

A
Practical Assignment
On
Python Programming Lab Assignment
Master of Computer Application -I Sem



RUNGTA INTERNATIONAL SKILLS UNIVERSITY

SESSION: 2025-26

Subject teacher name :-
Miss Kavita Kanwar

Submitted By:-
Shashank Sinha
ERP :- RU-25-11337

**RUNGTA INTERNATIONAL SKILLS
UNIVERSITY,CG
SCHOOL OF INFORMATION TECHNOLOGY**

INDEX

S.No	Name of Practical	Submission Date	Remarks
1.	Print following patterns: 1 12 123 1234 12345 * *** ***** ***** * * * * * * * * * * * * * *** * * * * *** * * * * *		
2.	Write to find the probability of rolling a dice/flipping a coin.		
3.	Write a program to find factorial of a number using recursion.		
4.	Write a program to search for an item in a user-provided list and display the position if found, otherwise print “item not found.”		

5.	Given a list of employee records as dictionaries, sort them by salary and display the sorted list.		
6.	Write a program that reads a text file and counts the number of lines, words and characters.		
7.	Read a sentence and display how many times each word appears, ignoring case and punctuation.		
8.	Write a Python program that lists all files in a directory and categorizes them by file extension.		
9.	Inside the circle.py module, create a class Circle that demonstrates by using a separate Point class (for center coordinates). Write the code for both classes.		
10.	Add an <code>_init_.py</code> file inside the shapes package that exposes only Circle and Rectangle using <code>_all_</code> . Write code to show how from <code>shapes import *</code> will work after this.		
11.	Write a program in main.py to import the shapes package using both: <ul style="list-style-type: none"> • Absolute imports • Relative imports (inside package) 		
12.	Write a Python program that reads two numbers from the user and performs division. Use try-except to handle the following exceptions: <ul style="list-style-type: none"> • ValueError (if the user enters non-numeric input) 		

	<ul style="list-style-type: none"> • ZeroDivisionError (if the second number is zero) • Display appropriate error messages. 		
13.	<p>Create a Python function <code>read_file(filename)</code> that opens and reads a text file. Use try-except-finally to handle:</p> <ul style="list-style-type: none"> • FileNotFoundError if the file does not exist • PermissionError if the file cannot be opened <p>Finally FileNotFoundError block should print “File read attempt completed.”</p>		
14.	Use California Housing datasets to build a Linear Regression model and print MAE and R ² score.		
15.	Perform Agglomerative Hierarchical Clustering on the Iris dataset.		

Practical-1

Aim: Print following patterns:

(a) 1

12

123

1234

12345

(b) *

(c) *

* * *

* * * * *

* * *

*

(d) * * *

* *

* *

* *

(e) * *

*

* *

Code :-

Shashank Sinha > Python > Lab Assignment > prac1.ipynb > # Pattern 1: Number Triangle
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jup

```
# Pattern 1: Number Triangle
print("Pattern 1:\n")
for i in range(1, 6):
    for j in range(1, i + 1):
        print(j, end="")
    print()

# small gap
print()

# Pattern 2: Diamond-like star pattern
print("Pattern 2:\n")
n = 4
for i in range(1, n + 1):
    print(" " * (n - i) + "* " * (2*i - 1))
print()

# Pattern 3: Bigger diamond pattern
print("Pattern 3:\n")
rows = 3
for i in range(1, rows + 1):
    print(" " * (rows - i) + "*" * (2 * i - 1))
for i in range(rows - 1, 0, -1):
    print(" " * (rows - i) + "*" * (2 * i - 1))

print()
```

```
# Pattern 4: Square border pattern
print("Pattern 4:\n")
n = 4
for i in range(n):
    for j in range(n):
        if i == 0 or i == n - 1 or j == 0 or j == n - 1:
            print("*", end="")
        else:
            print(" ", end="")
    print()

# Pattern 5: Cross star pattern
print("Pattern 5:\n")
n = 3
for i in range(n):
    for j in range(n):
        if i == j or i + j == n - 1:
            print("*", end="")
        else:
            print(" ", end="")
    print()

print()
```

Output :-

Pattern 1:

```
1  
12  
123  
1234  
12345
```

Pattern 2:

```
*  
* * *  
* * * * *  
* * * * * *
```

Pattern 3:

```
*  
***  
*****  
***  
*
```

Pattern 4:

```
****  
* *  
* *  
****
```

Pattern 5:

```
* *  
*  
* *
```

Practical-2

Aim: Write to find the probability of rolling a dice/flipping a coin

Code :-

```
Shashank.Sinha > Python > Lab Assignment > prac2.ipynb > # Program to find probability of rolling a dice or flipping a coin
↳ Generate + Code + Markdown | ⌂ Run All ⌂ Restart ⌂ Clear All Outputs | ⌂ Jupyter Variables ⌂ Outline ...
```

```
if choice == 1:
    # For a fair dice
    total_outcomes = 6
    print("For a fair dice, possible outcomes are: 1, 2, 3, 4, 5, 6")
    print(f"Total number of possible outcomes (total_outcomes) is: {total_outcomes}")

    event = int(input("Enter the number you want to get (1-6): "))

    if 1 <= event <= 6:
        # Each number has only one occurrence
        favorable_outcomes = 1
        probability = favorable_outcomes / total_outcomes

        print(f"\nProbability of getting {event} on a dice = (favorable_outcomes) / (total_outcomes) = {favorable_outcomes}/{total_outcomes} = {probability}")
    else:
        print("Invalid number. Please enter a number between 1 and 6.")

elif choice == 2:
    # For flipping a coin
    total_outcomes = 2
    print("For a fair coin, possible outcomes are: Head, Tail")
    event = input("Enter 'H' for Head or 'T' for Tail: ").upper()

    if event == 'H' or event == 'T':
        # Head or Tail each has one occurrence
        favorable_outcomes = 1
        probability = favorable_outcomes / total_outcomes

        print(f"\nProbability of getting {event} = (favorable_outcomes) / (total_outcomes) = {favorable_outcomes}/{total_outcomes} = {probability}")
    else:
        print("Invalid input. Please enter H or T.")

else:
    print("Invalid choice.")
```

Output :-

```
Choose an experiment:  
1. Roll a dice  
2. Flip a coin  
For a fair dice, possible outcomes are: 1, 2, 3, 4, 5, 6  
Total number of possible outcomes (total_outcomes) is: 6  
  
Probability of getting 5 on a dice = (favorable_outcomes) / (total_outcomes) = 1/6 = 0.1666666666666666
```

```
Choose an experiment:  
1. Roll a dice  
2. Flip a coin  
For a fair coin, possible outcomes are: Head, Tail
```

```
Probability of getting H = (favorable_outcomes) / (total_outcomes) = 1/2 = 0.5
```

Practical-3

Aim: Write a program to find factorial of a number using recursion.

Code :-

Shashank Sinha > Python > Lab Assignment > prac3.ipynb > # Program to find factorial of a number using recursion
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outlin

```
# Program to find factorial of a number using recursion
def factorial(n):
    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1
    # Recursive case: n! = n * (n-1)!
    else:
        return n * factorial(n - 1)

# Taking input from user
num = int(input("Enter a number: "))

# Checking for negative numbers
if num < 0:
    print("Factorial is not defined for negative numbers.")
else:
    result = factorial(num)
    print(f"The factorial of {num} is: {result}")
```

[1]

Output :-

... The factorial of 5 is: 120

Practical-4

Aim: Write a program to search for an item in a user-provided list and display the position if found, otherwise print “Item not found.”

Code :-

```
Shashank Sinha > Python > Lab Assignment > prac4.ipynb > # Program to search for an item in a user-provided list
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ...
```

```
# Program to search for an item in a user-provided list

# Predefined list
my_list = [12, 45, 98, 65]

# Taking input from the user
item = int(input("Enter the item to search: "))

# Searching item
found = False
for i in range(len(my_list)):
    if my_list[i] == item:
        print(f"Item found at position {i+1}")
        found = True
        break

if not found:
    print("Item not found in the list.")

[1] ✓ 3.8s
```

Output :-

```
45
ip Enter the item to search: (Press 'Enter' to confirm or 'Escape' to cancel)
```

```
Item found at position 2
```

Practical-5

Aim: Given a list of employee records as dictionaries, sort them by salary and display the sorted list.

Code:

```
Shashank Sinha > Python > Lab Assignment > prac5.ipynb > # Program to sort a list of employee records by salary
↳ Generate + Code + Markdown | ⚡ Run All ⚡ Restart ✖ Clear All Outputs | 📁 Jupyter Variables ⚡ Out

# Program to sort a list of employee records by salary

# List of employee dictionaries
employees = [
    {'name': 'John', 'salary': 46000},
    {'name': 'Alice', 'salary': 35000},
    {'name': 'Bob', 'salary': 54000},
    {'name': 'David', 'salary': 42000}
]

print("Original Employee List:")
for emp in employees:
    print(emp)

# Sorting by salary (ascending)
employees.sort(key=lambda x: x['salary'])

print("\nSorted Employee List (by Salary):")
for emp in employees:
    print(emp)

[4]
```

... Original Employee List:
{'name': 'John', 'salary': 46000}
{'name': 'Alice', 'salary': 35000}
{'name': 'Bob', 'salary': 54000}
{'name': 'David', 'salary': 42000}

Sorted Employee List (by Salary):
{'name': 'Alice', 'salary': 35000}
{'name': 'David', 'salary': 42000}
{'name': 'John', 'salary': 46000}
{'name': 'Bob', 'salary': 54000}

Output :-

Practical-6

Aim: Write a program that reads a text file and counts the number of lines, words and characters.

Code :-

```
Shashank Sinha > Python > Lab Assignment > prac6.ipynb > # Program to count number of lines, words, and characters in a file
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline

# Program to count number of lines, words, and characters in a file

filename = "sample.txt"

# Displaying file name
print("Reading file:", filename)

try:
    with open(filename, "r") as file:
        lines = file.readlines()

    line_count = len(lines)
    word_count = 0
    char_count = 0

    for line in lines:
        words = line.split()
        word_count += len(words)
        char_count += len(line) # includes newline

    # Display file content
    print("\nFile content:")
    for line in lines:
        print(line.strip())

    # Print statistics
    print("\n--- File Statistics ---")
    print(f"Total Lines: {line_count}")
    print(f"Total Words: {word_count}")
    print(f"Total Characters: {char_count}")

except FileNotFoundError:
    print("File not found. Please make sure 'sample.txt' exists.")

[1] ✓ 0.0s
```

Output :-

```
... Reading file: sample.txt
File content:
Hello World!

--- File Statistics ---
Total Lines: 1
Total Words: 2
Total Characters: 12
```

Practical-7

Aim: Read a sentence and display how many times each word appears, ignoring case and punctuation.

Code :-

```
Shashank Sinha > Python > Lab Assignment > prac7.ipynb > import string
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ...
```

```
import string

# Program to count how many times each word appears in a sentence (ignoring case and punctuation)

text = "Rain rain go away, come again another day."
print("Input Text:", text)

# Converting to lowercase and removing punctuation
clean_text = text.lower().translate(str.maketrans("", "", string.punctuation))

# Splitting into words
words = clean_text.split()

# Counting frequency using dictionary
word_count = {}
for word in words:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1

# Displaying result
print("\nWord Frequency:")
print(word_count)
```

[1]

Output :-

```
Input Text: Rain rain go away, come again another day.
```

```
Word Frequency:
```

```
{'rain': 2, 'go': 1, 'away': 1, 'come': 1, 'again': 1, 'another': 1, 'day': 1}
```

Practical-8

Aim: Write a Python program that lists all files in a directory and categorizes them by file extension.

Code:

Shashank Sinha > Python > Lab Assignment > prac8.ipynb > import os
Generate + Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ...

```
import os
from collections import defaultdict

# Program to list all files in a directory and categorize them by file extension.

# Directory to scan (you can change this path as needed)
directory = "."

print("Scanning directory:", os.path.abspath(directory))
# Dictionary to store files by extension
files_by_extension = defaultdict(list)

# Loop through all files in the directory
for filename in os.listdir(directory):
    if os.path.isfile(os.path.join(directory, filename)):
        # Split the file into name and extension
        _, ext = os.path.splitext(filename)

        # Convert extension to lowercase and remove the dot
        ext = ext.lower().lstrip('.')

        # Use a placeholder if there is no extension
        if ext:
            files_by_extension[ext].append(filename)
        else:
            files_by_extension["no_extension"].append(filename)

# Display categorized files
print("\nFiles categorized by extension:")
for ext, files in files_by_extension.items():
    print(f"\n{ext}:")
    for f in files:
        print(f"{f}")
```

Output :-

```
Scanning directory: c:\Users\22sha\OneDrive\Desktop\MCA\Shashank Sinha\Python\Lab Assigment
```

```
Files categorized by extension:
```

```
ipynb:  
prac1.ipynb  
prac2.ipynb  
prac3.ipynb  
prac4.ipynb  
prac5.ipynb  
prac6.ipynb  
prac7.ipynb  
prac8.ipynb
```

```
txt:  
sample.txt
```

Practical-9

Aim:- Inside the **circle.py** module, create a class **Circle** that demonstrates by using a separate **Point** class (for center coordinates). Write the code for both classes.

Code:-

point.py

```
Shashank Sinha > Python > Lab Assigment > point.py > ↗
1   class A:
2       def __init__(self, x, y):
3           self.x = x
4           self.y = y
5
6       def get_coordinates(self):
7           return (self.x, self.y)
8
9
```

circle.py

```
Shashank Sinha > Python > Lab Assigment > circle.ipynb > import math
Generate + Code + Markdown | Run All Restart Clear All Out
```

```
▶
    import math
    from point import A

    class Circle:
        def __init__(self, center: A, radius):
            self.center = center
            self.radius = radius

        def area(self):
            return math.pi * self.radius ** 2

        def perimeter(self):
            return 2 * math.pi * self.radius

        def center_coordinates(self):
            return self.center.get_coordinates()
```

[1] ✓ 0.0s

Practical-10

Aim:- Add an `__init__.py` file inside the `shapes` package that exposes only `Circle` and `Rectangle` using `__all__`. Write code to show how `from shapes import *` will work after this.

Module1(`__init__.py`)

Shashank Sinha > Python > Lab Assignment > shape > `__init__.py` > ...

```
1  from shape.circle import Circle
2  from shape.rectangle import Rectangle
3
4  __all__ = ["Circle", "Rectangle"]
```

Module2(`circle.py`)

Shashank Sinha > Python > Lab Assignment > shape > `circle.py` > `Circle`

```
1  import math
2  from shape.point import Point
3
4  class Circle:
5      def __init__(self, center, radius):
6          self.center = center
7          self.radius = radius
8
9      def area(self):
10         return math.pi * self.radius ** 2
11
12     def perimeter(self):
13         return 2 * math.pi * self.radius
```

Module3(rectangle.py)

Shashank Sinha > Python > Lab Assigment > shape > rectangle.py > Rectangle > perimeter

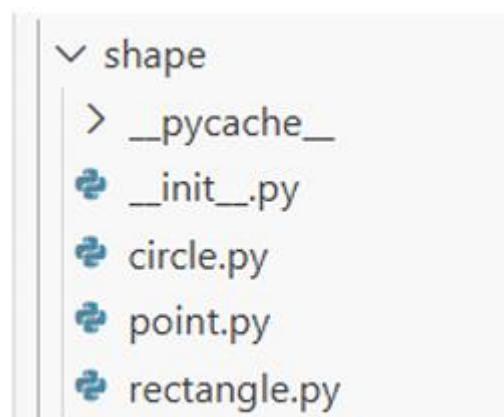
```
1  class Rectangle:
2      def __init__(self, length, width):
3          self.length = length
4          self.width = width
5
6      def area(self):
7          return self.length * self.width
8
9      def perimeter(self):
10         return 2 * (self.length + self.width)
```

Module 4(point.py)

Shashank Sinha > Python > Lab Assigment > shape > point.py > Point > _init_

```
1  class Point:
2      def __init__(self, x=0, y=0):
3          self.x = x
4          self.y = y
```

Directory Samples



Code:-

Shashank Sinha > Python > Lab Assigment > `prac10.ipynb` > `from shape import *`
❖ Generate + Code + Markdown | ▶ Run All ⚡ Restart ✖ Clear All Outputs | 🗑

```
▶ from shape import *

# Circle usage
c = Circle(None, 5)
print("Circle Area:", c.area())

# Rectangle usage
r = Rectangle(10, 4)
print("Rectangle Perimeter:", r.perimeter())

# Point is NOT accessible
# p = Point(2, 3) # ✖ NameError
```

[16]

Output:-

```
... Circle Area: 78.53981633974483
      Rectangle Perimeter: 28
```

Practical-11

Aim:- Write a program in **main.py** to import the **shapes** package using both:

- Absolute imports
- Relative imports (inside package)

Module 1(point.py)

```
Shashank Sinha > Python > Lab Assigment > shapes > point.py > Point > display
1  class Point:
2      def __init__(self, x=0, y=0):
3          self.x = x
4          self.y = y
5
6      def display(self):
7          return f"({self.x}, {self.y})"
```

Module 2(circle.py)

```
Shashank Sinha > Python > Lab Assigment > shapes > circle.py > Circle > display
1  import math
2  from shapes.point import Point
3
4  class Circle:
5      def __init__(self, center, radius):
6          self.center = center      # center is a Point object
7          self.radius = radius
8
9      def area(self):
10         return math.pi * self.radius ** 2
11
12     def perimeter(self):
13         return 2 * math.pi * self.radius
14
15     def display(self):
16         print("Center:", self.center.display())
17         print("Radius:", self.radius)
18         print("Area:", self.area())
19         print("Perimeter:", self.perimeter())
```

Module 3(rectangle.py)

Shashank Sinha > Python > Lab Assignment > shapes > rectangle.py > Rectangle > perimeter

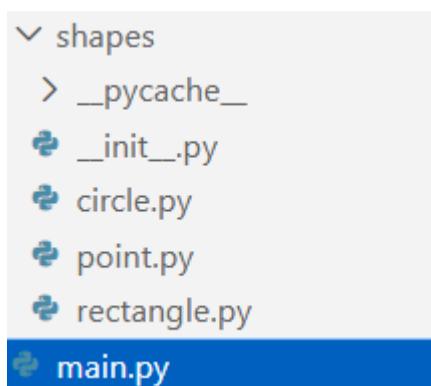
```
1  class Rectangle:
2      def __init__(self, length, width):
3          self.length = length
4          self.width = width
5
6      def area(self):
7          return self.length * self.width
8
9      def perimeter(self):
10         return 2 * (self.length + self.width)
```

main.py

Shashank Sinha > Python > Lab Assignment > main.py > ...

```
1  # Absolute imports
2  from shapes.circle import Circle
3  from shapes.rectangle import Rectangle
4  from shapes.point import Point
5
6  # Create objects
7  p = Point(0, 0)
8  c = Circle(p, 5)
9  r = Rectangle(10, 4)
10
11 print("Circle Area:", c.area())
12 print("Circle Perimeter:", c.perimeter())
13 print("Rectangle Area:", r.area())
14 print("Rectangle Perimeter:", r.perimeter())
```

Directory Sample



Output:-

```
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
Rectangle Area: 40
Rectangle Perimeter: 28
```

Practical-12

Aim:- Write a Python program that reads two numbers from the user and performs division.

use **try-except** to handle the following exceptions:

- **ValueError** (if the user enters non-numeric input)
- **ZeroDivisionError** (if the second number is zero)
- Display appropriate error messages.

Code:-

Shashank Sinha > Python > Lab Assignment > `prac12.ipynb` > `try:`

❖ Generate + Code + Markdown | Run All Clear All Outputs | Outline



```
try:  
    # Read two numbers from the user  
    num1 = float(input("Enter the first number: "))  
    num2 = float(input("Enter the second number: "))  
  
    # Perform division  
    result = num1 / num2  
    print("Result of division:", result)  
  
except ValueError:  
    print("Error: Please enter valid numeric values.")  
  
except ZeroDivisionError:  
    print("Error: Division by zero is not allowed.")
```

[4]

Input:-

d

Enter the second number: (Press 'Enter' to confirm or 'Escape' to cancel)

Output:-

Error: Division by zero is not allowed.

Input:-

a

Enter the second number: (Press 'Enter' to confirm or 'Escape' to cancel)

Output:-

Error: Please enter valid numeric values.

Practical-13

Aim:- Create a Python function **read_file(filename)** that opens and reads a text file.

Use **try-except-finally** to handle:

- **FileNotFoundException** if the file does not exist
 - **PermissionError** if the file cannot be opened
- Finally **FileNotFoundException** block should print "**File read attempt completed.**"

Code:-

The screenshot shows a Jupyter Notebook interface with the following details:

- Header: Shashank Sinha > Python > Lab Assignment > prac13.ipynb > def read_file(filename):
- Toolbar: Generate, + Code, + Markdown, Run All, Restart, Clear All Outputs, Jupyter Variables, Outline.
- Code Cell:

```
def read_file(filename):
    try:
        with open(filename, "r") as file:
            print(file.read())

    except FileNotFoundError:
        print("Error: File not found. Make sure the file exists in the same folder.")

    except PermissionError:
        print("Error: Permission denied. Cannot open the file.")

    finally:
        print("File read attempt completed.")

# Function call
filename = input("Enter file name (example: data.txt): ")
read_file(filename)
```

Input:-

sample.txt

Enter file name (example: data.txt): (Press 'Enter' to confirm or 'Escape' to cancel)

Output:-

hello
File read attempt completed.

Input:-

```
shass.txt
```

Enter file name (example: data.txt): (Press 'Enter' to confirm or 'Escape' to cancel)

Output:-

Error: File not found. Make sure the file exists in the same folder.
File read attempt completed.

Practical-14

Aim:- Use California Housing datasets to build a Linear Regression model and print MAE and R² score.

Code:-

Shashank Sinha > Python > Lab Assigment > prac14.ipynb > import pandas as pd

Generate Code Markdown | Run All Restart Clear All Outputs | Jupyter Variables

Generate Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.datasets import fetch_california_housing

# Load dataset
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['MedHouseValue'] = data.target

# Feature and target
X = df[['MedInc']]
y = df['MedHouseValue']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Model
model = LinearRegression()
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)
```

```

# Evaluation
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
print("Coefficient (Slope):", model.coef_)
print("Intercept:", model.intercept_)

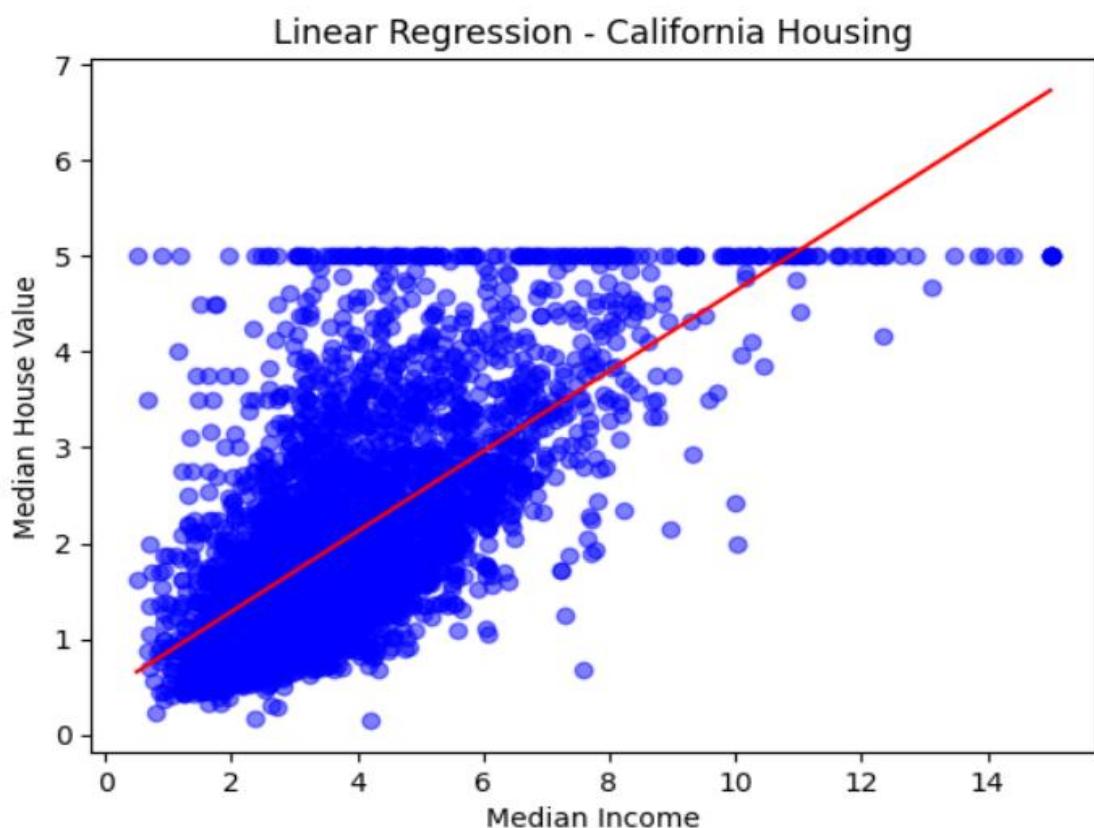
# Plot (sorted for clean line)
X_test_sorted = X_test.sort_values(by='MedInc')
y_pred_sorted = model.predict(X_test_sorted)

plt.scatter(X_test, y_test, color='blue', alpha=0.5)
plt.plot(X_test_sorted, y_pred_sorted, color='red')
plt.xlabel("Median Income")
plt.ylabel("Median House Value")
plt.title("Linear Regression - California Housing")
plt.show()

```

Output:-

Mean Absolute Error (MAE): 0.629908653009376
 R² Score: 0.45885918903846656
 Coefficient (Slope): [0.41933849]
 Intercept: 0.44459729169078677



Practical-15

Aim:- Perform Agglomerative Hierarchical Clustering on the Iris dataset .

Code:-

```
Shashank Sinha > Python > Lab Assigment > prac15.ipynb > import numpy as np
❖ Generate + Code + Markdown | ▶ Run All ⚡ Restart ✖ Clear All Outputs | [?] Ju
❖ Gener

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

# Load Iris dataset
iris = load_iris()
X = iris.data
df = pd.DataFrame(X, columns=iris.feature_names)

# -----
# Dendrogram (Hierarchical Tree)
# -----
linked = linkage(X, method='ward')

plt.figure(figsize=(10, 6))
dendrogram(linked)
plt.title("Dendrogram for Iris Dataset")
plt.xlabel("Data Points")
plt.ylabel("Euclidean Distance")
plt.show()

# -----
# Agglomerative Hierarchical Clustering
# -----
model = AgglomerativeClustering(
    n_clusters=3,
    linkage='ward'
)
```

```

clusters = model.fit_predict(X)

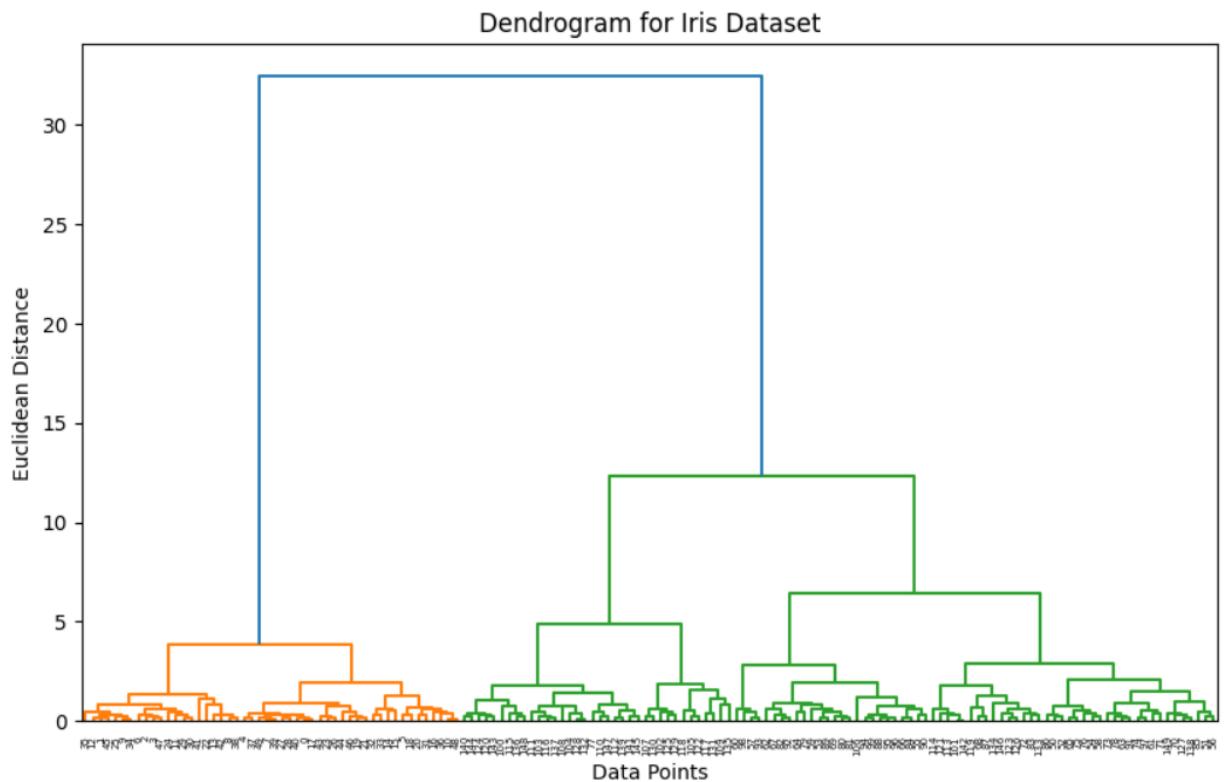
# Add cluster labels to dataframe
df['cluster'] = clusters

print(df.head())

# -----
# Visualization using first two features
# -----
plt.figure(figsize=(7, 5))
plt.scatter(X[:, 0], X[:, 1], c=clusters)
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("Agglomerative Clustering on Iris Dataset")
plt.show()

```

Output:-



```
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm) \
0           5.1          3.5         1.4          0.2
1           4.9          3.0         1.4          0.2
2           4.7          3.2         1.3          0.2
3           4.6          3.1         1.5          0.2
4           5.0          3.6         1.4          0.2
```

```
Cluster
```

```
0      1
1      1
2      1
3      1
4      1
```

```
...
```

