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 _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)		

	• ZeroDvisionError (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 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 Assigment > 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()
```

```python
# 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()

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 :-

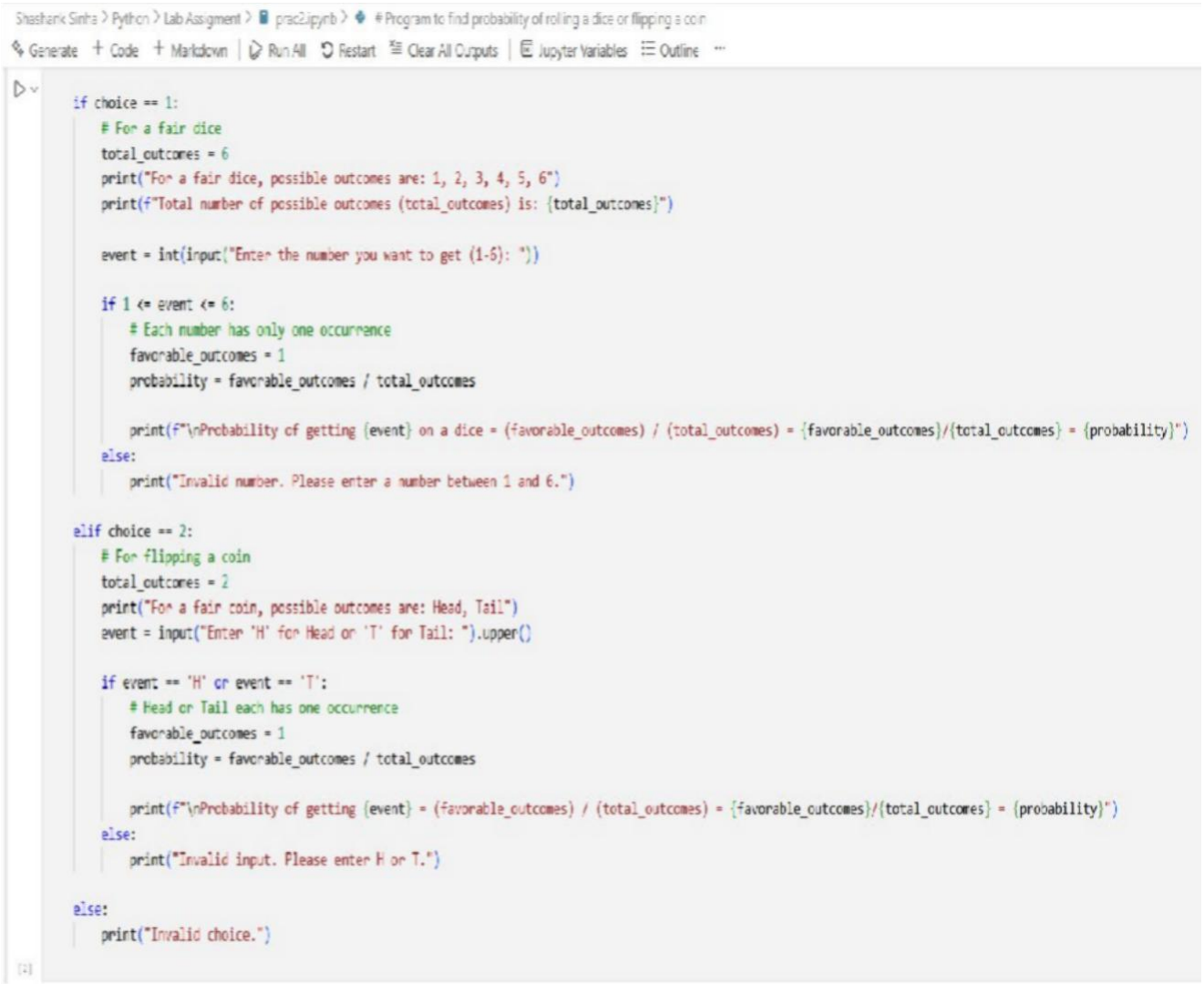
```
Pattern 1:

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 :-



Output :-

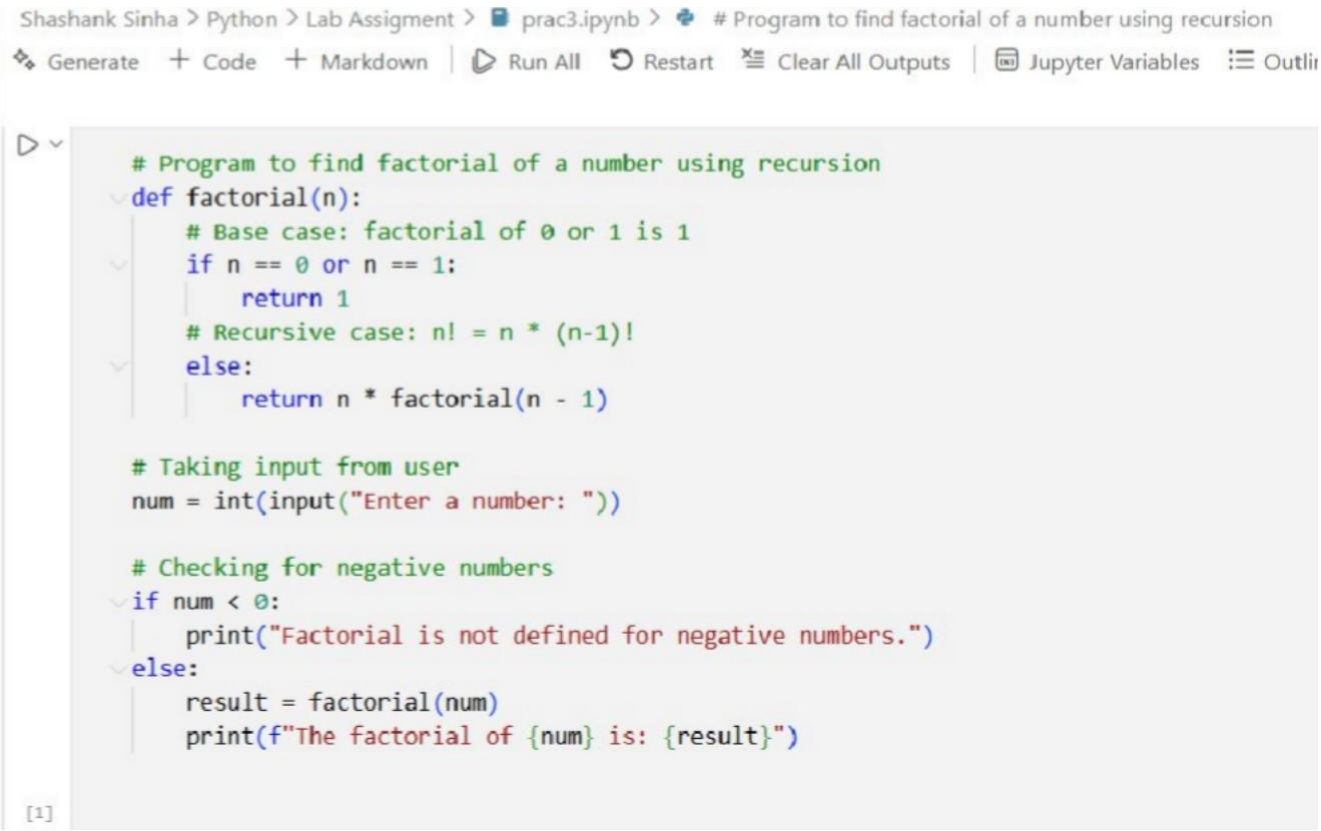
```
Choose an experiment:
1. Roll a dice
2. Flip a coin
For a fair dice, possible outcomes are: 1, 2, 3, 4, 5, 6
Total number of possible outcomes (total_outcomes) is: 6

Probability of getting 5 on a dice = (favorable_outcomes) / (total_outcomes) = 1/6 = 0.16666666666666666
```

```
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 :-



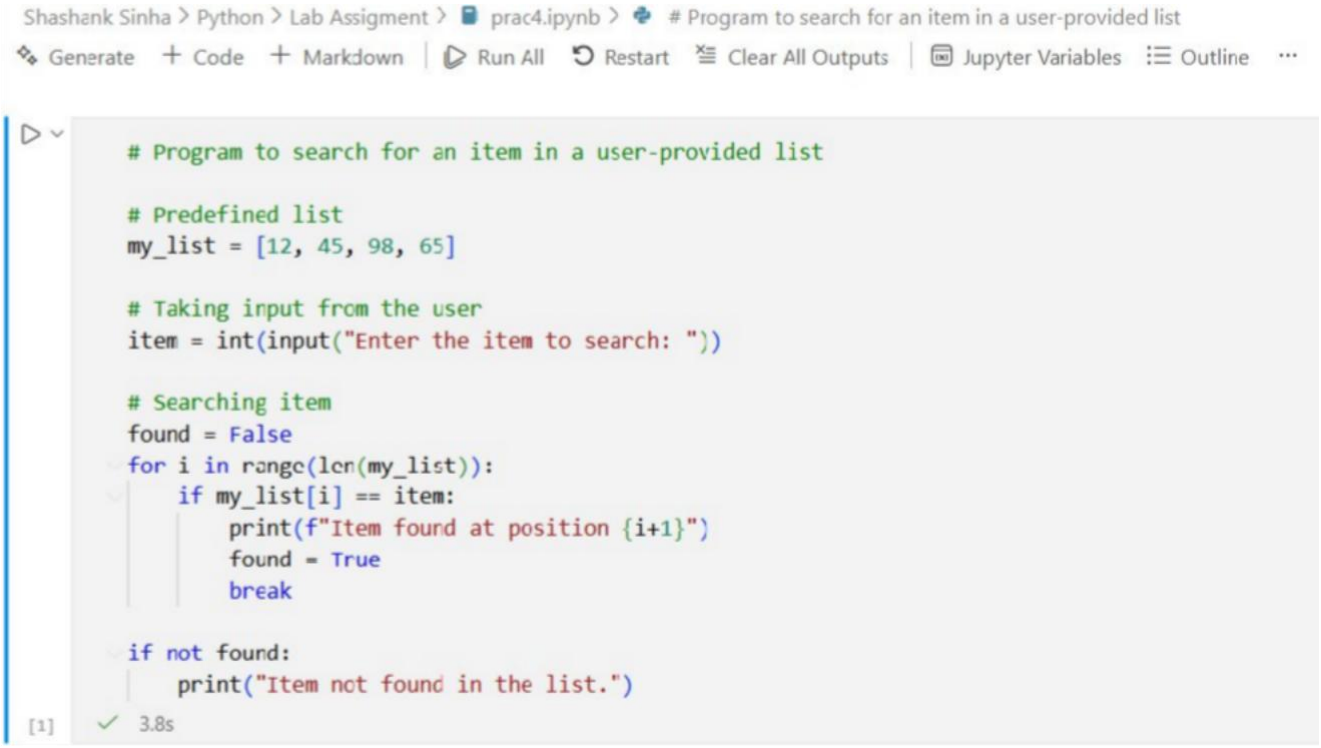
Output :-

```
...  The factorial of 5 is: 120
```

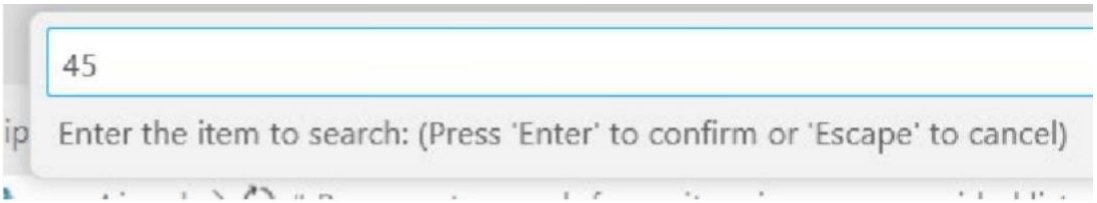
Practical-4

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 :-



Output :-




```
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:

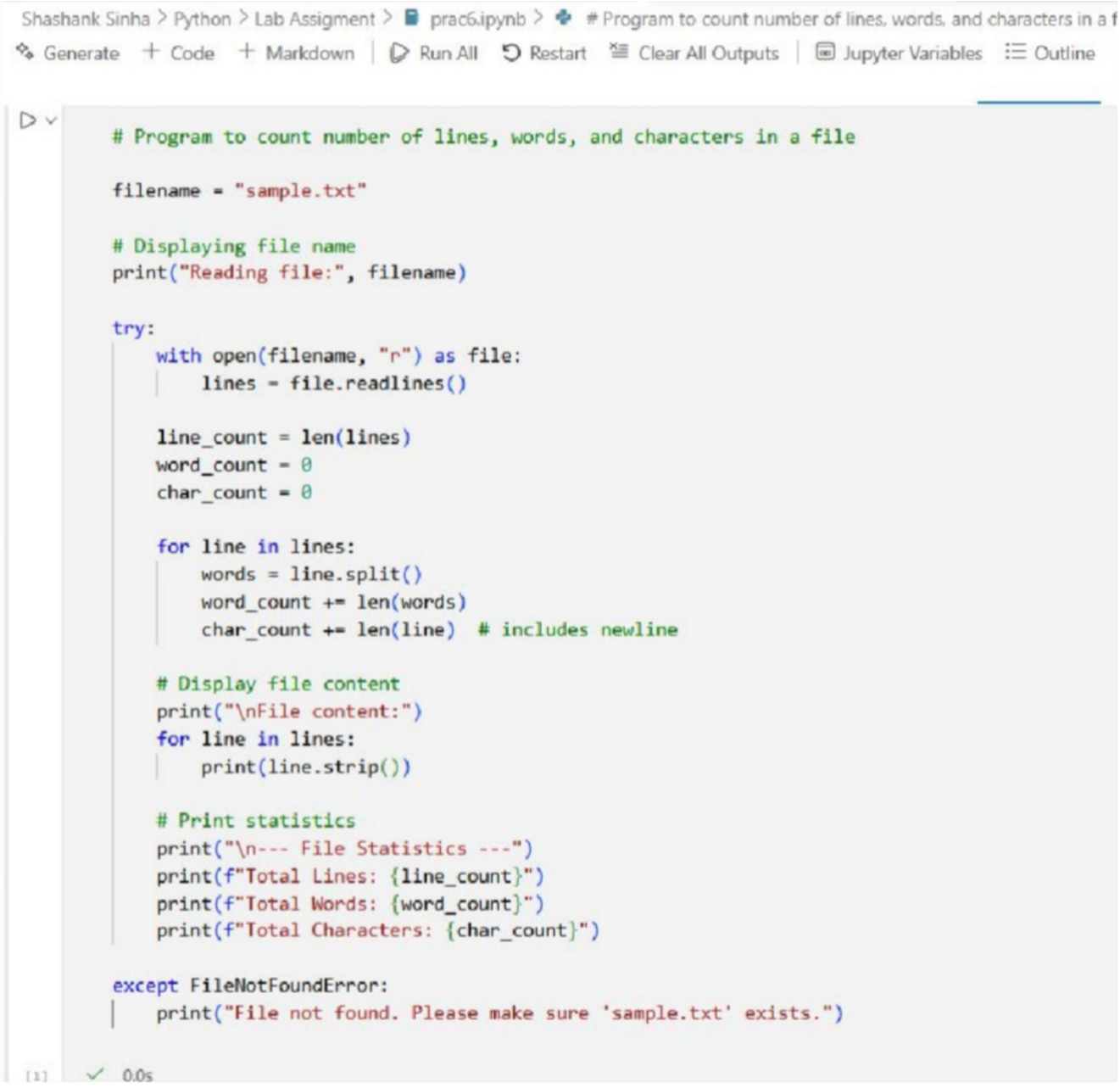
Output :-

```
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}
```

Practical-6

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

Code :-



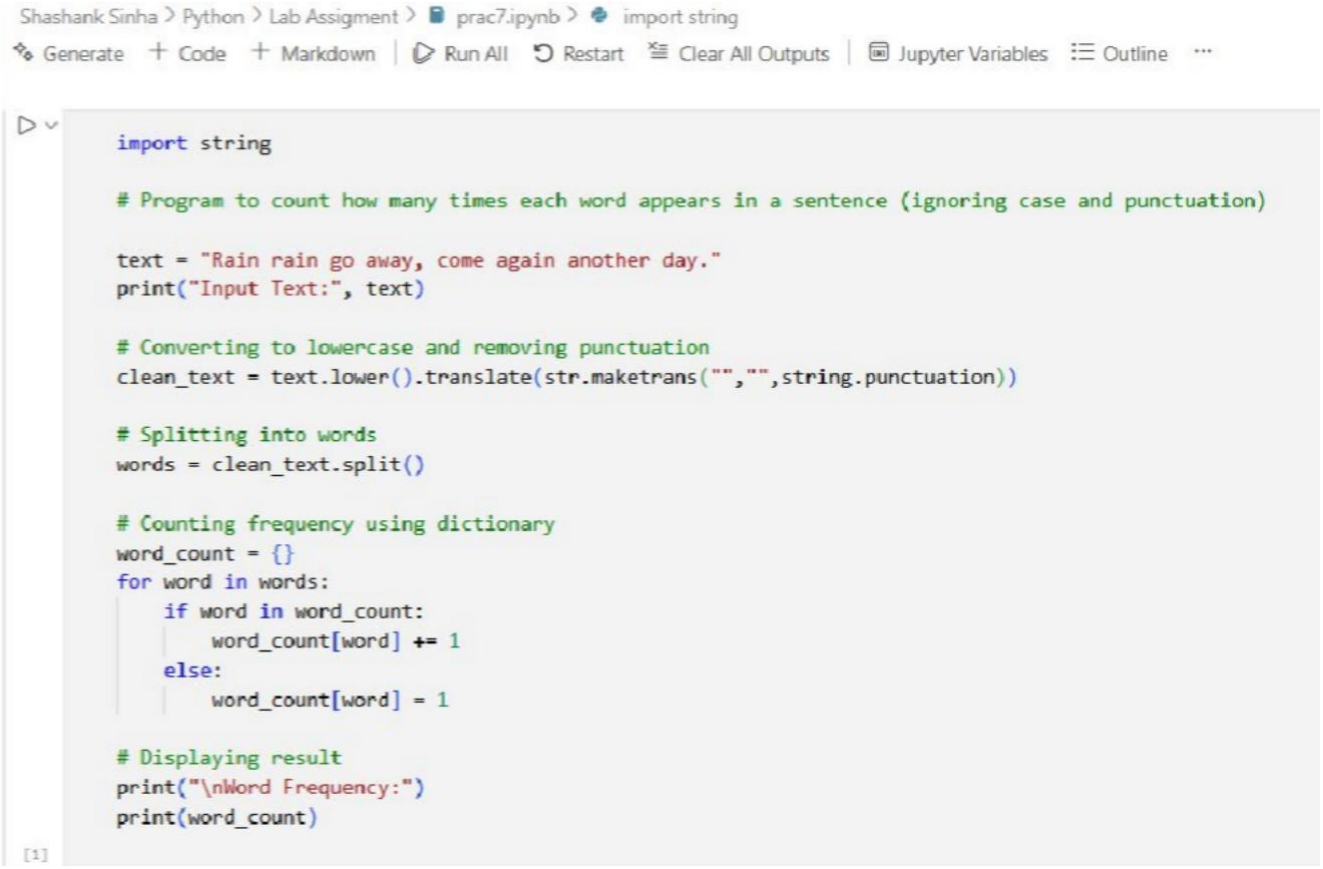
Output :-

```
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 : -




```python
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)
```

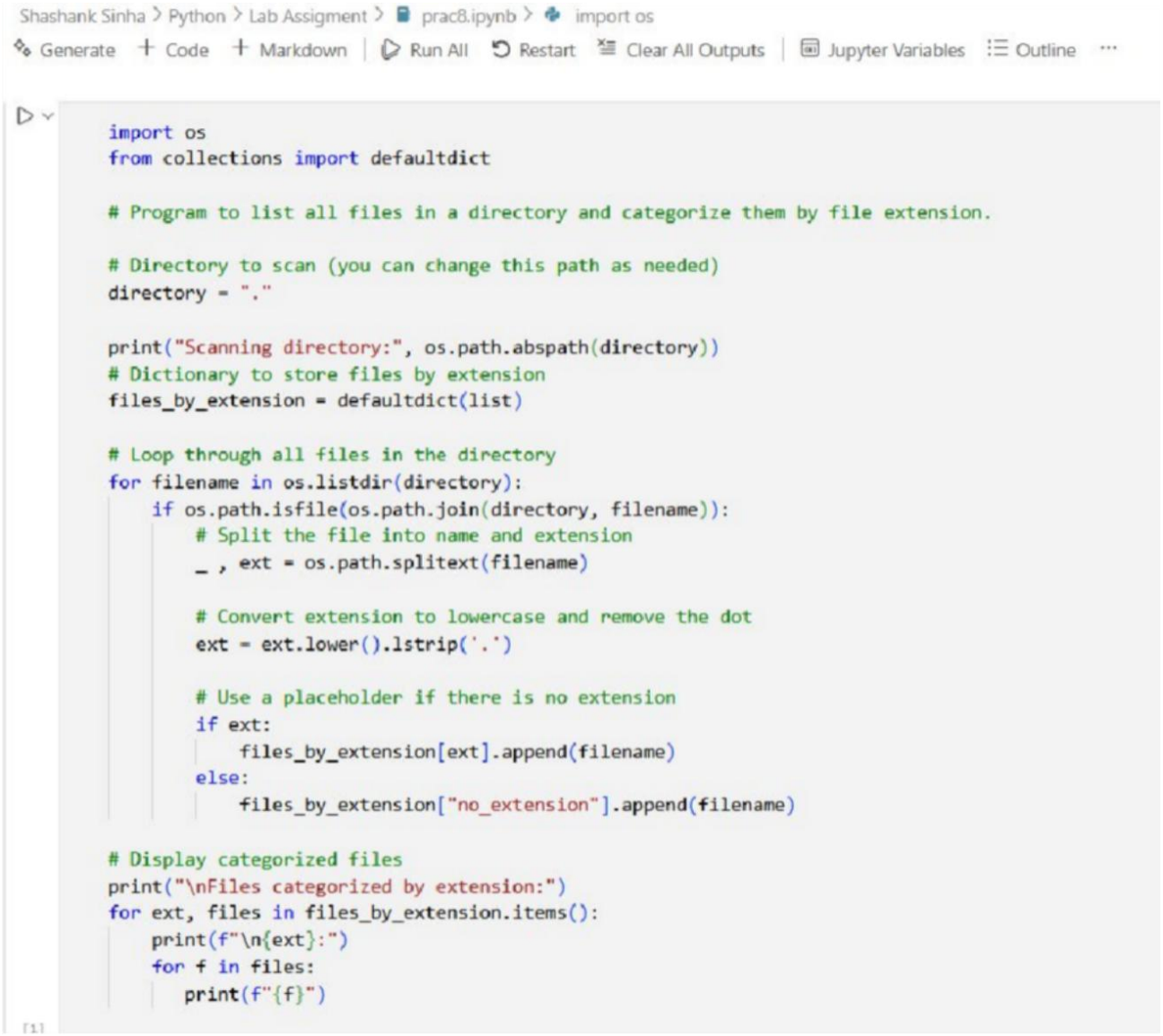
Output :-

```
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:



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 >

```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def get_coordinates(self):
        return (self.x, self.y)


```

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 Assigment > 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 Assigment > 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

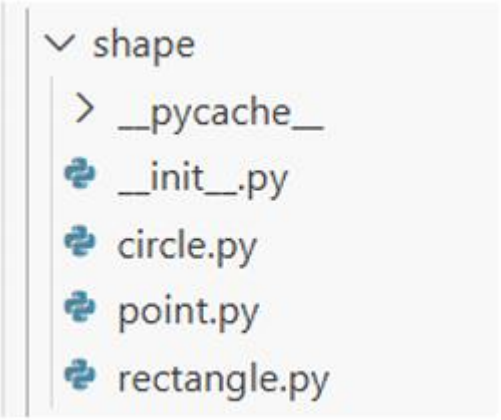
```python
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)
```

Module 4(point.py)

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

```python
class Point:
    def _init_(self, x=0, y=0):
        self.x = x
        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

```
[16]
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
```

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 Assigment > shapes > rectangle.py > Rectangle > perimeter

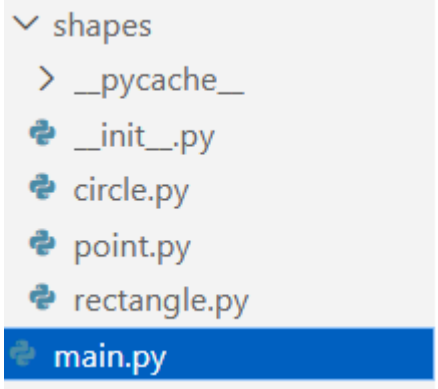
```python
class Rectangle:
    def _init_(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)
```

main.py

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

```python
# Absolute imports
from shapes.circle import Circle
from shapes.rectangle import Rectangle
from shapes.point import Point

# Create objects
p = Point(0, 0)
c = Circle(p, 5)
r = Rectangle(10, 4)

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

Directory Sample



Output:-

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

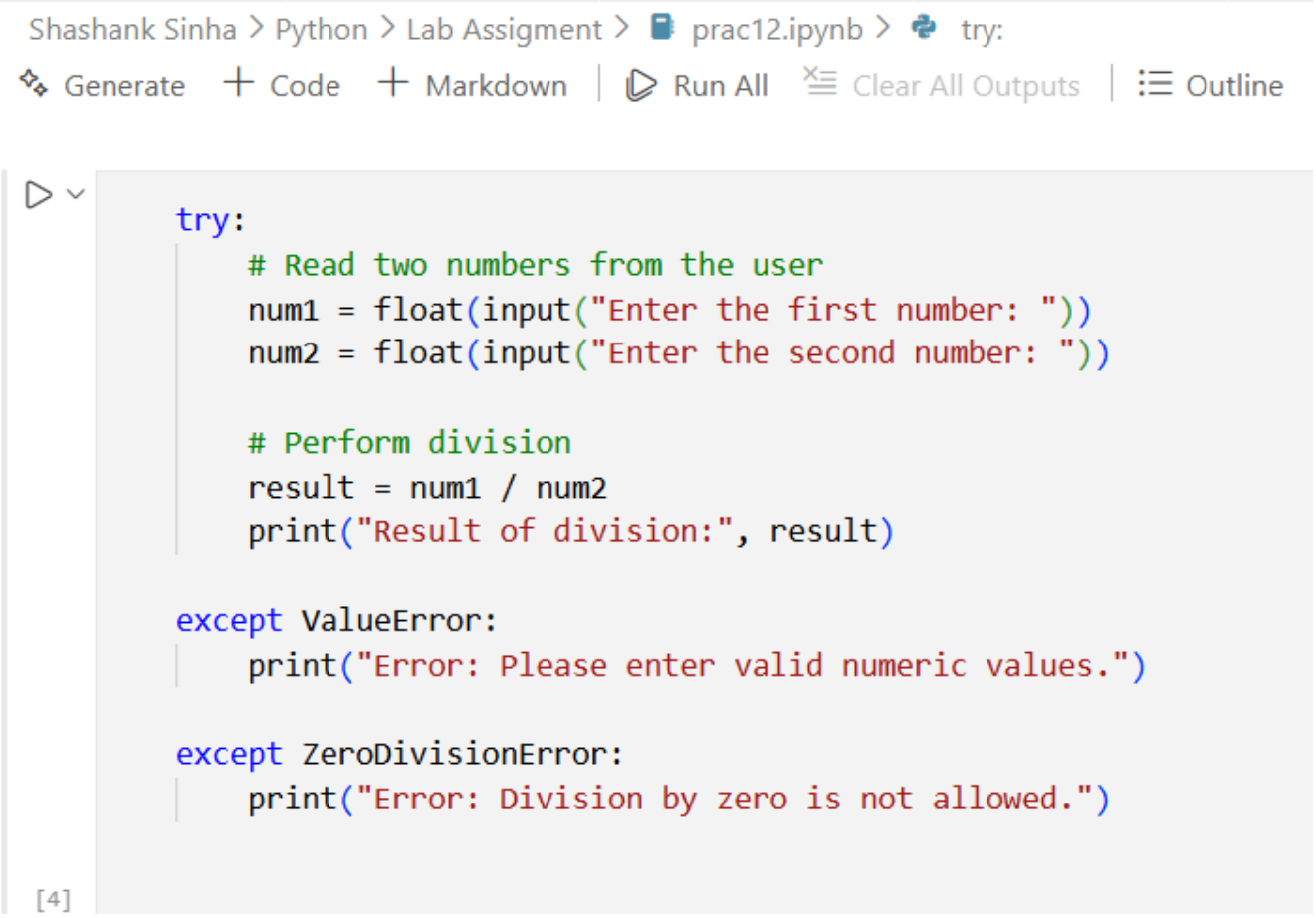
Practical-12

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

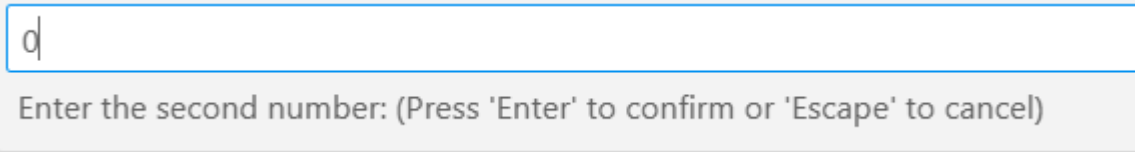
use **try-except** to handle the following exceptions:

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

Code:-



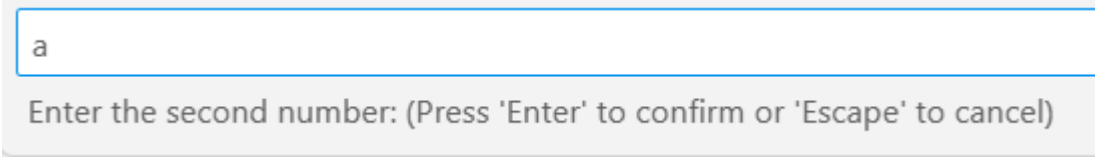
Input:-



Output:-

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

Input:-



Output:-

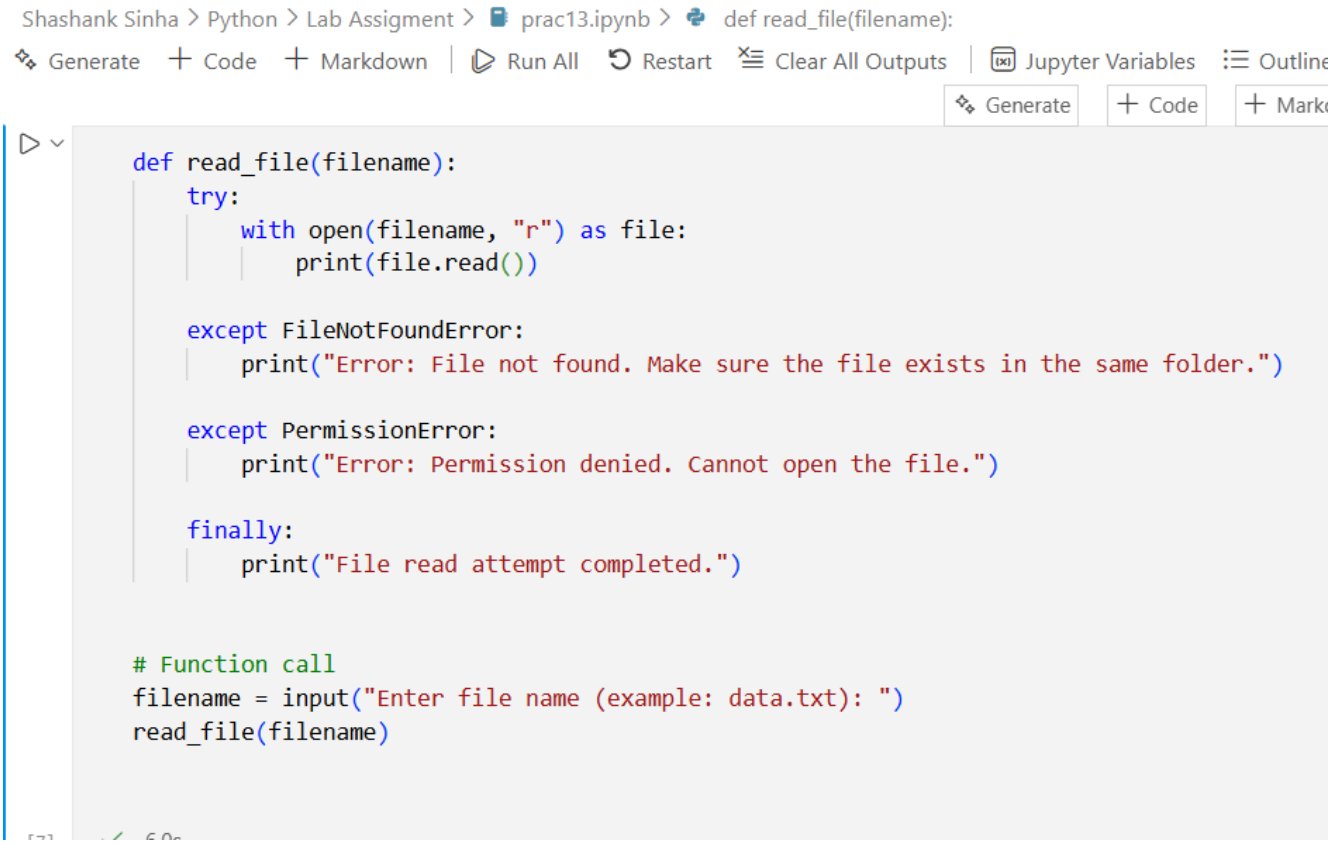
```
Error: Please enter valid numeric values.
```

Practical-13

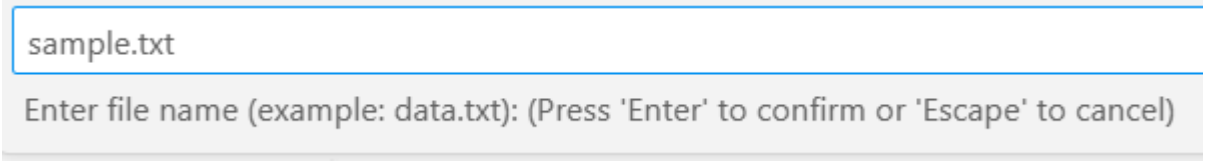
Aim:- Create a Python function **read_file(filename)** that opens and reads a text file.
Use **try-except-finally** to handle:

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

Code:-



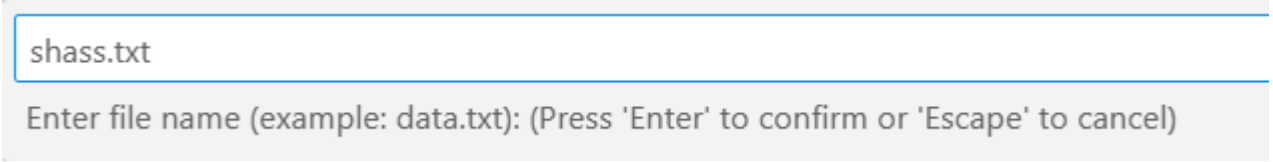
Input:-



Output:-

```
hello
File read attempt completed.
```

Input:-



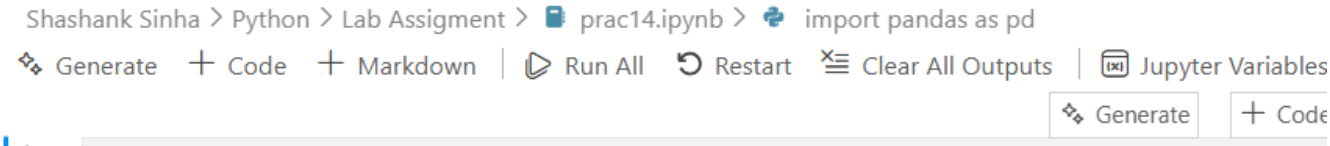
Output:-

```
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 R^2 score.

Code:-

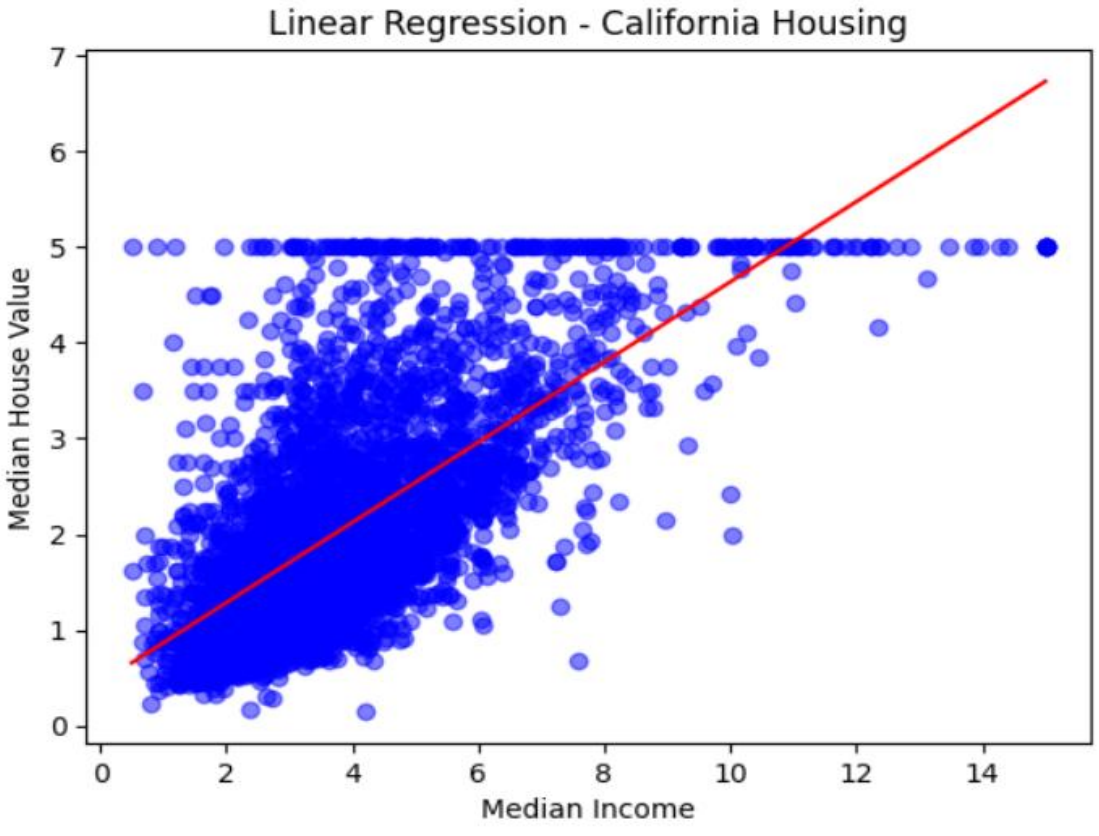



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.datasets import fetch_california_housing

# 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)
```

```python
# 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

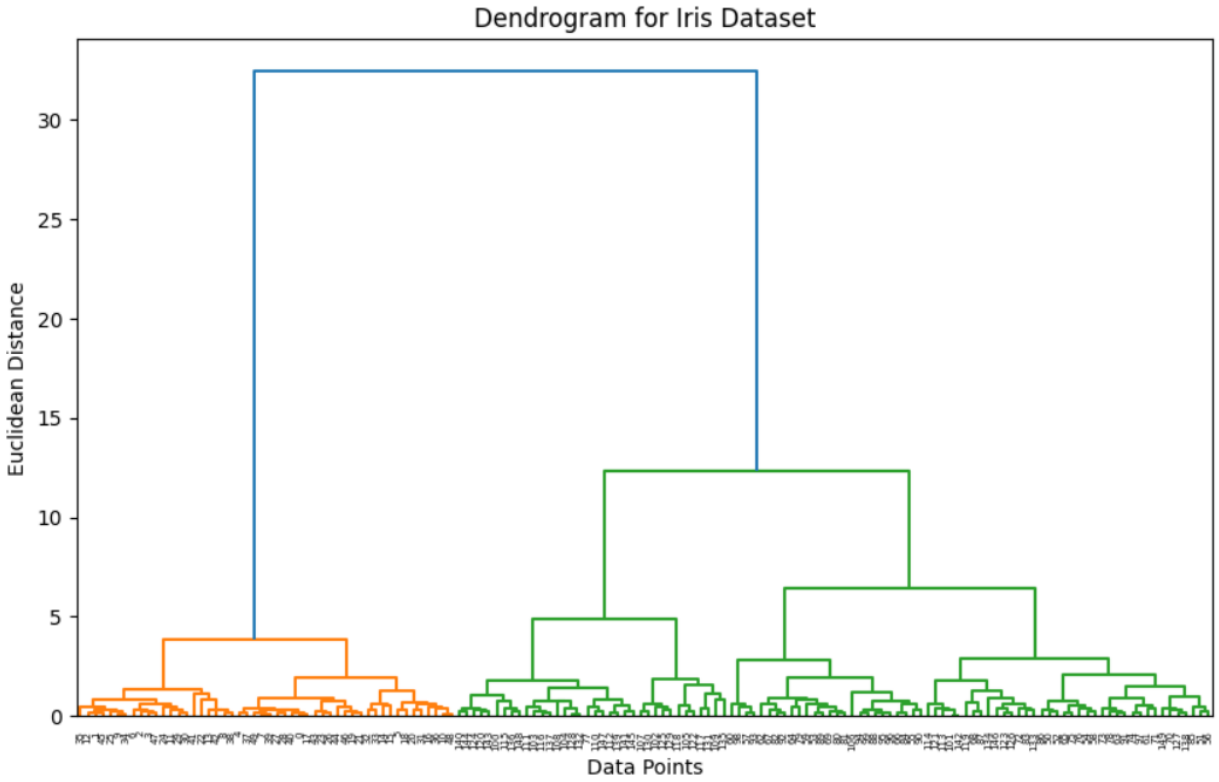
Generate | Code | Markdown | Run All | Restart | Clear All Outputs | Ju

Genera

```
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
```