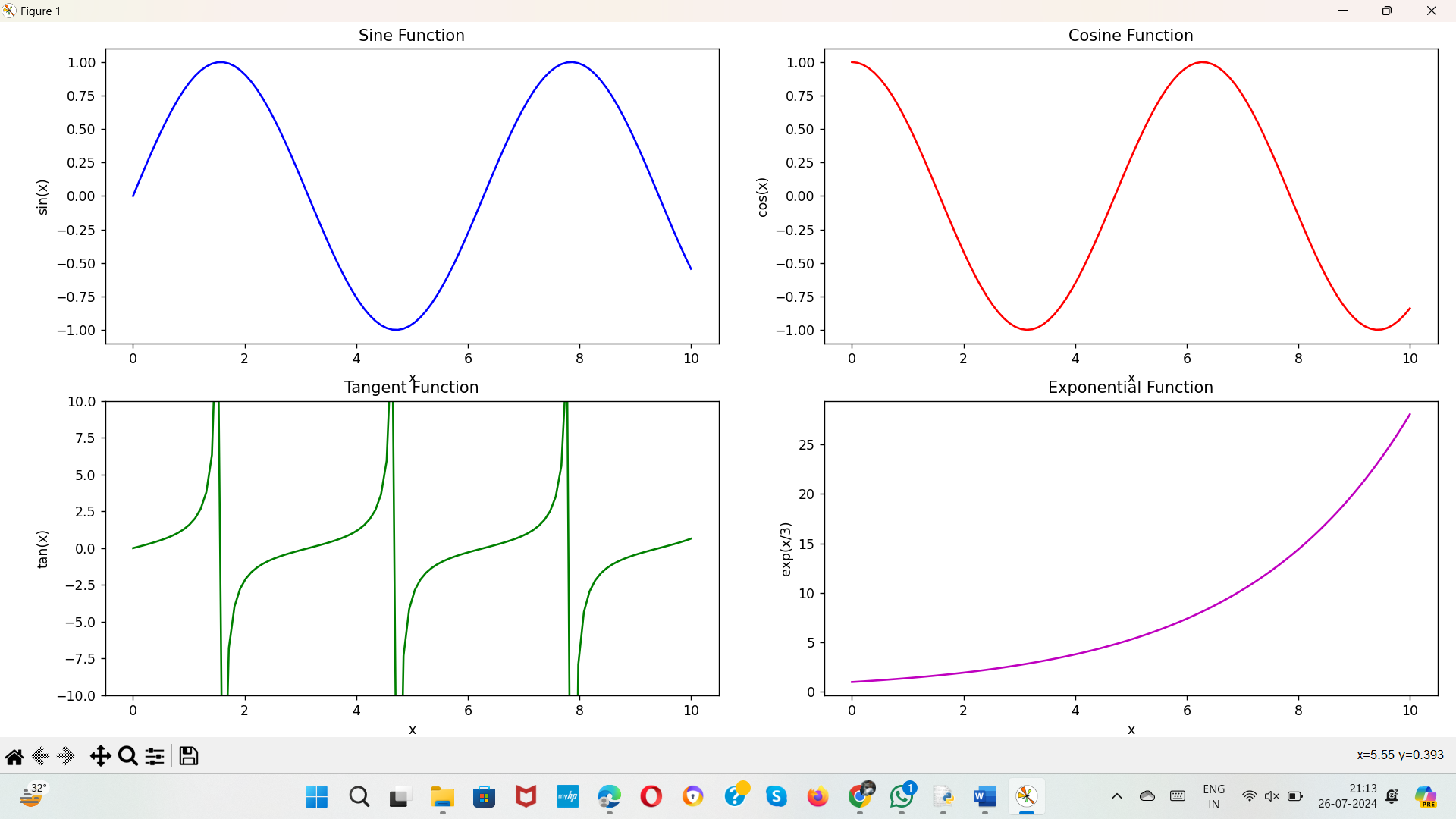
plt.xlabel('x')

plt.ylabel('exp(x/3)')

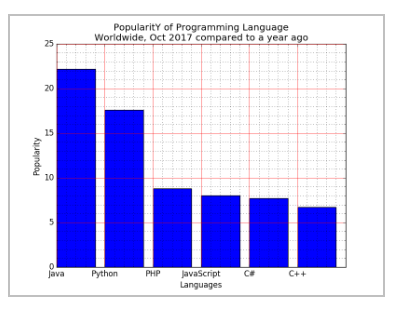
plt.tight\_layout()

plt.show()

**SAMPLE OUTPUT:**



27.Write a Python programming to display a bar chart of the popularity of programming Languages.  
Sample data:  
Programming languages: Java, Python, PHP, JavaScript, C#, C++  
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



**AIM:**

To develop a a Python programming to display a bar chart of the popularity of programming Languages.

**ALGORITHM:**

1. Import the `matplotlib.pyplot` library as `plt`.

2. Define lists for programming languages and their popularity.

3. Create a bar plot with `plt.bar`, specifying the languages, popularity, and color.

4. Set the plot title, x-axis label, and y-axis label.

5. Display the plot using `plt.show()`.

**PROGRAM:**

import matplotlib.pyplot as plt

languages = ['Python', 'Java', 'JavaScript', 'C#', 'PHP', 'C++', 'Ruby']

popularity = [30, 20, 25, 15, 10, 18, 12]

plt.figure(figsize=(10, 6))

plt.bar(languages, popularity, color='skyblue')

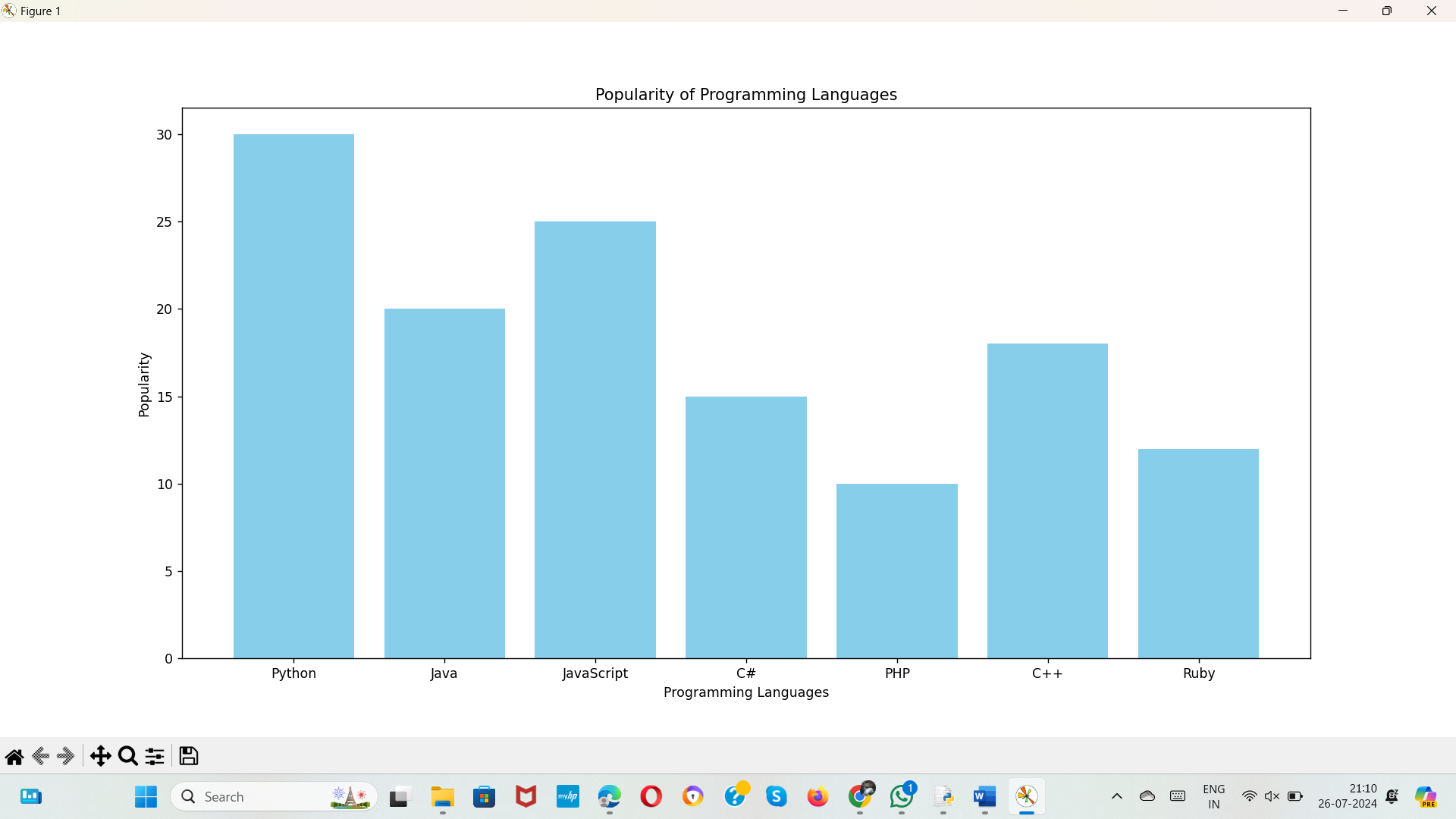
plt.title('Popularity of Programming Languages')

plt.xlabel('Programming Languages')

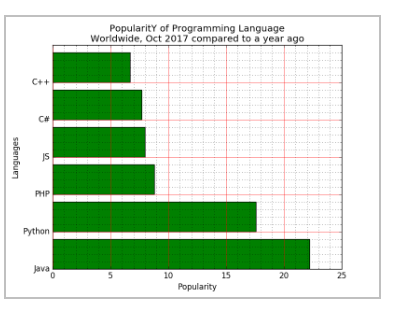
plt.ylabel('Popularity')

plt.show()

**SAMPLE OUTPUT:**



28.Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.  
Sample data:  
Programming languages: Java, Python, PHP, JavaScript, C#, C++  
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



**AIM:**

To develop a a python programming to display a horizontal bar chart of the popularity of programming Languages.

**ALGORITHM:**

1. Import the `matplotlib.pyplot` library as `plt`.

2. Define lists for programming languages, their popularity, and corresponding colors.

3. Create a horizontal bar plot with `plt.barh`, specifying the languages, popularity, and colors.

4. Set the plot title, x-axis label, and y-axis label.

5. Display the plot using `plt.show()`.

**PROGRAM:**

import matplotlib.pyplot as plt

languages = ['Python', 'Java', 'JavaScript', 'C#', 'PHP', 'C++', 'Ruby']

popularity = [30, 20, 25, 15, 10, 18, 12]

colors = ['blue', 'green', 'red', 'purple', 'orange', 'brown', 'pink']

plt.figure(figsize=(10, 6))

plt.barh(languages, popularity, color=colors)

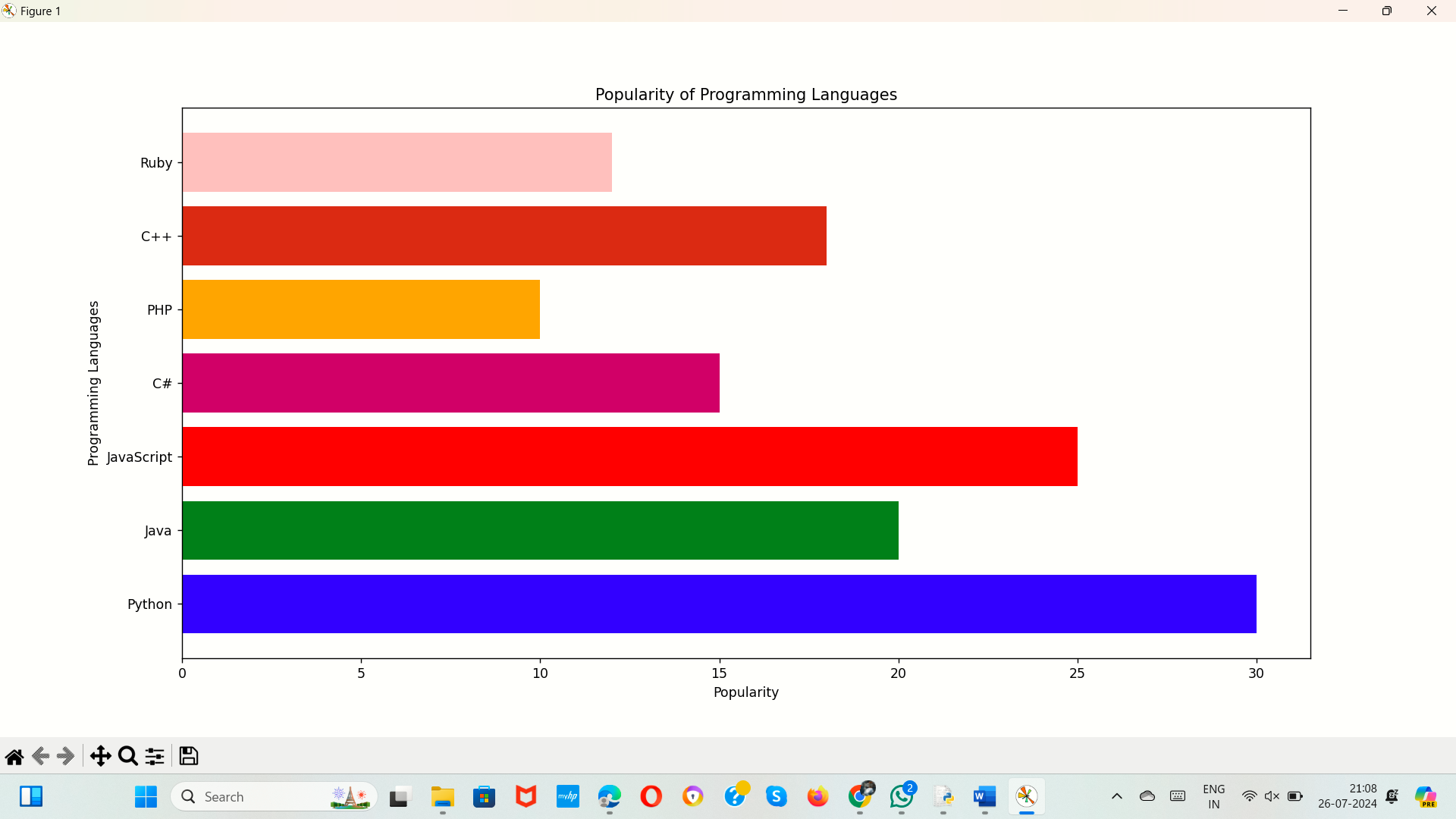
plt.title('Popularity of Programming Languages')

plt.xlabel('Popularity')

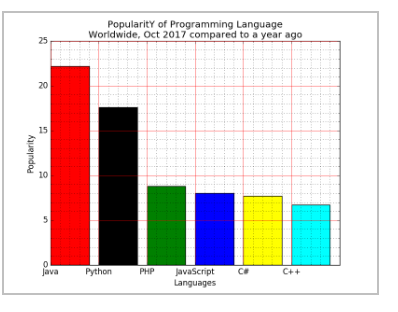
plt.ylabel('Programming Languages')

plt.show()

**SAMPLE OUTPUT:**



29.Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.  
Sample data:  
Programming languages: Java, Python, PHP, JavaScript, C#, C++  
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



**AIM:**

To develop a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.

**ALGORITHM:**

1. Import the `matplotlib.pyplot` library as `plt`.

2. Define lists for programming languages, their popularity, and corresponding colors.

3. Create a vertical bar plot with `plt.bar`, specifying the languages, popularity, and colors.

4. Set the plot title, x-axis label, and y-axis label.

5. Display the plot using `plt.show()`.

**PROGRAM:**

import matplotlib.pyplot as plt

languages = ['Python', 'Java', 'JavaScript', 'C#', 'PHP', 'C++', 'Ruby']

popularity = [30, 20, 25, 15, 10, 18, 12]

colors = ['blue', 'green', 'red', 'purple', 'orange', 'brown', 'pink']

plt.figure(figsize=(10, 6))

plt.bar(languages, popularity, color=colors)

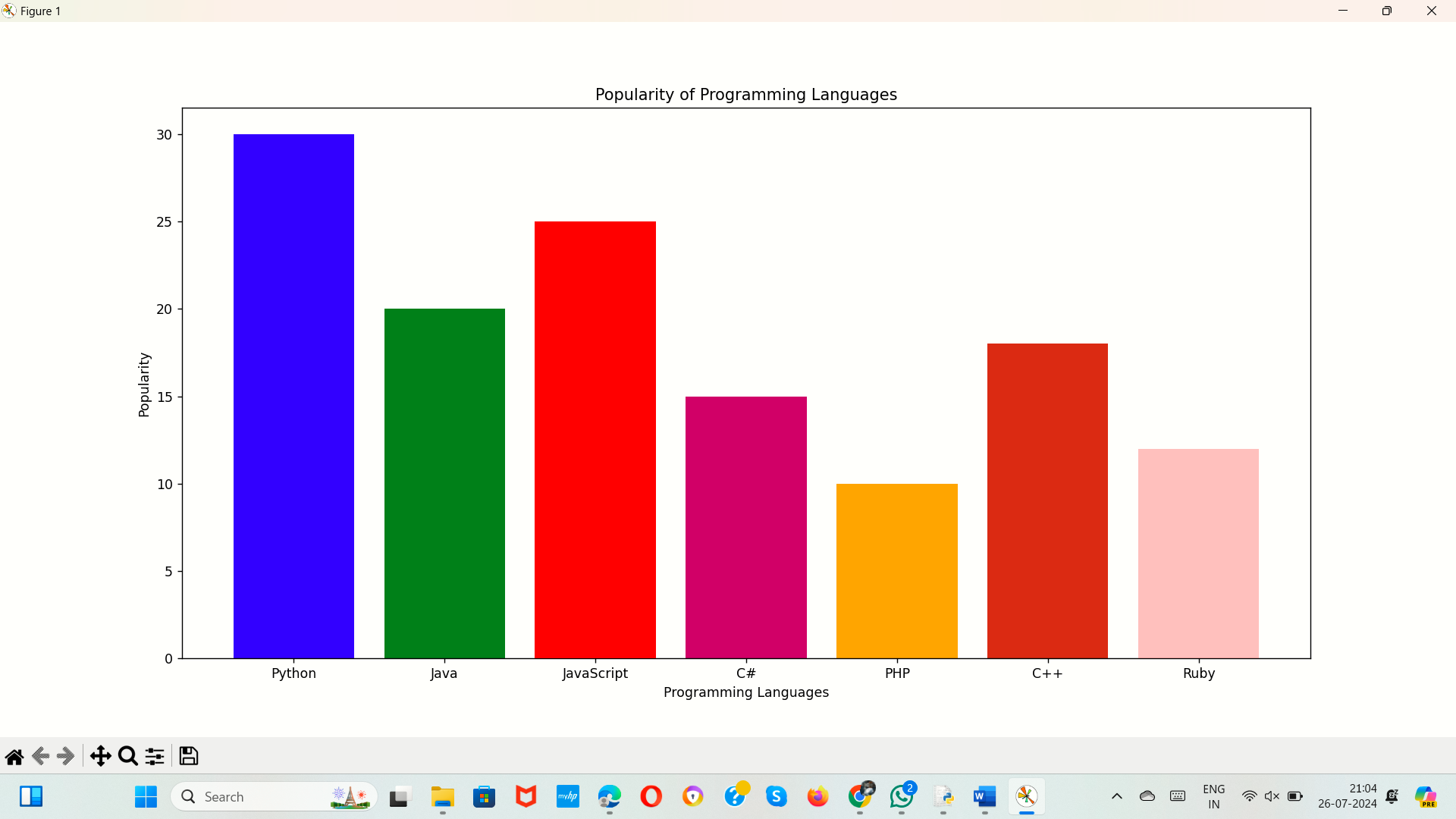
plt.title('Popularity of Programming Languages')

plt.xlabel('Programming Languages')

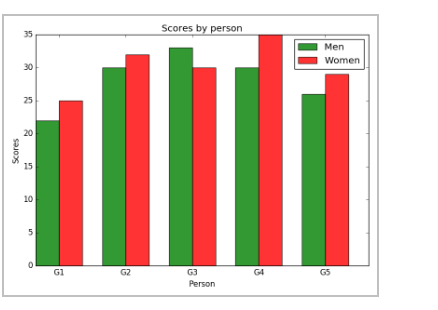
plt.ylabel('Popularity')

plt.show()

**SAMPLE OUTPUT:**



30.Write a Python program to create bar plot of scores by group and gender. Use multiple X values on the same chart for men and women.  
  
Sample Data:  
Means (men) = (22, 30, 35, 35, 26)  
Means (women) = (25, 32, 30, 35, 29)



**AIM:**

To develop a Python program to create bar plot of scores by group and gender. Use multiple X values on the same chart for men and women

**ALGORITHM:**

1. Import the `matplotlib.pyplot` and `numpy` libraries.

2. Define lists for group names, and the means for men and women.

3. Set the bar width and calculate the positions for each bar group.

4. Create two bar plots with `plt.bar` for men and women, offsetting the positions for side-by-side bars.

5. Add labels, a title, and a legend, then display the plot using `plt.show()`.

**PROGRAM:**

import matplotlib.pyplot as plt

import numpy as n

groups = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']

means\_men = [22, 30, 35, 35, 26]

means\_women = [25, 32, 30, 35, 29]

bar\_width = 0.3

index = np.arange(len(groups))

bars1 = plt.bar(index, means\_men, bar\_width, label='Men', color='blue')

bars2 = plt.bar(index + bar\_width, means\_women, bar\_width, label='Women', color='pink')

plt.xlabel('Groups')

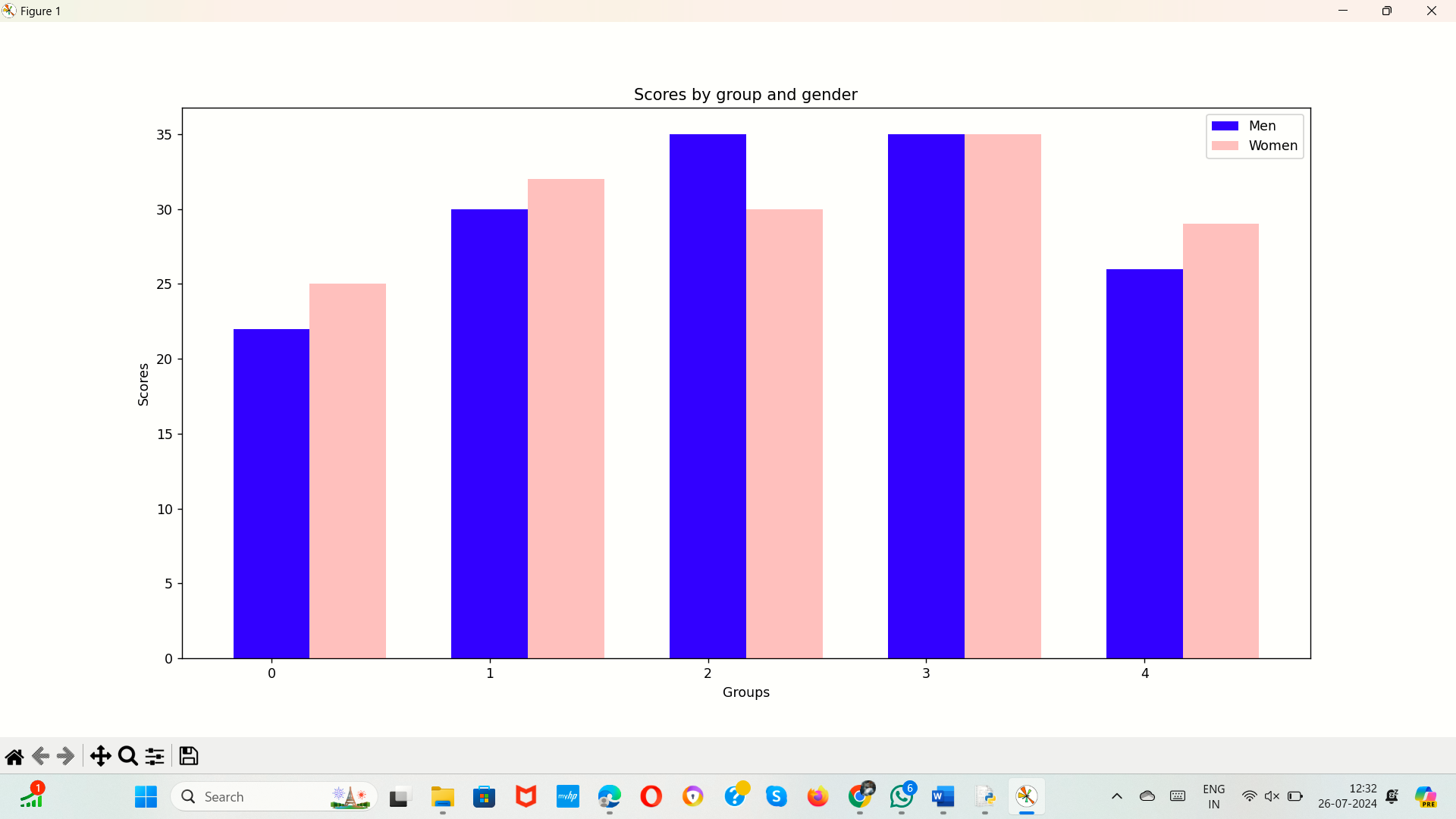
plt.ylabel('Scores')

plt.title('Scores by group and gender')

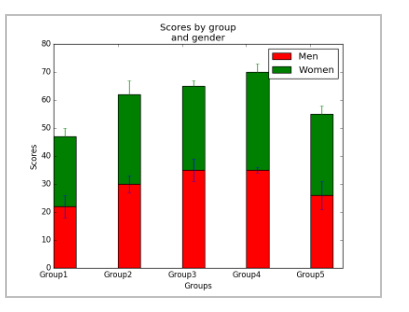
plt.legend()

plt.show()

**SAMPLE OUTPUT:**



31.Write a Python program to create a stacked bar plot with error bars.  
Note: Use bottom to stack the women?s bars on top of the men?s bars.  
Sample Data:  
Means (men) = (22, 30, 35, 35, 26)  
Means (women) = (25, 32, 30, 35, 29)  
Men Standard deviation = (4, 3, 4, 1, 5)  
Women Standard deviation = (3, 5, 2, 3, 3)



**AIM:**

TO develop a python program to create a stacked bar plot with error bars.

**ALGORITHM:**

1. Import the necessary libraries.

2. Define the groups and their corresponding data for men and women.

3. Define the error values for men and women.

4. Set the width of the bars.

5. Create an array of indices based on the number of groups.

6. Plot the bar chart for men's scores with error bars.

7. Plot the bar chart for women's scores with error bars, stacking them on top of men's scores.

8. Set the x-axis label to "groups".

9. Set the y-axis label to "scores".

10. Display the plot.

**PROGRAM:**

import numpy as np

import matplotlib.pyplot as plt

groups=["Group1","Group2","Group3","Group4"]

men=[12,23,34,45]

women=[23,34,45,56]

err\_men=[4,3,4,1]

err\_women=[3,5,2,3]

bar\_width=0.35

index=np.arange(len(groups))

plt.bar(index,men,bar\_width,yerr=err\_men,capsize=5)

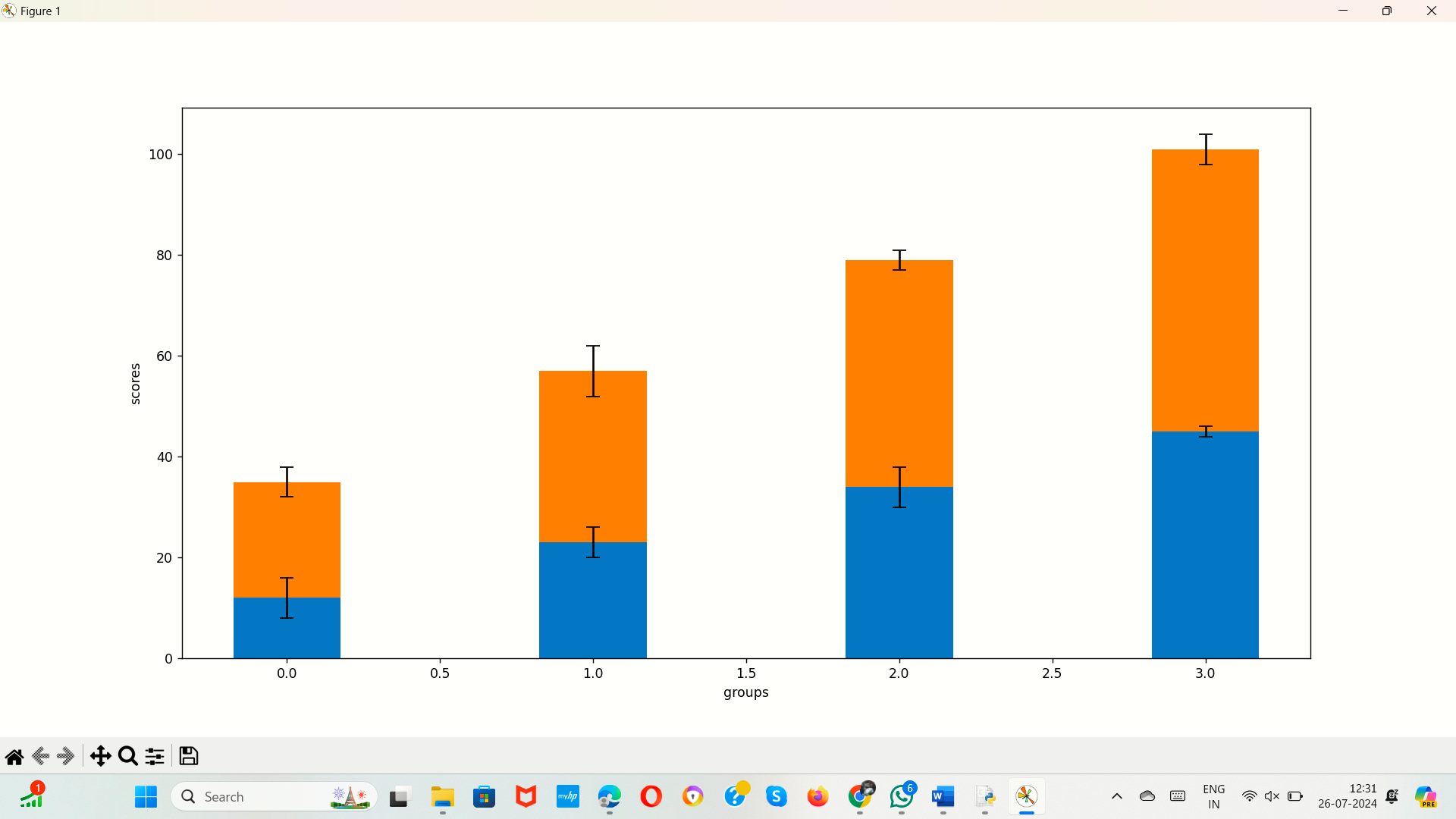
plt.bar(index,women,bar\_width,yerr=err\_women,bottom=men,capsize=5)

plt.xlabel("groups")

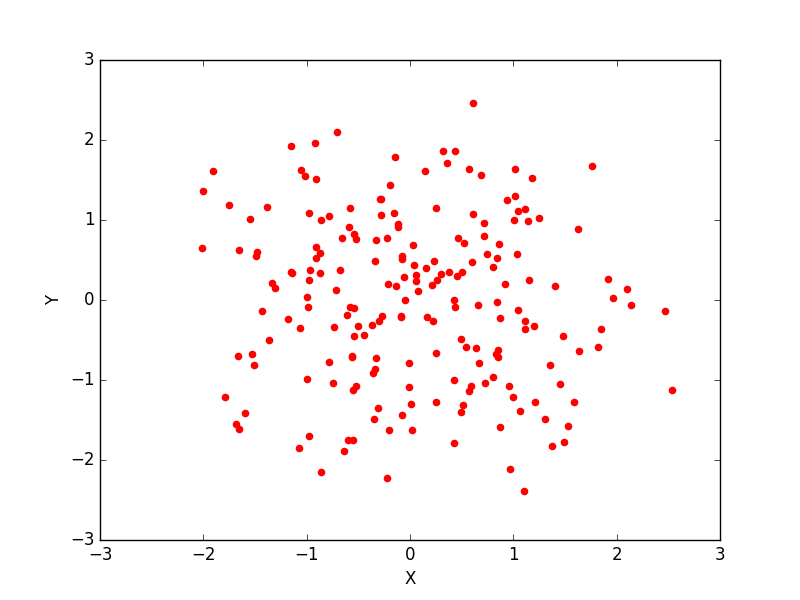
plt.ylabel("scores")

plt.show()

**SAMPLE OUTPUT:**



32.Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.



**AIM:**

To develop a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.

**ALGORITHM:**

1. Import the necessary library.

2. Define the x values.

3. Define the y values.

4. Create a scatter plot with the x and y values.

5. Set the x-axis label to "x".

6. Set the y-axis label to "y".

7. Display the plot.

**PROGRAM:**

import matplotlib.pyplot as plt

x=[12,23,34,45,56]

y=[23,34,45,56,67]

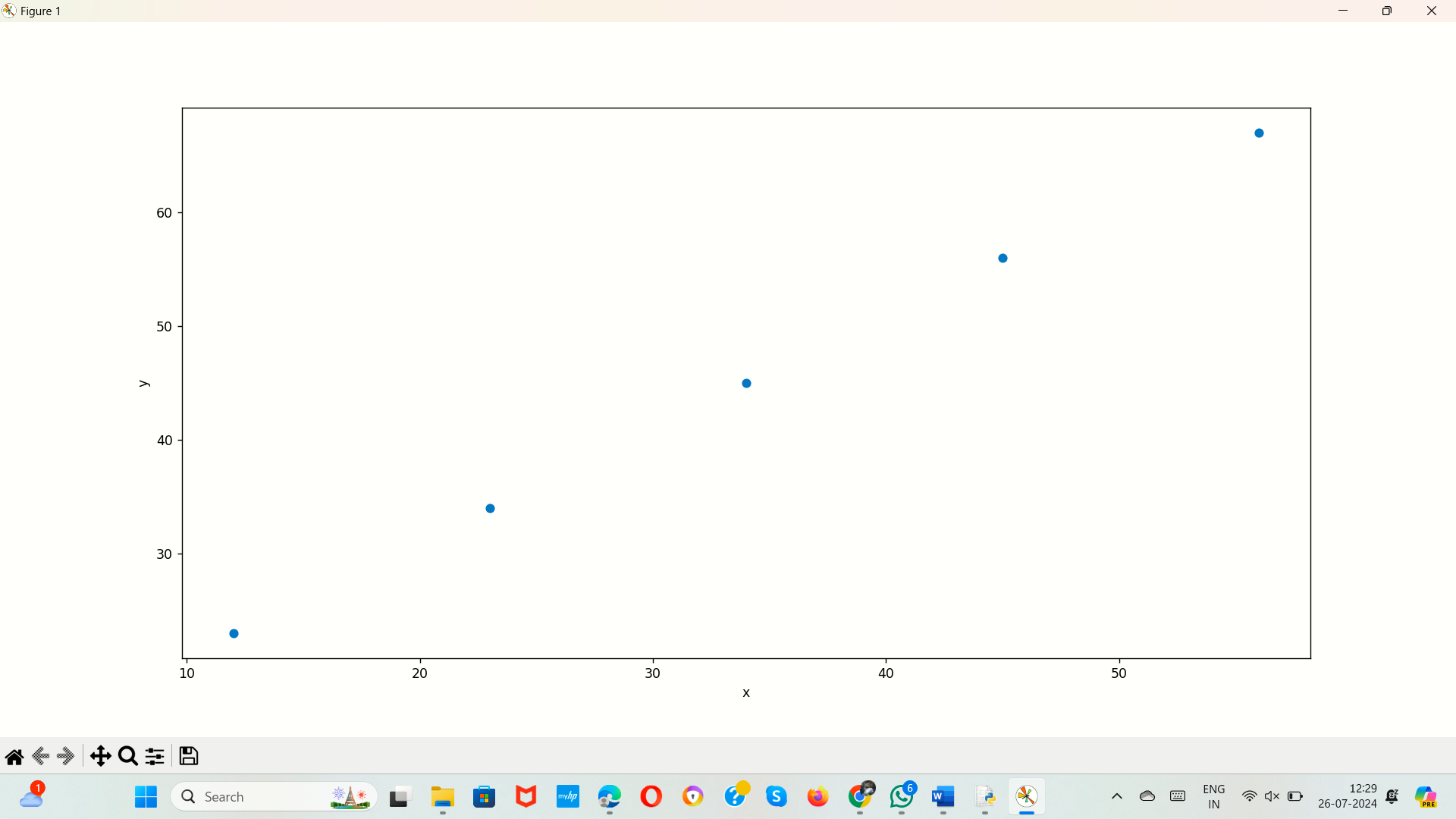
plt.scatter(x,y)

plt.xlabel("x")

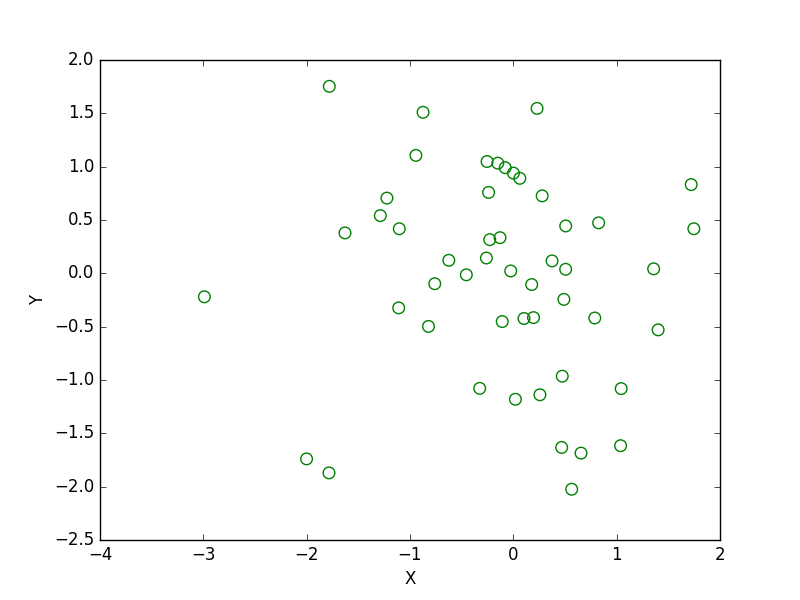
plt.ylabel("y")

plt.show()

**SAMPLE OUTPUT:**



33.Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.



**AIM:**

To develop a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

**ALGORITHM:**

1. Import the necessary library.

2. Define the sizes for the scatter plot markers.

3. Define the x values.

4. Define the y values.

5. Create a scatter plot with the x and y values, using green outlines and no face color for the markers, and sizes based on the defined sizes.

6. Display the plot.

**PROGRAM:**

import matplotlib.pyplot as plt

sizes=[50,120,146,173,187,193,137,174,219,616]

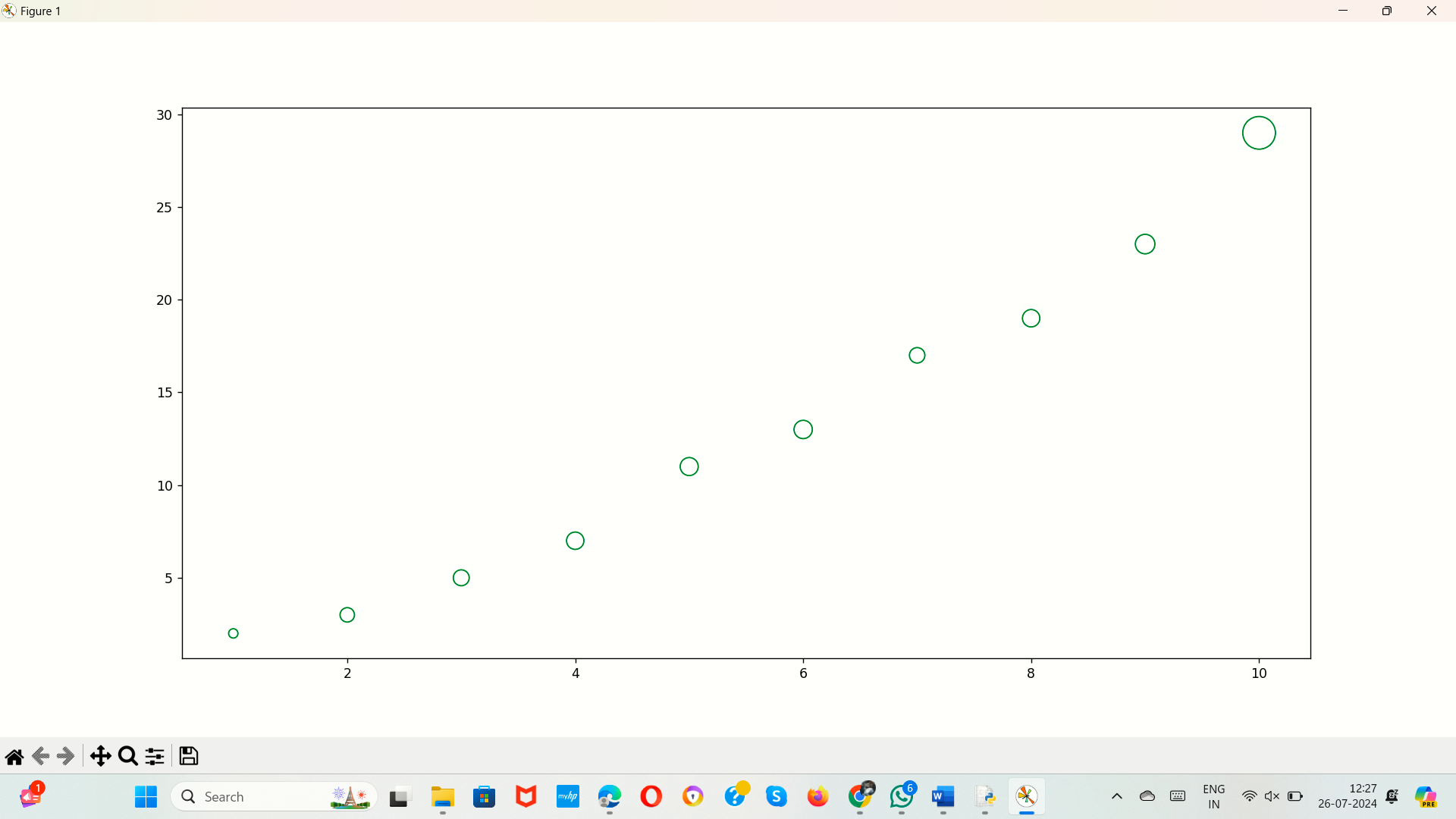
x=[1,2,3,4,5,6,7,8,9,10]

y=[2,3,5,7,11,13,17,19,23,29]

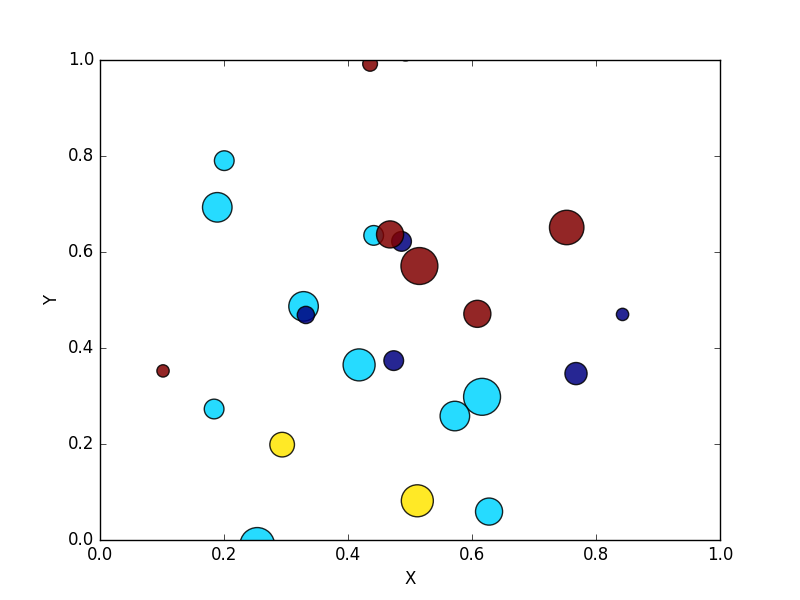
plt.scatter(x,y,color="Green",facecolors="none",s=sizes)

plt.show()

**SAMPLE OUTPUT:**



34.Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.



**AIM:**

To develop a python program to draw a scatter plot using random distributions to generate balls of different sizes.

**ALGORITHM:**

1. Import the necessary libraries.

2. Set the random seed for reproducibility.

3. Define the number of points.

4. Define the x values.

5. Define the y values.

6. Define the sizes for the scatter plot markers.

7. Generate random colors for the scatter plot markers.

8. Create a scatter plot with the x and y values, using the defined sizes and colors, with a specified alpha and colormap.

9. Set the x-axis label to 'X-axis'.

10. Set the y-axis label to 'Y-axis'.

11. Set the title of the plot.

12. Add a color bar to the plot.

13. Display the plot.

**PROGRAM:**

import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)

num\_points = 10

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

y = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]

sizes = [50, 120, 146, 173, 187, 193, 137, 174, 219, 616]

colors = np.random.rand(num\_points)

plt.scatter(x, y, s=sizes, c=colors, alpha=0.5, cmap='viridis')

plt.xlabel('X-axis')

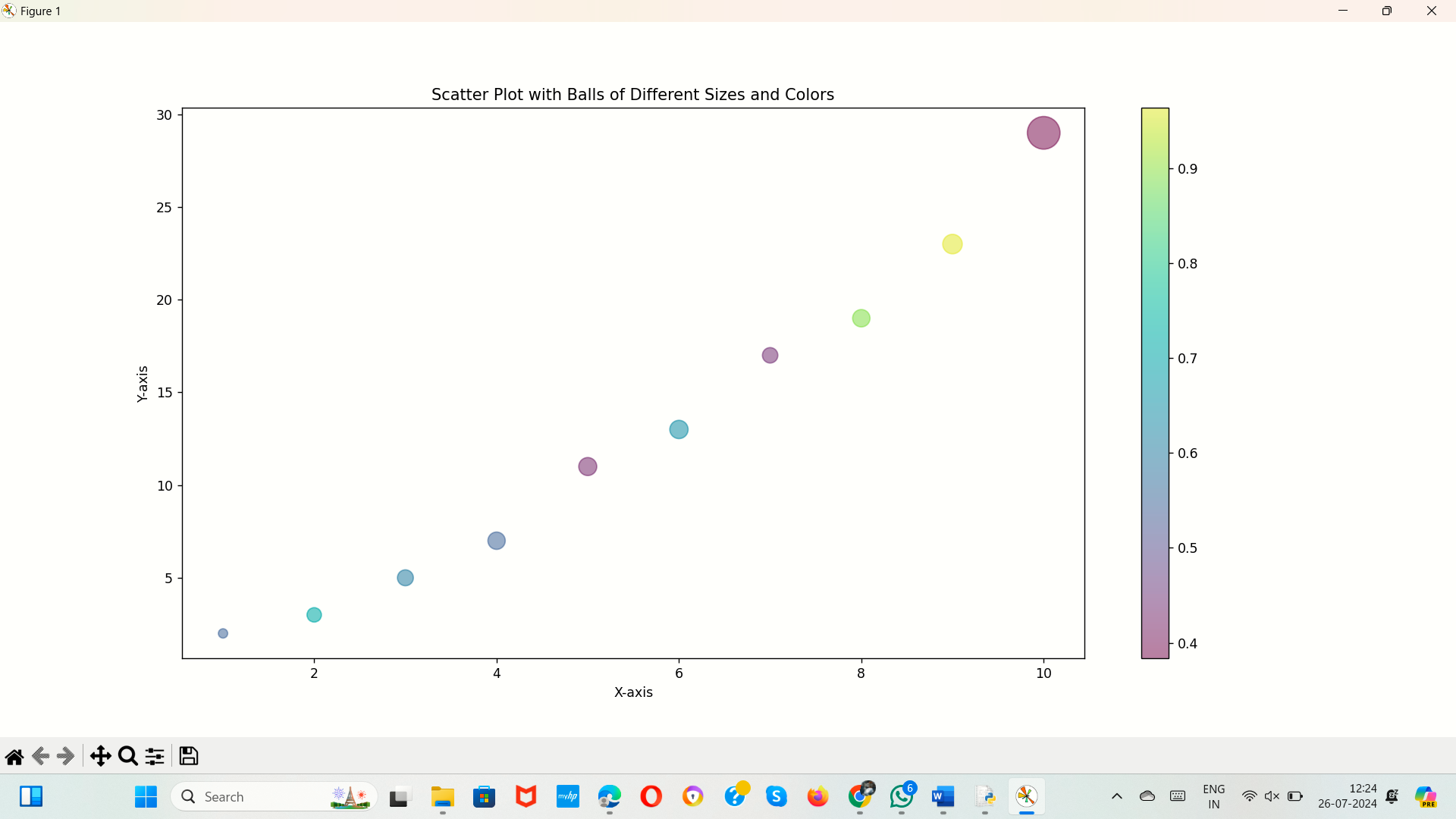
plt.ylabel('Y-axis')

plt.title('Scatter Plot with Balls of Different Sizes and Colors')

plt.colorbar()

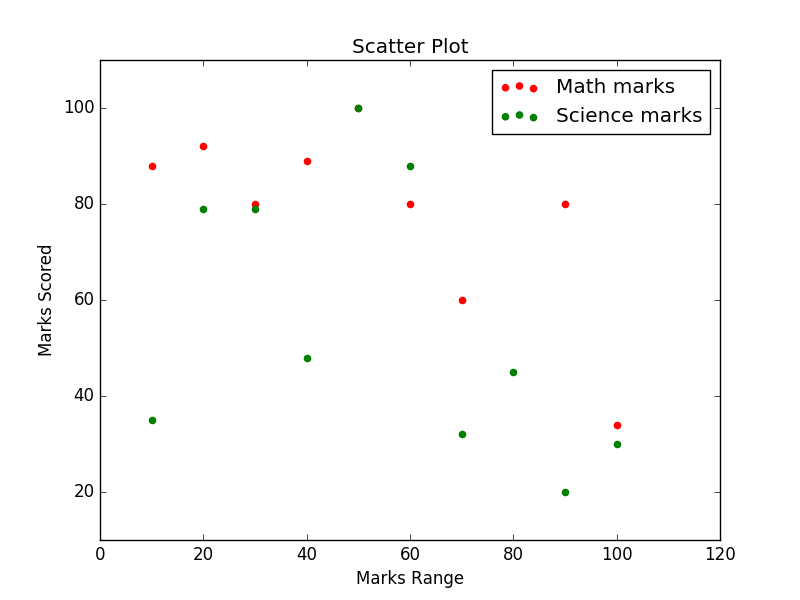
plt.show()

**SAMPLE OUTPUT:**



35.Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students.  
Sample data:

Test Data:  
math\_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]  
science\_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]  
marks\_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]



**AIM:**

To develop a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students.

**ALGORITHM:**

1. Import the necessary library.

2. Define the marks range.

3. Define the math marks.

4. Define the science marks.

5. Create a scatter plot for math marks with a star marker.

6. Create a scatter plot for science marks with a star marker.

7. Set the x-axis label to "marks range".

8. Set the y-axis label to "marks scored".

9. Display the plot.

**PROGRAM:**

import matplotlib.pyplot as plt

marks\_range=[10,20,30,40,50,60,70,80,90,100]

math\_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]

science\_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]

plt.scatter(marks\_range,math\_marks,marker="\*")

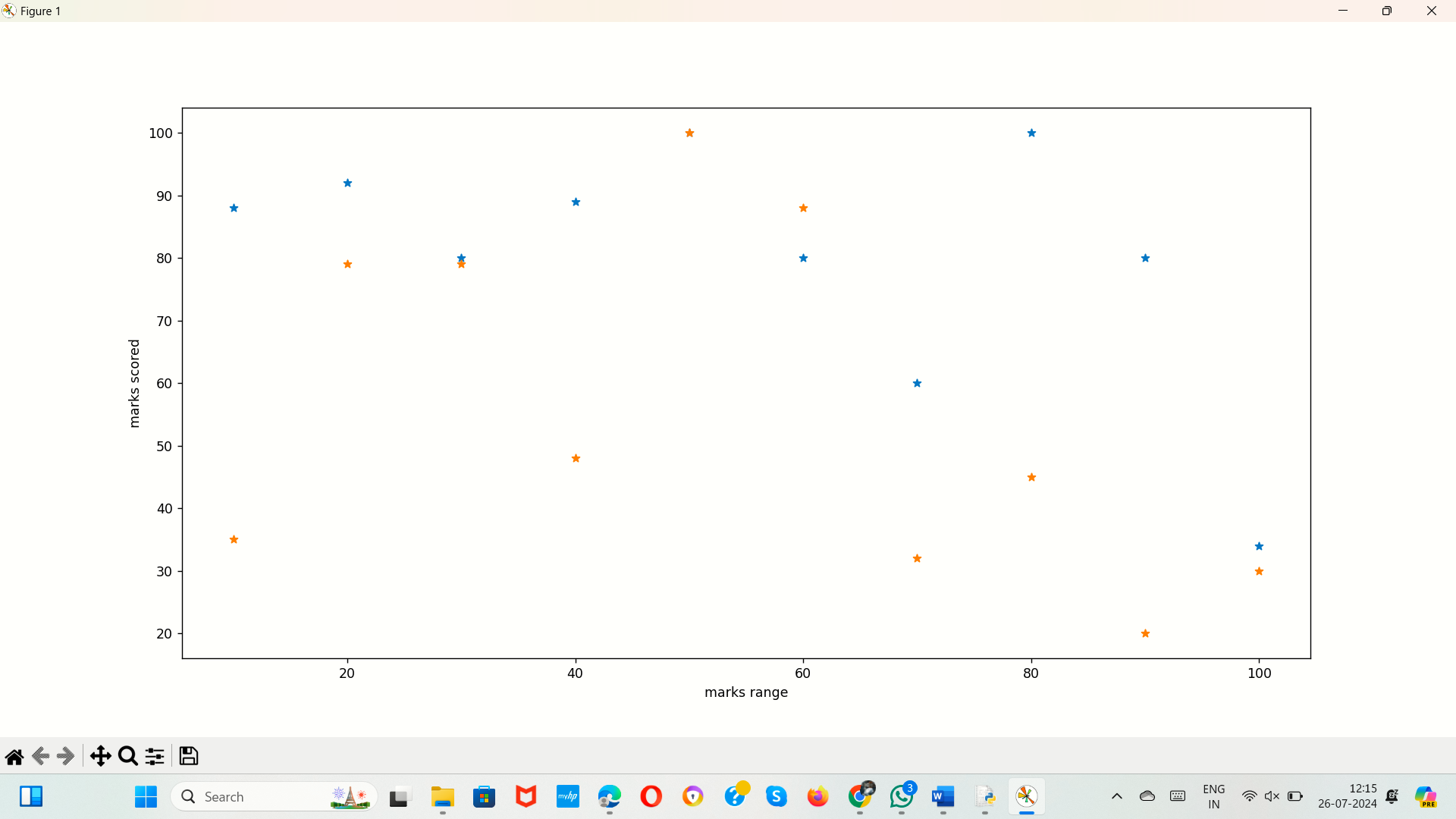
plt.scatter(marks\_range,science\_marks,marker="\*")

plt.xlabel("marks range")

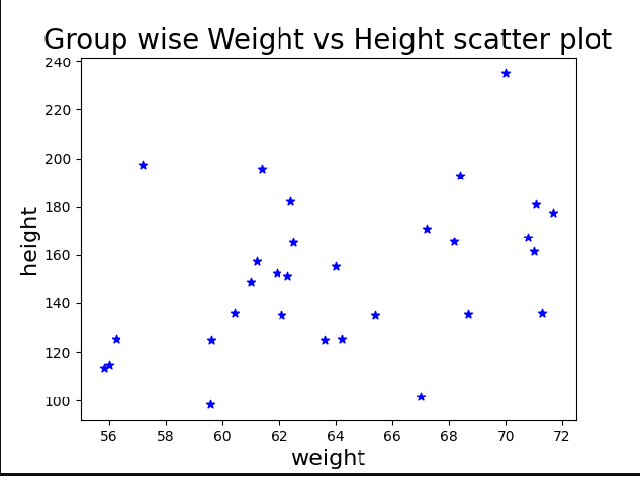
plt.ylabel("marks scored")

plt.show()

**SAMPLE OUTPUT:**



36.Write a Python program to draw a scatter plot for three different groups comparing weights and heights.



**AIM:**

To develop a Python program to draw a scatter plot for three different groups comparing weights and heights.

**ALGORITHM:**

1. Import the necessary library.

2. Define the weights and heights for Group 1.

3. Define the weights and heights for Group 2.

4. Define the weights and heights for Group 3.

5. Create a scatter plot for Group 1 with circle markers and a label.

6. Create a scatter plot for Group 2 with cross markers and a label.

7. Create a scatter plot for Group 3 with triangle markers and a label.

8. Set the x-axis label to 'Weight (kg)'.

9. Set the y-axis label to 'Height (cm)'.

10. Set the title of the plot.

11. Add a legend to the plot.

12. Display the plot.

**PROGRAM:**

import matplotlib.pyplot as plt

group1\_weights = [50, 55, 60, 65, 70, 75, 80]

group1\_heights = [150, 160, 155, 170, 175, 180, 165]

group2\_weights = [60, 65, 70, 75, 80, 85, 90]

group2\_heights = [155, 165, 160, 175, 180, 185, 170]

group3\_weights = [70, 75, 80, 85, 90, 95, 100]

group3\_heights = [160, 170, 165, 180, 185, 190, 175]

plt.scatter(group1\_weights, group1\_heights, marker='o', label='Group 1')

plt.scatter(group2\_weights, group2\_heights, marker='x', label='Group 2')

plt.scatter(group3\_weights, group3\_heights, marker='^', label='Group 3')

plt.xlabel('Weight (kg)')

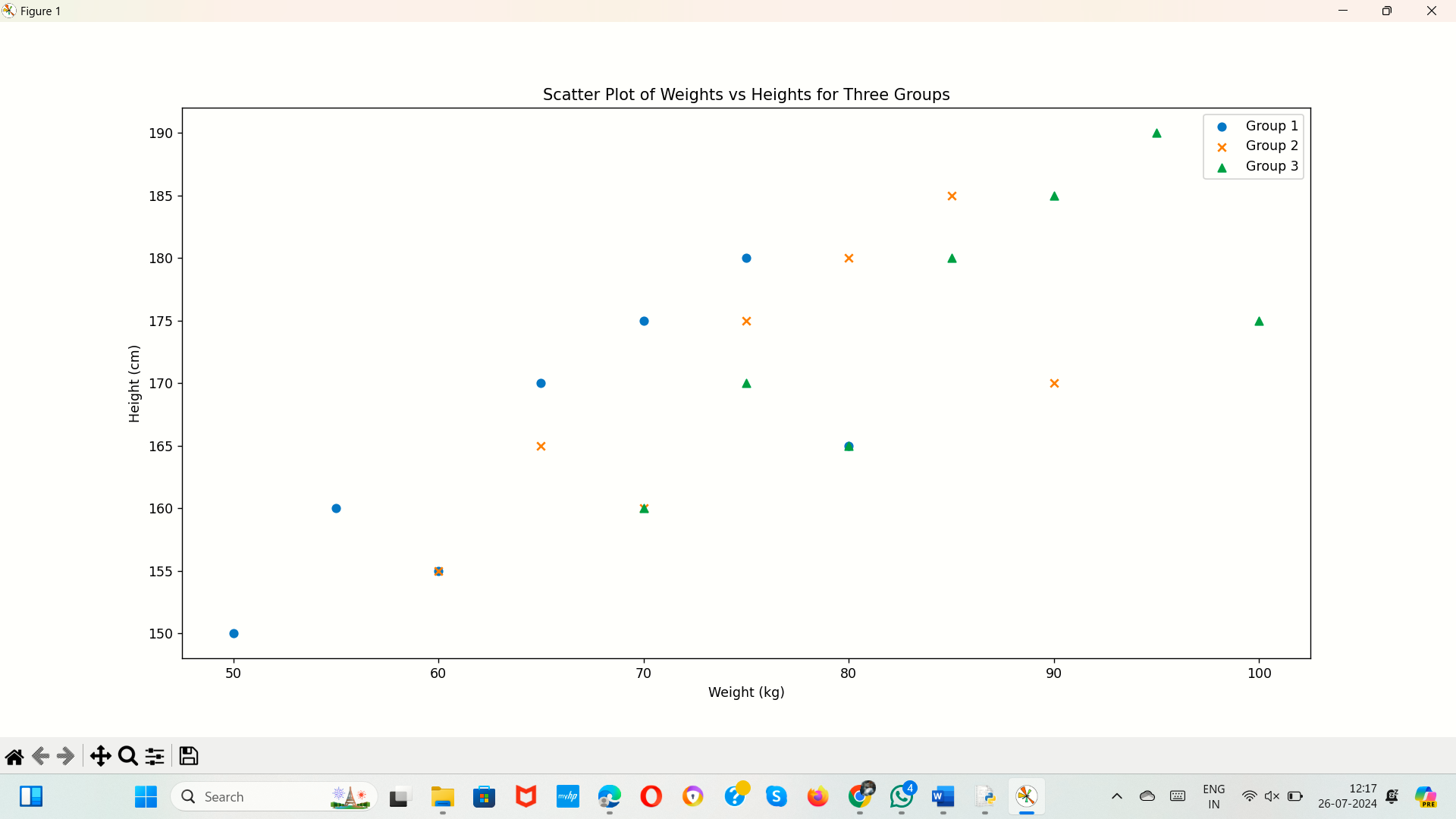
plt.ylabel('Height (cm)')

plt.title('Scatter Plot of Weights vs Heights for Three Groups')

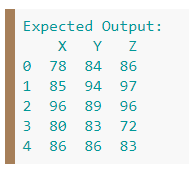
plt.legend()

plt.show()

**SAMPLE OUTPUT:**



37.Write a Pandas program to create a dataframe from a dictionary and display it.  
Sample data: {'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]}



**AIM:**

To develop a Pandas program to create a dataframe from a dictionary and display it.

**ALGORITHM:**

1. Import the necessary library.

2. Define the data dictionary with columns 'X', 'Y', and 'Z'.

3. Convert the dictionary to a Pandas DataFrame.

4. Print the DataFrame.

**PROGRAM:**

import pandas as pd

data = {'X': [78, 85, 96, 80, 86],

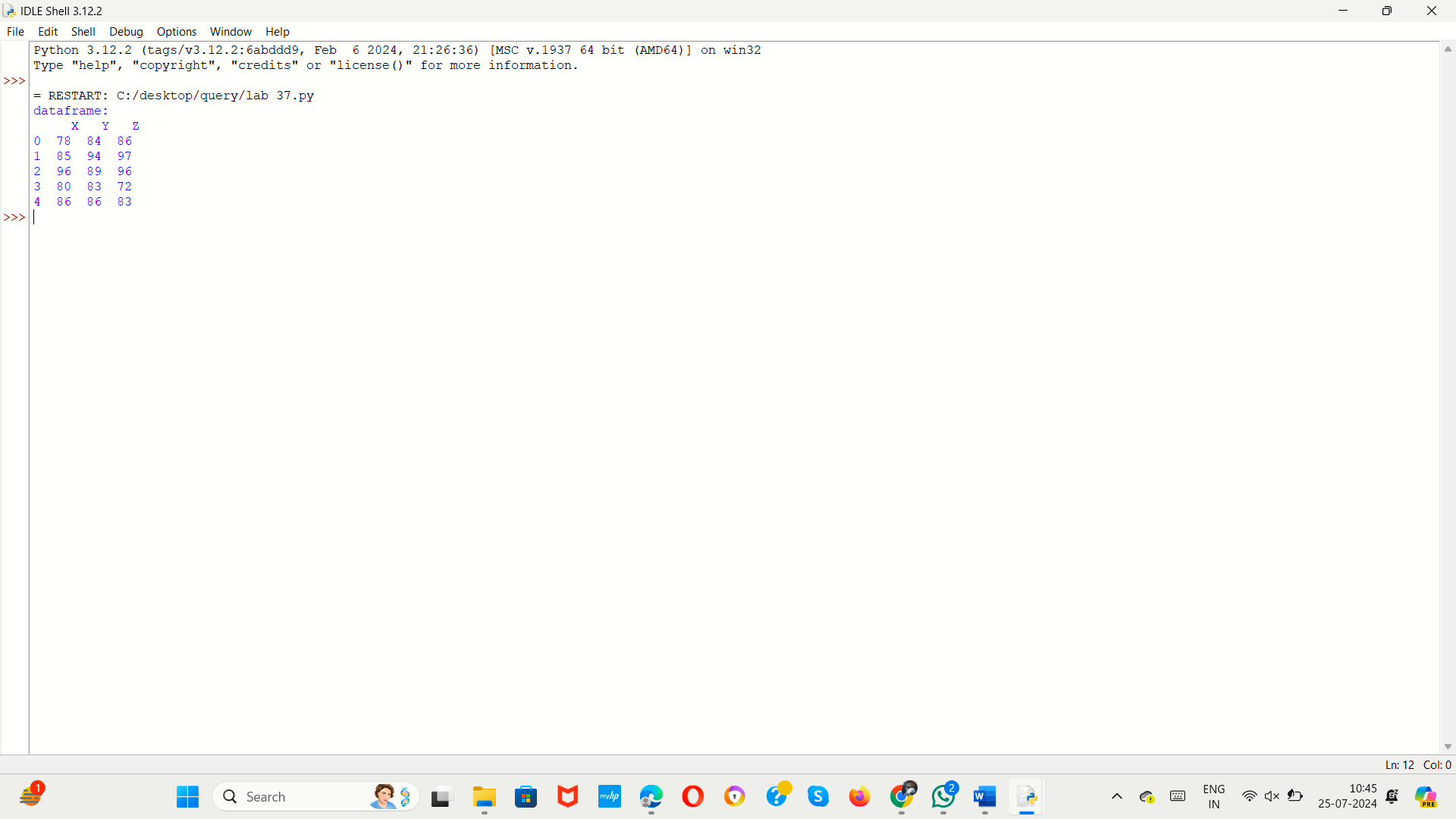
'Y': [84, 94, 89, 83, 86],

'Z': [86, 97, 96, 72, 83]}

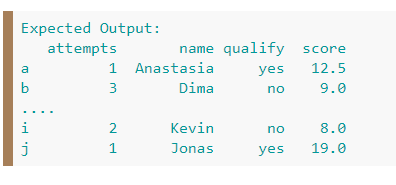
df = pd.DataFrame(data)

print("dataframe:\n",df)

**SAMPLE OUTPUT:**



38.Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.  
Sample Python dictionary data and list labels:  
exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],  
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],  
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],  
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}  
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']



**AIM:**

To develop a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

**ALGORITHM:**

1. Import the necessary libraries.

2. Define the exam data dictionary with 'name', 'score', 'attempts', and 'qualify' columns.

3. Convert the dictionary to a Pandas DataFrame.

4. Set the index of the DataFrame.

5. Drop rows with missing values.

6. Print the resulting DataFrame.

**PROGRAM:**

import pandas as pd

import numpy as np

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

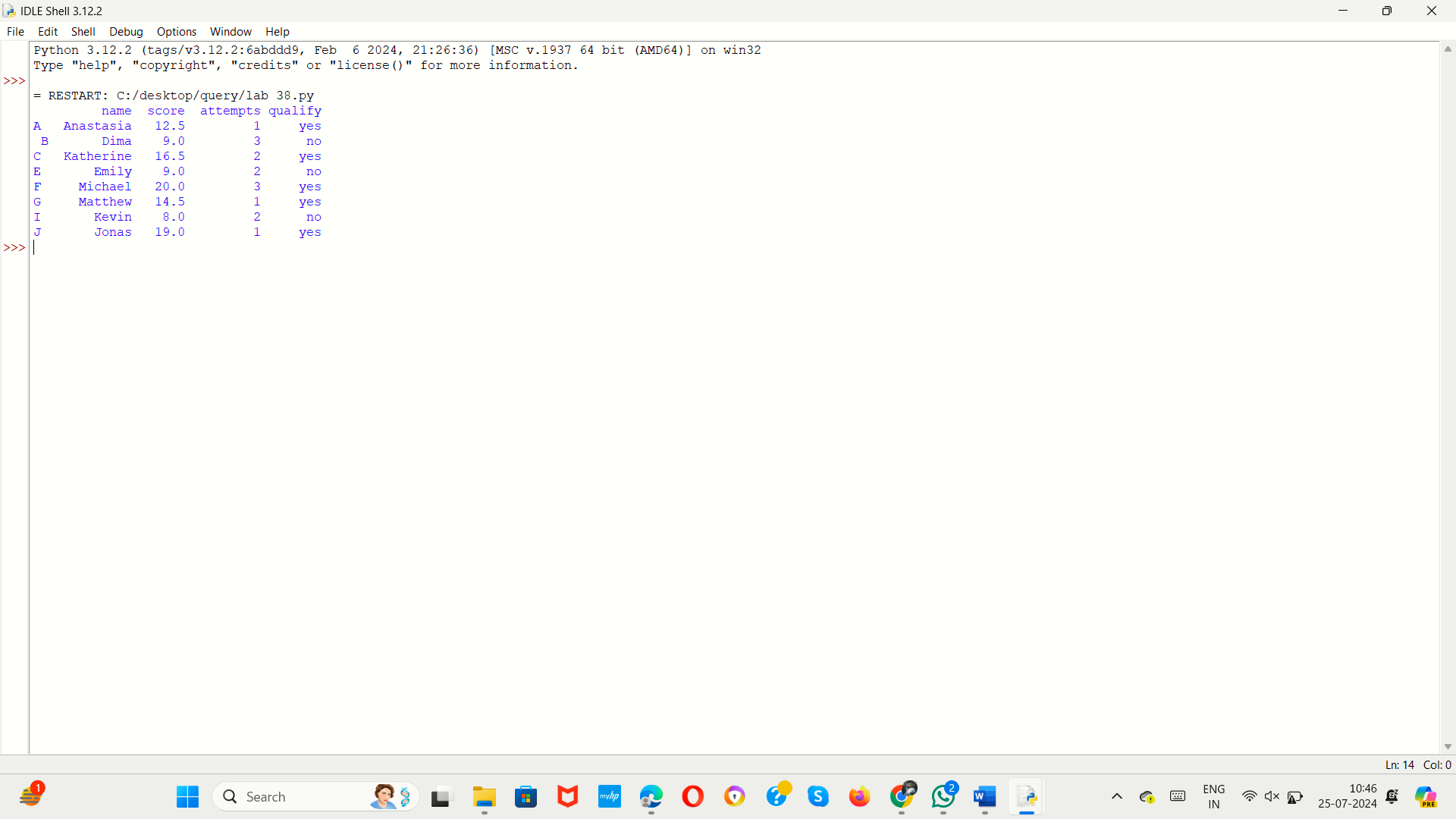
df=pd.DataFrame(exam\_data)

df.index= ['A',' B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

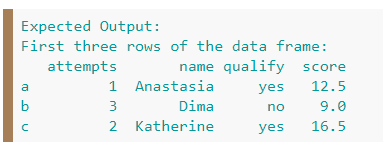
select=df.dropna()

print(select)

**SAMPLE OUTPUT:**



39.Write a Pandas program to get the first 3 rows of a given DataFrame.  
Sample Python dictionary data and list labels:  
exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],  
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],  
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],  
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}  
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']



**AIM:**

To develop a pandas program to get the first 3 rows of a given DataFrame.

**ALGORITHM:**

1. Import the necessary libraries.

2. Define the exam data dictionary with 'name', 'score', 'attempts', and 'qualify' columns.

3. Define the labels for the DataFrame index.

4. Convert the dictionary to a Pandas DataFrame with the specified index.

5. Select the first three rows of the DataFrame.

6. Print the first three rows of the DataFrame.

**PROGRAM:**

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']'''

import pandas as pd

import numpy as np

exam\_data = {

'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']

}

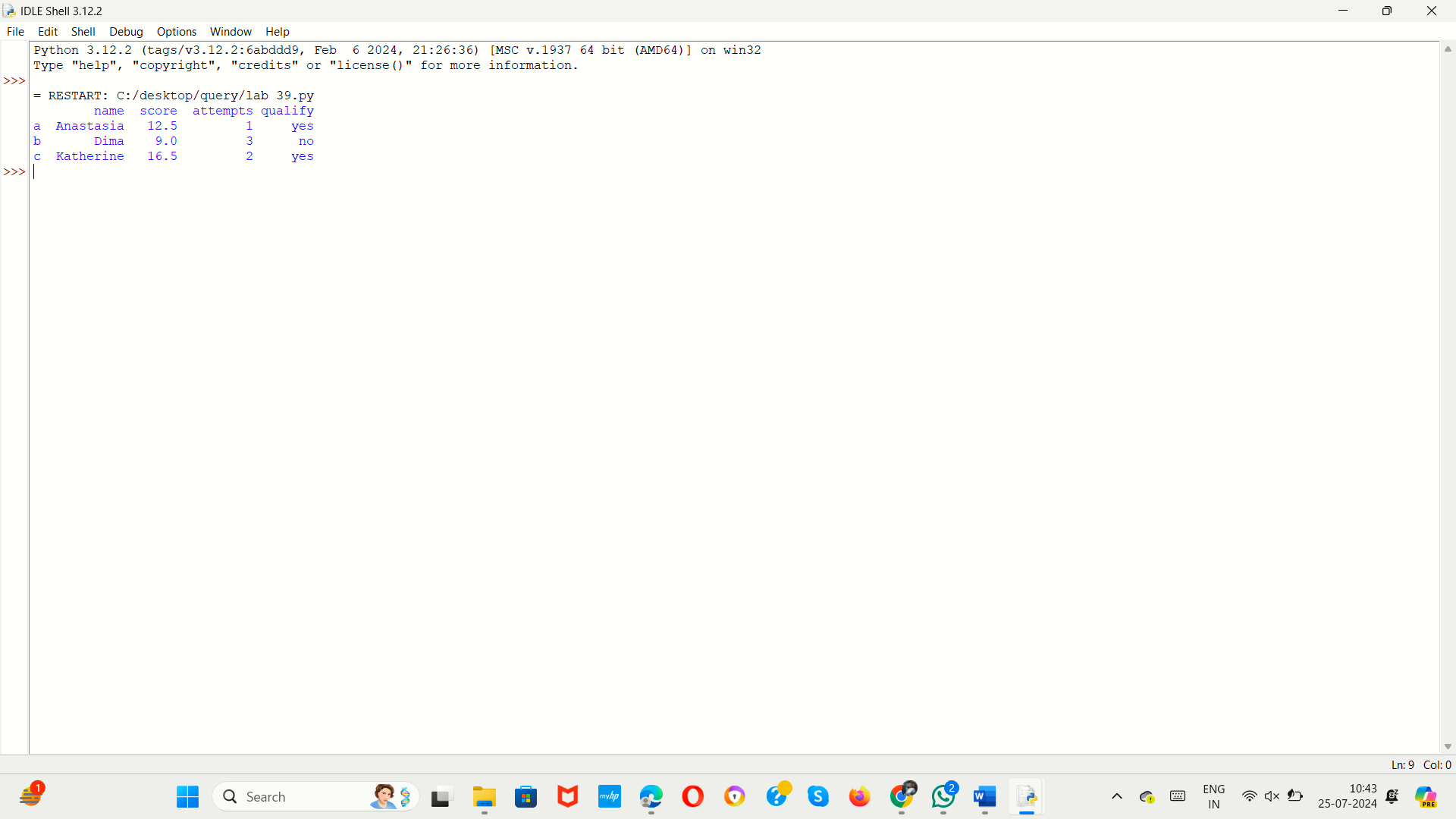
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data, index=labels)

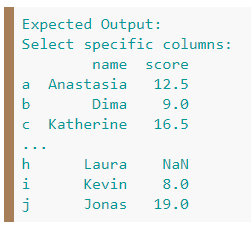
first\_three\_rows = df.head(3)

print(first\_three\_rows)

**SAMPLE OUTPUT:**



1. Write a Pandas program to select the 'name' and 'score' columns from the following DataFrame.  
   Sample Python dictionary data and list labels:  
   exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],  
   'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],  
   'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],  
   'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}  
   labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']



**AIM:**

To develop a a Pandas program to select the 'name' and 'score' columns from the following DataFrame.

**ALGORITHM:**

1. Import the necessary libraries.

2. Define the exam data dictionary with 'name', 'score', 'attempts', and 'qualify' columns.

3. Define the labels for the DataFrame index.

4. Convert the dictionary to a Pandas DataFrame with the specified index.

5. Select the 'name' and 'score' columns from the DataFrame.

6. Print the selected columns.

**PROGRAM:**

import pandas as pd

import numpy as np

exam\_data = {

'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']

}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data, index=labels)

selected\_columns = df[['name', 'score']]

print(selected\_columns)

**SAMPLE OUTPUT:**

