A

Lab Assignment

On

Python Programming With Problem Solving Lab

Master Of Computer Application - I Sem

RUNGTA INTERNATIONAL SKILLS UNIVERSITY

SESSION: 2025-26

Submitted To:-
Ms. KAVITA KANWAR

Submitted By:-
TANU CHAUDHARY
Ref No. RU-25-11704

RUNGTA INTERNATIONAL SKILLS UNIVERSITY ,CG
SCHOOL OF INFORMATION TECHNOLOGY

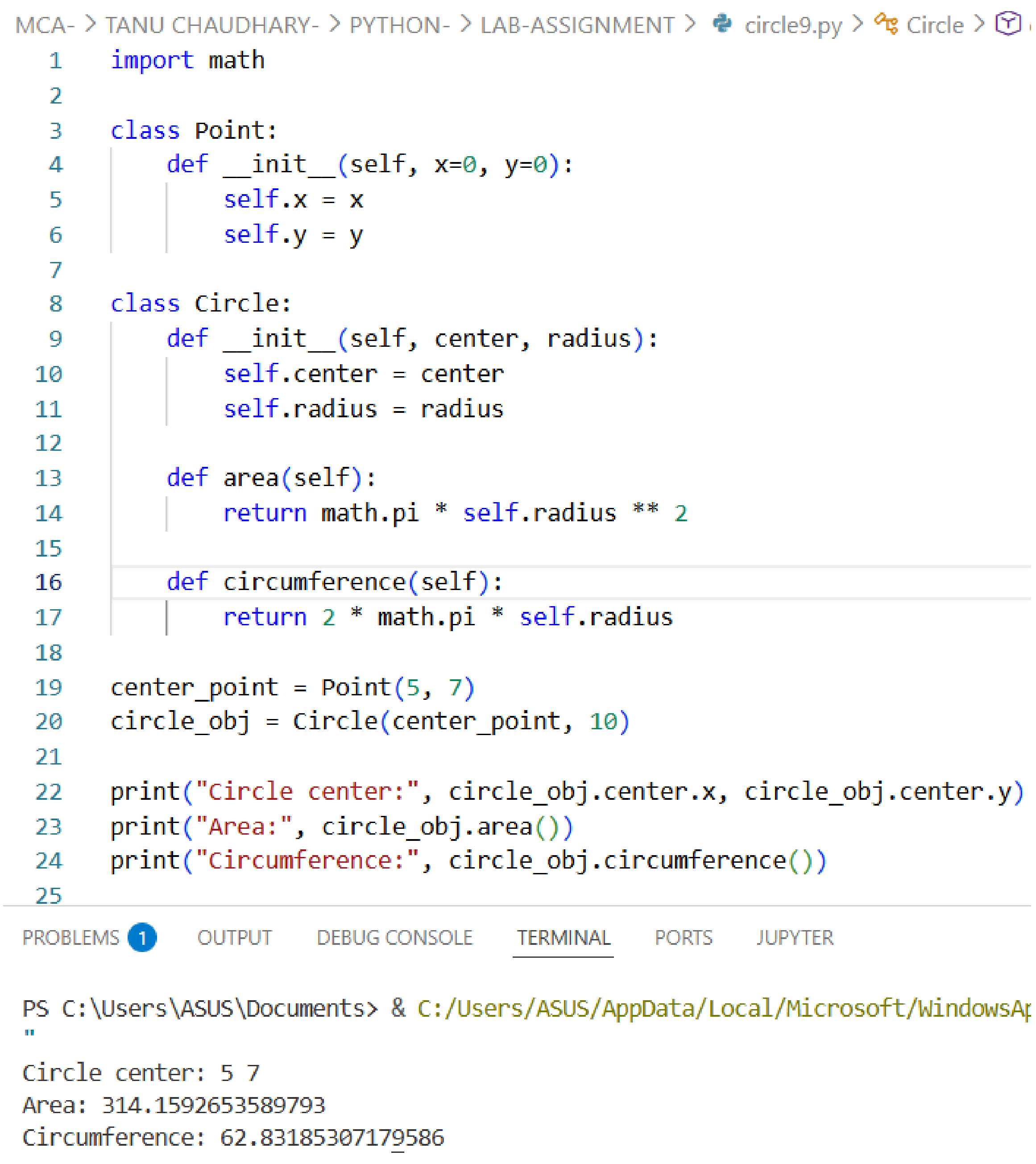
INDEX

S. No	Name of Practical	Submission Date	Remarks
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 _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.		
11	Write a program in main.py to import the shapes package using both: • 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: • ValueError (if the user enters non-numeric input) • ZeroDivisionError (if the second number is zero) • Display appropriate error messages.		
13	Create a Python function read_file(filename) that opens and reads a text file. Use try-except-finally to handle: • FileNotFoundError if the file does not exist • PermissionError if the file cannot be opened Finally FileNotFoundError block should print "File read attempt completed."		
14	Use California Housing datasets to build a Linear Regression model and print MAE and R2 score.		
15	Perform Agglomerative Hierarchical Clustering on the Iris dataset.		

AIM 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.

Source Code:



MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > circle9.py > Circle >

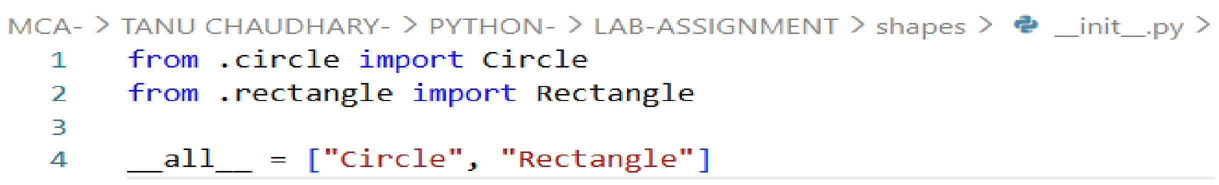
```python
import math

class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

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

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

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

center_point = Point(5, 7)
circle_obj = Circle(center_point, 10)

print("Circle center:", circle_obj.center.x, circle_obj.center.y)
print("Area:", circle_obj.area())
print("Circumference:", circle_obj.circumference())

```

```
PS C:\Users\ASUS\Documents> & C:/Users/ASUS/AppData/Local/Microsoft/WindowsAp
"

Circle center: 5 7
Area: 314.1592653589793
Circumference: 62.83185307179586
```

AIM 10:

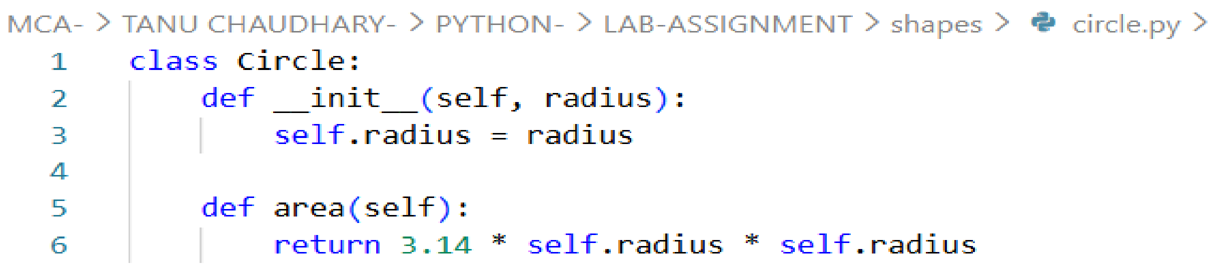
Add an _init_.py file inside the shapes package that exposes only Circle and Rectangle using _all_. Write code to show how from shapes import * will work after this.

Source Code:



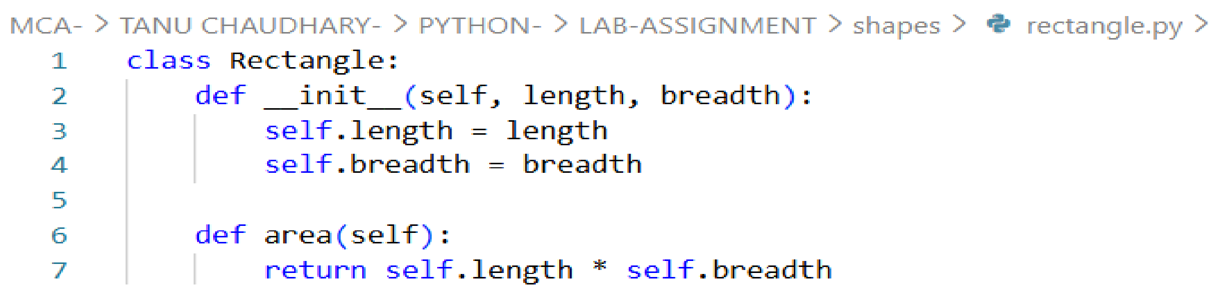
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > shapes > __init__.py >

```python
from .circle import Circle
from .rectangle import Rectangle

__all__ = ["Circle", "Rectangle"]
```



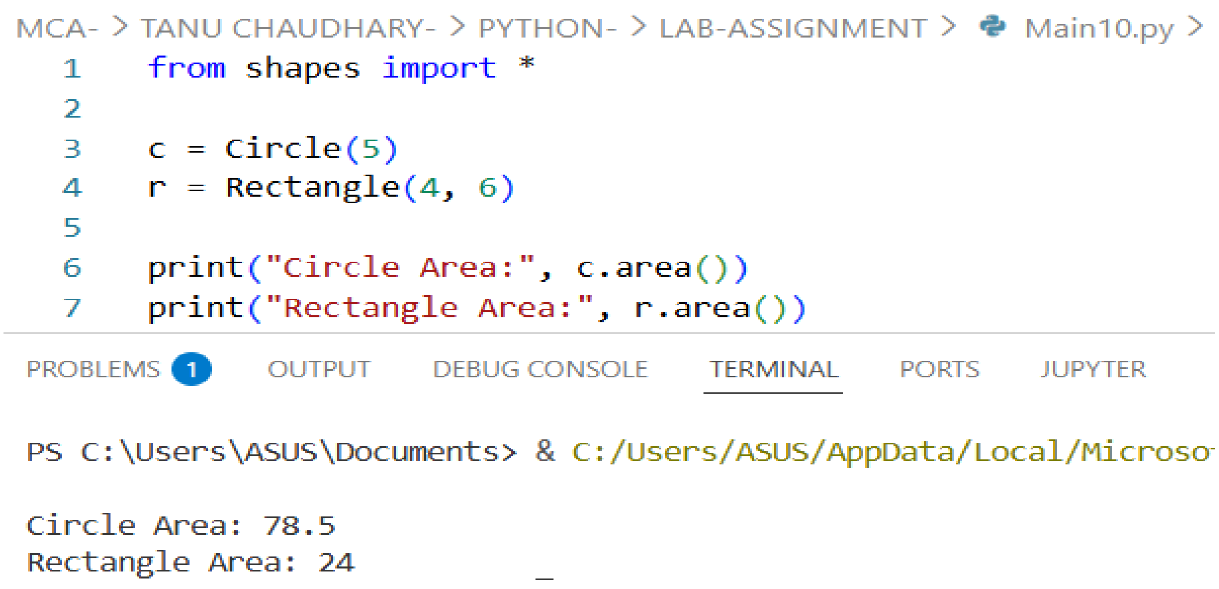
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > shapes > circle.py >

```python
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius
```



MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > shapes > rectangle.py >

```python
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def area(self):
        return self.length * self.breadth
```



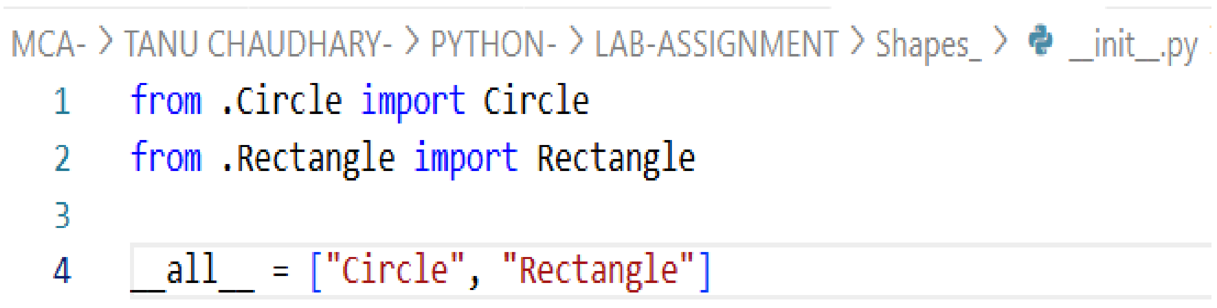
MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Main10.py >

```python
from shapes import *

c = Circle(5)
r = Rectangle(4, 6)

print("Circle Area:", c.area())
print("Rectangle Area:", r.area())
```

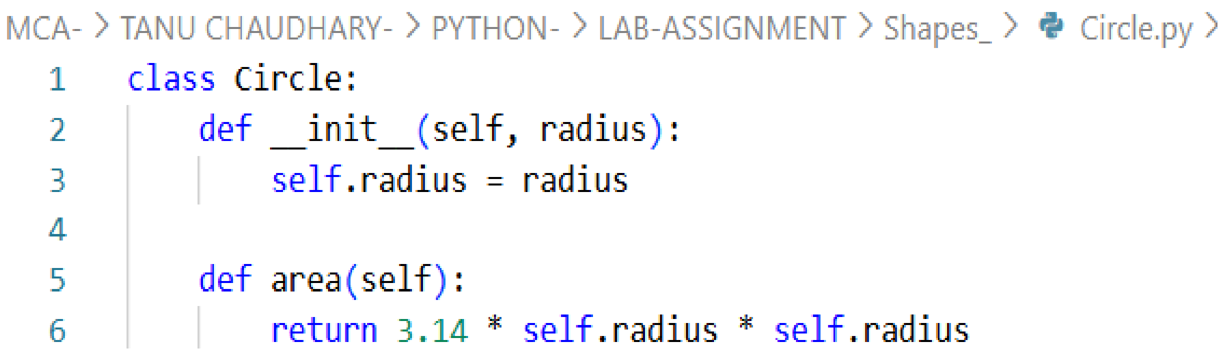
```
PS C:\Users\ASUS\Documents> & C:/Users/ASUS/AppData/Local/Microso

Circle Area: 78.5
Rectangle Area: 24
```

AIM 11:
Write a program in main.py to import the shapes package using both: • Absolute imports • Relative imports (inside package)

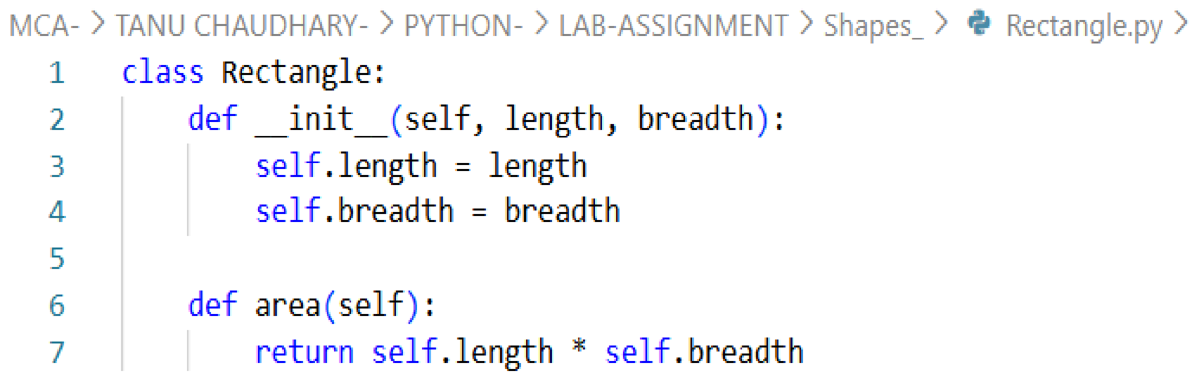
Source Code:


MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Shapes_ > __init__.py

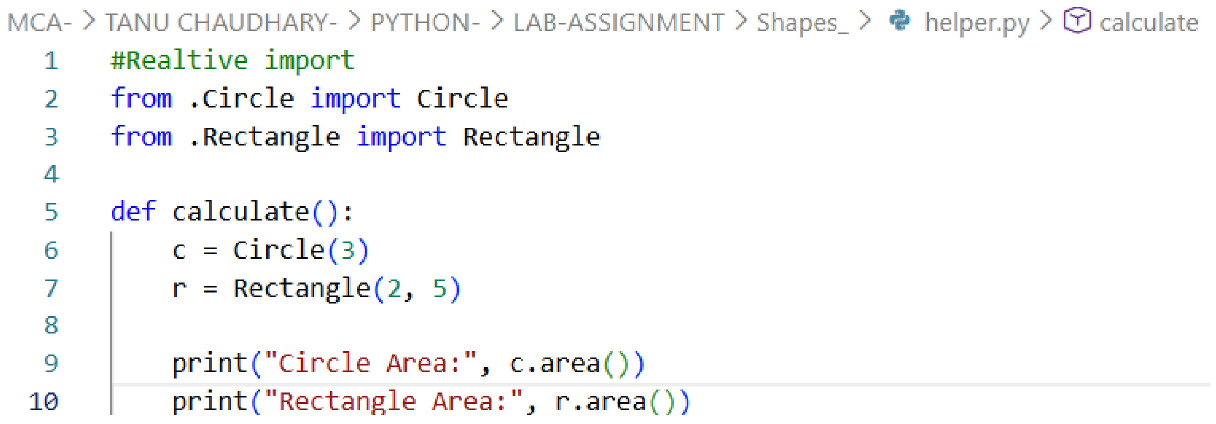
```python
from .Circle import Circle
from .Rectangle import Rectangle

__all__ = ["Circle", "Rectangle"]
```


MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Shapes_ > Circle.py >

```python
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius
```

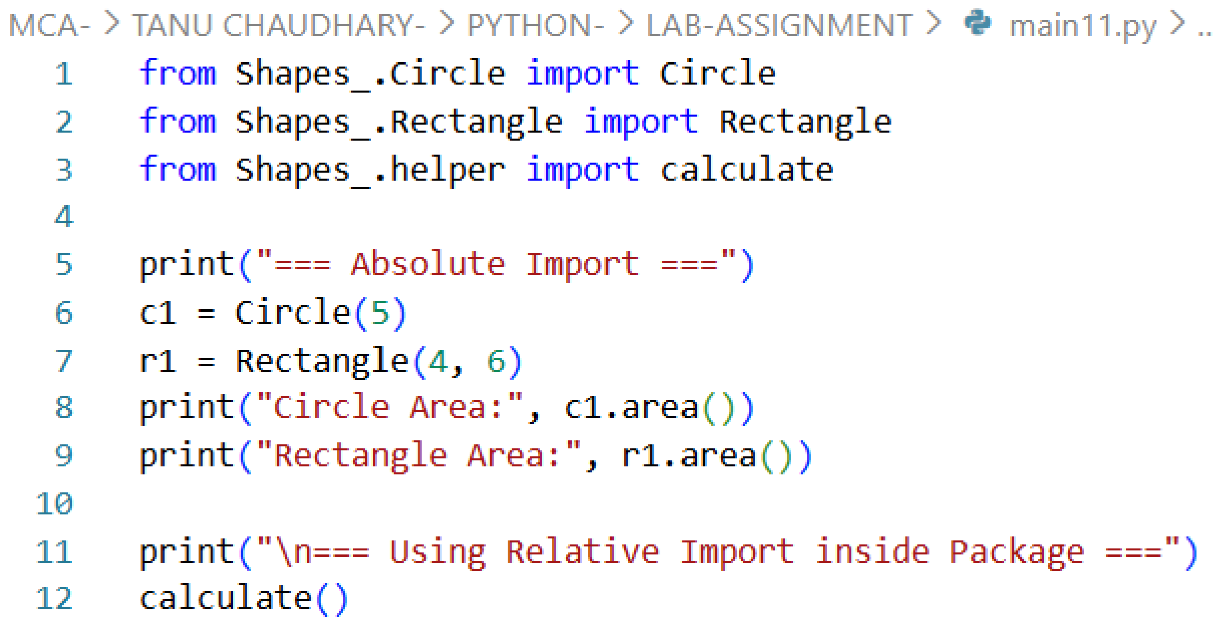

MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Shapes_ > Rectangle.py >

```python
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth

    def area(self):
        return self.length * self.breadth
```



MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Shapes_ > helper.py > calculate

```python
#Realtive import
from .Circle import Circle
from .Rectangle import Rectangle

def calculate():
    c = Circle(3)
    r = Rectangle(2, 5)

    print("Circle Area:", c.area())
    print("Rectangle Area:", r.area())
```



MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > main11.py > ..

```python
from Shapes_.Circle import Circle
from Shapes_.Rectangle import Rectangle
from Shapes_.helper import calculate

print("=== Absolute Import ===")
c1 = Circle(5)
r1 = Rectangle(4, 6)
print("Circle Area:", c1.area())
print("Rectangle Area:", r1.area())

print("\n=== Using Relative Import inside Package ===")
calculate()
```

AIM 12:

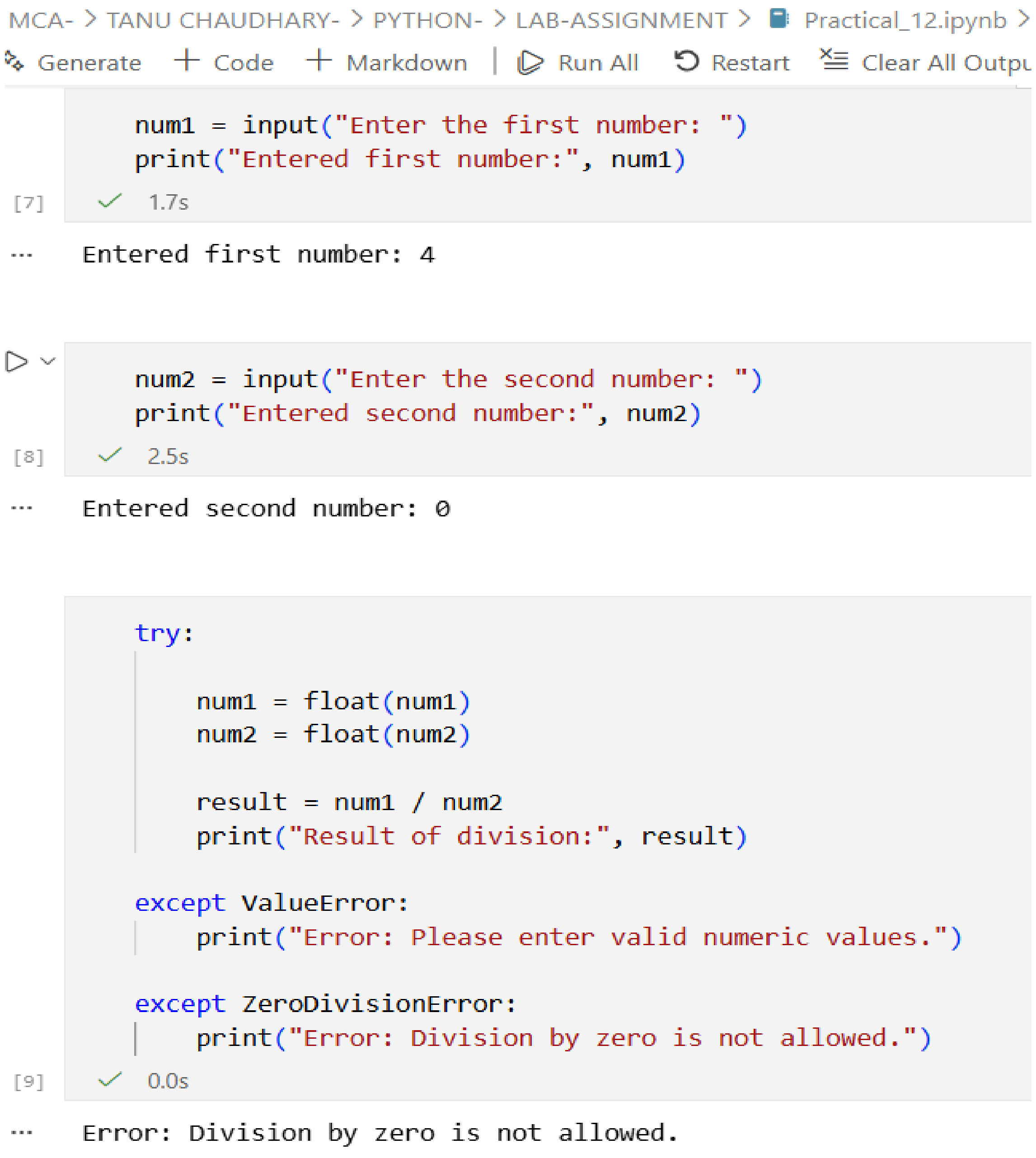
Write a Python program that reads two numbers from the user and performs division. Use try-except to handle the following exceptions:

• ValueError (if the user enters non-numeric input)

• ZeroDivisionError (if the second number is zero)

• Display appropriate error messages.

Source Code:


MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Practical_12.ipynb >
Generate + Code + Markdown | Run All Restart Clear All Outpu

```
num1 = input("Enter the first number: ")
print("Entered first number:", num1)
```

[7] ✓ 1.7s

```
Entered first number: 4
```

```
num2 = input("Enter the second number: ")
print("Entered second number:", num2)
```

[8] ✓ 2.5s

```
Entered second number: 0
```

```
try:

    num1 = float(num1)
    num2 = float(num2)

    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.")
```

[9] ✓ 0.0s

```
Error: Division by zero is not allowed.
```

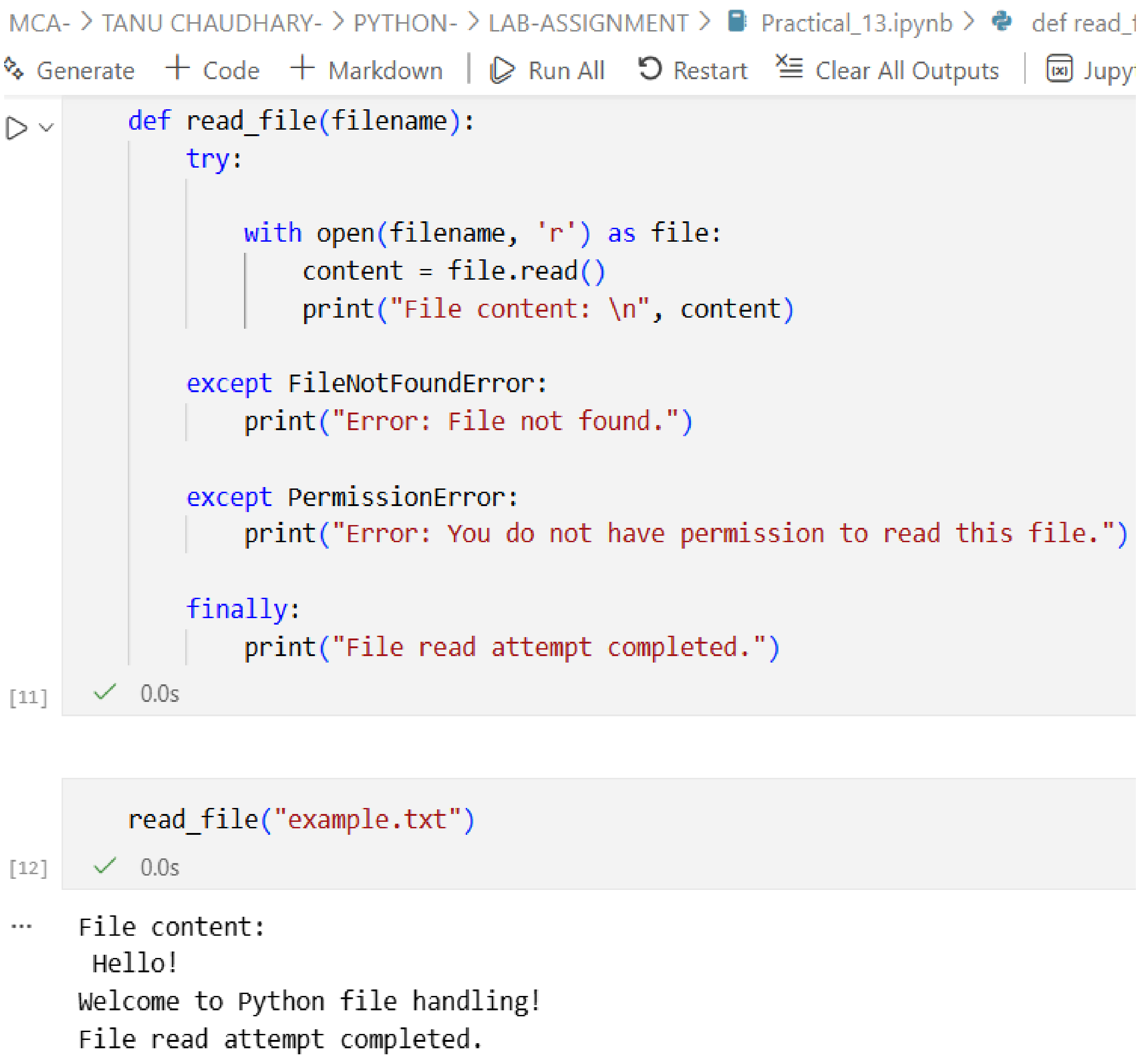
AIM 13:
Create a Python function read_file(filename) that opens and reads a text file. Use try-except-finally to handle:

• FileNotFoundError if the file does not exist

• PermissionError if the file cannot be opened

Finally FileNotFoundError block should print “File read attempt completed.”

Source Code:


MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Practical_13.ipynb > def read_f
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupy

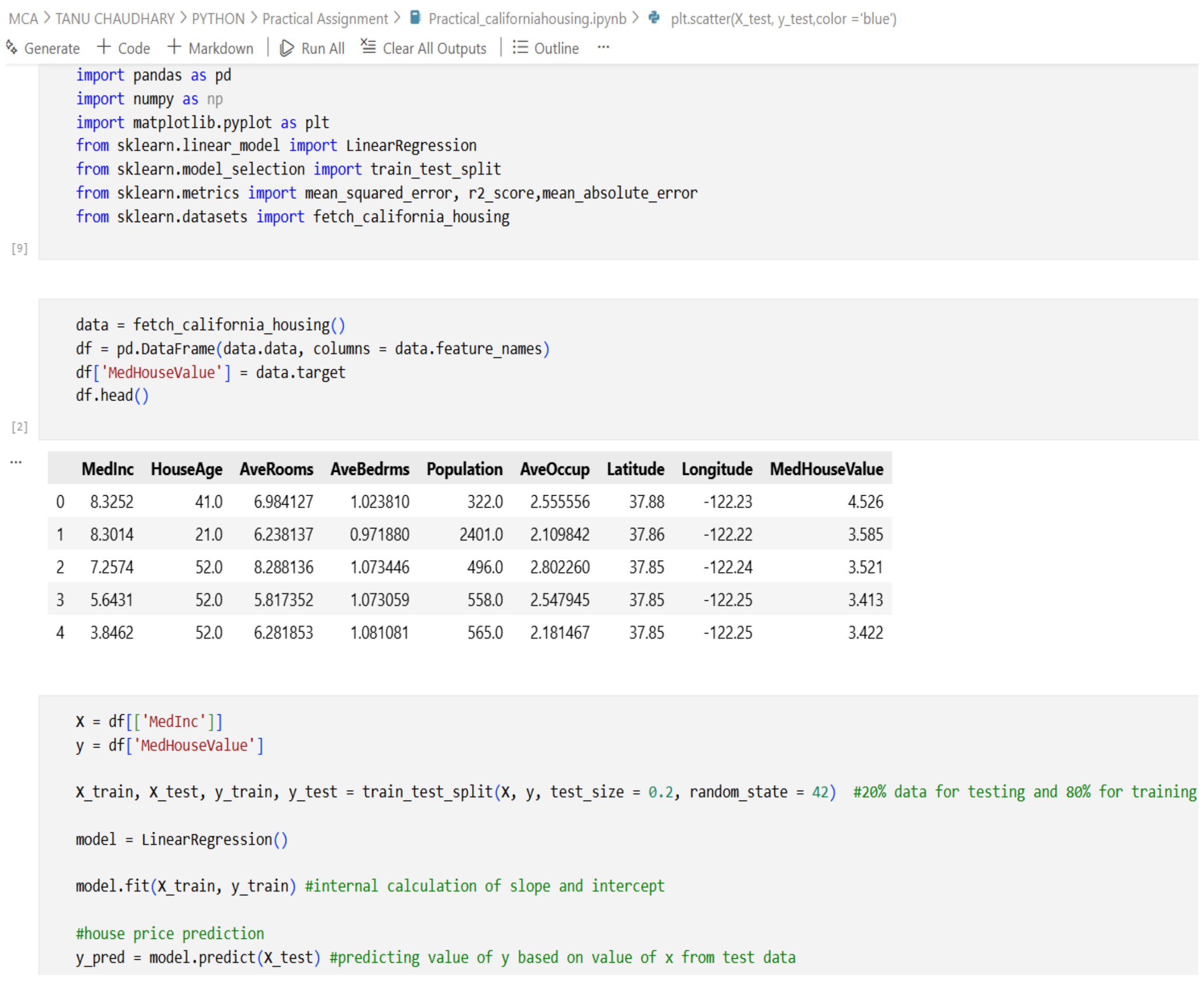
```python
def read_file(filename):
    try:

        with open(filename, 'r') as file:
            content = file.read()
            print("File content: \n", content)

    except FileNotFoundError:
        print("Error: File not found.")

    except PermissionError:
        print("Error: You do not have permission to read this file.")

    finally:
        print("File read attempt completed.")
```
[11] ✓ 0.0s

```python
read_file("example.txt")
```
[12] ✓ 0.0s

```
File content: 
 Hello!
Welcome to Python file handling!
File read attempt completed.
```

AIM 14:
Use California Housing datasets to build a Linear Regression model and print MAE and R2 score.

Source Code:

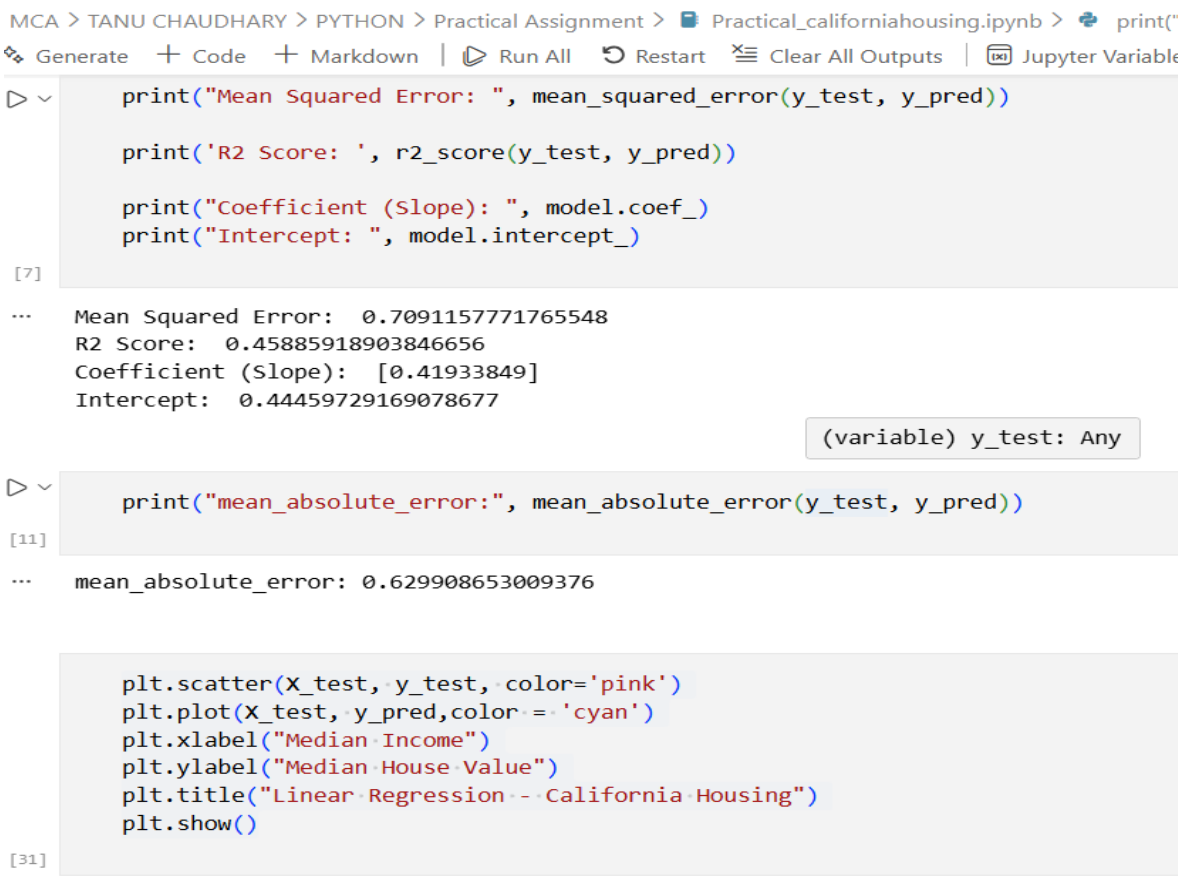

MCA > TANU CHAUDHARY > PYTHON > Practical Assignment > Practical_californiahousing.ipynb > plt.scatter(X_test, y_test,color ='blue')
Generate
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
[9]
[2]

```python
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_squared_error, r2_score,mean_absolute_error
from sklearn.datasets import fetch_california_housing
```

```python
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns = data.feature_names)
df['MedHouseValue'] = data.target
df.head()
```

```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude  MedHouseValue
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88    -122.23          4.526
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86    -122.22          3.585
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85    -122.24          3.521
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85    -122.25          3.413
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85    -122.25          3.422
```

```python
X = df[['MedInc']]
y = df['MedHouseValue']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)  #20% data for testing and 80% for training

model = LinearRegression()

model.fit(X_train, y_train) #internal calculation of slope and intercept

#house price prediction
y_pred = model.predict(X_test) #predicting value of y based on value of x from test data
```


MCA > TANU CHAUDHARY > PYTHON > Practical Assignment > Practical_californiahousing.ipynb > print("
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variable

```
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))

print('R2 Score: ', r2_score(y_test, y_pred))

print("Coefficient (Slope): ", model.coef_)
print("Intercept: ", model.intercept_)
```

[7]

```
Mean Squared Error:  0.7091157771765548
R2 Score:  0.45885918903846656
Coefficient (Slope):  [0.41933849]
Intercept:  0.44459729169078677
```

(variable) y_test: Any

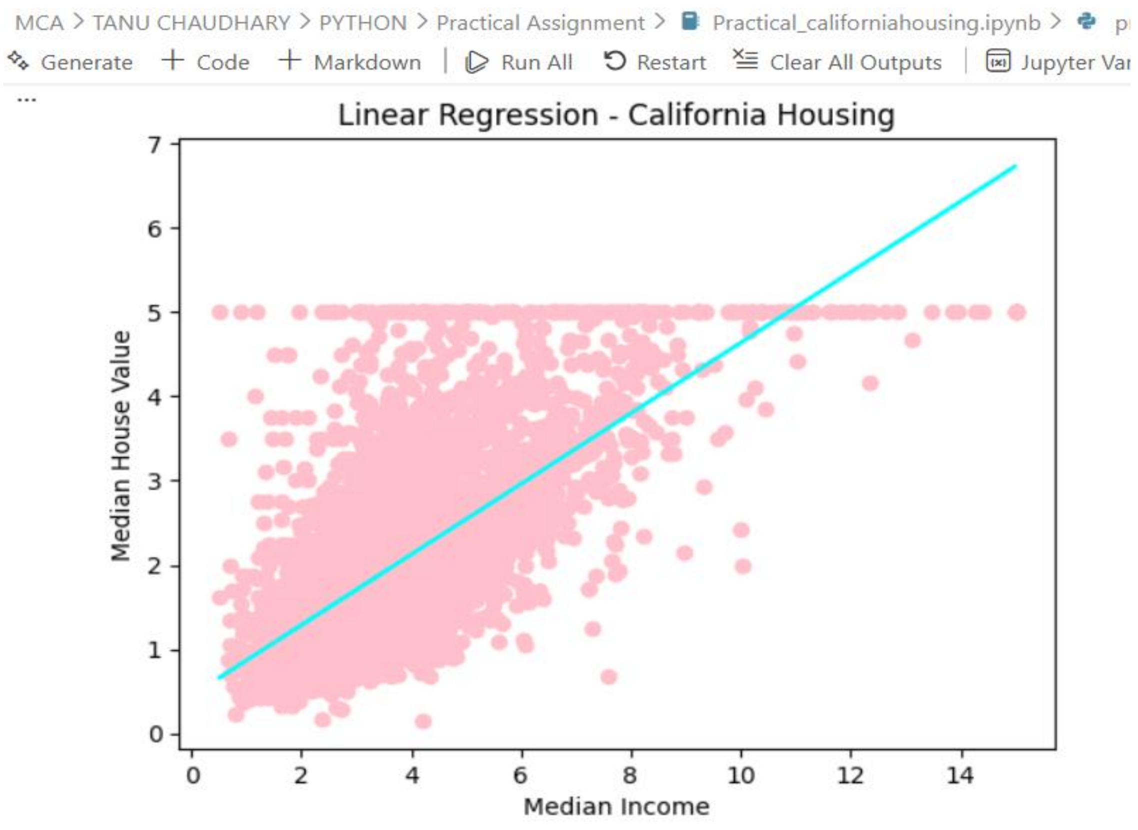
```
print("mean_absolute_error:", mean_absolute_error(y_test, y_pred))
```

[11]

```
mean_absolute_error: 0.629908653009376
```

```
plt.scatter(X_test, y_test, color='pink')
plt.plot(X_test, y_pred,color = 'cyan')
plt.xlabel("Median Income")
plt.ylabel("Median House Value")
plt.title("Linear Regression - California Housing")
plt.show()
```

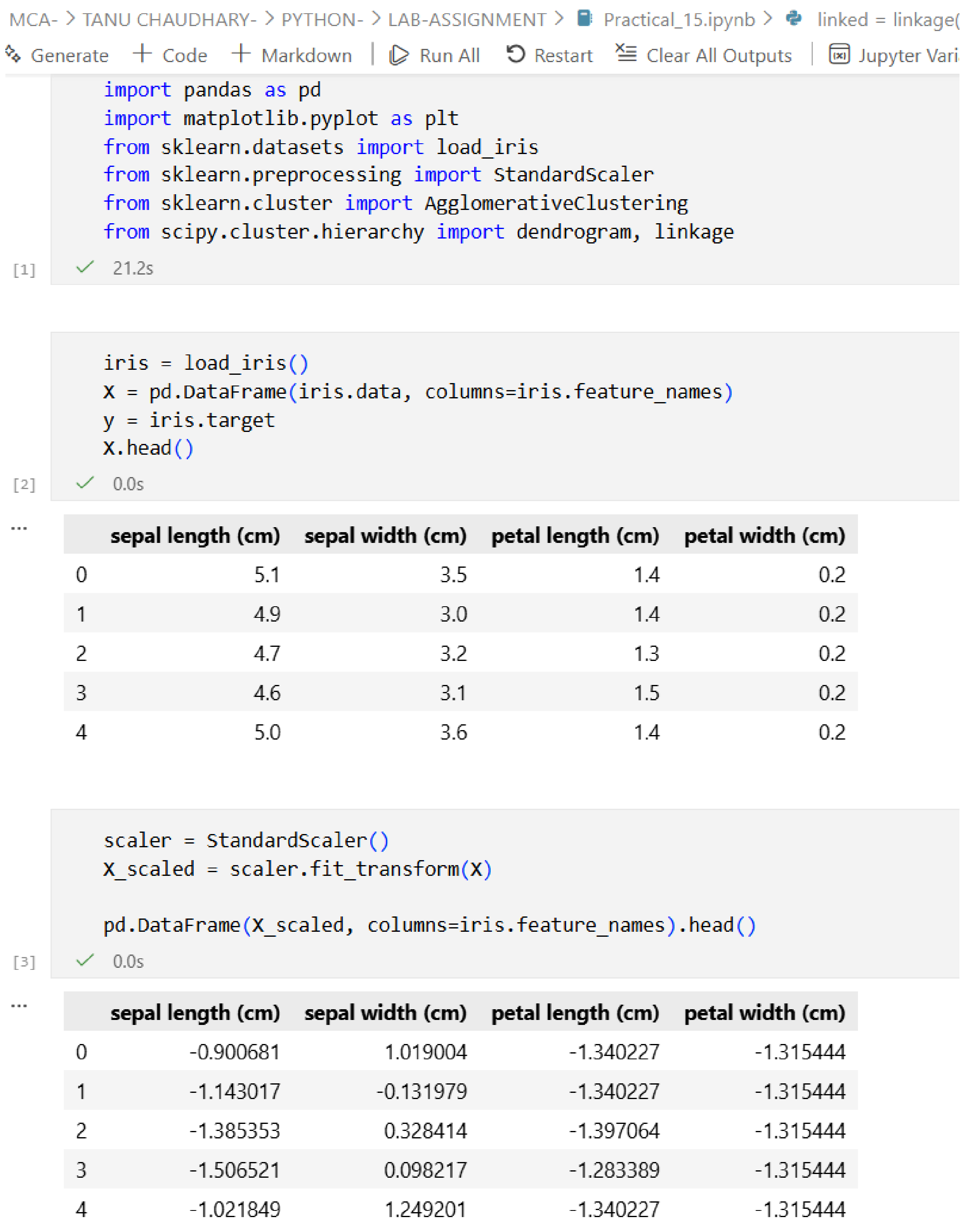
[31]



AIM 15:

Perform Agglomerative Hierarchical Clustering on the Iris dataset.

Source Code:


MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Practical_15.ipynb > linked = linkage(
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Vari

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
```

[1] ✓ 21.2s

```python
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target
X.head()
```

[2] ✓ 0.0s

	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

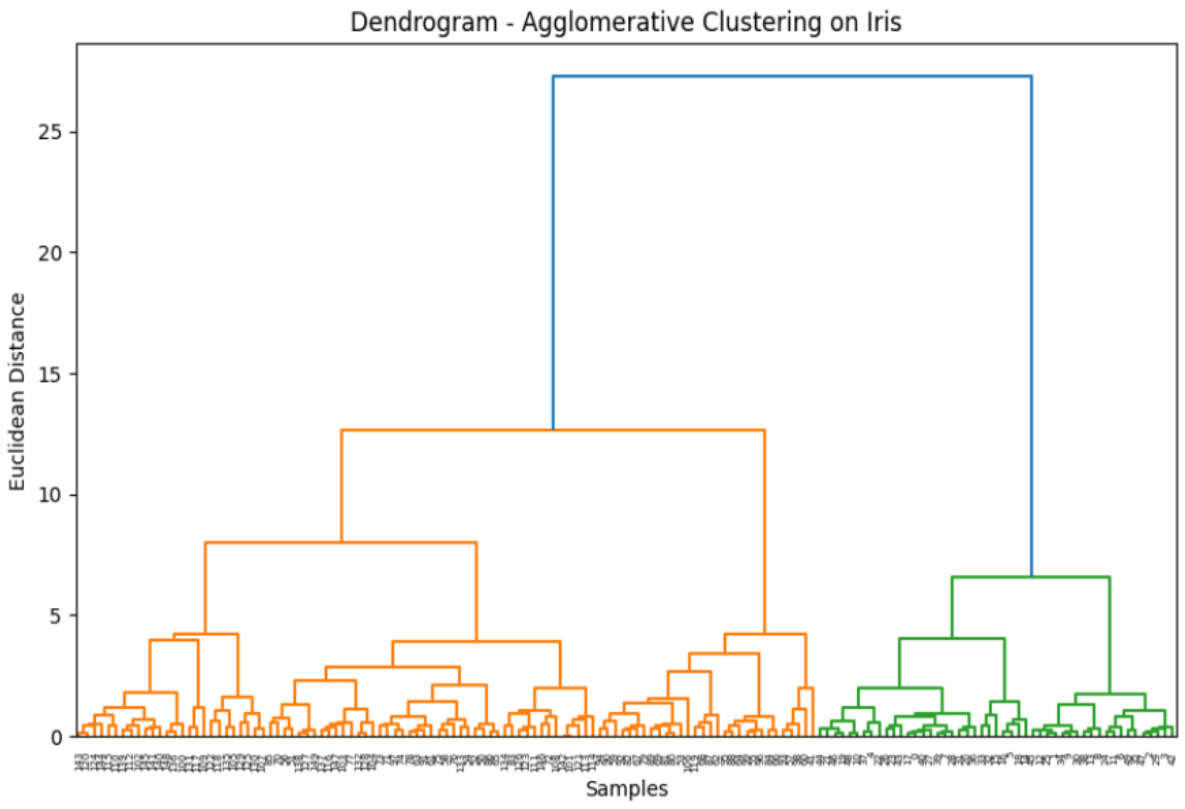
```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pd.DataFrame(X_scaled, columns=iris.feature_names).head()
```

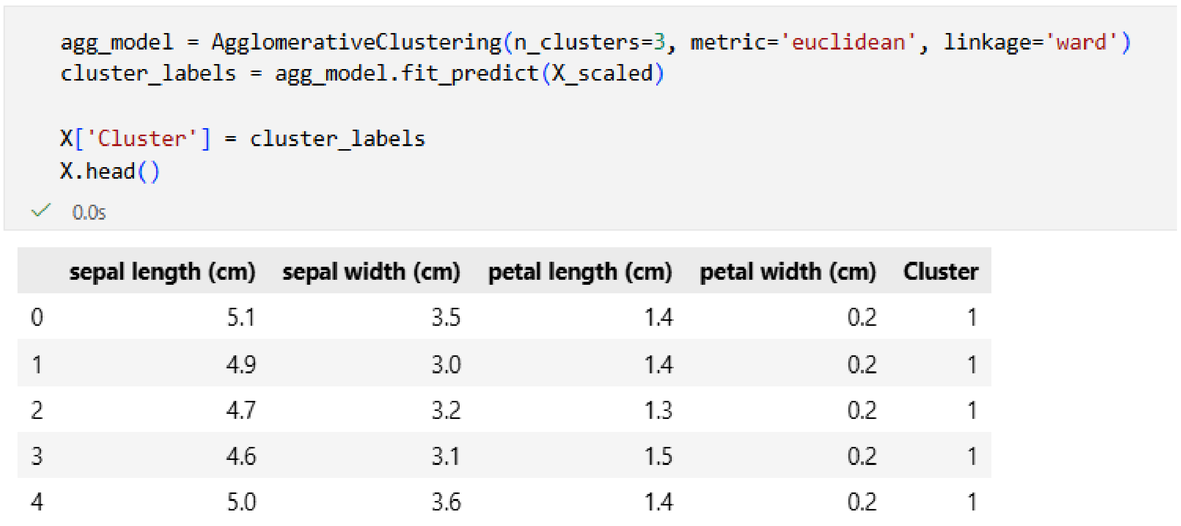
[3] ✓ 0.0s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

```
linked = linkage(X_scaled, method='ward')

plt.figure(figsize=(10,6))
dendrogram(linked,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True)
plt.title("Dendrogram - Agglomerative Clustering on Iris")
plt.xlabel("Samples")
plt.ylabel("Euclidean Distance")
plt.show()
```

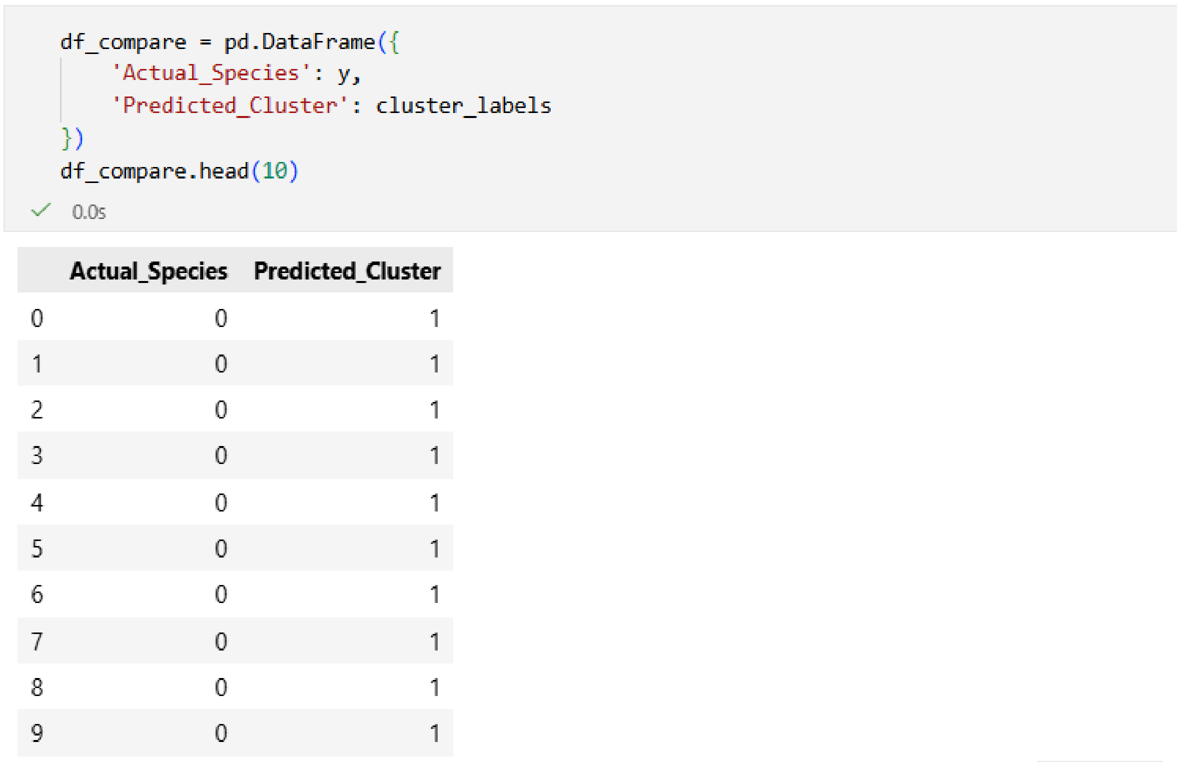
✓ 0.3s





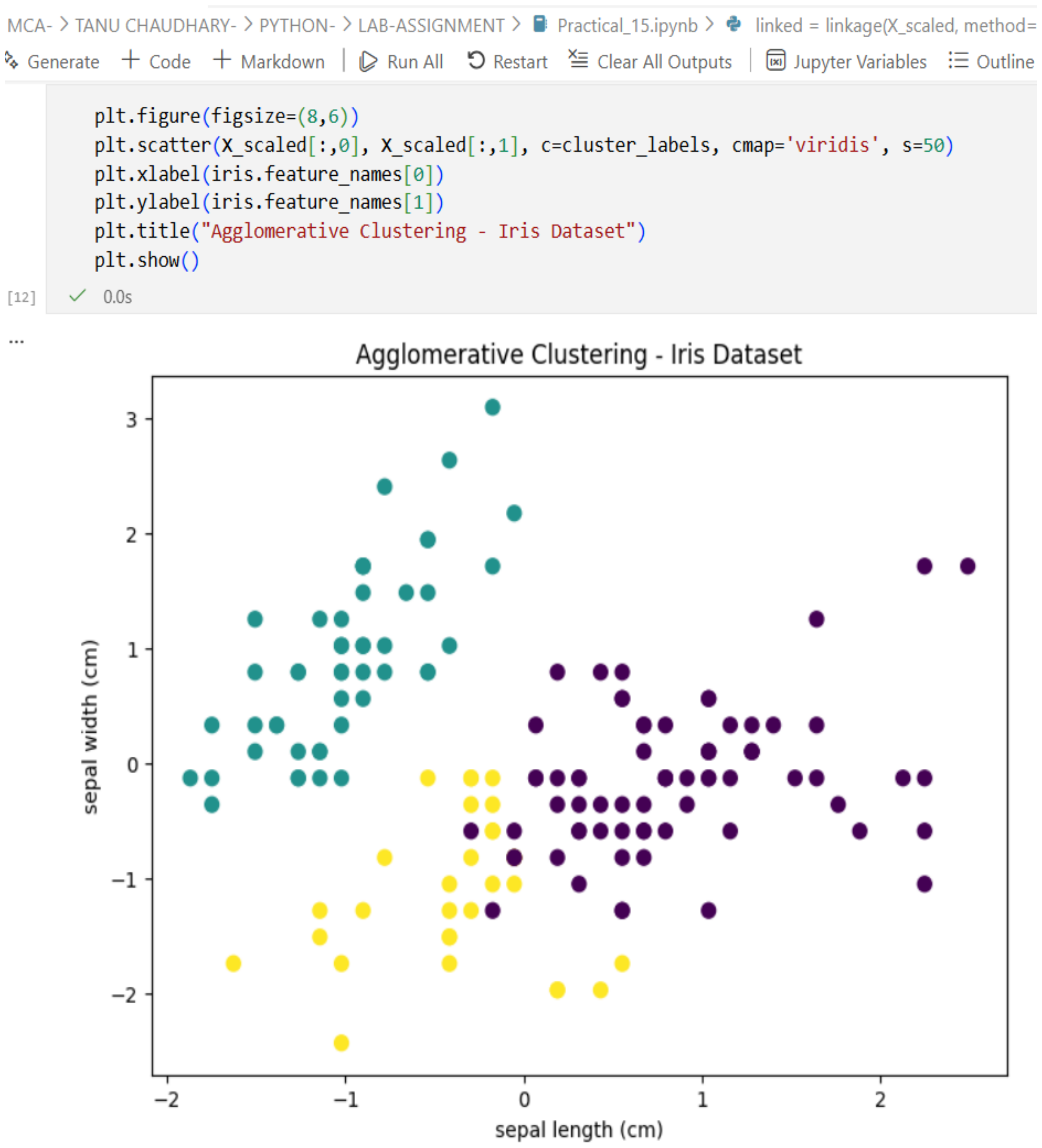
```python
agg_model = AgglomerativeClustering(n_clusters=3, metric='euclidean', linkage='ward')
cluster_labels = agg_model.fit_predict(X_scaled)

X['Cluster'] = cluster_labels
X.head()
```
✓ 0.0s

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



```python
df_compare = pd.DataFrame({
    'Actual_Species': y,
    'Predicted_Cluster': cluster_labels
})
df_compare.head(10)
```
✓ 0.0s

	Actual_Species	Predicted_Cluster
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
5	0	1
6	0	1
7	0	1
8	0	1
9	0	1


MCA- > TANU CHAUDHARY- > PYTHON- > LAB-ASSIGNMENT > Practical_15.ipynb > linked = linkage(X_scaled, method=
Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline

```
plt.figure(figsize=(8,6))
plt.scatter(X_scaled[:,0], X_scaled[:,1], c=cluster_labels, cmap='viridis', s=50)
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title("Agglomerative Clustering - Iris Dataset")
plt.show()
```

[12] ✓ 0.0s

...

Agglomerative Clustering - Iris Dataset
sepal width (cm)
sepal length (cm)
3
2
1
0
−1
−2
−2
−1
0
1
2