BANGALORE INSTITUTE OF TECHNOLOGY

K.R. ROAD, V.V PURAM, BANGALORE – 560 004

(AFFILIATED TO VTU, BELAGAVI)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

COURSE CODE: BCS358D

DATA VISUALIZATION WITH PYTHON LABORATORY

As per Choice Based Credit System Scheme (CBCS)

FOR III SEMESTER ISE AS PRESCRIBED BY VTU

Academic Year 2023-24

Prepared By:

Deeksha Chandra,
Assistant Professor
Dept. of ISE, BIT

BANGALORE INSTITUTE OF TECHNOLOGY

VISION:

Establish and develop the Institute as the Centre of higher learning, ever abreast with expanding horizon of knowledge in the field of Engineering and Technology with entrepreneurial thinking, leadership excellence for life-long success and solve societal problems.

MISSION:

- Provide high quality education in the Engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.
- Develop the Institute as a leader in Science, Engineering, Technology, Management and Research and apply knowledge for the benefit of society.
- Establish mutual beneficial partnerships with Industry, Alumni, Local, State and Central Governments by Public Service Assistance and Collaborative Research.
- Inculcate personality development through sports, cultural and extracurricular activities and engage in social, economic and professional challenges.

Bangalore Institute of Technology

K. R. Road, V. V. Pura, Bengaluru- 560004

Department of Information Science and Engineering

VISION:

Empower every student to be innovative, creative and productive in the field of Information Technology by imparting quality technical education, developing skills and inculcating human values.

MISSION:

M1	To evolve continually as a Centre of Excellence in offering quality Information Technology **Education**.
M2	To nurture the students to meet the global competency in industry for **Employment**.
M3	To promote collaboration with industry and academia for constructive interaction to empower **Entrepreneurship.**
M4	To provide reliable, contemporary and integrated technology to support and facilitate **Teaching, Learning, Research and Service.**

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

PEO-1	Uplift the students through Information Technology **Education.**
PEO-2	Provide exposure to emerging technologies and train them to **Employable** in multi-disciplinary industries.
PEO-3	Motivate them to become good professional Engineers and **Entrepreneur.**
PEO-4	Inspire them to prepare for **Higher Learning and Research.**

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1	To provide our graduates **with Core Competence in Information Processing and Management.**
PSO-2	To provide our graduates with Higher Learning in **Computing Skills.**

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Bangalore Institute of Technology

K. R. Road, V.V. Pura, Bengaluru 560004

Department of Information Science and Engineering

DATA VISUALIZATION WITH PYTHON LABORATORY (BCS358D)

PRE-REQUISITES:

Python programming concepts.

COURSE LEARNING OBJECTIVES (CLO)

This laboratory course enables students to get practical experience in design, develop, implement, analyze and evaluation/testing of

CLO 1: Demonstrate the use of IDLE or PyCharm IDE to create Python Applications.

CLO 2: Using Python programming language to develop programs for solving real-world problems.

CLO 3: Implement of Matplotlib for drawing different Plots.

CLO 4: Demonstrate working with Seaborn, Bokeh.

CLO 5: Working with Plotly for 3D, Time Series and Maps.

COURSE OUTCOMES (CO)

On the completion of this laboratory course, the students will be able to:

CO 1: Demonstrate the use of IDLE or PyCharm IDE to create Python Applications

CO 2: Use Python programming constructs to develop programs for solving real-world problems.

CO 3: Use Matplotlib for drawing different Plots.

CO 4: Demonstrate working with Seaborn, Bokeh for visualization.

CO 5: Use Plotly for drawing, Time Series and Maps.

		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
BCS358D	CO1	2				3							
	CO2	2	3	2		3							
	CO3	2				3							
	CO4	2				3							
	CO5	2				3							

		PSO1	PSO2
BCS358D	CO1	1	2
	CO2	2	2
	CO3	2	2
	CO4	2	2
	CO5	1	2

DATA VISUALIZATION WITH PYTHON LABORATORY

Subject Code: BCS358D **CIE Marks: 50**
Hours/Week: 0:0:2:0 **Exam Hours: 03**
Total Hours: 24

List of Programs

Sl. No.	Name of Experiment (Part-A)
1.	a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user. b) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.
2.	a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed. b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.
3.	a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters. 03092022 b) Write a Python program to find the string similarity between two given strings Sample Output: Sample Output: Original string: Original string: Python Exercises Python Exercises Python Exercises Python Exercises Similarity between two said strings: Similarity between two said strings: 1.0 0.967741935483871
4.	a) Write a Python program to Demonstrate how to Draw a Bar Plot using Matplotlib.. b) Write a Python program to Demonstrate how to Draw a Scatter Plot using Matplotlib.
5.	a) Write a Python program to Demonstrate how to Draw a Histogram Plot using Matplotlib. b) Write a Python program to Demonstrate how to Draw a Pie Chart using Matplotlib.
6.	a) Write a Python program to illustrate Linear Plotting using Matplotlib. b) Write a Python program to illustrate liner plotting with line formatting using Matplotlib.
7.	a) Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions.
8.	Write a Python program to explain working with bokeh line graph using Annotations and Legends. a) Write a Python program for plotting different types of plots using Bokeh.
9.	a) Write a Python program to draw 3D Plots using Plotly Libraries.
10.	a) Write a Python program to draw Time Series using Plotly Libraries. b) Write a Python program for creating Maps using Plotly Libraries.

DATA VISUALIZATION WITH PYTHON LABORATORY

Subject Code: BCS358D **CIE Marks: 50**
Hours/Week: 0:0:2:0 **Total Hours: 24**

Lesson Planning / Schedule of Experiments

Sl. No	Name of Experiment	WEEK
1	Sample Programs	Week1
2	a. To find the best of two test average marks out of three test'smarks. b. To check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.	Week2
3	a. Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. b. Program to convert binary to decimal, octal to hexadecimal using functions	Week3
	a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters. 03092022 b) Write a Python program to find the string similarity between two given strings Sample Output: Sample Output: Original string: Original string: Python Exercises Python Exercises Python Exercises Python Exercises Similarity between two said strings: Similarity between two said strings: 1.0 0.967741935483871	Week4
4	a. Demonstrate how to Draw a Bar Plot using Matplotlib b. Demonstrate how to Draw a Pie Chart using Matplotlib	Week5
5	a. Demonstrate how to Draw a Histogram Plot using Matplotlib. b. Demonstrate how to Draw a Pie Chart using Matplotlib.	Week6
6	**LAB TEST - 1**	Week7
7	a. Illustrate Linear Plotting using Matplotlib. b. Illustrate liner plotting with line formatting using Matplotlib.	Week8
8	a. Program which explains uses of customizing seaborn plots with Aesthetic functions.	Week9
9	Program to explain working with bokeh line graph using Annotations and Legends. a. Program for plotting different types of plots using Bokeh.	Week10
10	Program to draw 3D Plots using Plotly Libraries.	Week11
11	a. Python program to draw Time Series using Plotly Libraries. b. Program for creating Maps using Plotly Libraries.	Week12
12	**LAB TEST - 2**	Week13

Conduction of Practical Examination:

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from Part A with lot.
3. Marks distribution: **Procedure (15%) + Execution (70%) + Viva (15%)**

Part A: (15+70+15=100)

Final Exam Marks:_____/50 Marks (100 scaled down to 50)

4. Change of experiment is allowed only once and marks allotted to the write up part to be made Zero.

Instructions to the Candidates

1. Students should come with thorough preparation for the experiment to be conducted.
2. Students will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiment conducted in the previous class.
3. Practical record should be neatly maintained.
4. They should obtain the signatures of the staff in-charge in the observation book after completing each experiment.
5. Theory regarding each experiment should be written in the practical record before procedure in your own words.
6. Ask lab technician for assistance if you have any problem.
7. Save your class work, assignments in system.
8. Do not download or install software without the assistance of the laboratory technician.
9. Do not alter the configuration of the system.
10. Turn off the systems after use.

PROGRAM-1

1a) Write a Python program to find the best of two test average marks out of three test marksaccepted by the user.

```
def get_valid_test_score():
    while True:
        try:
            score = float(input("Enter a test score: "))
            if 0 <= score <= 100:
                return score
            else:
                print("Please enter a valid test score between 0 and 100.")
        except ValueError:
            print("Invalid input. Please enter a valid number.")

# Input three test scores for a single student
test_scores = []

for i in range(3):
    test_scores.append(get_valid_test_score())

# Sort the test scores in descending order
test_scores.sort(reverse=True)

# Calculate the average of the best two test scores
best_two_averages = sum(test_scores[:2]) / 2

# Output the result
print("The average of the best two test scores is: {best_two_averages:.2f}")
```

OUTPUT:

1. Enter a test score: 98
   Enter a test score: 65
   Enter a test score: 85
   The average of the best two test scores is: 91.50

2. Enter a test score: 52
   Enter a test score: 52
   Enter a test score: 48
   The average of the best two test scores is: 52.00

3. Enter a test score: -2
   Please enter a valid test score between 0 and 100.

1b) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.

```
def is_palindrome(number):
    number_str = str(number)
    return number_str == number_str[::-1]
def count_digits(number):
    digit_count = [0] * 10              # Initialize a list to count each digit from 0 to 9
    while number >0:
        digit = number % 10             # Get the last digit
        digit_count[digit] += 1         # Increment the count for that digit
        number //= 1                    # Remove the last digit
    return digit_count
try:
    num = int(input("Enter a number: "))
    if num <0:
        print("Please enter a non-negative number.")
    else:
        if is_palindrome(num):
            print(f"{num} is a palindrome.")
        else:
            print(f"{num} is not a palindrome.")
        digit_count = count_digits(num)
        for digit, count in enumerate(digit_count):
            if count >0:
                print(f"Digit {digit} appears {count} times in the number.")
except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

OUTPUT:

1. Enter a number: 818
   818 is a palindrome.
   Digit 1 appears 1 times in the number.
   Digit 8 appears 2 times in the number.

2. Enter a number: 123
   123 is not a palindrome.
   Digit 1 appears 1 times in the number.
   Digit 2 appears 1 times in the number.
   Digit 3 appears 1 times in the number.

3. Enter a number: -125
   Please enter a non-negative number.

Palindrome Program for strings

```
def is_palindrome(s):
    s = s.lower()     # Convert to lowercase to make it case-insensitive
    s = list(s)        #Convert the string to a list
    length = len(s)
    for i in range(length // 2):
        if s[i].isalnum() and s[length - i - 1].isalnum():
            if s[i].lower() != s[length - i - 1].lower():
                return False
        elif not s[i].isalnum():
            while nots[i].isalnum():
                i += 1
        elif not s[length - i - 1].isalnum():
            while nots[length-i-1].isalnum():
                length -= 1
    return True

# Input string from the user
string = input("Enter a string: ")
if is_palindrome(string):
    print(f"'{string}' is a palindrome.")
else:
        print(f"'{string}' is not a palindrome.")
```

OUTPUT:

1. Enter a string: rar
   'rar' is a palindrome.

2. Enter a string: ram
   'ram' is not a palindrome.

PROGRAM-2

2a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program that accepts a value for N (where N >0) as input and pass this value to the function. Display a suitable error messageif the condition for input value is not followed.

```
def fn(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fn(n-1) + fn(n-2)
num = int(input("Enter a number : "))
if num > 0:
    print("fn(", num, ") = ",fn(num) , sep ="")
else:
        print("Error in input")
```

OUTPUT:

1. Enter a number : 5
   fn(5) = 3

2. Enter a number : -4
   Error in input

2b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.

```
def BinToDec(x):
    dec = 0
    i = 0
    while x>0:
        r = x%10
        if r!=0 and r!=1:
            print("Enter a valid Binary number")
            return 0
        else:
            dec = dec + r*2**i
            x = x // 10
            i += 1
    return dec
def OctaToHexa(n):
    num = n
    dec = 0
    base = 1
    temp = num
    while temp:
        r = temp % 10
        temp = temp // 10
        dec += r * base
        base = base * 8
    result = ' '
    while dec != 0:
        temp = 0
        temp = dec % 16
        if temp < 10:
            result = str(temp) + result
        else:
            result = chr(temp + 55) + result
        dec = dec // 16
    return result

x = int(input("Enter a Binary number "))
result = BinToDec(x)
if result:
    print("The Decimal equivalent of {0} is {1}".format(x, result))
```

```
y = int(input("Enter a Octal number "))
result = OctaToHexa(y)
print(result)
if result:
   print("The Hexa Decimal equivalent of {0} is {1}".format(y, result))
```

OUTPUT:

```
Enter a Binary number 1111
The Decimal equivalent of 1111 is 15
Enter a Octal number 146523
CD53
The Hexa Decimal equivalent of 146523 is CD53
```

PROGRAM-3

3a) Write a Python program that accepts a sentence and find the number of words, digits,uppercase letters, and lowercase letters.

```
x = input("Enter a sentence")
y = x
print("There are",len(x.split())," words in the sentence")
digits,upper,lower=0,0,0
for i in x:
    if i.isdigit():
        digits+=1
    elif i.isupper():
        upper+=1
    elif i.islower():
        lower+=1
print("There are {0} digits, {1} upper case characters and {2} lower case characters in
the sentence".format(digits,upper,lower))
```

OUTPUT:

```
Enter a sentence Rama went to Devaraja market to pick 2 kgs of vegetable
There are 11 words in the sentence
There are 1 digits, 2 upper case characters and 42 lower case characters in the sentence
```

3b) Write a Python program to find the string similarity between two given strings.

```
x = input("Enter first String")
y = input("Enter second String")
x = x.strip()
y = y.strip()
sim=0
if len(x)>len(y):
    xx = x
    yy = y
else:
    xx = y
    yy = x
j=0
for i in yy:
    if i==xx[j]:
        sim+=1
    else:
        pass
    j+=1
similarity = (sim/len(xx))
print("The similarity between the two given strings is", similarity)
```

OUTPUT:

1. Enter first String Python Exercises
   Enter second String Python Exercises
   The similarity between the two given strings is 1.0

2. Enter first String Python Lab
   Enter second String Python Laboratory
   The similarity between the two given strings is 0.5882352941176471

MATPLOTLIB

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in a variety of formats. It is widely used for producing high-quality plots and charts in scientific computing, data analysis, and machine learning. Matplotlib provides a range of functions for creating different types of plots, including line plots, scatter plots, bar plots, histograms, and more. Different visualizations plots are as follows:

Scatter plots: Scatter plots are particularly useful when exploring the relationship between two continuous variables. They excel at revealing patterns, trends, and correlations between data points. These visualizations are adept at identifying outliers, showcasing them as points deviating from the main cluster. By providing a clear picture of the distribution of data points along two axes, scatter plots aid in understanding the spread and density of values. Moreover, they are valuable for comparing different datasets, recognizing similarities or differences.

Histogram: A histogram is a graphical representation of the distribution of a dataset, typically used for continuous or discrete data. It provides a way to visualize the frequency or count of data points within specific intervals or bins. In a histogram, the data is divided into contiguous, non-overlapping intervals, and the height of each bar in the chart represents the frequency or count of data points falling within that interval.

To create a histogram, you divide the range of the data into bins or intervals and then count the number of data points that fall into each bin. The resulting bar chart, with the bars representing these counts, provides a visual summary of the data's distribution.

Bar chart: A bar chart is a graphical representation of data in which rectangular bars are used to represent the values of different categories. Each bar's length is proportional to the value it represents. Bar charts are effective for comparing discrete categories or groups and are particularly useful for showing the distribution of categorical data.

Pie chart: Pie charts are a type of data visualization that is commonly used to represent the proportions of different parts of a whole. The primary purpose of a pie chart is to show the relationship of parts to a whole and to illustrate how each part contributes to the total.

SEABORN

Seaborn is a statistical data visualization library built on top of Matplotlib in Python. It provides an interface for creating informative and attractive statistical graphics. Seaborn comes with several built-in themes and color palettes to make it easy to create aesthetically pleasing visualizations. It is particularly useful for exploring complex datasets and understanding relationships between variables.

BOKEH

Bokeh is a Python interactive visualization library that targets modern web browsers for presentation. It allows you to create interactive, web-ready visualizations in Python. Bokeh generates HTML and JavaScript code that can be embedded into web pages. This allows you to create interactive visualizations that can be easily shared on the web.

PLOTLY

Plotly is a versatile Python library for creating interactive and publication-quality plots and dashboards. It supports a wide range of chart types. Plotly excels at creating interactive plots. Users can zoom, pan, hover over data points for additional information, and perform other interactive actions directly within the plot. Its ability to create web-based dashboards makes it a powerful tool for building data-driven applications.

PROGRAM-4

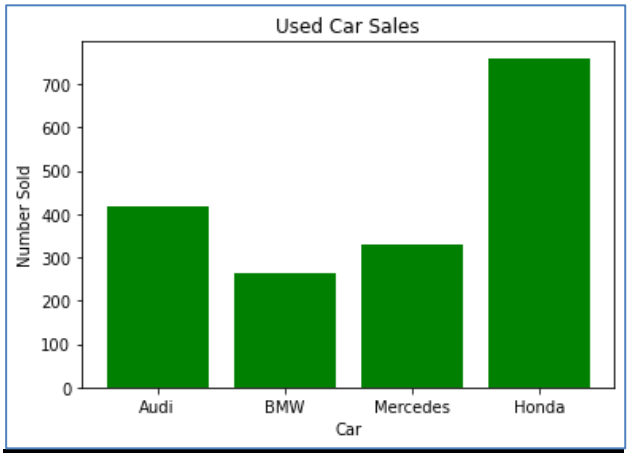
4a) Write a Python program to demonstrate how to draw a Bar Plot using Matplotlib.

Dataset (Cars_Barplot.csv)

Car	Sales
Audi	419
BMW	263
Mercedes	330
Honda	760

```
import matplotlib.pyplot as plt
import pandas as pd
# Initialize the lists for X and Y
data = pd.read_csv("Cars_Barplot.csv")
df = pd.DataFrame(data)
X = list(df.iloc[:, 0])
Y = list(df.iloc[:, 1])
# Plot the data using bar() method
plt.bar(X, Y, color='g')
plt.title("Used Car Sales")
plt.xlabel("Car")
plt.ylabel("Number Sold")
# Show the plot
plt.show()
```

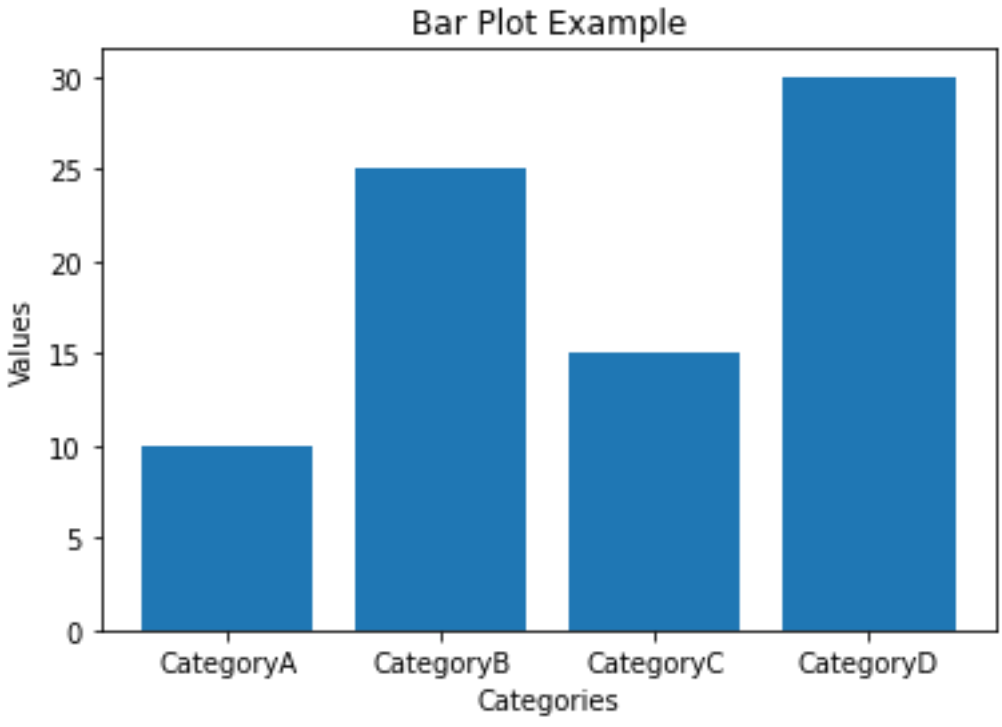
OUTPUT



OR

```
import matplotlib.pyplot as plt
# Data for the bar plot
categories=['CategoryA','CategoryB','CategoryC','CategoryD']
values = [10, 25, 15, 30]
# Create a bar plot
plt.bar(categories, values)
#Addlabelsandatitle
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot Example')
# Display the plot
 plt.show()
```

OUTPUT:



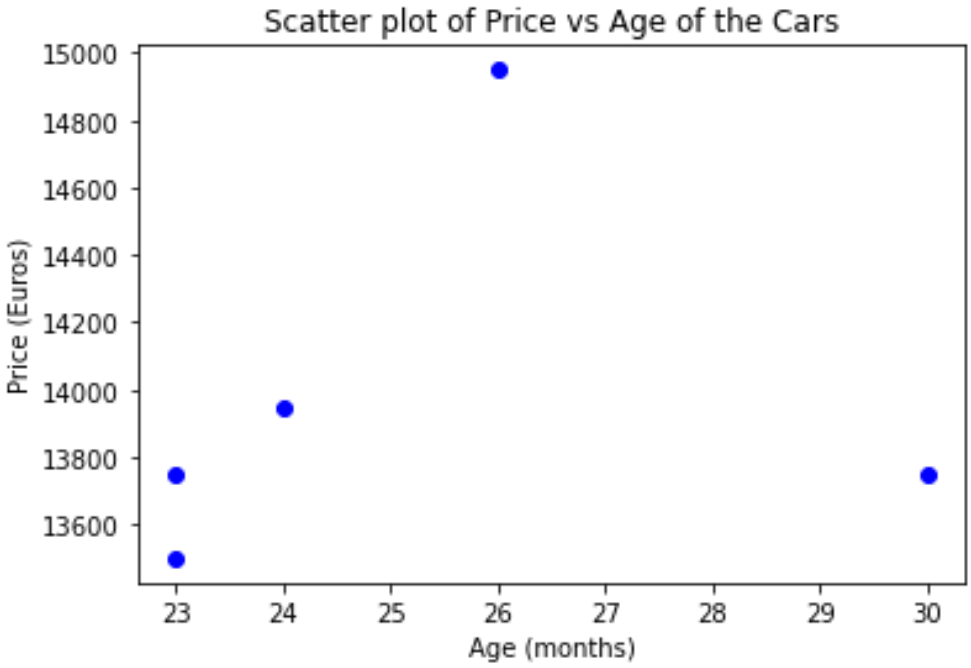
4b) Write a Python program to demonstrate how to draw a Scatter Plot using Matplotlib.

Dataset (Cars.csv)

Id	Model	Price	Age	Mfg._Month	Mfg._ Year	KM	Fuel_ Type	HP	Met_ Color	Auto matic	cc	Doors
1	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13500	23	10	2002	46986	Diesel	90	1	0	2000	3
2	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	13750	23	10	2002	72937	Diesel	90	1	0	2000	3
3	?TOYOTACorolla 2.0 D4D HATCHB TERRA 2/3-Doors	13950	24	9	2002	41711	Diesel	90	1	0	2000	3
4	TOYOTA Corolla 2.0 D4D HATCHB TERRA 2/3-Doors	14950	26	7	2002	48000	Diesel	90	0	0	2000	3
5	TOYOTA Corolla 2.0 D4D HATCHB SOL 2/3-Doors	13750	30	3	2002	38500	Diesel	90	0	0	2000	3

```
# import the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Importing data.
cars_data = pd.read_csv("Cars.csv")
# Create scatter plot using two variables, Age and Price.
plt.scatter(cars_data['Age'],cars_data['Price'],c='blue')
# To set the title
plt.title('Scatter plot of Price vs Age of the Cars')
# To set the x and y axis labels.
plt.xlabel('Age (months)')
plt.ylabel('Price (Euros)')
# To show the scatter plot
plt.show()
```

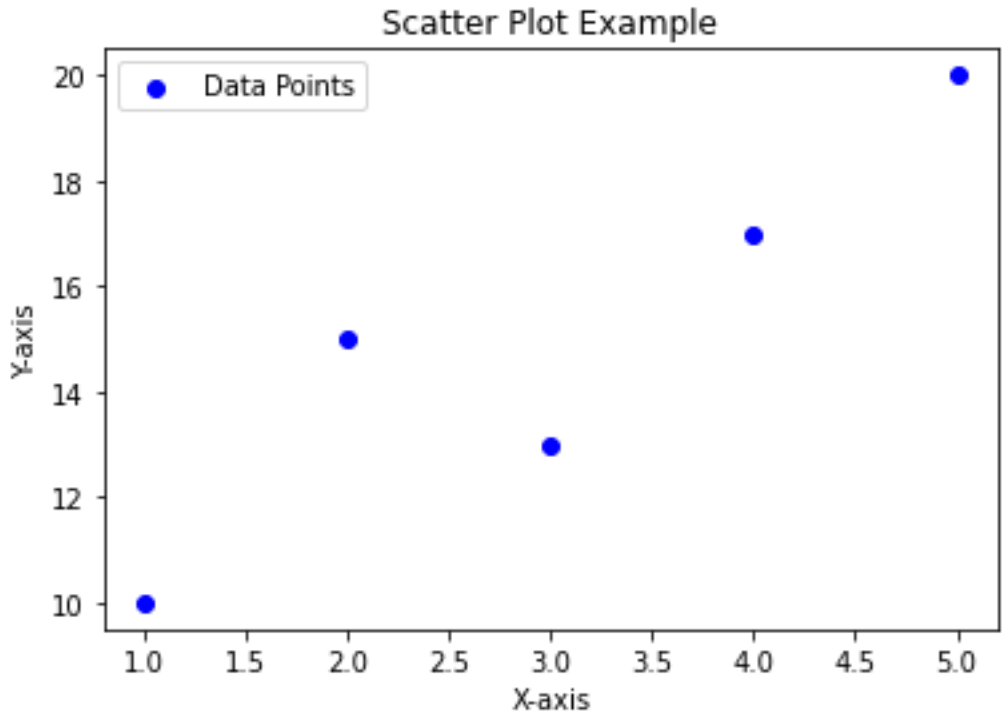
OUTPUT:



OR

```
import matplotlib.pyplot as plt
# Data for the scatter plot
x = [1, 2, 3, 4, 5]
y = [10, 15, 13, 17, 20]
# Create a scatter plot
plt.scatter(x, y, marker='o', color='blue', label='Data Points')
# Add labels and a title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot Example')
# Add a legend
plt.legend()
# Display the plot
plt.show()
```

OUTPUT:

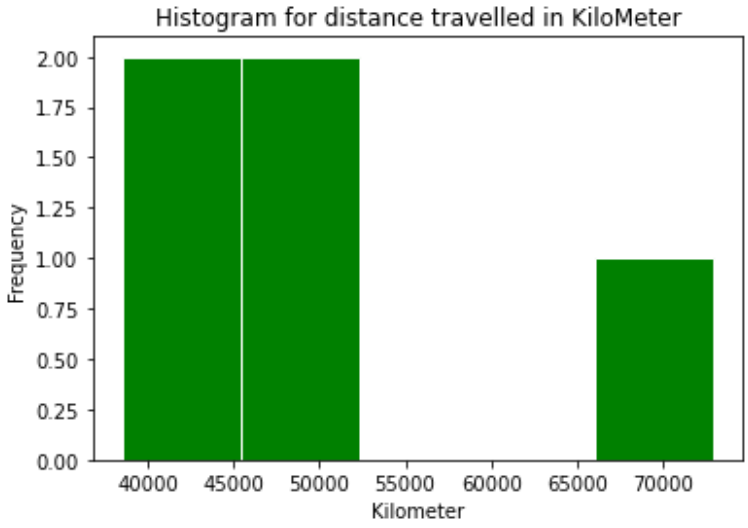


PROGRAM-5

5a) Write a Python program to demonstrate how to draw a Histogram using Matplotlib

```
# import the necessary libraries

# Pandas library for data frames

import pandas as pd

# numpy library to do numerical operations

import numpy as np

import matplotlib.pyplot as plt

cars_data = pd.read_csv("cars.csv")

plt.title('Histogram for distance travelled in KiloMeter')

plt.hist(cars_data ['KM'], color='green', edgecolor='white', bins=5)

plt.xlabel('Kilometer')

plt.ylabel('Frequency')

plt.show()
```

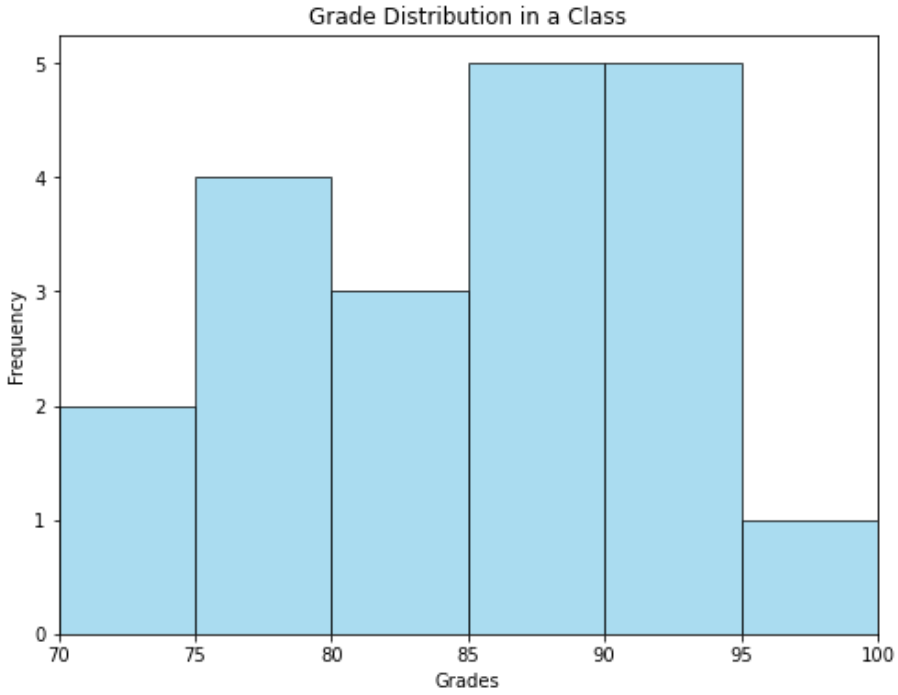
OUTPUT:



OR

```
import numpy as np
# Sample data: Student grades
grades = [75, 82, 92, 88, 78, 90, 85, 95, 70, 87, 93, 79, 81, 86, 89, 94, 73, 84, 91, 77]
# Create a histogram
plt.figure(figsize=(8, 6))
plt.hist(grades, bins=[70, 75, 80, 85, 90, 95, 100], edgecolor='black', alpha=0.7, color='skyblue')
# Customize labels and title
plt.xlabel('Grades')
plt.ylabel('Frequency')
plt.title('Grade Distribution in a Class')
# Set the x-axis limits
plt.xlim(70,100)
# Display the plot
plt.show()
```

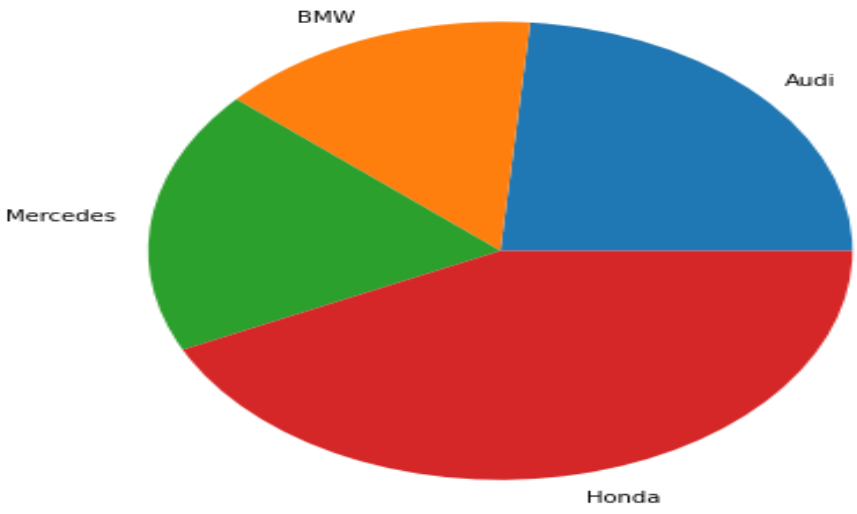
OUTPUT:



5b) Write a Python program to demonstrate how to draw a Pie chart using Matplotlib

```
# Import libraries

import matplotlib.pyplot as plt

import pandas as pd

# Creating dataset

cars_data = pd.read_csv("Cars_Barplot.csv")

cars = cars_data["Car"]

data = cars_data["Sales"]

# Creating plot

fig = plt.figure(figsize =(10, 7))

plt.pie(data, labels = cars)

# show plot

plt.show()
```

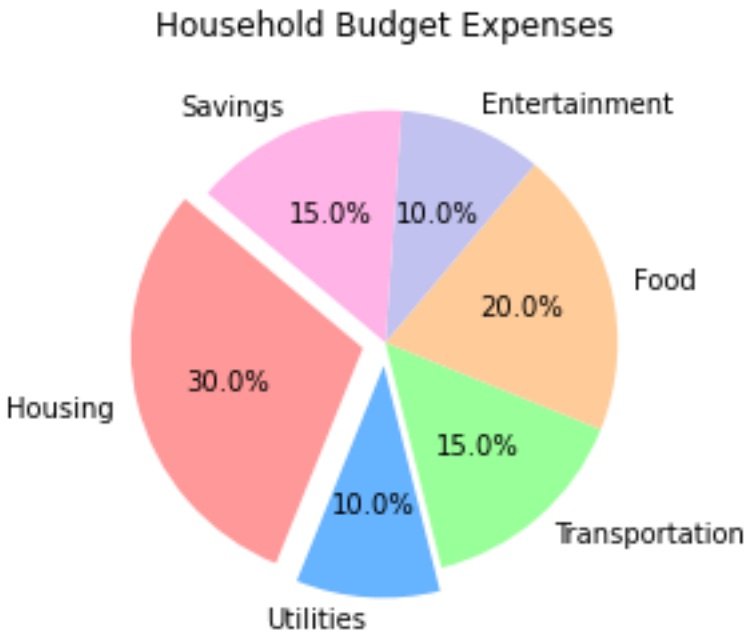
OUTPUT:



OR

```
import matplotlib.pyplot as plt
# Data for the household budget pie chart
categories=['Housing','Utilities','Transportation','Food','Entertainment','Savings']
expenses = [30, 10, 15, 20, 10, 15]
colors=['#ff9999','#66b3ff','#99ff99','#ffcc99','#c2c2f0','#ffb3e6']
explode = (0.1, 0.1, 0, 0, 0, 0)
# Create a pie chart
plt.pie(expenses, labels=categories, autopct='%1.1f%%', startangle=140, colors=colors, explode=explode)
# Add a title
plt.title('Household Budget Expenses')
# Display the plot
plt.show()
```

OUTPUT:

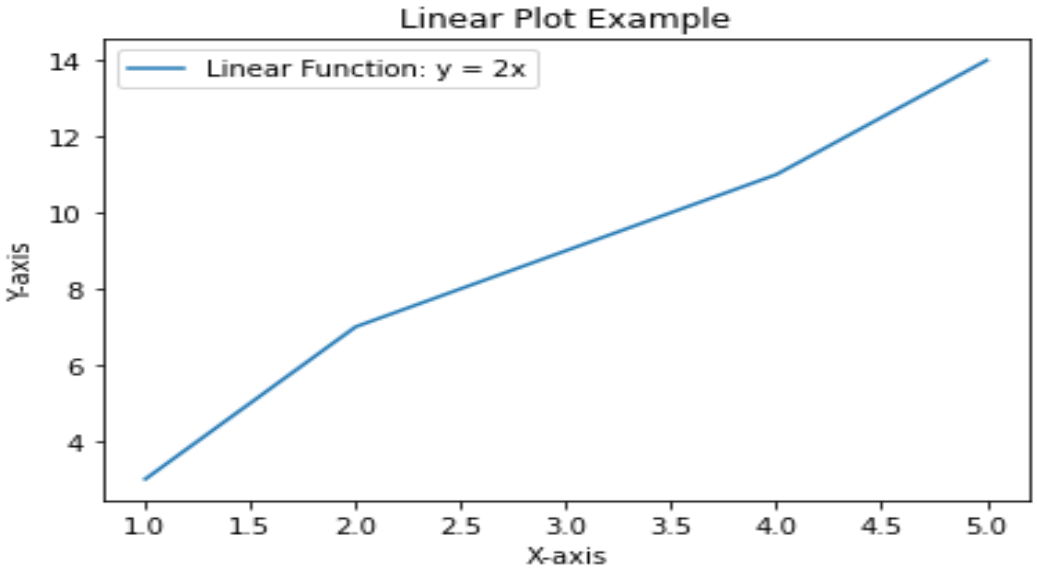


PROGRAM-6

6a) Write a Python program to illustrate Linear Plotting using Matplotlib

```
import matplotlib.pyplot as plt
def linear_plot():
# Sample data
    x = [1, 2, 3, 4, 5]
    y = [3, 7, 9, 11, 14]
    # Plotting the data
    plt.plot(x, y, label='Linear Function: y = 2x')
    # Adding labels and title
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title('Linear Plot Example')
    plt.legend()
    plt.show()
    # Call the function to generate the plot
linear_plot()
```

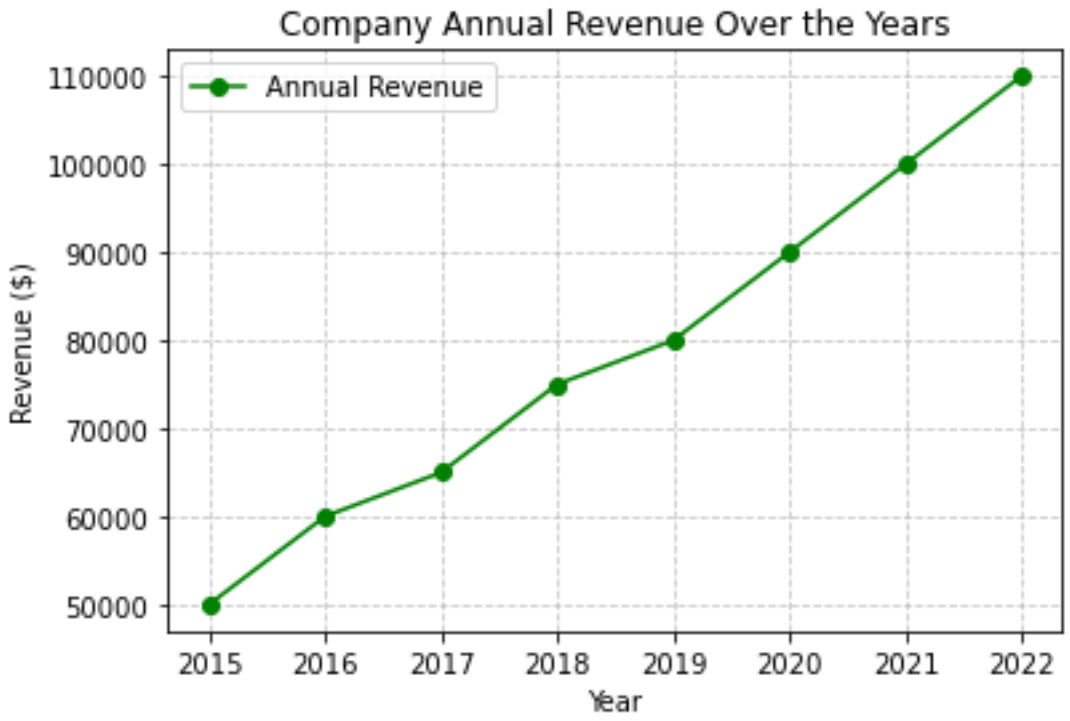
OUTPUT:



OR

```
import matplotlib.pyplot as plt
# Data for the linear plot
years = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
revenue = [50000, 60000, 65000, 75000, 80000, 90000, 100000, 110000]
# Create a linear plot
plt.plot(years,revenue, marker='o', color='green', linestyle='-', label='Annual Revenue')
# Add labels and a title
plt.xlabel('Year')
plt.ylabel('Revenue ($)')
plt.title('Company Annual Revenue Over the Years')
# Add a grid
plt.grid(True, linestyle='--', alpha=0.7)
# Add a legend
plt.legend()
# Display the plot
plt.show()
```

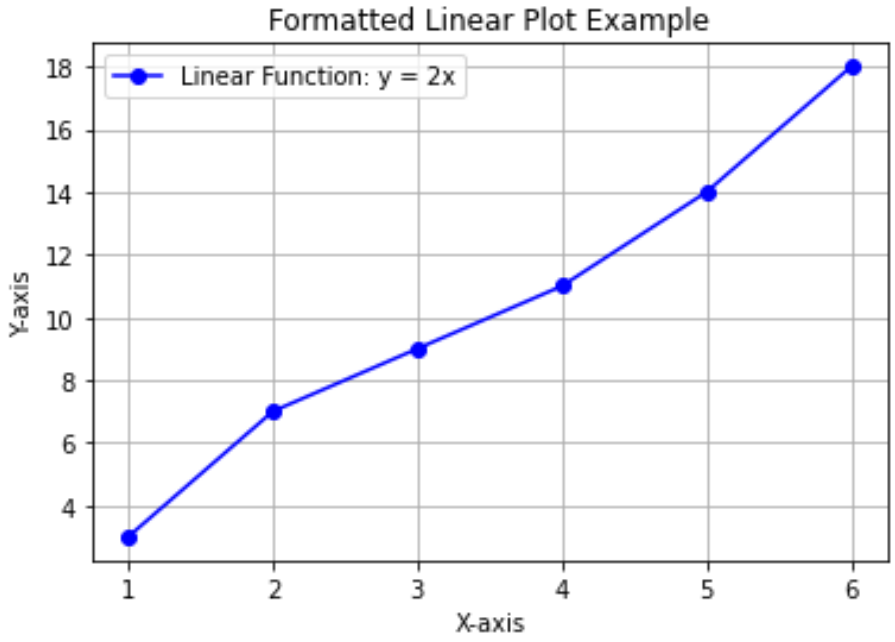
OUTPUT:



6b) Write a Python program to illustrate liner plotting with line formatting using Matplotlib

```
import matplotlib.pyplot as plt
def formatted_linear_plot():
# Sample data
    x = [1, 2, 3, 4, 5, 6]
    y = [3, 7, 9, 11, 14, 18]
    plt.plot(x, y, marker='o', linestyle='-', color='b', label='Linear Function: y = 2x')
    # Adding labels and title
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title('Formatted Linear Plot Example')
    plt.legend()
    plt.grid(True) # Add a grid for better readability
    plt.show()
# Call the function to generate the formatted linear plot
formatted_linear_plot()
```

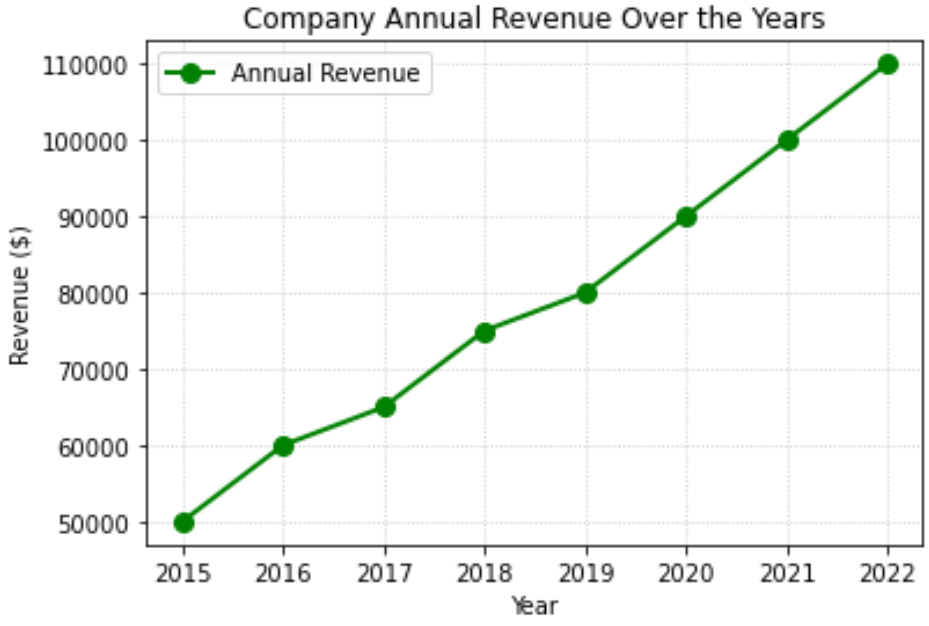
OUTPUT:



OR

```
import matplotlib.pyplot as plt
# Data for the linear plot
years = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
revenue = [50000, 60000, 65000, 75000, 80000, 90000, 100000, 110000]
# Create a linear plot with line formatting
plt.plot(years,revenue, marker='o', linestyle='-', color='green', label='Annual Revenue', linewidth=2, markersize=8)
# Add labels and a title
plt.xlabel('Year')
plt.ylabel('Revenue ($)')
plt.title('Company Annual Revenue Over the Years')
# Add a grid
plt.grid(True, linestyle=':', alpha=0.7)
# Add a legend
plt.legend()
# Display the plot
plt.show()
```

OUTPUT:

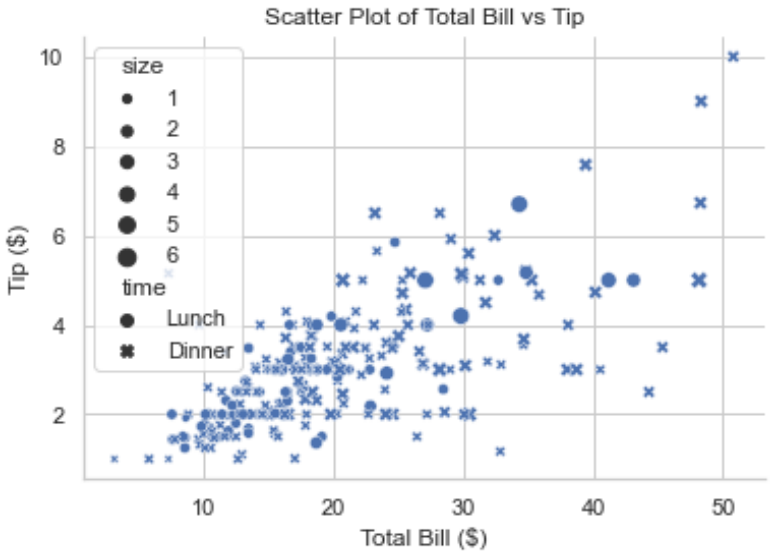


PROGRAM-7

7a) Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions.

```
import seaborn as sns
import matplotlib.pyplot as plt
# Load a sample dataset
tips = sns.load_dataset("tips")
# Set the aesthetic style of the plot
sns.set(style="whitegrid")
# Create a scatter plot using Seaborn
sns.scatterplot(x="total_bill", y="tip", style="time", size="size", data=tips)
# Customize the plot further using Seaborn aesthetic functions
sns.despine() # Remove the top and right spines from the plot
# Set custom labels and title
plt.xlabel("Total Bill ($)")
plt.ylabel("Tip ($)")
plt.title("Scatter Plot of Total Bill vs Tip")
# Show the plot
plt.show()
```

OUTPUT:

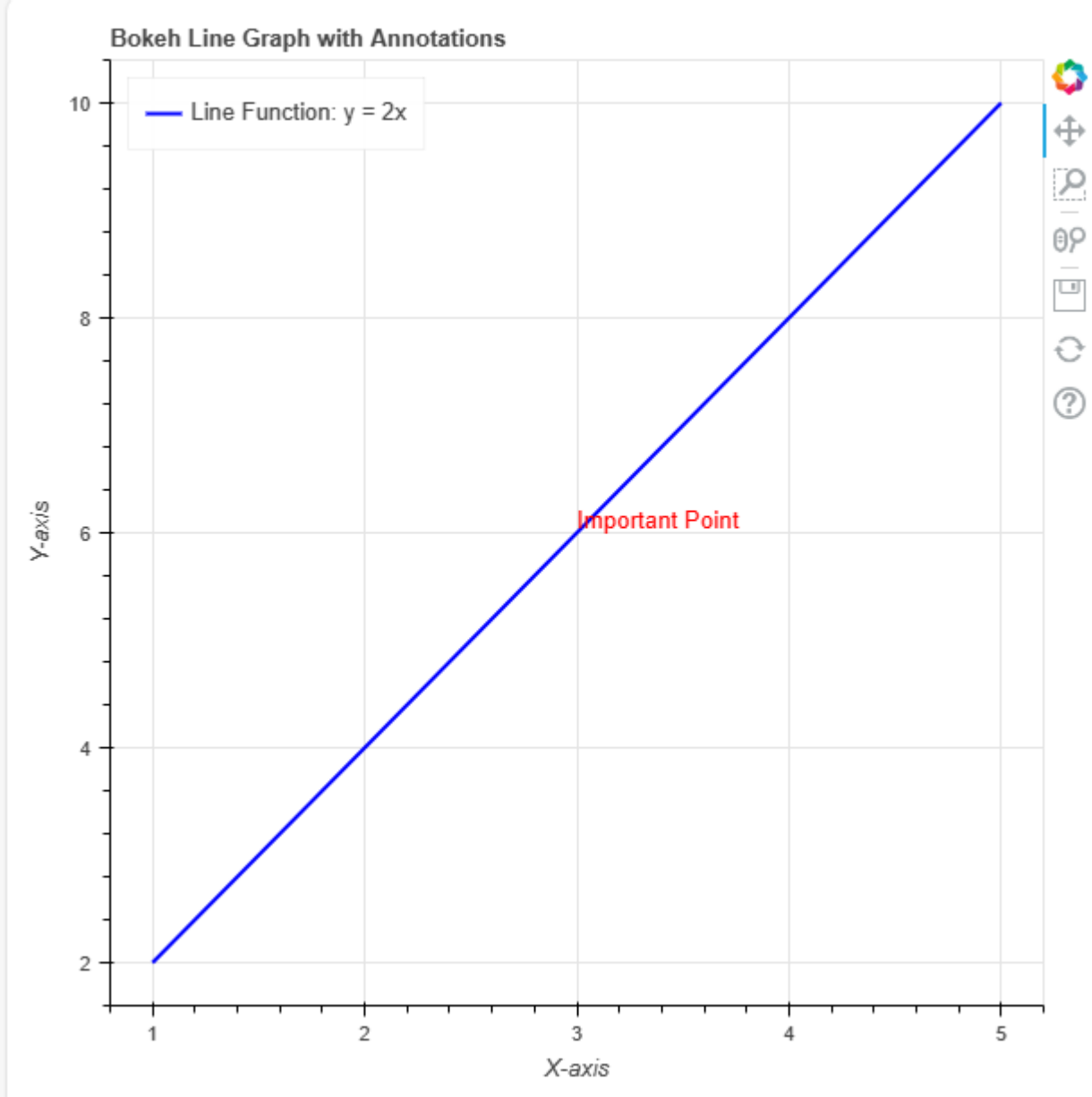


PROGRAM-8

8a) Write a Python program to explain working with bokeh line graph using Annotations and Legends.

```
from bokeh.plotting import figure, output_file, show
from bokeh.models import Label
# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# Output to static HTML file
output_file("line_graph_with_annotations.html")
# Create a figure
p = figure(title="Bokeh Line Graph with Annotations", x_axis_label='X-axis', y_axis_label='Y-axis')
# Plot the line
p.line(x, y, line_width=2, line_color="blue", legend_label="Line Function: y = 2x")
# Add an annotation
annotation = Label(x=3, y=6, text="Important Point", text_font_size="10pt", text_color="red")
p.add_layout(annotation)
# Add legend
p.legend.location = "top_left"
p.legend.click_policy = "hide"
# Show the plot
show(p)
```

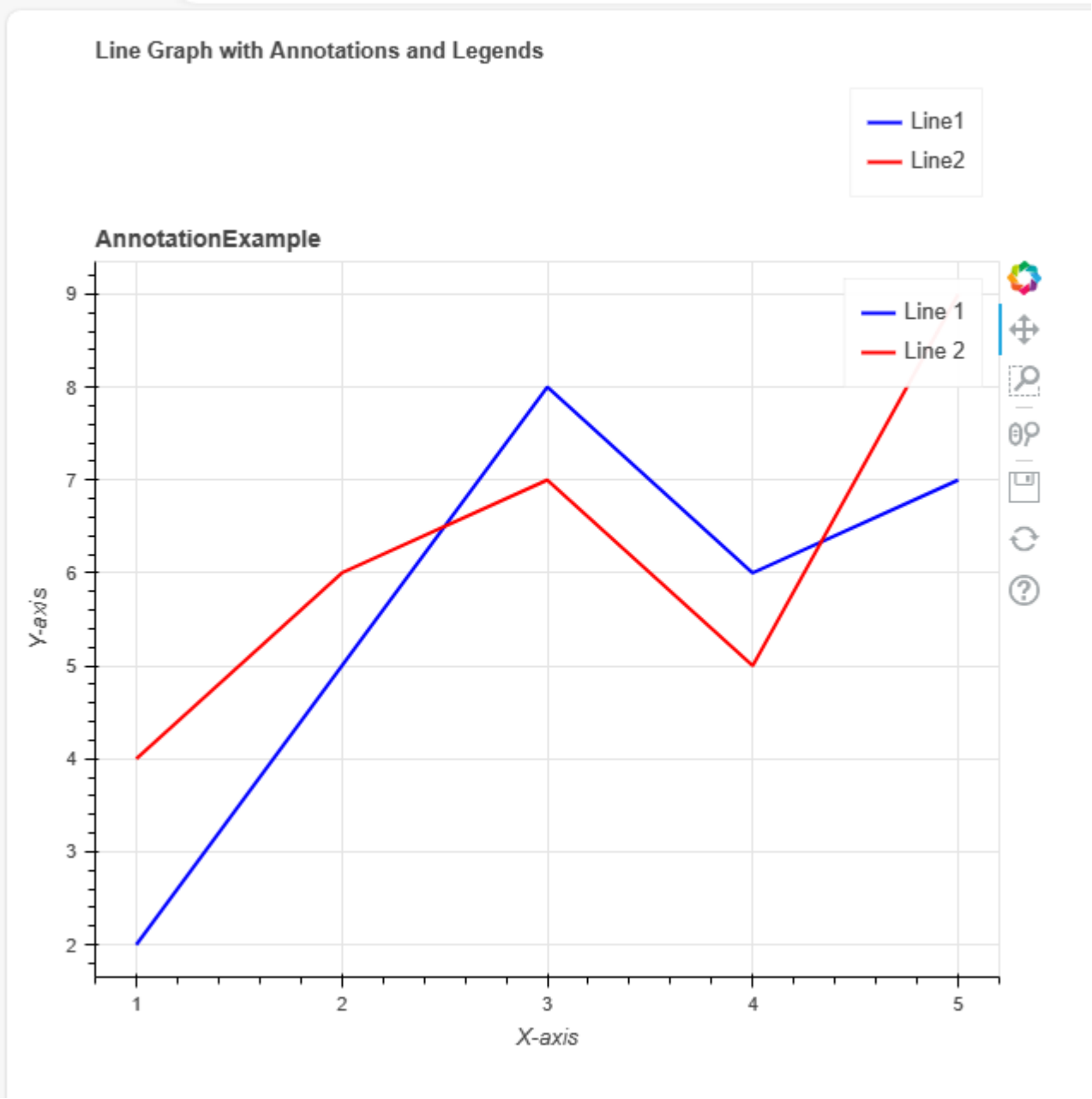
OUTPUT:


Bokeh Line Graph with Annotations
Line Function: y = 2x
Important Point
Y-axis
X-axis
10
8
6
4
2
1
2
3
4
5

OR

```
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource
from bokeh.models.annotations import Title, Legend, LegendItem
from bokeh.io import output_notebook
# Sample data
x = [1, 2, 3, 4, 5]
y1 = [2, 5, 8, 6, 7]
y2 = [4, 6, 7, 5, 9]
# Create a Bokeh figure
p = figure(title="Line Graph with Annotations and Legends", x_axis_label="X-axis", y_axis_label="Y-axis")
# Add data sources
source1=ColumnDataSource(data=dict(x=x,y=y1))
source2 = ColumnDataSource(data=dict(x=x, y=y2))
# Plot the first line
line1 = p.line('x', 'y', source=source1, line_width=2, line_color="blue", legend_label="Line 1")
# Plot the second line
line2 = p.line('x', 'y', source=source2, line_width=2, line_color="red", legend_label="Line 2")
# Add an annotation
annotation=Title(text="AnnotationExample",text_font_size="14px")
p.add_layout(annotation, 'above')
# Create a legend
legend=Legend(items=[LegendItem(label="Line1",renderers=[line1]),LegendItem(label="Line2",renderers=[line2])])
p.add_layout(legend, 'above')
# Show the plot
output_notebook()
show(p)
```

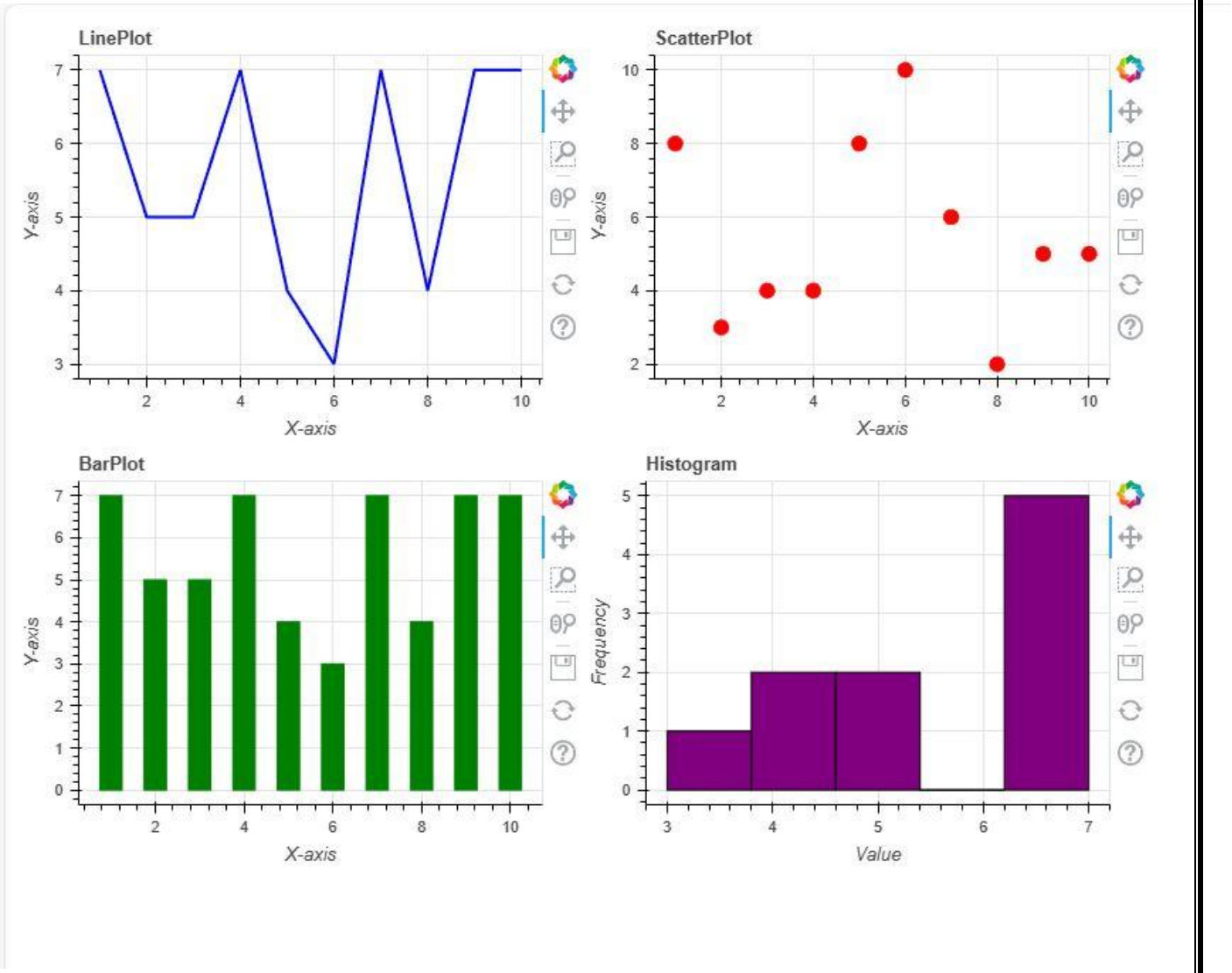
OUTPUT:


Line Graph with Annotations and Legends
Line1
Line2
AnnotationExample
Line 1
Line 2
Y-axis
X-axis
9
8
7
6
5
4
3
2
1
2
3
4
5

8b) Write a Python program for plotting different types of plots using Bokeh

```
from bokeh.plotting import figure, show, output_file
from bokeh.models import ColumnDataSource
from bokeh.layouts import layout
import random
import numpy as np

# Generate some sample data
x = list(range(1, 11))
y1 = [random.randint(1, 10) for inx in x]  # Corrected typo here
y2 = [random.randint(1, 10) for _ in x]

# Create a Bokeh figure with custom plot dimensions
p1 = figure(title="LinePlot", x_axis_label="X-axis", y_axis_label="Y-axis", plot_width=400, plot_height=300)
p1.line(x, y1, line_width=2, line_color="blue")

p2 = figure(title="ScatterPlot", x_axis_label="X-axis", y_axis_label="Y-axis", plot_width=400, plot_height=300)
p2.scatter(x, y2, size=10, color="red", marker="circle")

p3 = figure(title="BarPlot", x_axis_label="X-axis", y_axis_label="Y-axis", plot_width=400, plot_height=300)
p3.vbar(x=x, top=y1, width=0.5, color="green")

p4 = figure(title="Histogram", x_axis_label="Value", y_axis_label="Frequency", plot_width=400, plot_height=300)
hist, edges = np.histogram(y1, bins=5)
p4.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:], fill_color="purple", line_color="black")

# Create a layout with the plots
layout = layout([
    [p1, p2],
    [p3, p4]
])
# Output to an HTML file

output_file("bokeh_plots.html")
# Show the plots
show(layout)
```

OUTPUT:

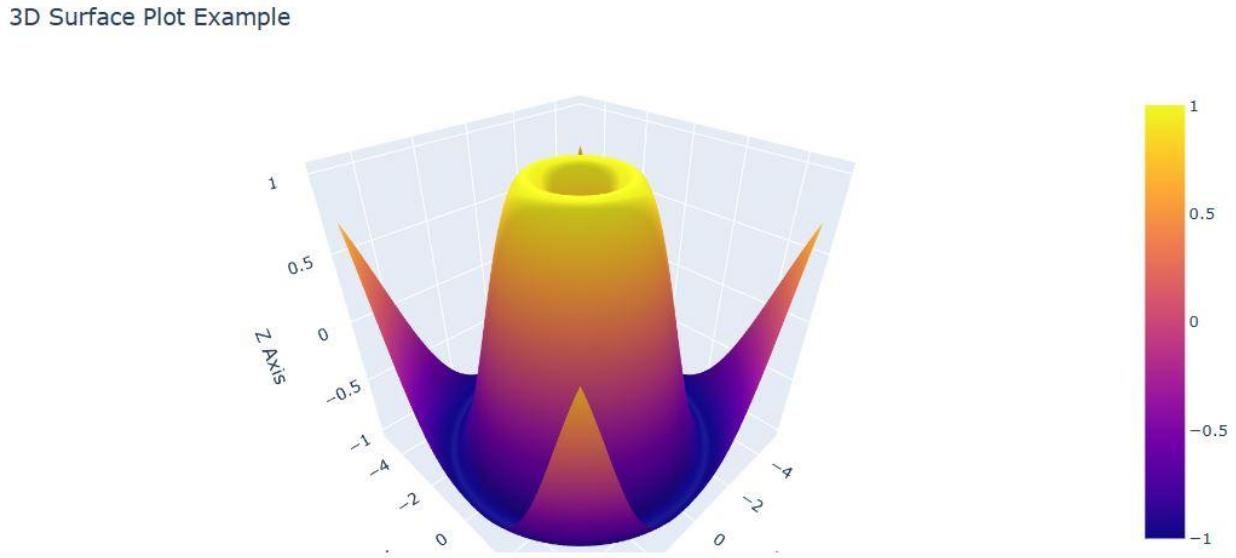

LinePlot
Y-axis
X-axis
ScatterPlot
Y-axis
X-axis
BarPlot
Y-axis
X-axis
Histogram
Frequency
Value

PROGRAM-9

9a) Write a Python program to draw 3D Plots using Plotly Libraries.

```
import plotly.graph_objects as go

import numpy as np

# Create data for the 3D surface plot

x = np.linspace(-5, 5, 100)

y = np.linspace(-5, 5, 100)

x, y = np.meshgrid(x, y)

z = np.sin(np.sqrt(x**2 + y**2))

fig = go.Figure(data=[go.Surface(z=z, x=x, y=y)])

fig.update_layout(scene=dict(xaxis_title='XAxis',  yaxis_title='YAxis',  zaxis_title='Z Axis',) title='3D Surface Plot Example',)

fig.show()
```

OUTPUT:



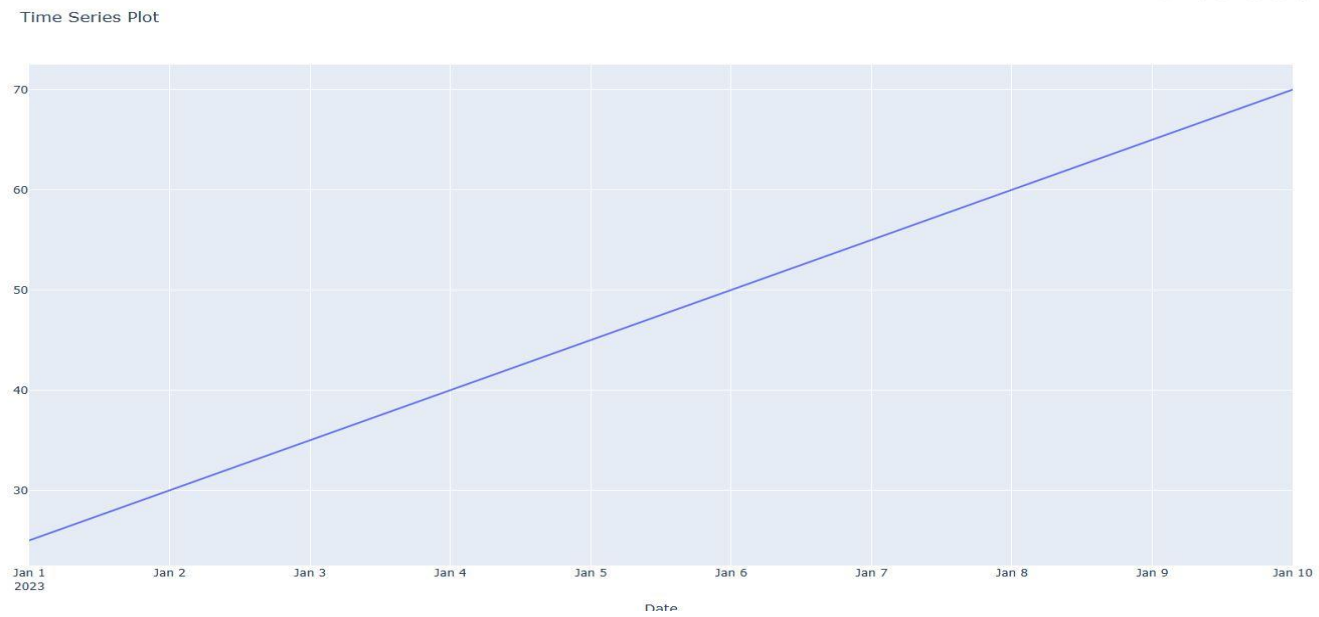
PROGRAM-10

10a). Write a Python program to draw Time Series using Plotly Libraries.

```
import plotly.graph_objs as go
import plotly.offline as pyo
import pandas as pd

# Sample time series data (you can load your own data)
data = {'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'), 'Value': [25, 30, 35, 40, 45, 50, 55, 60, 65, 70]}
df = pd.DataFrame(data)

# Create a time series plot
trace = go.Scatter(x=df['Date'], y=df['Value'], mode='lines', name='Time Series')

layout = go.Layout(title='Time Series Plot', xaxis=dict(title='Date'), yaxis=dict(title='Value'))

# Include a line break or proper indentation
fig = go.Figure(data=[trace], layout=layout)

# Show the plot in a Jupyter Notebook or export it as an HTML file
pyo.plot(fig, filename='time_series_plot.html')
```

OUTPUT:

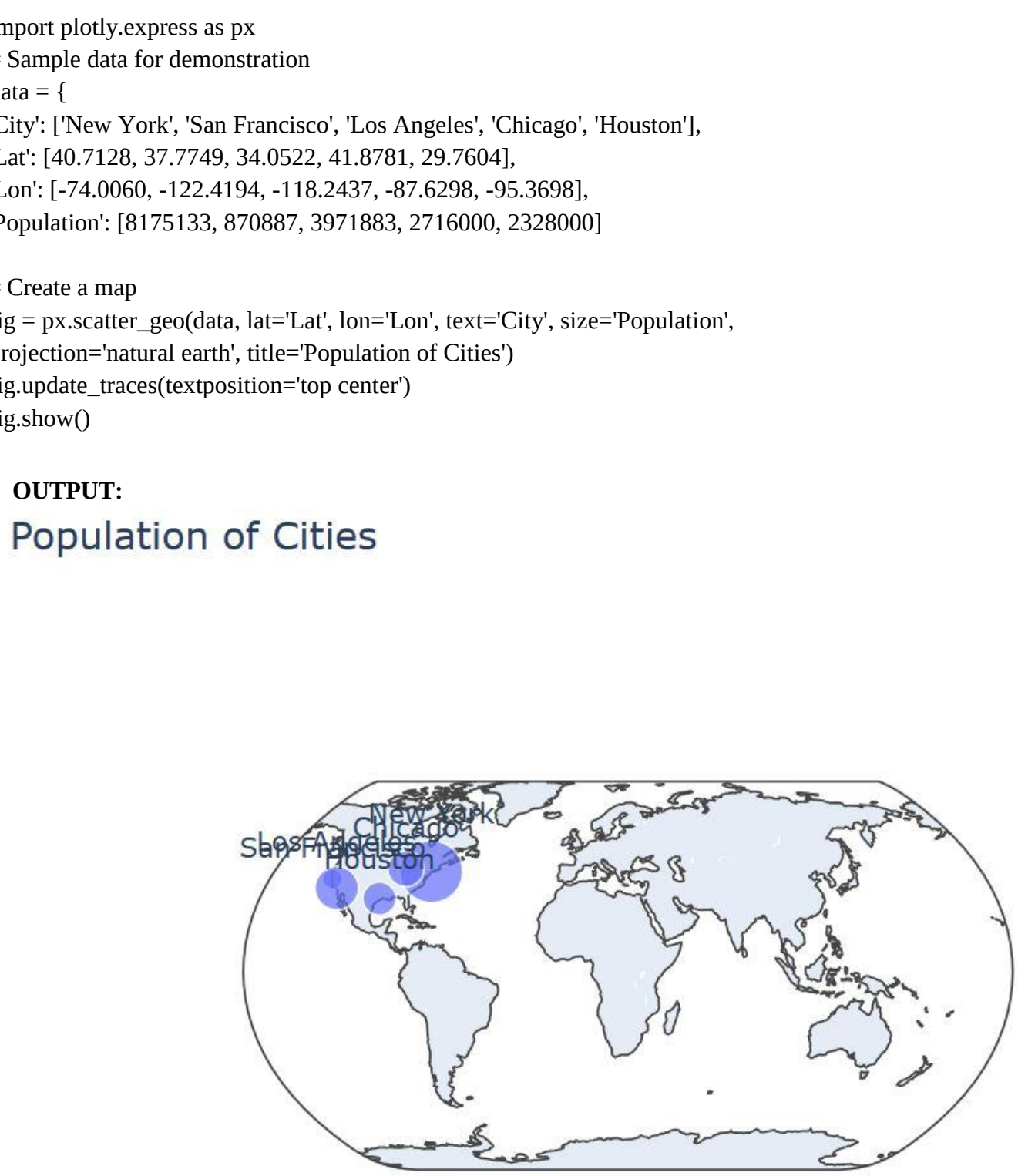
```
'time_series_plot.html'
```



10 b) Write a Python program for creating Maps using Plotly Libraries.

```
import plotly.express as px
# Sample data for demonstration
data = {
'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Houston'],
'Lat': [40.7128, 37.7749, 34.0522, 41.8781, 29.7604],
'Lon': [-74.0060, -122.4194, -118.2437, -87.6298, -95.3698],
'Population': [8175133, 870887, 3971883, 2716000, 2328000]
}
# Create a map
fig = px.scatter_geo(data, lat='Lat', lon='Lon', text='City', size='Population',
projection='natural earth', title='Population of Cities')
fig.update_traces(textposition='top center')
fig.show()
```

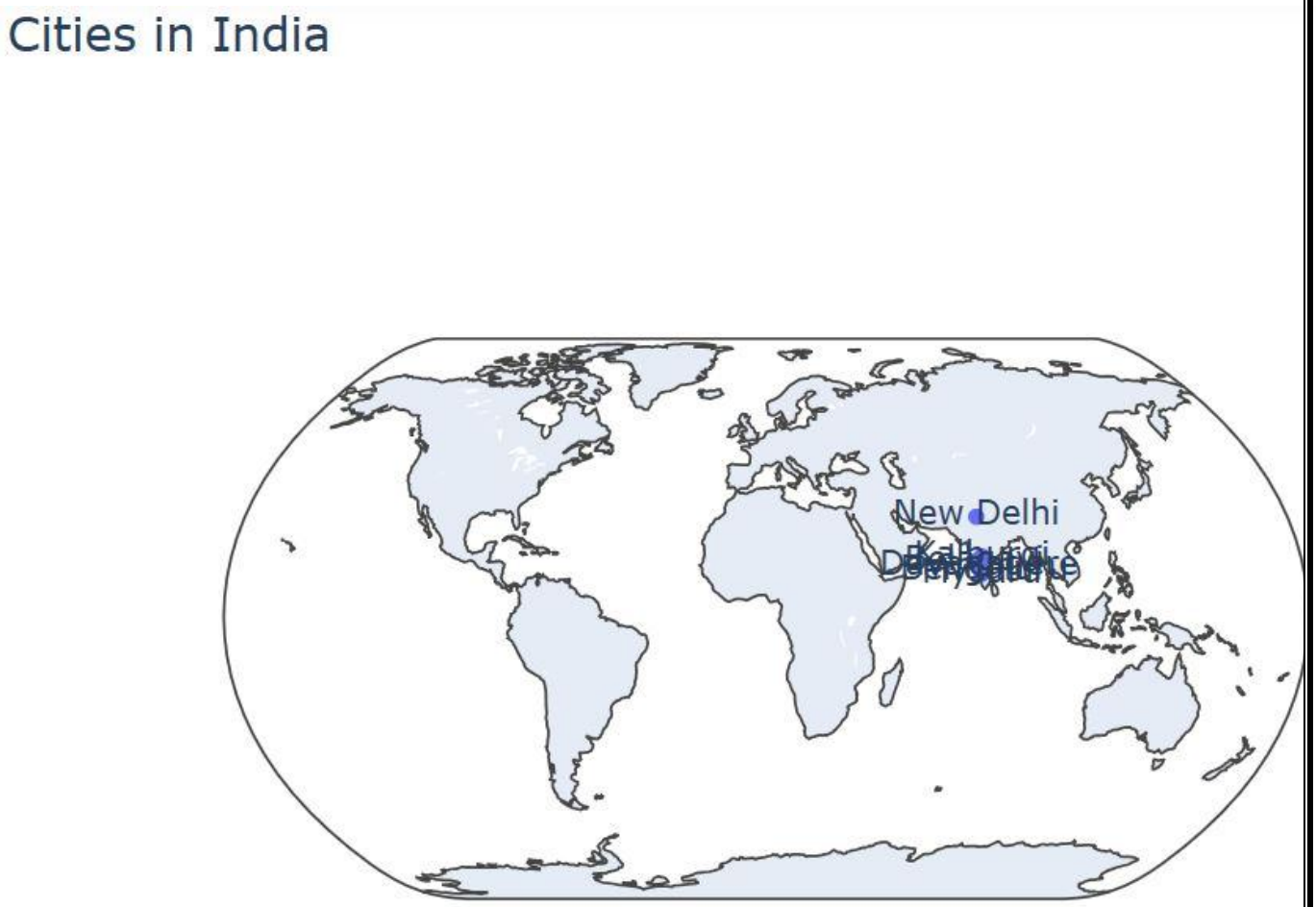
OUTPUT:



OR

```
import plotly.express as px
import pandas as pd
# Load your dataset (ensure it has 'lat' and 'lng' columns for latitude and longitude)
data = {
"City": ["Bengaluru", "Mysuru", "Belagavi", "Kalburgi", "Davanagere", "New Delhi"],
"Lat": [12.9716, 12.2958, 15.8497, 17.3291, 14.4644, 28.6139],
"Lon": [77.5946, 76.6394, 74.5048, 77.2471, 75.9224, 77.2090]
}
df = pd.DataFrame(data)
# Create a scatter map plot
fig = px.scatter_geo(df, lat="Lat", lon="Lon", text="City",
title="Cities in India", projection="natural earth")
# Show the map
fig.show()
```

OUTPUT:



VIVA QUESTIONS

What is Python?

1. Python is one of the most widely-used and popular programming languages, was developed by Guido van Rossum and released first on February 20, 1991.
2. Python is a free and open-source language with a very simple and clean syntax which makes it easyfor developers to learn Python.
3. It supports object-oriented programming and is most commonly used to perform general-purpose programming.
4. Python is used in several domains like Data Science, Machine Learning, Deep Learning, Artificial Intelligence, Scientific Computing Scripting, Networking, Game Development Web Development, Web Scraping, and various other domains, System Scripting, Software Development, and Complex Mathematics.

What are the benefits of using Python language as a tool in the present scenario?

The following are the benefits of using Python language:

Object-Oriented Language, High-Level Language, Dynamically Typed language, Extensive support Libraries, Presence of third-party modules, Open source and community development, Portable and Interactive, Portable across Operating systems.

Is Python a compiled language or an interpreted language?

Python is a partially compiled language and partially interpreted language. '#' is used to comment oneverything that comes after on the line.

Difference between a Mutable datatype and an Immutable data type?

Mutable data types can be edited i.e., they can change at runtime. Eg – List, Dictionary, etc. Immutable data types can not be edited i.e., they can not change at runtime. Eg – String, Tuple, etc.

What is a lambda function?

A lambda function is an anonymous function. This function can have any number of parameters but,can have just one statement.

Pass means performing no operation or in other words, it is a placeholder in the compound statement,where there should be a blank left and nothing has to be written there.

Python provides various web frameworks to develop web applications.

The popular python web frameworks are **Django, Pyramid, Flask.**

1. Python's standard library supports for E-mail processing, FTP, IMAP, and other Internet protocols.
2. Python's **SciPy and NumPy**help in scientific and computational application development.

3. Python's **Tkinter** library supports to create desktop-based GUI applications.

What is the difference between / and // in Python?

// represents floor division whereas / represents precise division. 5//2 = 2 5/2 = 2.5

Yes, **indentation is required in Python**. A Python interpreter can be informed that a group of statements belongs to a specific block of code by using Python indentation. Indentations make the code easy to read for developers in all programming languages but in Python, it is very important to indentthe code in a specific order.

What is Scope in Python?

The location where we can find a variable and also access it if required is called the scope of a variable. **Python Local variable:** Local variables are those that are initialized within a function and are uniqueto that function. It cannot be accessed outside of the function.

Python Global variables: Global variables are the ones that are defined and declared outside any function and are not specified to any function.

Module-level scope: It refers to the global objects of the current module accessible in the program. Outermost scope: It refers to any built-in names that the program can call. The name referenced islocated last among the objects in this scope.

Python documentation strings(or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods.

Declaring Docstrings: The docstrings are declared using ”’triple single quotes”’ or “”””triple double quotes””””” just below the class, method, or function declaration. All functions should have a docstring. **Accessing Docstrings:** The docstrings can be accessed using the___doc method of the object or using the help function.

What is slicing in Python?

Python Slicing is a string operation for extracting a part of the string, or some part of a list. With this operator, one can specify where to start the slicing, where to end, and specify the step. List slicing returns a new list from the existing list.

PIP is an acronym for Python Installer Package which provides a seamless interface to install various Python modules. It is a command-line tool that can search for packages over the internet and install them without any user interaction.

SAMPLE PROGRAMS

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Example1: Creating the array

```
import numpy as np
a = np.array([1,2,3])
print a
```

Example2: Creating array by specifying data type

```
# dtype parameter
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print a
```

Example3: Getting the shape (No. of rows and No. columns)

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print  a.shape
```

Example 4: Reshaping the array

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
b = a.reshape(3,2)
print b
```

Example 5: Create array values from 1 to 10

```
import numpy as np
a = np.arange(12)
a.ndim
# now reshape it
b = a.reshape(4,3)
print b
```

Example 6: Slicing and indexing

```
import numpy as np
a = np.arange(10)
s = slice(2,7,2)
print(a[s])
```

Example 7: Create array values from 1 to 10 and Slice items between indexes

```
import numpy as np
a = np.arange(10)
print(a[2:5])
```

Example 8: Arithmetic operations on array

```
import numpy as np
a = np.arange(9, dtype = np.float_).reshape(3,3)
print ('First array:' )
print (a)
print ('\n')
print ('Second array:')
b = np.array([10,10,10])
print (b)
print ('\n')
print ('Add the two arrays:')
print (np.add(a,b))
print('\n')
print ('Subtract the two arrays:')
print (np.subtract(a,b))
print ('\n')
print ('Multiply the two arrays:' )
print (np.multiply(a,b))
print ('\n' )
print ('Divide the two arrays:' )
print (np.divide(a,b))
```

Example 9: Applying Numpy array function

```
import numpy as np
a = np.array([10,100,1000])
print 'Our array is:'
print a
print '\n'
print 'Applying power function:'
print np.power(a,2)
print '\n'
print 'Second array:'
b = np.array([1,2,3])
print b
print '\n'
print 'Applying power function again:'
print np.power(a,b)
```

PANDAS

Pandas library, it provides high performance, easy to use data structures, and analysis tools for the python programming language. It is an open source python library which provides high performance data manipulation and analysis We use Pandas library to work with data frames. Whenever we read any data into Spyder that becomes a data frame. That is what we call it as a data frame, where the data frames being represented in terms of a tabular fashioned data where each row will be represented as sample and each column will be present are so variable. Import Pandas library as pd

1: Reading .csv file and 1st column is made the index column and replace missing values with "??"

```
Import pandas as pd
data_csv= pd.read_csv ("cars_data.csv")
data_csv=pd.read_csv("cars_data.csv",index_col=0,na_values=["??","???"])
```

2: Knowing the number of rows and colomns in the dataframe

```
data_csv.index
data_csv.columns
data_csv. shape
```

3: Data types

```
Print (data_csv.dtypes)                    ; returns the data type of each column
print (data_csv.info())                    ; returns the summary of data type
Print (np.unique (data_csv ['automatic'])) ; returns unique elements of the columns
data_csv.isnull ().sum ()                  ; counts the number of missing values
```

4: checking the relationship between two columns

```
pd.crosstab(index=data_csv["Automatic"],columns=data_csv["Fuel_Type"],dropna=True )
; relationship between the columns automatic and fueltype is checked by dropping the
        missing values.
```