A
Practical Assignment
On
Python Programming Lab Records

Master of Computer Application -I Sem

RUNGTA INTERNATIONAL SKILLS UNIVERSITY

SESSION: 2025-26

Submitted To:-
Miss Kavita Kanwar

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

RUNGTA INTERNATIONAL SKILLS
UNIVERSITY,CG
SCHOOL OF INFORMATION TECHNOLOGY

INDEX

	salary) and display the average salary by department.		
10.	Write a python program using math module to calculate: a) Square root of a number b) Factorial of a number c) Power of a number (take input from the user)		
11.	Write a python program to find the area and circumference of a circle using math module, take input from the user.		
12.	Create a list of student and select CR randomly, then shuffle the list and print it.		
13.	Write a python program to generate a random password of 12 characters using UPPERCASE, lowercase, digits and special symbols(!, @, #, $, *). Password should start with Capital letter or digits.		
14.	Create a package named shapes that contains two modules: circle.py and rectangle.py. Each module should compute area and		

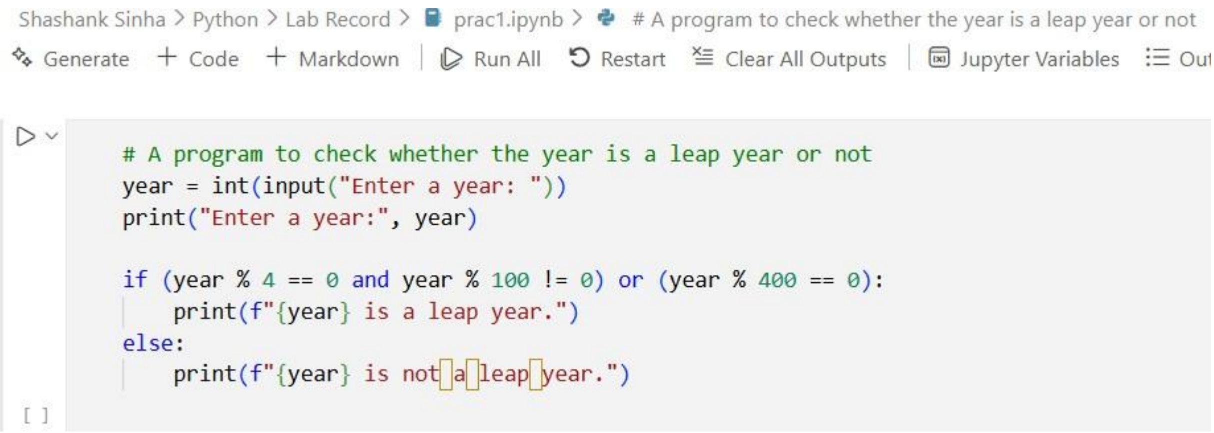
	perimeter of the shape. Write the complete directory structure and sample code.		
15.	Write a program that asks the user for an index and prints the element at that index from a predefined list. Handle: - IndexError (index out of range) - TypeError (if user enters a non-integer index) - Print the specific exception message using except Exception as e.		
16.	Implement a supervised learning model to classify flowers in the Iris dataset using decision tree classifier. Print the accuracy of the model.		
17.	Use Support Vector Machine (SVM) on the breast cancer datasets to classify malignant vs benign tumors.		
18.	Perform Principal Component Analysis (PCA) on		

	the digits dataset and reduce the dimension to 2. Print the explained variance ratio.		
19.	Apply K-Means clustering on the Iris Dataset and print cluster label		
20.	Write a Python program to perform basic EDA on a student performance dataset to understand its structure and summary statistics.		
21.	Write a Python program using pandas to analyze a sales dataset and extract meaningful insights.		
22.	Write a Python program to visualize an employee dataset using matplotlib and seaborn to understand data distribution and relationships.		

Practical-1

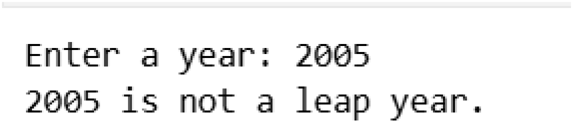
Aim: Write a program to check whether the year is leap or not

Code :-


Shashank Sinha > Python > Lab Record > prac1.ipynb > # A program to check whether the year is a leap year or not
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Out
[ ]

```python
# A program to check whether the year is a leap year or not
year = int(input("Enter a year: "))
print("Enter a year:", year)

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Output :-

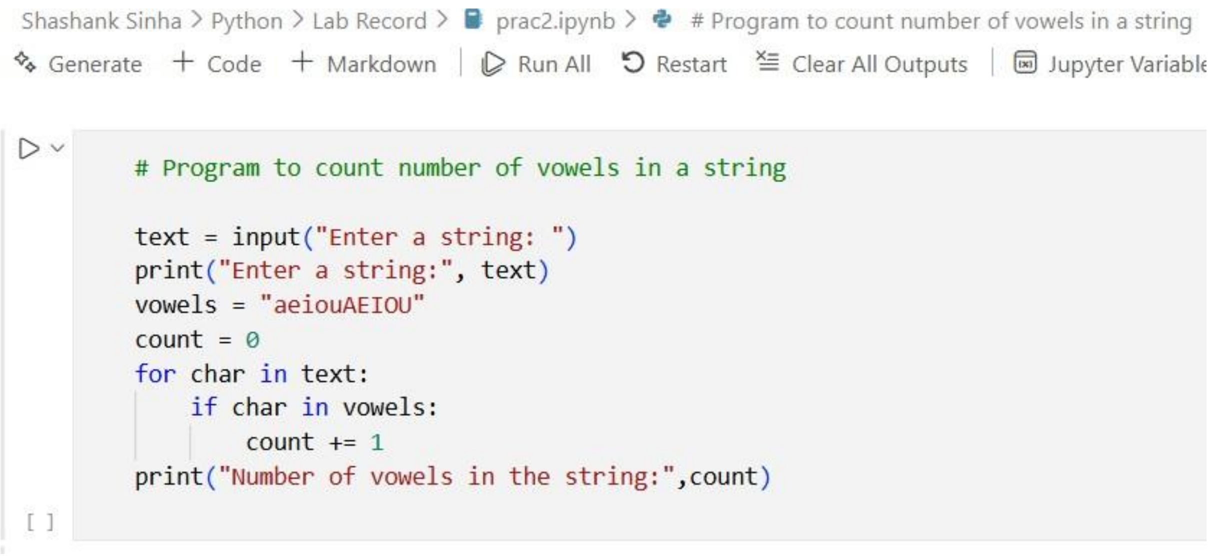


```
Enter a year: 2005
2005 is not a leap year.
```

Practical-2

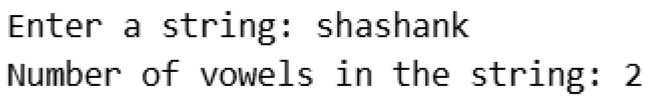
Aim: Write a program to count no. of vowels in a string.

Code :-


Shashank Sinha > Python > Lab Record > prac2.ipynb > # Program to count number of vowels in a string
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variable
[ ]

```python
# Program to count number of vowels in a string

text = input("Enter a string: ")
print("Enter a string:", text)
vowels = "aeiouAEIOU"
count = 0
for char in text:
    if char in vowels:
        count += 1
print("Number of vowels in the string:",count)
```

Output :-

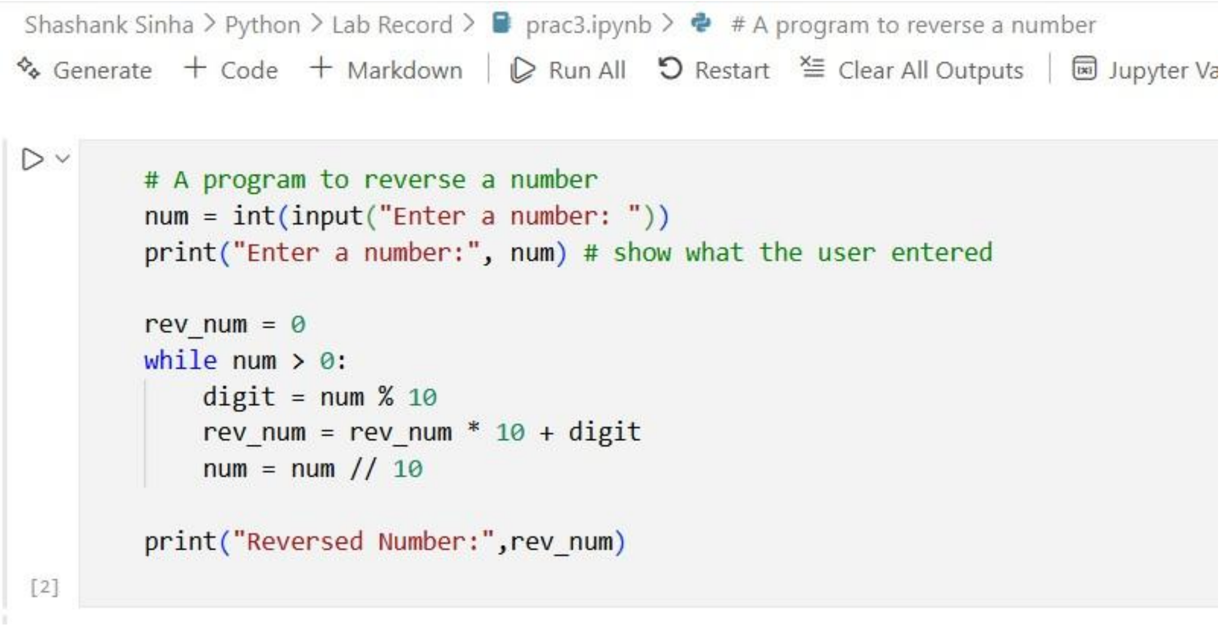


```
Enter a string: shashank
Number of vowels in the string: 2
```

Practical-3

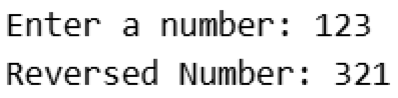
Aim: Write a program to reverse a number.

Code :-


Shashank Sinha > Python > Lab Record > prac3.ipynb > # A program to reverse a number
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Va
[2]

```python
# A program to reverse a number
num = int(input("Enter a number: "))
print("Enter a number:", num) # show what the user entered

rev_num = 0
while num > 0:
    digit = num % 10
    rev_num = rev_num * 10 + digit
    num = num // 10

print("Reversed Number:",rev_num)
```

Output :-

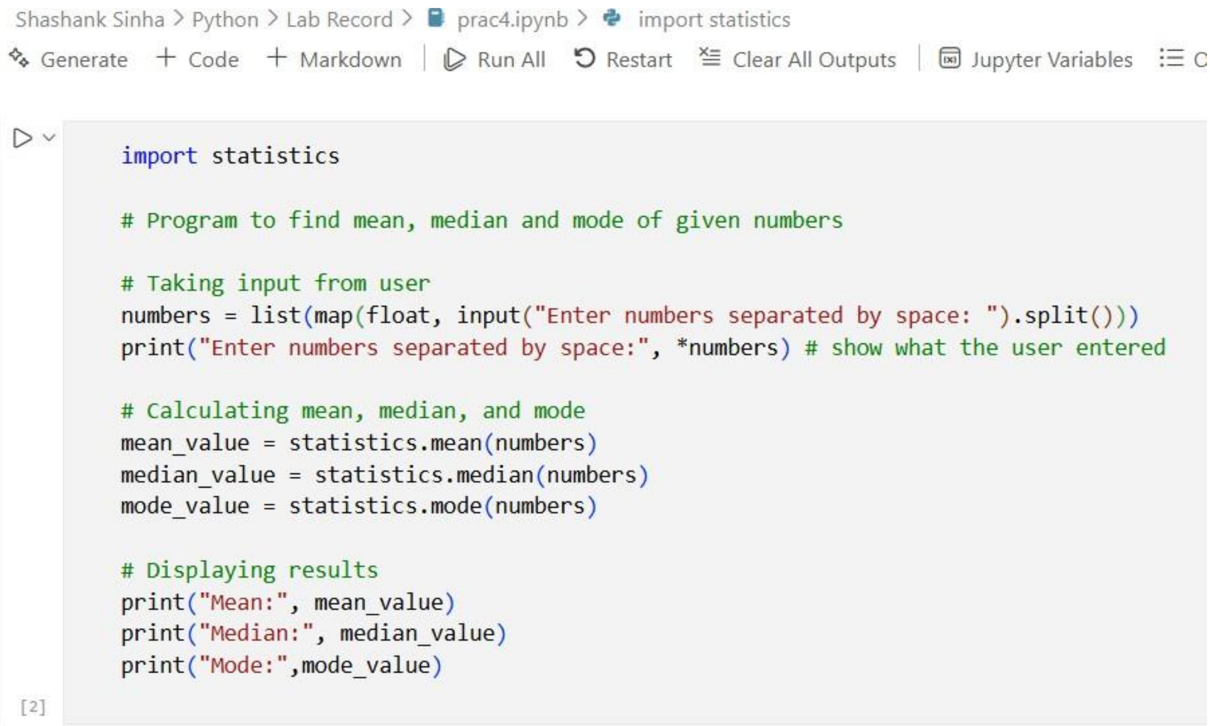


```
Enter a number: 123
Reversed Number: 321
```

Practical-4

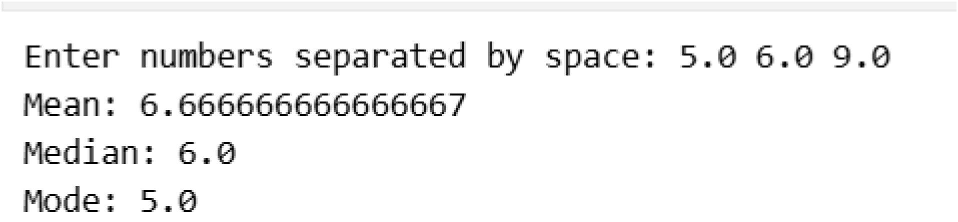
Aim: Write a program to find mean, median and mode of given number.

Code :-



```
import statistics

# Program to find mean, median and mode of given numbers

# Taking input from user
numbers = list(map(float, input("Enter numbers separated by space: ").split()))
print("Enter numbers separated by space:", *numbers) # show what the user entered

# Calculating mean, median, and mode
mean_value = statistics.mean(numbers)
median_value = statistics.median(numbers)
mode_value = statistics.mode(numbers)

# Displaying results
print("Mean:", mean_value)
print("Median:", median_value)
print("Mode:",mode_value)
```

Output :-

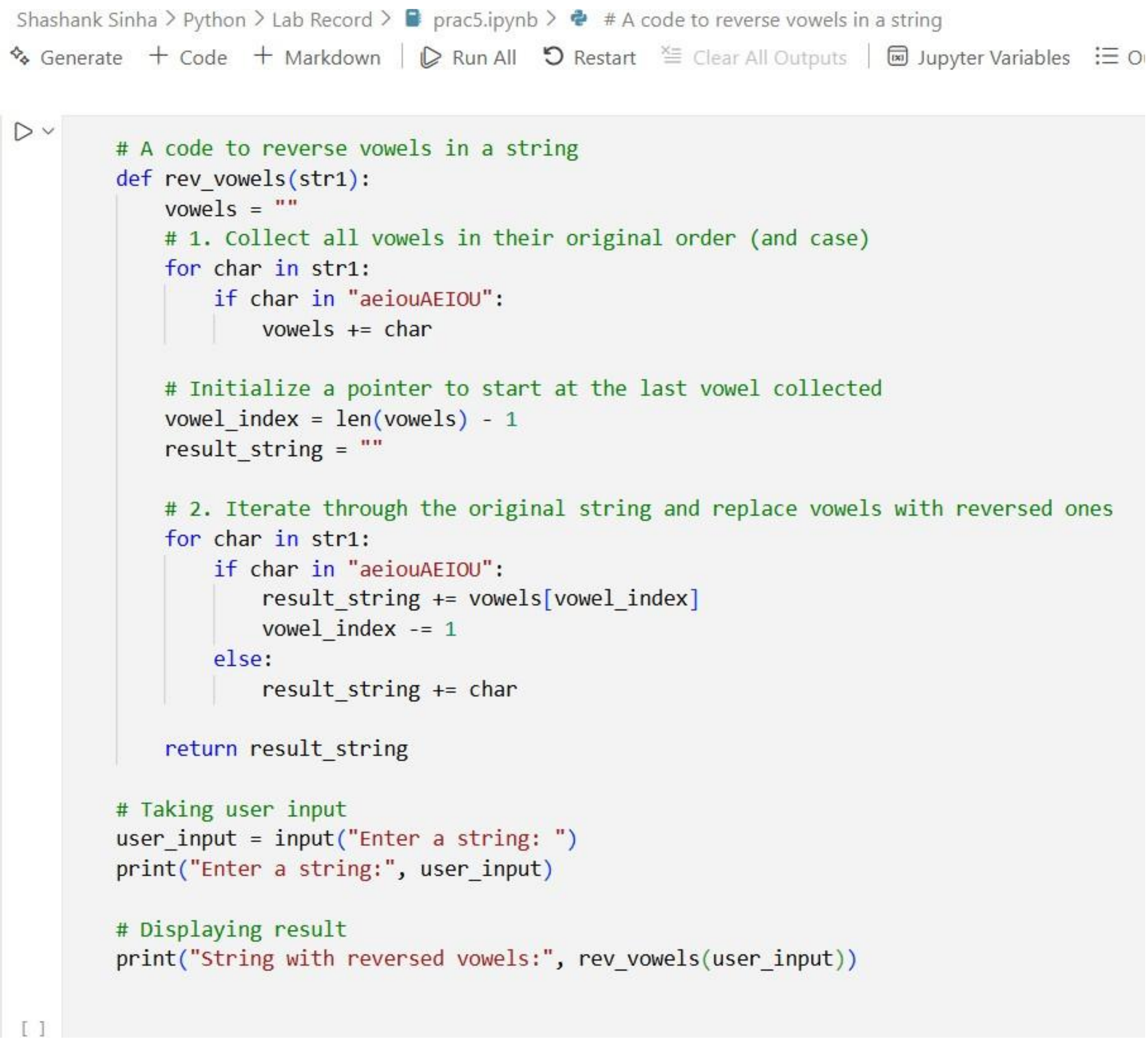


```
Enter numbers separated by space: 5.0 6.0 9.0
Mean: 6.666666666666667
Median: 6.0
Mode: 5.0
```

Practical-5

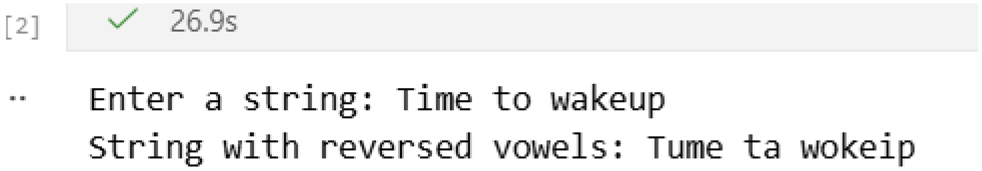
Aim: Write a Python program to reverse only the vowels in a given string, keeping other characters in their original positions.

Code :-


Shashank Sinha > Python > Lab Record > prac5.ipynb > # A code to reverse vowels in a string
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
O
[ ]

```python
# A code to reverse vowels in a string
def rev_vowels(str1):
    vowels = ""
    # 1. Collect all vowels in their original order (and case)
    for char in str1:
        if char in "aeiouAEIOU":
            vowels += char

    # Initialize a pointer to start at the last vowel collected
    vowel_index = len(vowels) - 1
    result_string = ""

    # 2. Iterate through the original string and replace vowels with reversed ones
    for char in str1:
        if char in "aeiouAEIOU":
            result_string += vowels[vowel_index]
            vowel_index -= 1
        else:
            result_string += char

    return result_string

# Taking user input
user_input = input("Enter a string: ")
print("Enter a string:", user_input)

# Displaying result
print("String with reversed vowels:", rev_vowels(user_input))
```

Output :-

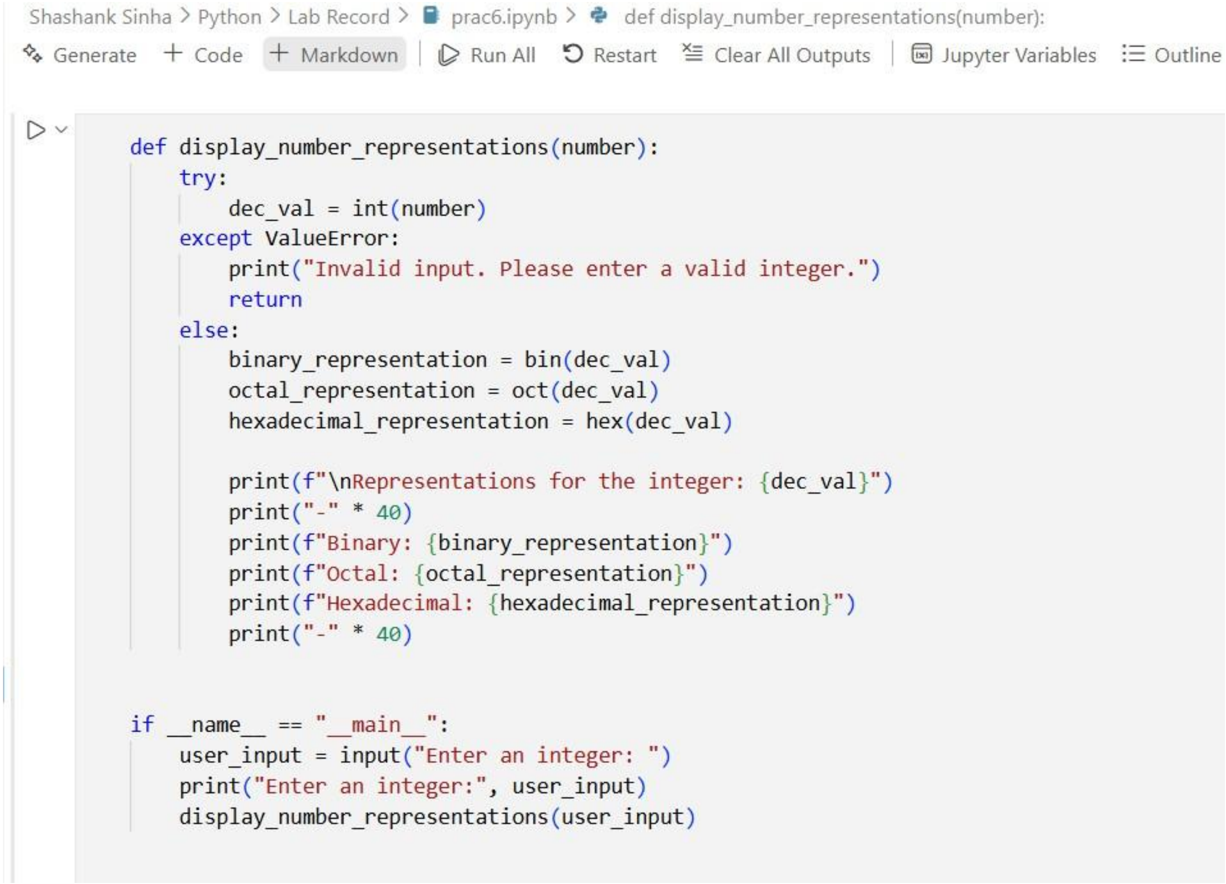


```
Enter a string: Time to wakeup
String with reversed vowels: Tume ta wokeip
```

Practical-6

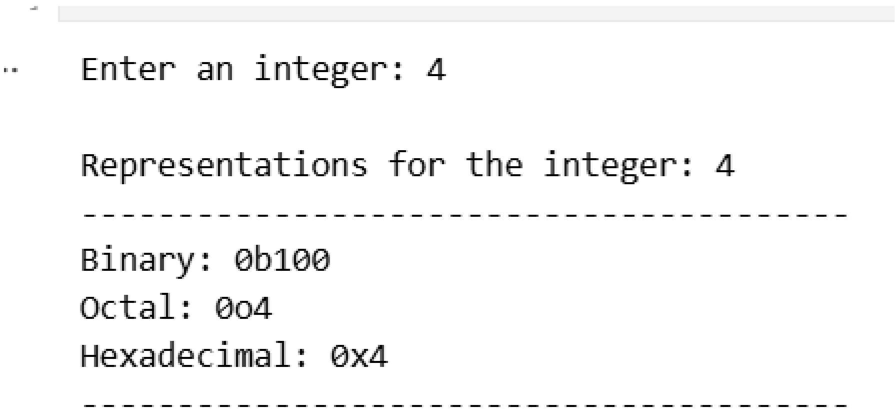
Aim: Create a script that takes an integer and displays its binary, octal and hexadecimal representations neatly formatted.

Code :-


Shashank Sinha > Python > Lab Record > prac6.ipynb > def display_number_representations(number):
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline

```python
def display_number_representations(number):
    try:
        dec_val = int(number)
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        return
    else:
        binary_representation = bin(dec_val)
        octal_representation = oct(dec_val)
        hexadecimal_representation = hex(dec_val)

        print(f"\nRepresentations for the integer: {dec_val}")
        print("-" * 40)
        print(f"Binary: {binary_representation}")
        print(f"Octal: {octal_representation}")
        print(f"Hexadecimal: {hexadecimal_representation}")
        print("-" * 40)


if __name__ == "__main__":
    user_input = input("Enter an integer: ")
    print("Enter an integer:", user_input)
    display_number_representations(user_input)
```

Output :-

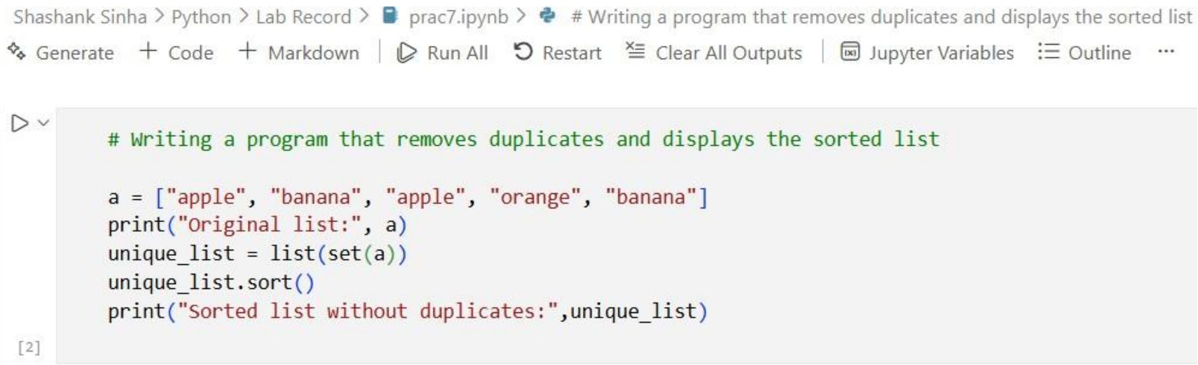


```
Enter an integer: 4

Representations for the integer: 4
----------------------------------------
Binary: 0b100
Octal: 0o4
Hexadecimal: 0x4
----------------------------------------
```

Practical-7

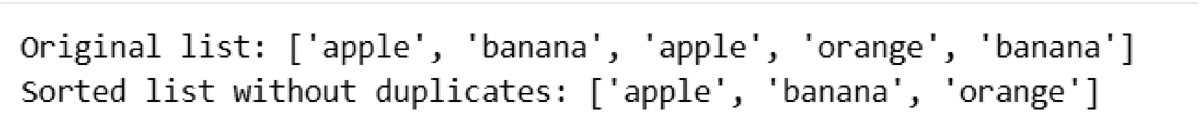
Aim: Given a list of items (possibly with duplicates), write a program that removes duplicates and displays that sorted list.

Code :-


Shashank Sinha > Python > Lab Record > prac7.ipynb > # Writing a program that removes duplicates and displays the sorted list
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
[2]

```python
# Writing a program that removes duplicates and displays the sorted list

a = ["apple", "banana", "apple", "orange", "banana"]
print("Original list:", a)
unique_list = list(set(a))
unique_list.sort()
print("Sorted list without duplicates:",unique_list)
```

Output :-

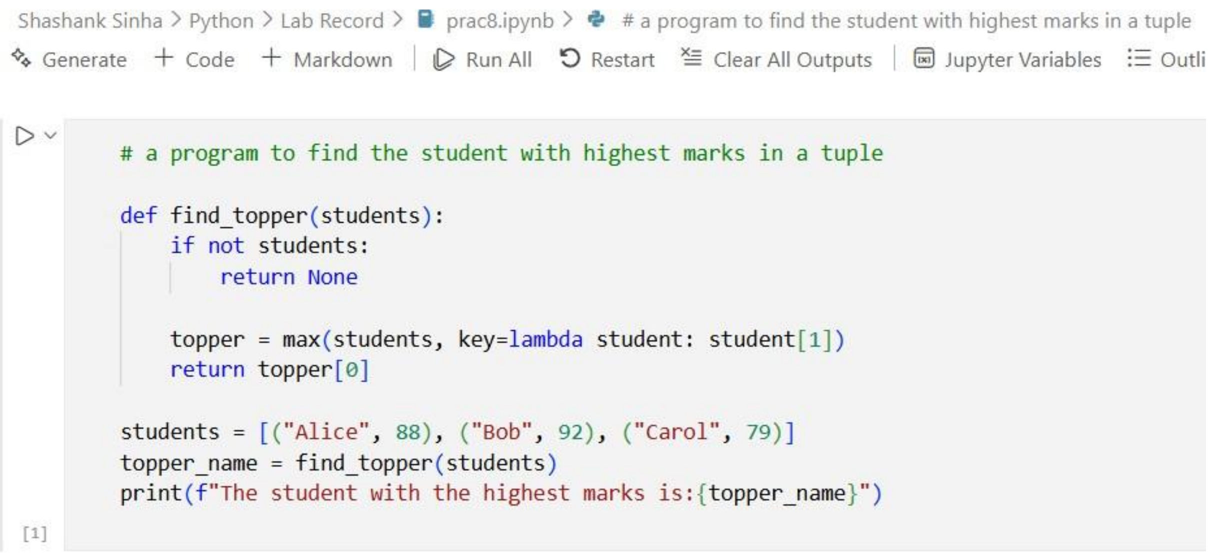


```
Original list: ['apple', 'banana', 'apple', 'orange', 'banana']
Sorted list without duplicates: ['apple', 'banana', 'orange']
```

Practical-8

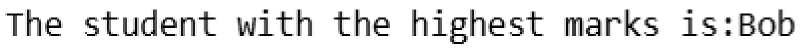
Aim: Accept a list of students and their marks as tuples. Display the name of the student with the highest marks.

Code :-


Shashank Sinha > Python > Lab Record > prac8.ipynb > # a program to find the student with highest marks in a tuple
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outli
[1]

```python
# a program to find the student with highest marks in a tuple

def find_topper(students):
    if not students:
        return None

    topper = max(students, key=lambda student: student[1])
    return topper[0]

students = [("Alice", 88), ("Bob", 92), ("Carol", 79)]
topper_name = find_topper(students)
print(f"The student with the highest marks is:{topper_name}")
```

Output :-

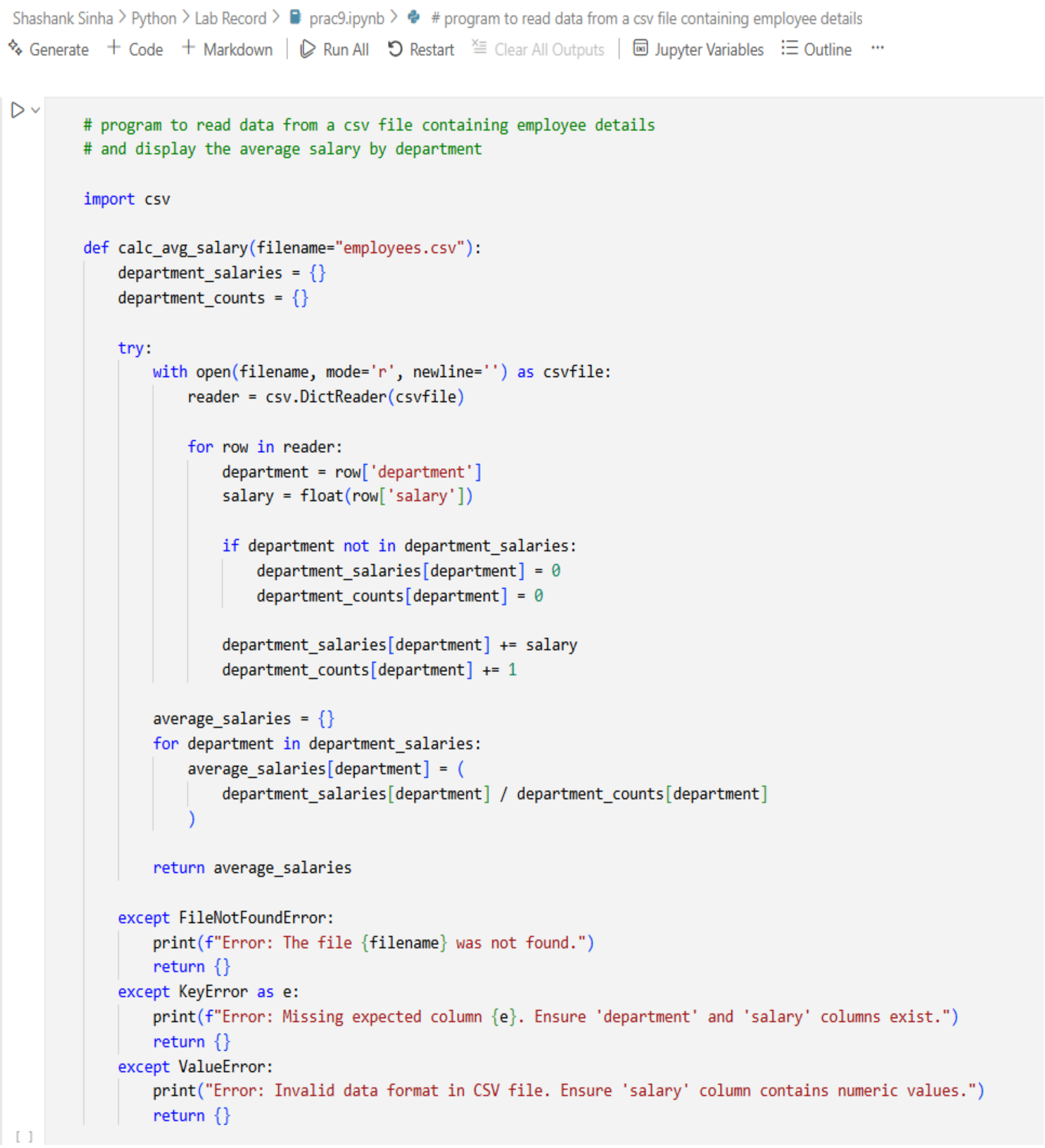


```
The student with the highest marks is:Bob
```

Practical-9

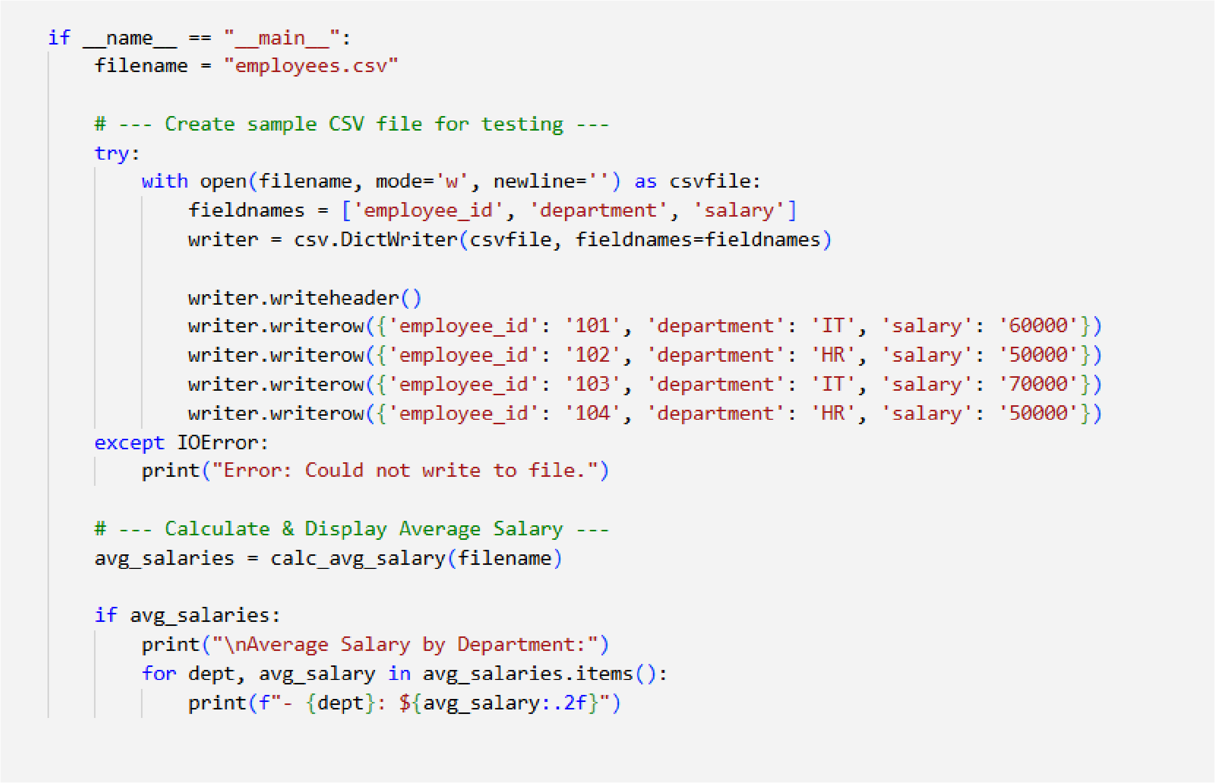
Aim: Read data from a CSV file containing employee details (name, department, salary) and display the average salary by department.

Code :-



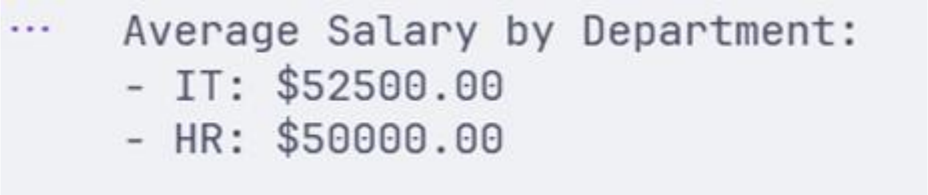
Shashank Sinha > Python > Lab Record > prac9.ipynb > # program to read data from a csv file containing employee details
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
···
[ ]

```python
# program to read data from a csv file containing employee details
# and display the average salary by department

import csv

def calc_avg_salary(filename="employees.csv"):
    department_salaries = {}
    department_counts = {}

    try:
        with open(filename, mode='r', newline='') as csvfile:
            reader = csv.DictReader(csvfile)

            for row in reader:
                department = row['department']
                salary = float(row['salary'])

                if department not in department_salaries:
                    department_salaries[department] = 0
                    department_counts[department] = 0

                department_salaries[department] += salary
                department_counts[department] += 1

        average_salaries = {}
        for department in department_salaries:
            average_salaries[department] = (
                department_salaries[department] / department_counts[department]
            )

        return average_salaries

    except FileNotFoundError:
        print(f"Error: The file {filename} was not found.")
        return {}
    except KeyError as e:
        print(f"Error: Missing expected column {e}. Ensure 'department' and 'salary' columns exist.")
        return {}
    except ValueError:
        print("Error: Invalid data format in CSV file. Ensure 'salary' column contains numeric values.")
        return {}
```



```python
if __name__ == "__main__":
    filename = "employees.csv"

    # --- Create sample CSV file for testing ---
    try:
        with open(filename, mode='w', newline='') as csvfile:
            fieldnames = ['employee_id', 'department', 'salary']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

            writer.writeheader()
            writer.writerow({'employee_id': '101', 'department': 'IT', 'salary': '60000'})
            writer.writerow({'employee_id': '102', 'department': 'HR', 'salary': '50000'})
            writer.writerow({'employee_id': '103', 'department': 'IT', 'salary': '70000'})
            writer.writerow({'employee_id': '104', 'department': 'HR', 'salary': '50000'})
    except IOError:
        print("Error: Could not write to file.")

    # --- Calculate & Display Average Salary ---
    avg_salaries = calc_avg_salary(filename)

    if avg_salaries:
        print("\nAverage Salary by Department:")
        for dept, avg_salary in avg_salaries.items():
            print(f"- {dept}: ${avg_salary:.2f}")
```

Output :-



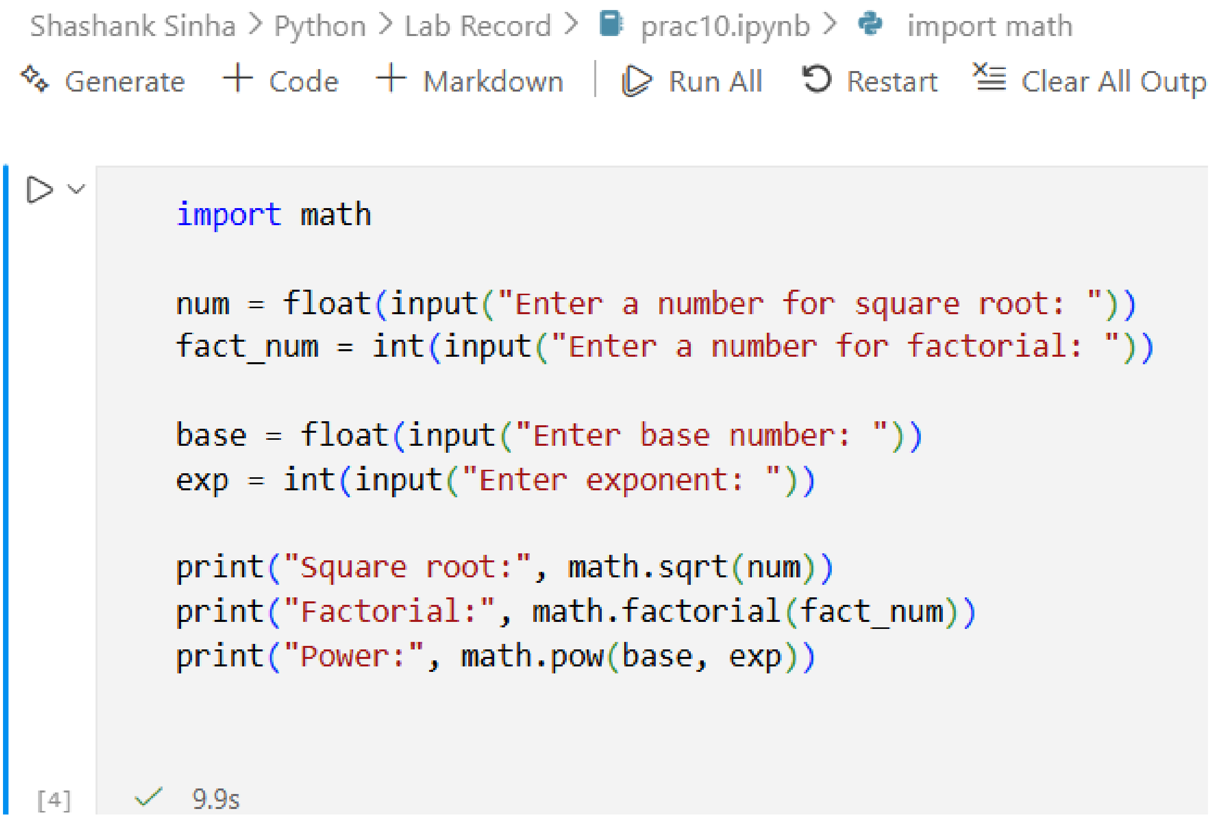
```
Average Salary by Department:
- IT: $52500.00
- HR: $50000.00
```

Practical-10

Aim:- Write a python program using math module to calculate:

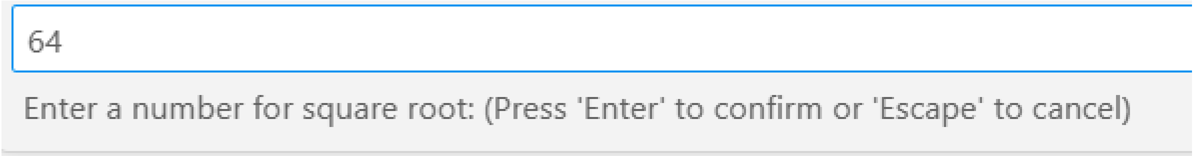
a) square root of a number
b) factorial of a number
c) power of a number
(take input from the user)

Code:-


Shashank Sinha > Python > Lab Record > prac10.ipynb > import math
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outp
[4]
✓ 9.9s

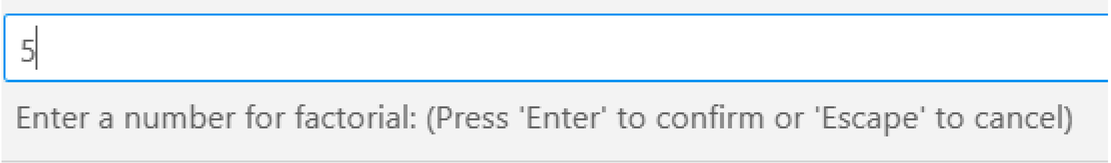
```python
import math

num = float(input("Enter a number for square root: "))
fact_num = int(input("Enter a number for factorial: "))

base = float(input("Enter base number: "))
exp = int(input("Enter exponent: "))

print("Square root:", math.sqrt(num))
print("Factorial:", math.factorial(fact_num))
print("Power:", math.pow(base, exp))
```

Input:-



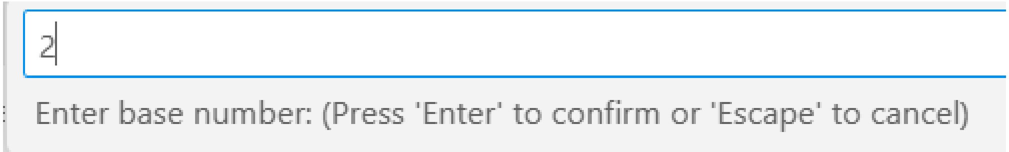
64

Enter a number for square root: (Press 'Enter' to confirm or 'Escape' to cancel)



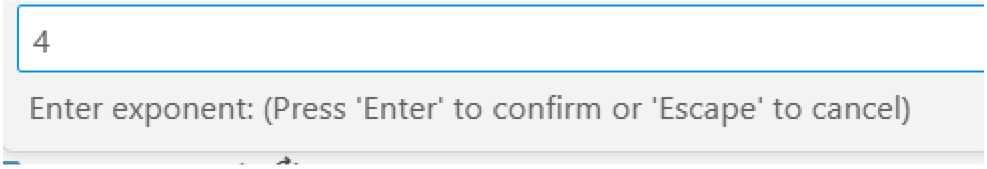
5

Enter a number for factorial: (Press 'Enter' to confirm or 'Escape' to cancel)



```
2
```

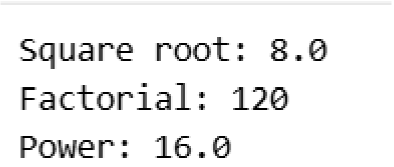
Enter base number: (Press 'Enter' to confirm or 'Escape' to cancel)



```
4
```

Enter exponent: (Press 'Enter' to confirm or 'Escape' to cancel)

Output:-

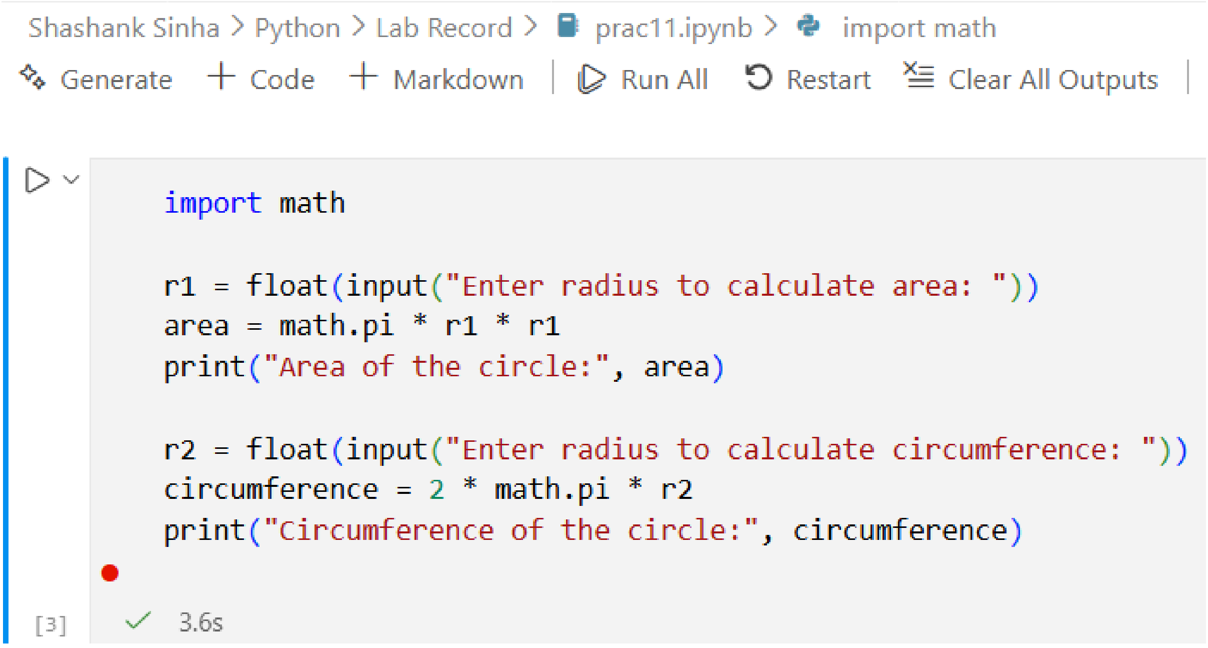


```
Square root: 8.0
Factorial: 120
Power: 16.0
```

Practical-11

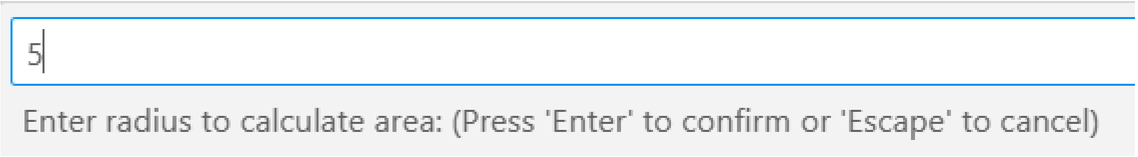
Aim:- Write a python program to find the area and circumference of a circle using math module, take input from the user

Code:-

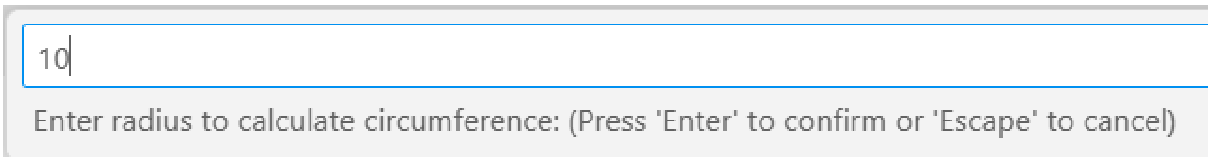

Shashank Sinha > Python > Lab Record > prac11.ipynb > import math
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
[3]
✓ 3.6s

```
import math

r1 = float(input("Enter radius to calculate area: "))
area = math.pi * r1 * r1
print("Area of the circle:", area)

r2 = float(input("Enter radius to calculate circumference: "))
circumference = 2 * math.pi * r2
print("Circumference of the circle:", circumference)
```

Input:-

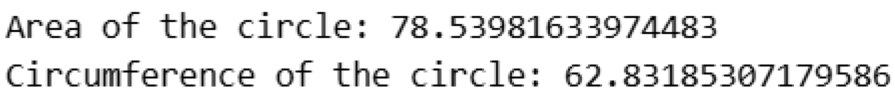


```
5
Enter radius to calculate area: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
10
Enter radius to calculate circumference: (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:-

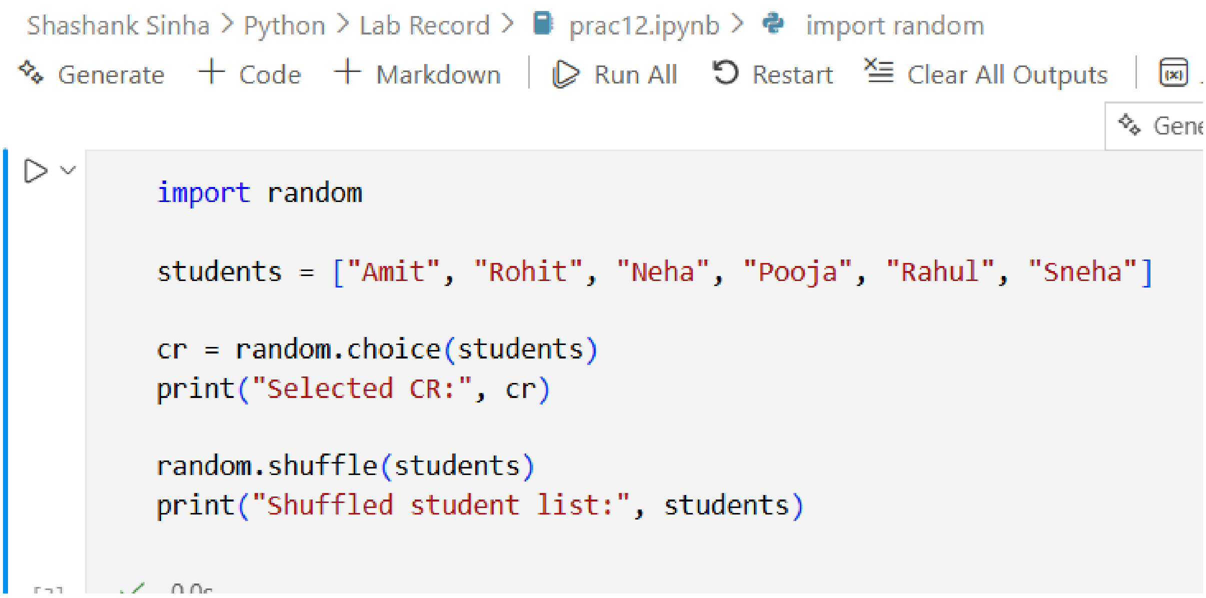


```
Area of the circle: 78.53981633974483
Circumference of the circle: 62.83185307179586
```

Practical-12

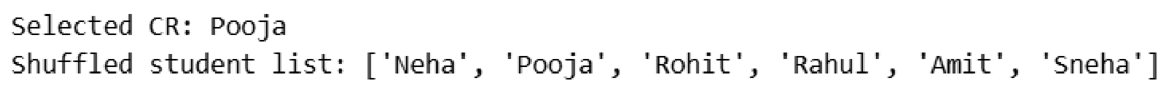
Aim:- Create a list of student and select CR randomly, then shuffle the list and print it.

Code:-


Shashank Sinha > Python > Lab Record > prac12.ipynb > import random
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Gene
0.0s

```python
import random

students = ["Amit", "Rohit", "Neha", "Pooja", "Rahul", "Sneha"]

cr = random.choice(students)
print("Selected CR:", cr)

random.shuffle(students)
print("Shuffled student list:", students)
```

Output:-

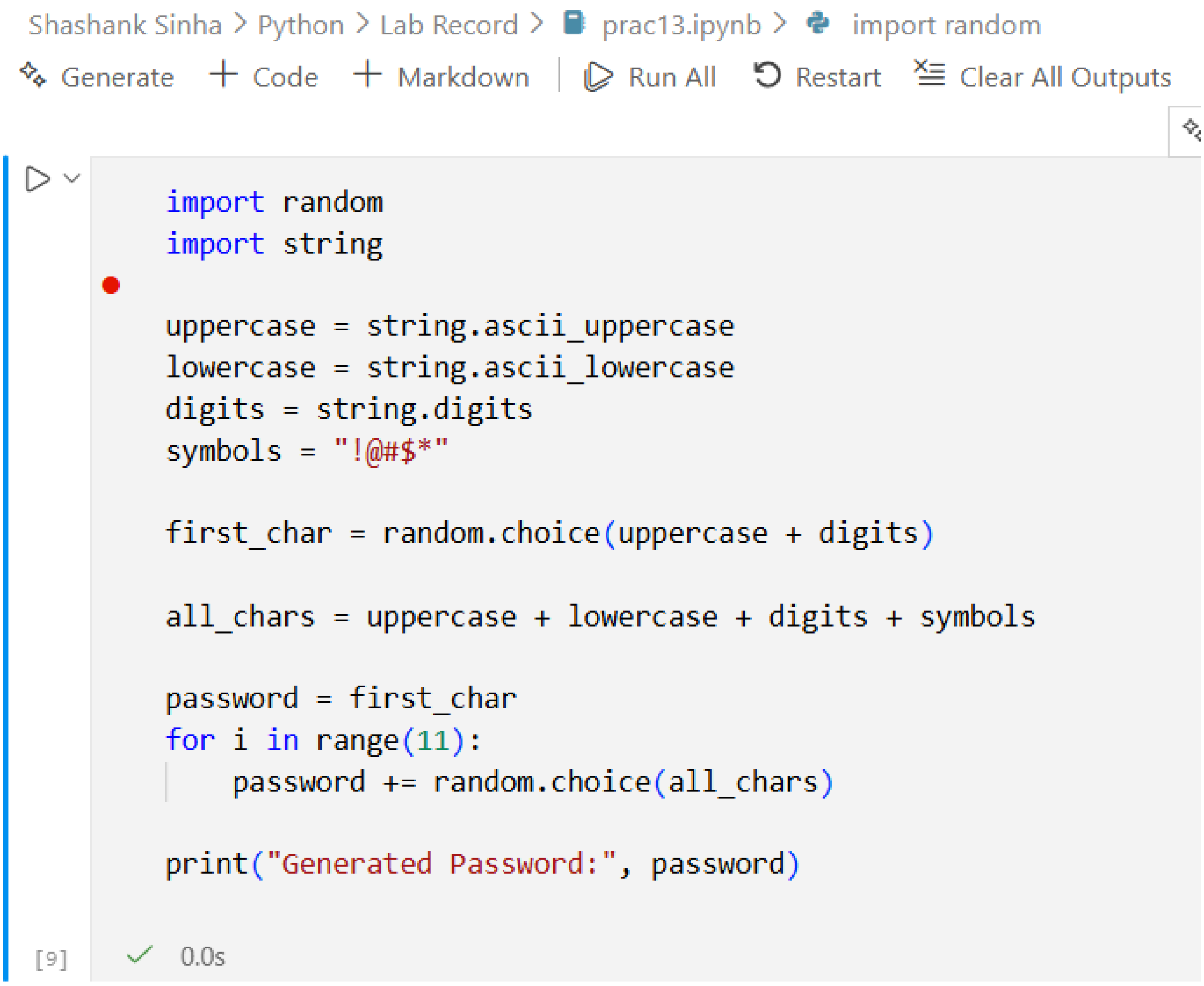


```
Selected CR: Pooja
Shuffled student list: ['Neha', 'Pooja', 'Rohit', 'Rahul', 'Amit', 'Sneha']
```

Practical-13

Aim:- Write a python program to generate a random password of 12 characters using UPPERCASE, lowercase, digits and special symbols(!, @, #, $, *). Password should start with Capital letter or digits.

Code:-


Shashank Sinha > Python > Lab Record > prac13.ipynb > import random
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
[9]
✓ 0.0s

```python
import random
import string

uppercase = string.ascii_uppercase
lowercase = string.ascii_lowercase
digits = string.digits
symbols = "!@#$*"

first_char = random.choice(uppercase + digits)

all_chars = uppercase + lowercase + digits + symbols

password = first_char
for i in range(11):
    password += random.choice(all_chars)

print("Generated Password:", password)
```

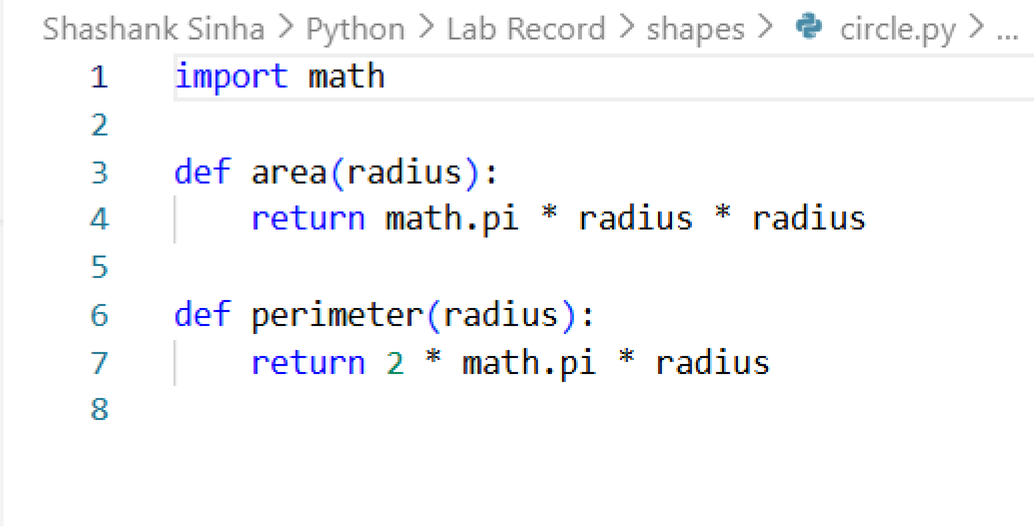
Output:-

```
Generated Password: 99Ngopq*aNdr
```

Practical-14

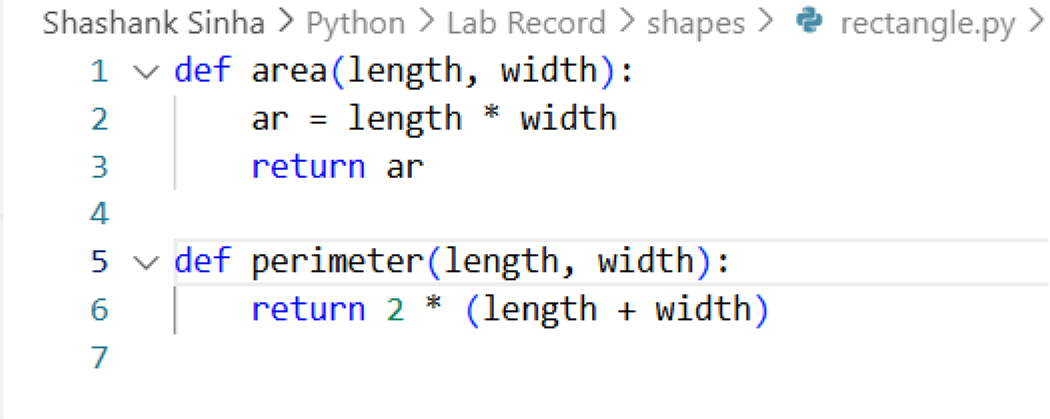
Aim:- Create a package named **shapes** that contains two modules: **circle.py** and **rectangle.py.** Each module should compute area and perimeter of the shape. Write the complete directory structure and sample code.

Module 1(circle.py)


Shashank Sinha > Python > Lab Record > shapes > circle.py > ...

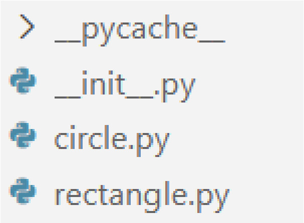
```python
import math

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

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

```

Module 2(rectangle.py)


Shashank Sinha > Python > Lab Record > shapes > rectangle.py >

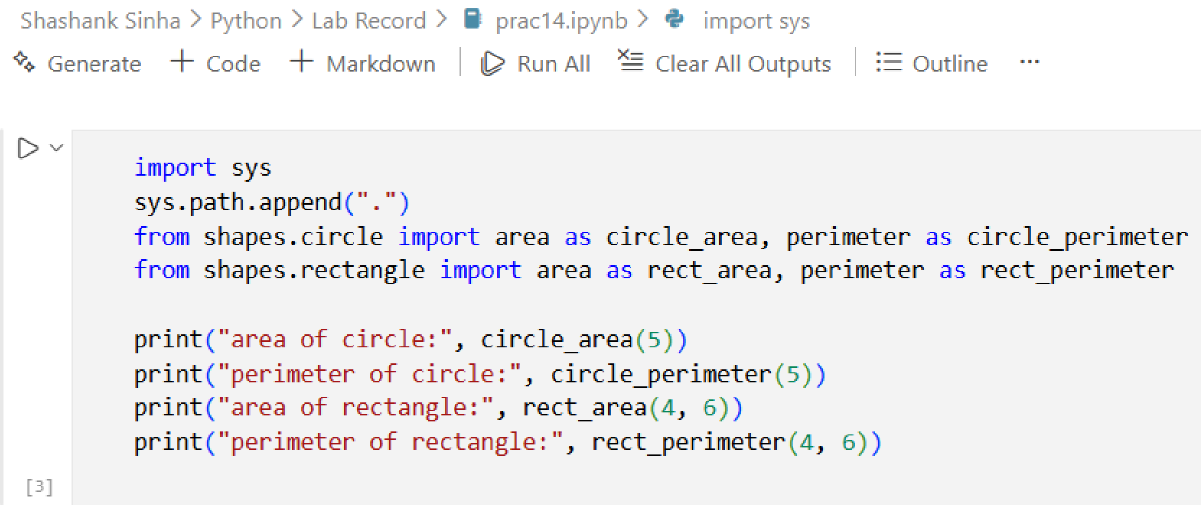
```python
def area(length, width):
    ar = length * width
    return ar

def perimeter(length, width):
    return 2 * (length + width)

```

Directory Sample



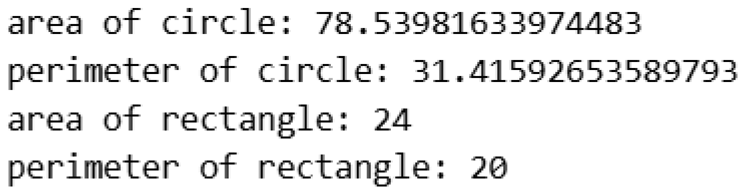
```
> __pycache__
  __init__.py
  circle.py
  rectangle.py
```

Code:-


Shashank Sinha > Python > Lab Record > prac14.ipynb > import sys
Generate
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
···
[3]

```
import sys
sys.path.append(".")
from shapes.circle import area as circle_area, perimeter as circle_perimeter
from shapes.rectangle import area as rect_area, perimeter as rect_perimeter

print("area of circle:", circle_area(5))
print("perimeter of circle:", circle_perimeter(5))
print("area of rectangle:", rect_area(4, 6))
print("perimeter of rectangle:", rect_perimeter(4, 6))
```

Output:-



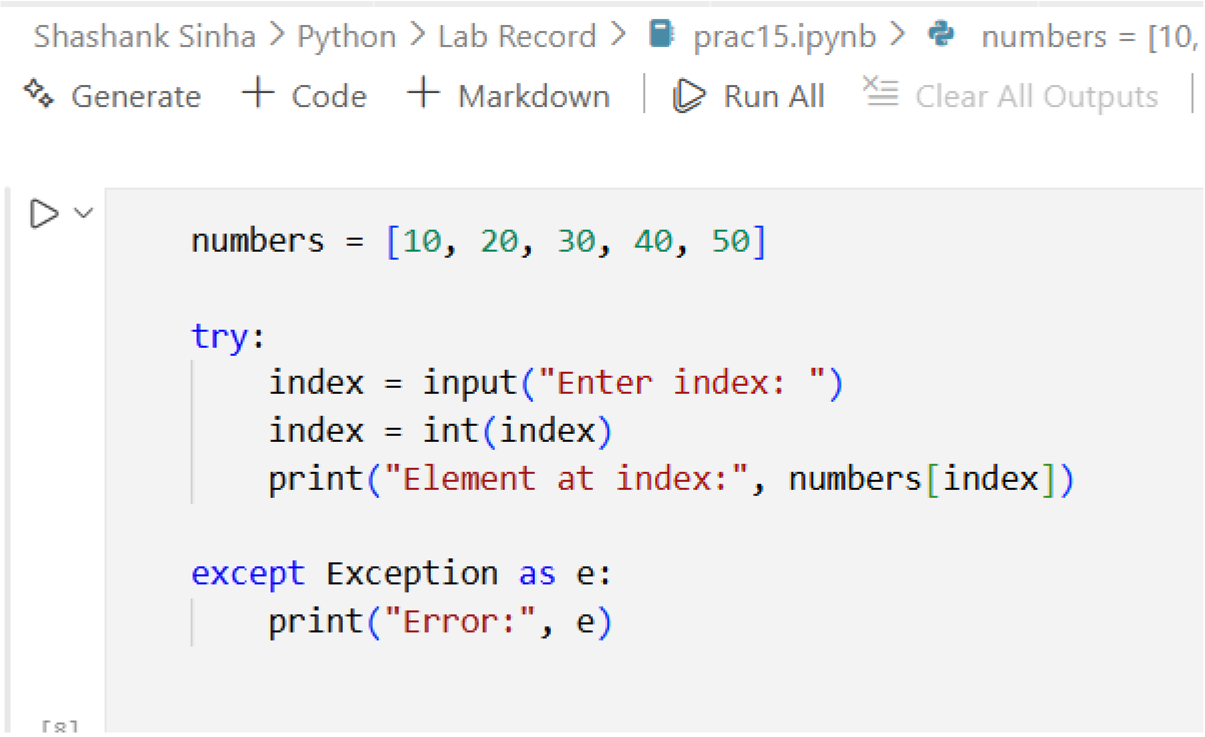
```
area of circle: 78.53981633974483
perimeter of circle: 31.41592653589793
area of rectangle: 24
perimeter of rectangle: 20
```

Practical-15

Aim:- Write a program that asks the user for an index and prints the element at that index from a predefined list.
Handle:

- **IndexError** (index out of range)
- **TypeError** (if user enters a non-integer index)
- Print the specific exception message using **except Exception as e.**

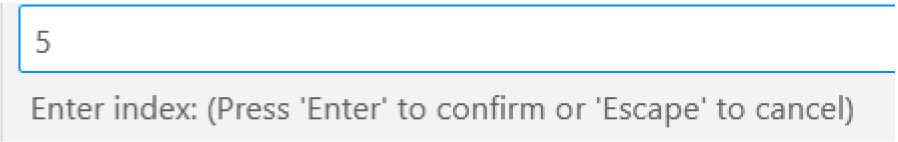
Code:-


Shashank Sinha > Python > Lab Record > prac15.ipynb > numbers = [10,
Generate
+ Code
+ Markdown
Run All
Clear All Outputs
[8]

```
numbers = [10, 20, 30, 40, 50]

try:
    index = input("Enter index: ")
    index = int(index)
    print("Element at index:", numbers[index])

except Exception as e:
    print("Error:", e)
```

Output:-

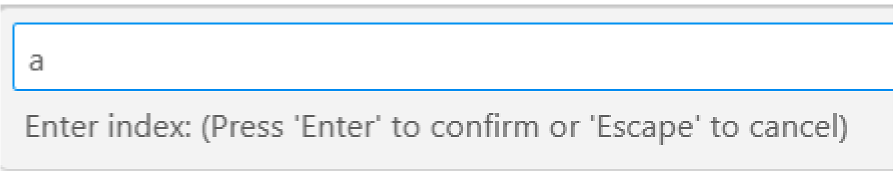
1



```
5
Enter index: (Press 'Enter' to confirm or 'Escape' to cancel)
```

```
Error: list index out of range
```

2

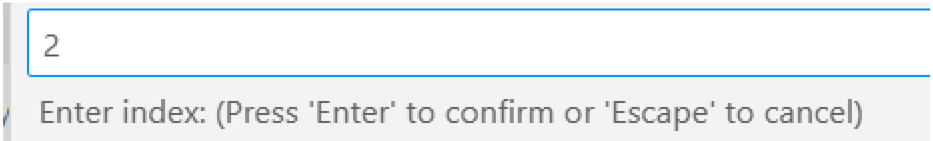


```
a
```

Enter index: (Press 'Enter' to confirm or 'Escape' to cancel)

```
Error: invalid literal for int() with base 10: 'a'
```

3



```
2
```

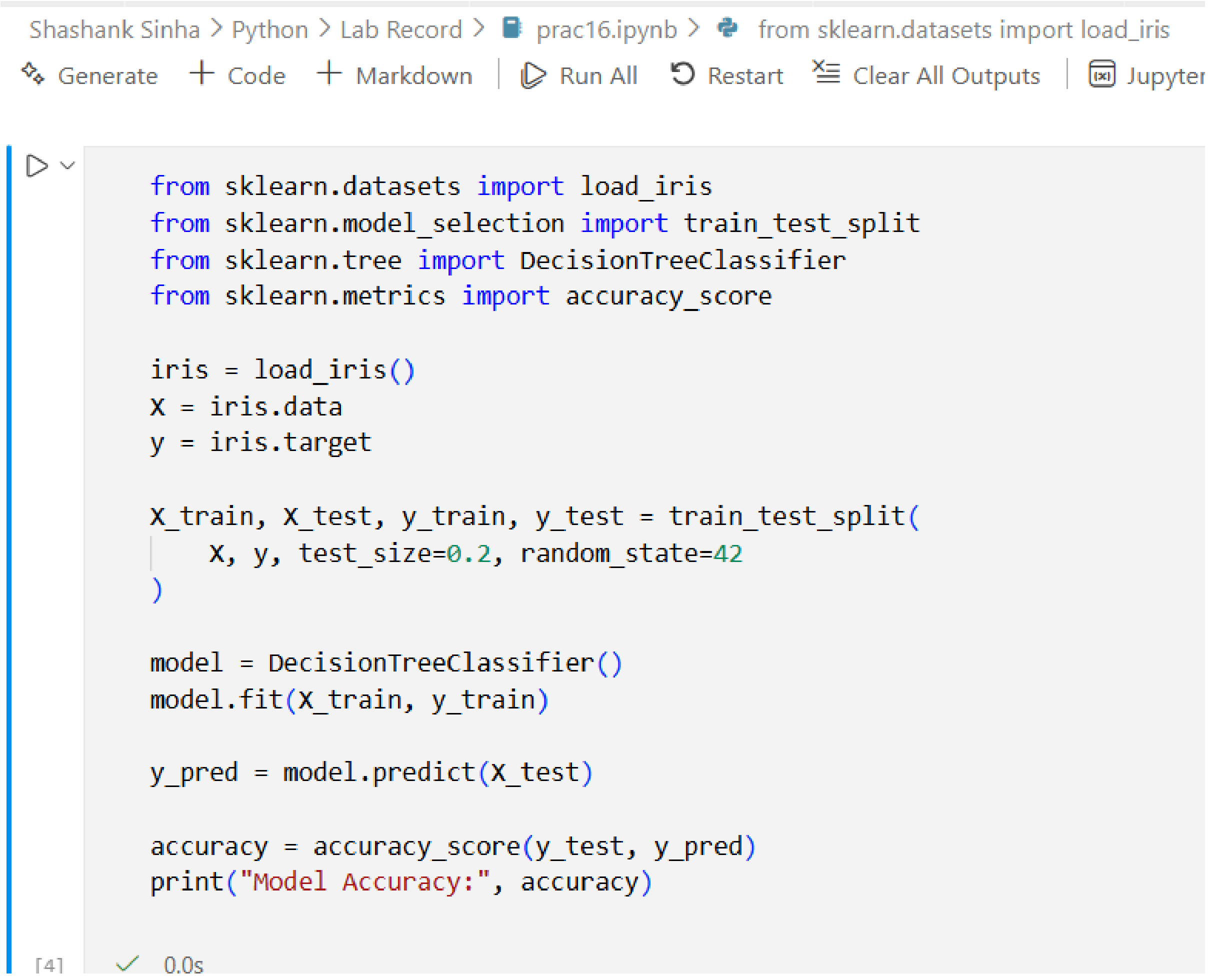
Enter index: (Press 'Enter' to confirm or 'Escape' to cancel)

```
Element at index: 30
```

Practical-16

Aim:- Implement a supervised learning model to classify flowers in the Iris dataset using Decision Tree Classifier. Print the accuracy of the model.

Code:-


Shashank Sinha > Python > Lab Record > prac16.ipynb > from sklearn.datasets import load_iris
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter
[4]
✓ 0.0s

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

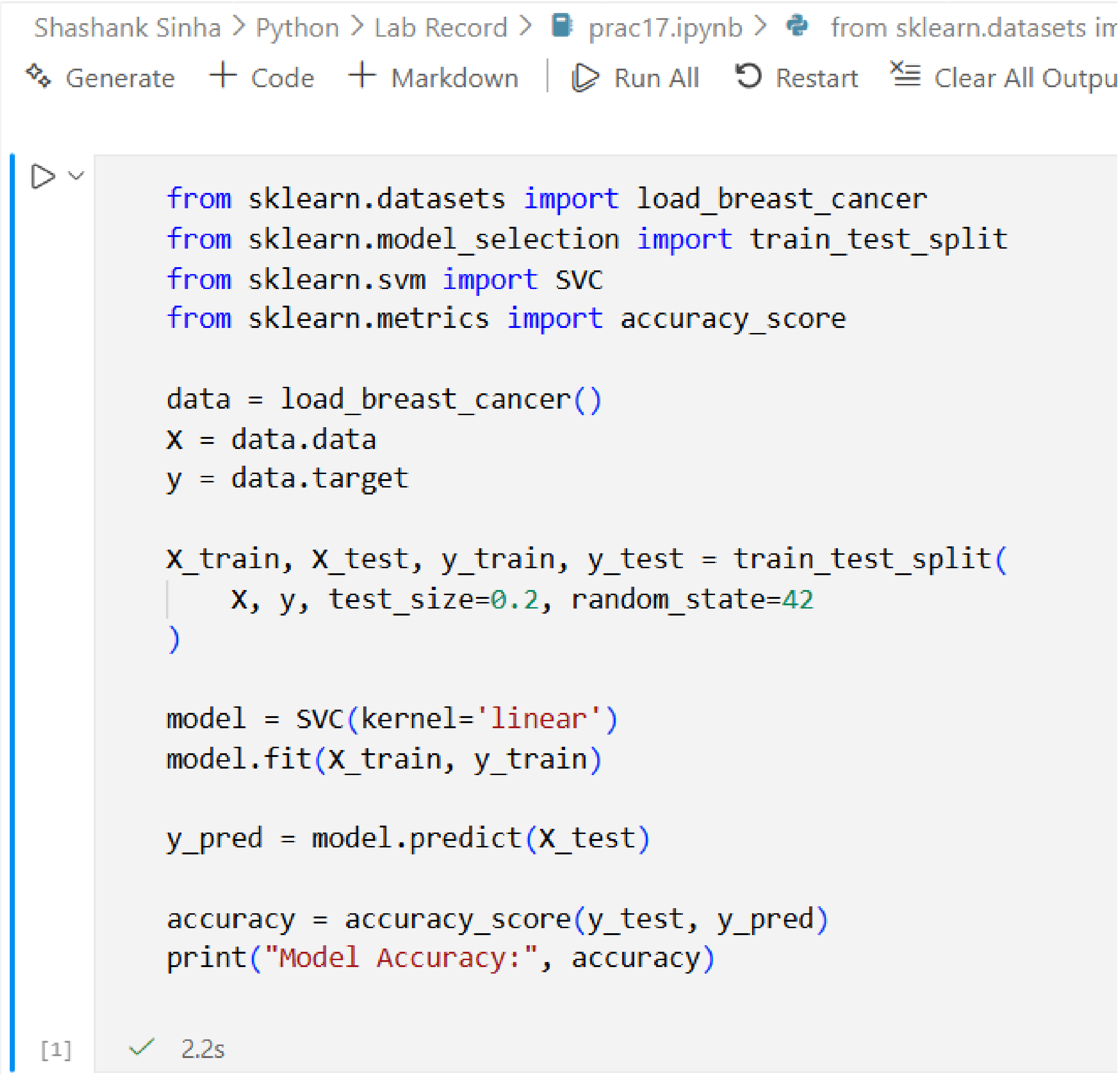
Output:-

```
Model Accuracy: 1.0
```

Practical-17

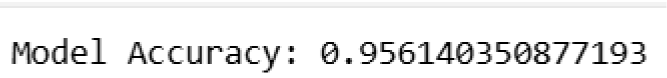
Aim:- Use Support Vector Machine (SVM) on the Breast Cancer datasets to classify malignant vs benign tumors.

Code:-


Shashank Sinha > Python > Lab Record > prac17.ipynb > from sklearn.datasets im
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outpu
[1]
✓ 2.2s

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = SVC(kernel='linear')
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

Output:-

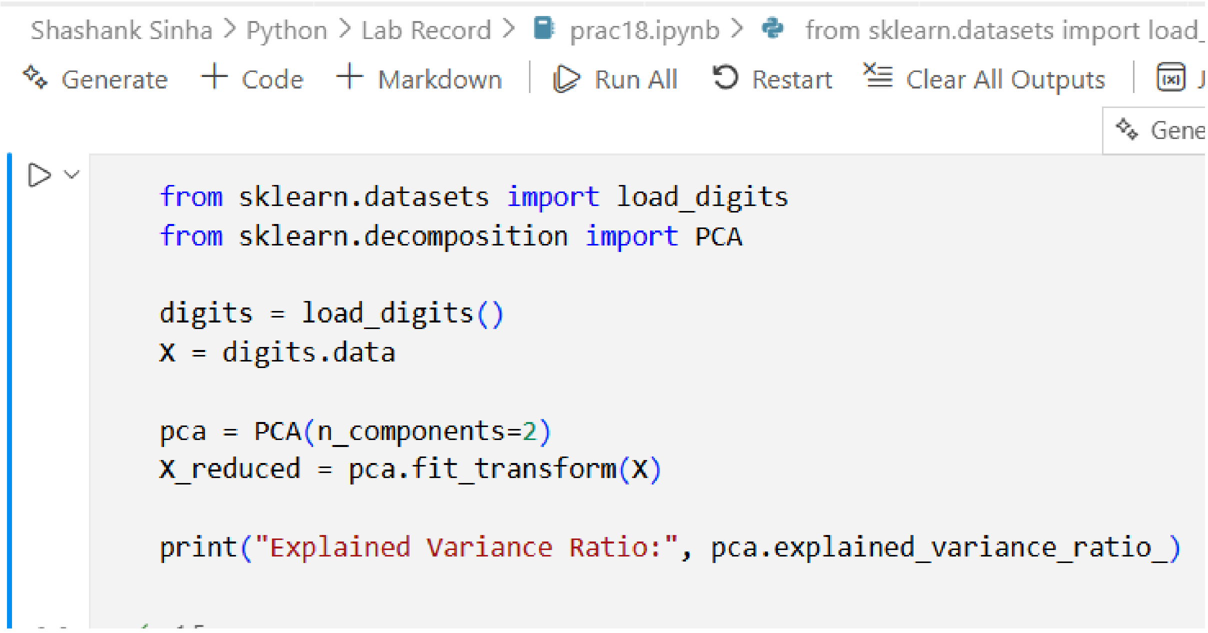


```
Model Accuracy: 0.956140350877193
```

Practical-18

Aim:- Perform Principal Component Analysis (PCA)on the Digits dataset and reduce the dimension to 2. Print the explained variance ratio.

Code:-



Shashank Sinha > Python > Lab Record > prac18.ipynb > from sklearn.datasets import load_
Generate
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Gene

```python
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA

digits = load_digits()
X = digits.data

pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```

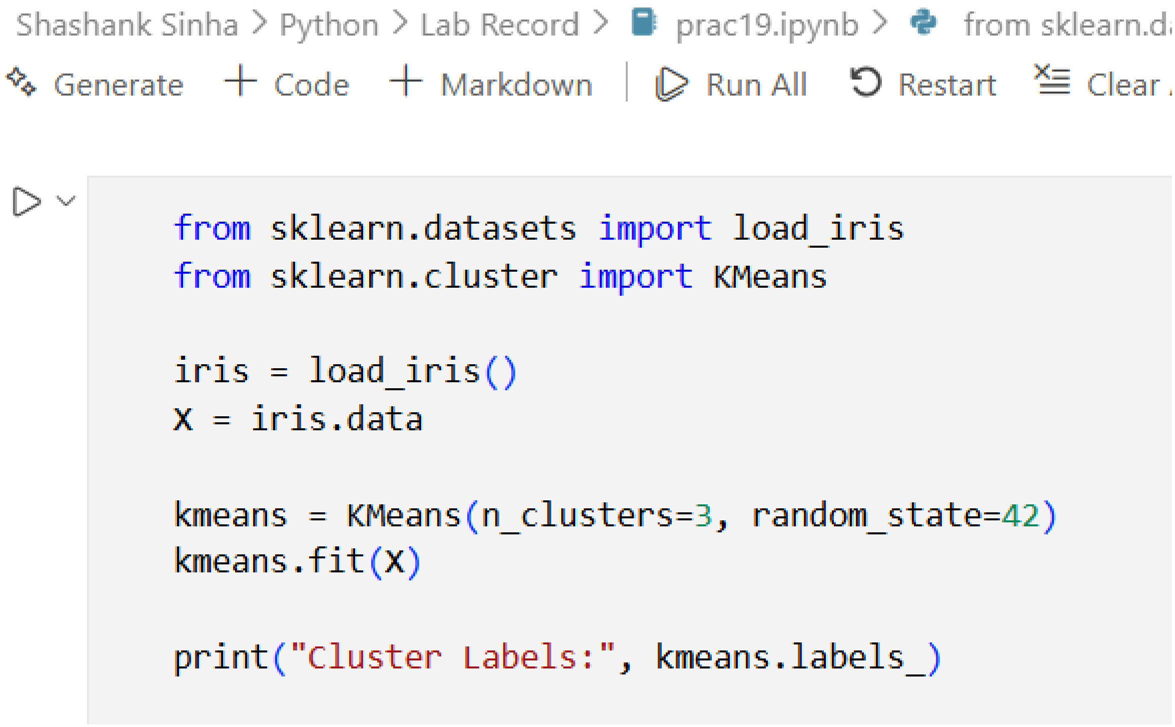
Output:-

```
Explained Variance Ratio: [0.14890594 0.13618771]
```

Practical-19

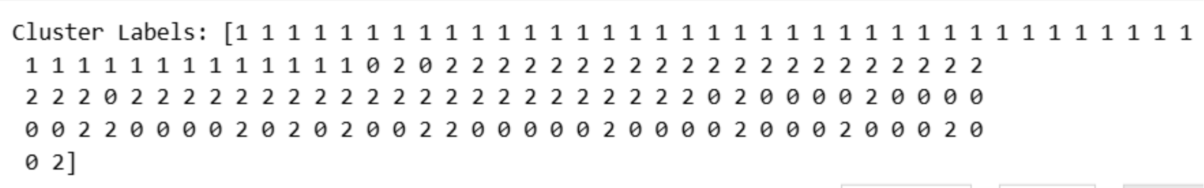
Aim:- Apply K-Means clustering on the Iris Dataset and print cluster labels.

Code:-


Shashank Sinha > Python > Lab Record > prac19.ipynb > from sklearn.d
Generate
+ Code
+ Markdown
Run All
Restart
Clear

```python
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

iris = load_iris()
X = iris.data

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

print("Cluster Labels:", kmeans.labels_)
```

Output:-



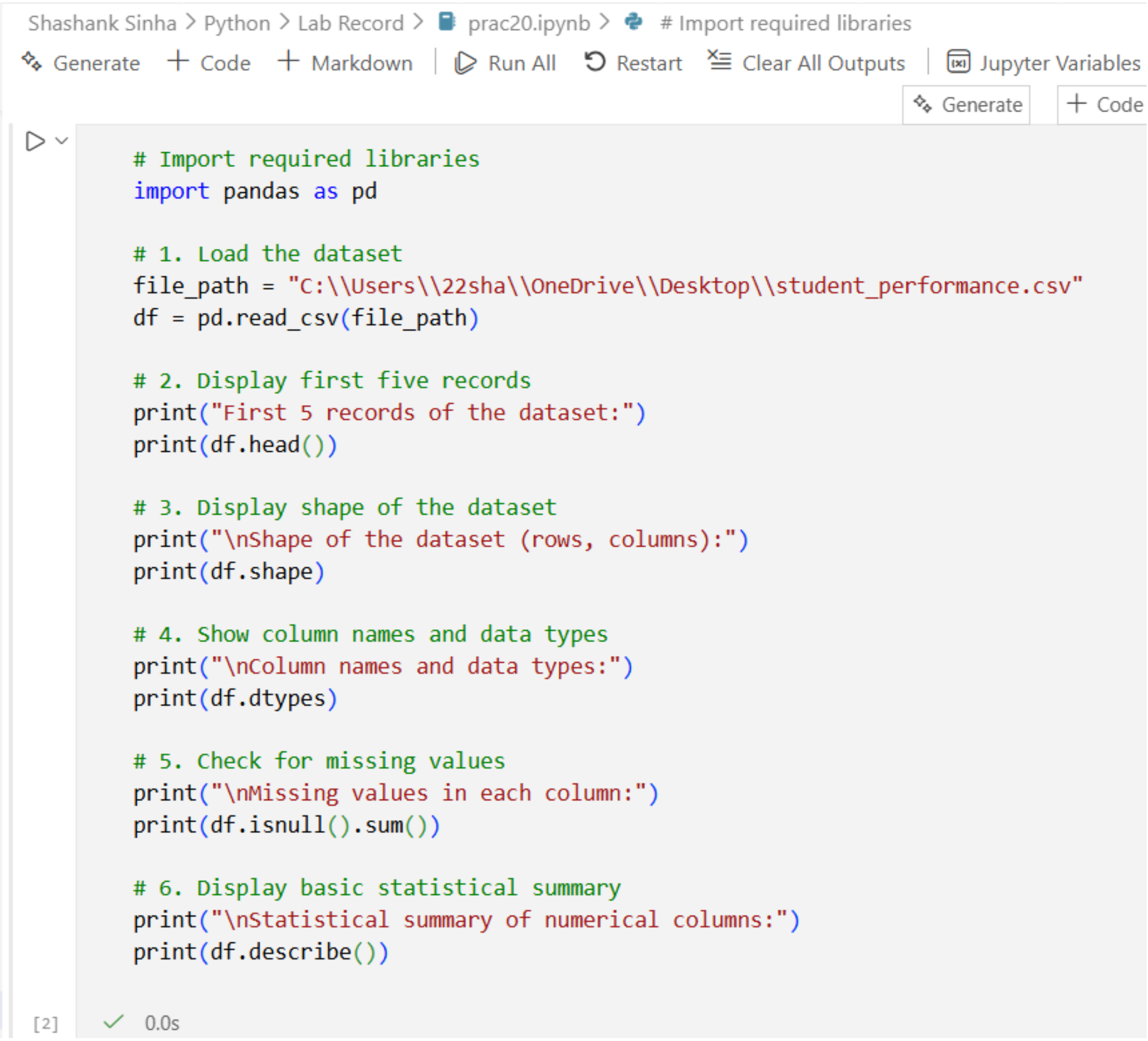
```
Cluster Labels: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
```

Practical-20

1. Aim:- Write a Python program to perform basic EDA on a student performance dataset to understand its structure and summary statistics. (dataset attached along with filename – **sample_dataset**)

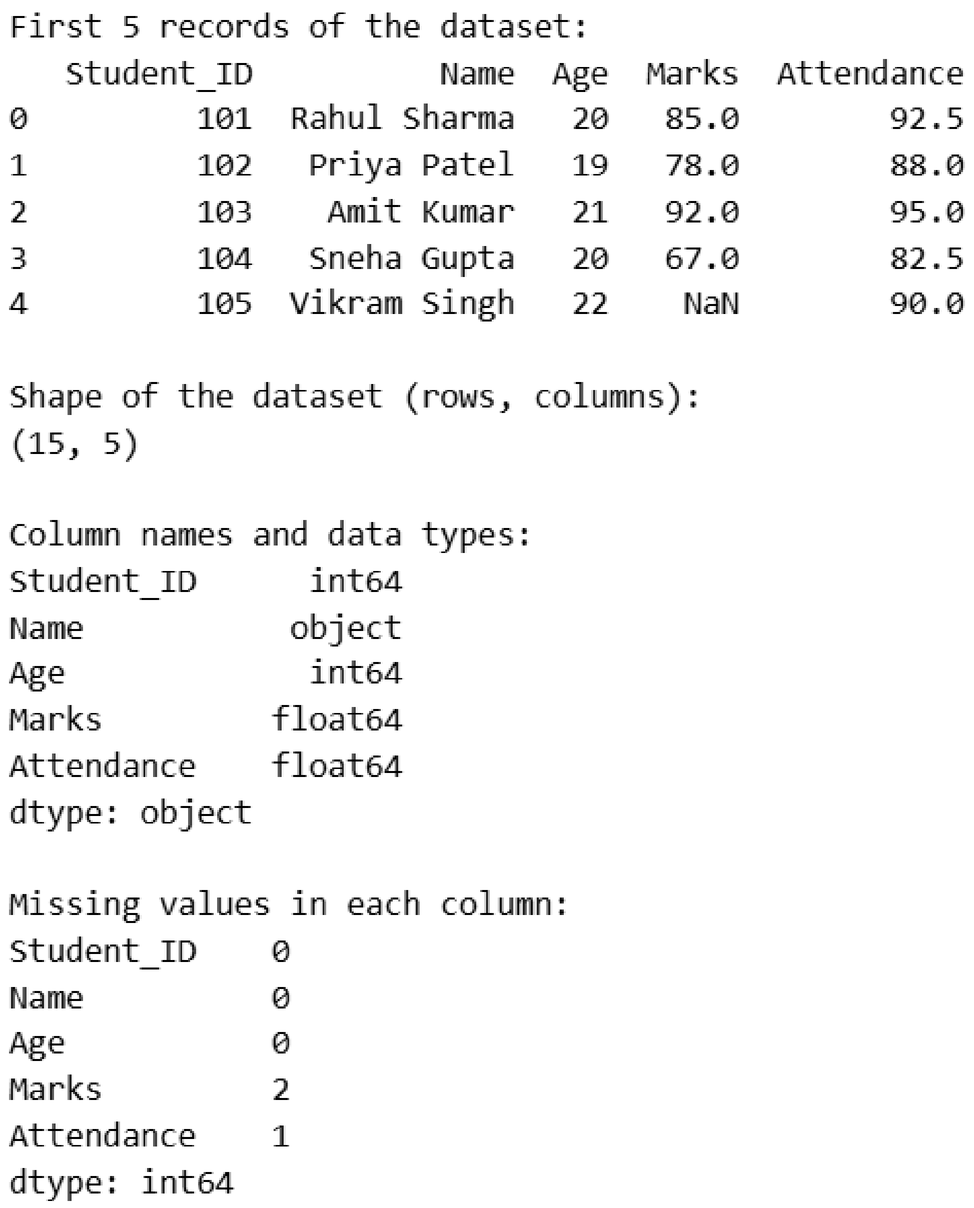
    - Load the dataset into Python
    - Display the first five records
    - Display the shape of the dataset
    - Show column names and data types
    - Check for missing values
    - Display basic statistical summary

Code:-

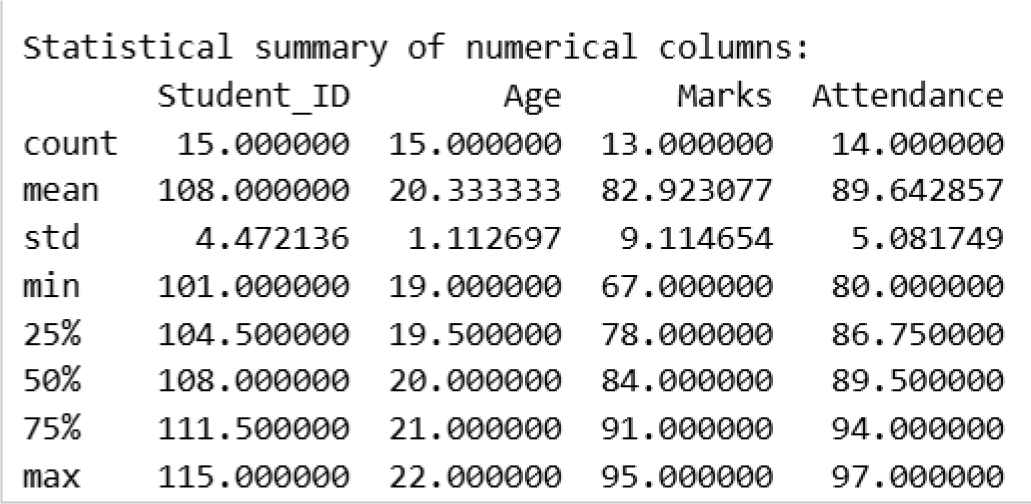


```python
# Import required libraries
import pandas as pd

# 1. Load the dataset
file_path = "C:\\Users\\22sha\\OneDrive\\Desktop\\student_performance.csv"
df = pd.read_csv(file_path)

# 2. Display first five records
print("First 5 records of the dataset:")
print(df.head())

# 3. Display shape of the dataset
print("\nShape of the dataset (rows, columns):")
print(df.shape)

# 4. Show column names and data types
print("\nColumn names and data types:")
print(df.dtypes)

# 5. Check for missing values
print("\nMissing values in each column:")
print(df.isnull().sum())

# 6. Display basic statistical summary
print("\nStatistical summary of numerical columns:")
print(df.describe())
```

Output:-



```
First 5 records of the dataset:
   Student_ID          Name  Age  Marks  Attendance
0         101  Rahul Sharma   20   85.0        92.5
1         102   Priya Patel   19   78.0        88.0
2         103    Amit Kumar   21   92.0        95.0
3         104   Sneha Gupta   20   67.0        82.5
4         105  Vikram Singh   22    NaN        90.0

Shape of the dataset (rows, columns):
(15, 5)

Column names and data types:
Student_ID      int64
Name           object
Age             int64
Marks         float64
Attendance    float64
dtype: object

Missing values in each column:
Student_ID    0
Name          0
Age           0
Marks         2
Attendance    1
dtype: int64
```



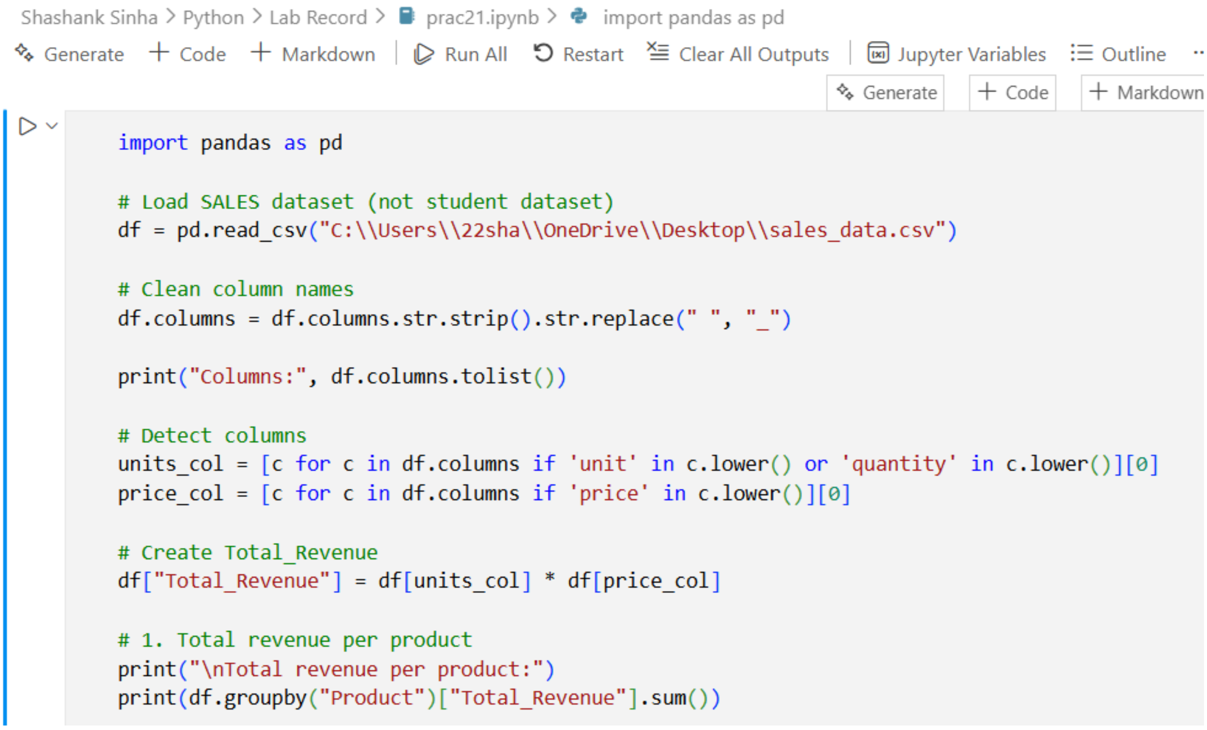
```
Statistical summary of numerical columns:
       Student_ID        Age      Marks  Attendance
count   15.000000  15.000000  13.000000   14.000000
mean   108.000000  20.333333  82.923077   89.642857
std      4.472136   1.112697   9.114654    5.081749
min    101.000000  19.000000  67.000000   80.000000
25%    104.500000  19.500000  78.000000   86.750000
50%    108.000000  20.000000  84.000000   89.500000
75%    111.500000  21.000000  91.000000   94.000000
max    115.000000  22.000000  95.000000   97.000000
```

Practical-21

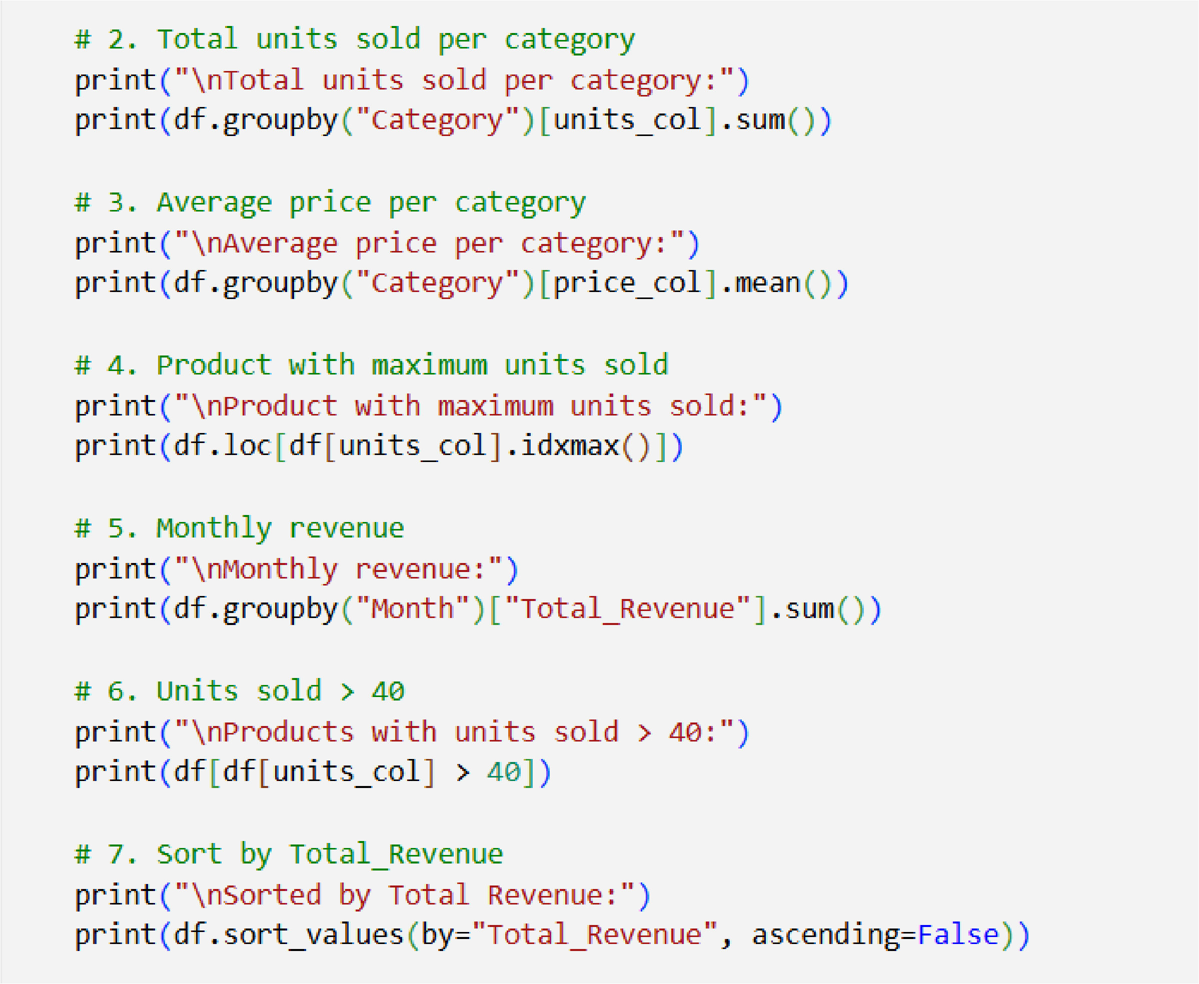
Aim:- Write a Python program using pandas to analyze a sales dataset and extract meaningful insights.

- Load the dataset into a pandas DataFrame
- Create a new column Total_Revenue by multiplying Units_Sold and Price_Per_Unit
- Find the total revenue generated for each product
- Find the total units sold for each category
- Calculate the average price per unit for each category
- Find the product with maximum units sold
- Group the data by Month and calculate the total revenue for each month
- Filter and display products where Units_Sold is greater than 40
- Sort the dataset by Total_Revenue in descending order

Code:-

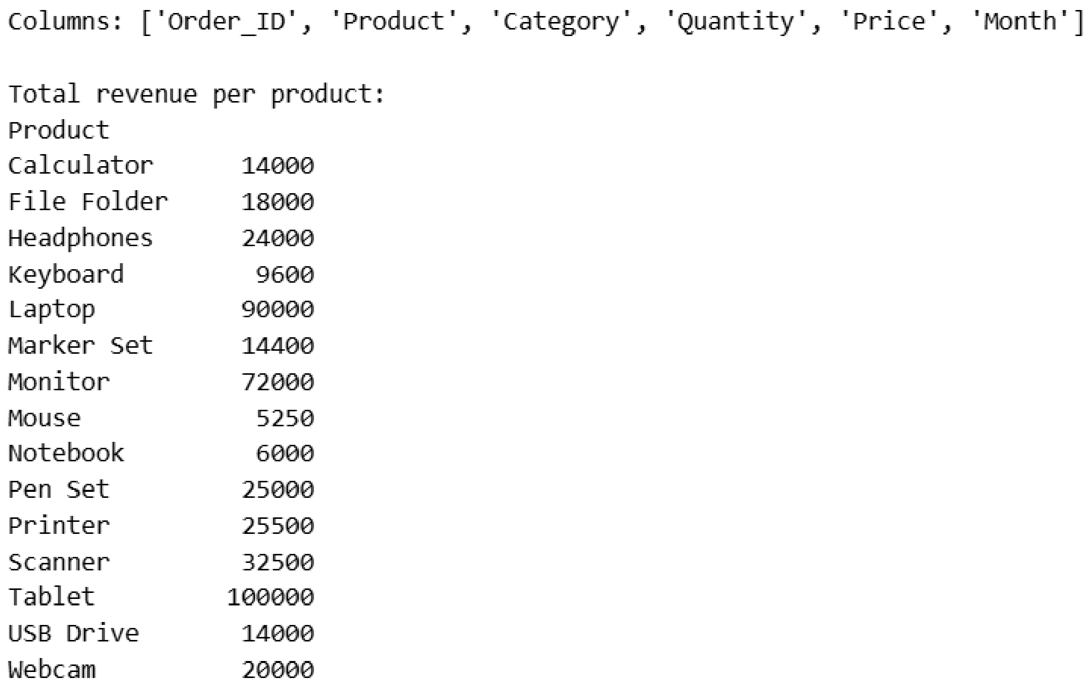


```
import pandas as pd

# Load SALES dataset (not student dataset)
df = pd.read_csv("C:\\Users\\22sha\\OneDrive\\Desktop\\sales_data.csv")

# Clean column names
df.columns = df.columns.str.strip().str.replace(" ", "_")

print("Columns:", df.columns.tolist())

# Detect columns
units_col = [c for c in df.columns if 'unit' in c.lower() or 'quantity' in c.lower()][0]
price_col = [c for c in df.columns if 'price' in c.lower()][0]

# Create Total_Revenue
df["Total_Revenue"] = df[units_col] * df[price_col]

# 1. Total revenue per product
print("\nTotal revenue per product:")
print(df.groupby("Product")["Total_Revenue"].sum())
```

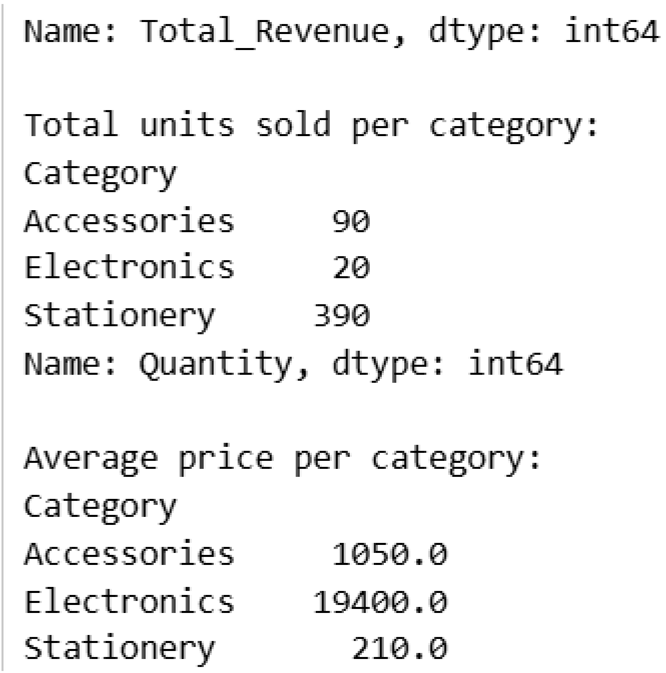


```
# 2. Total units sold per category
print("\nTotal units sold per category:")
print(df.groupby("Category")[units_col].sum())

# 3. Average price per category
print("\nAverage price per category:")
print(df.groupby("Category")[price_col].mean())

# 4. Product with maximum units sold
print("\nProduct with maximum units sold:")
print(df.loc[df[units_col].idxmax()])

# 5. Monthly revenue
print("\nMonthly revenue:")
print(df.groupby("Month")["Total_Revenue"].sum())

# 6. Units sold > 40
print("\nProducts with units sold > 40:")
print(df[df[units_col] > 40])

# 7. Sort by Total_Revenue
print("\nSorted by Total Revenue:")
print(df.sort_values(by="Total_Revenue", ascending=False))
```

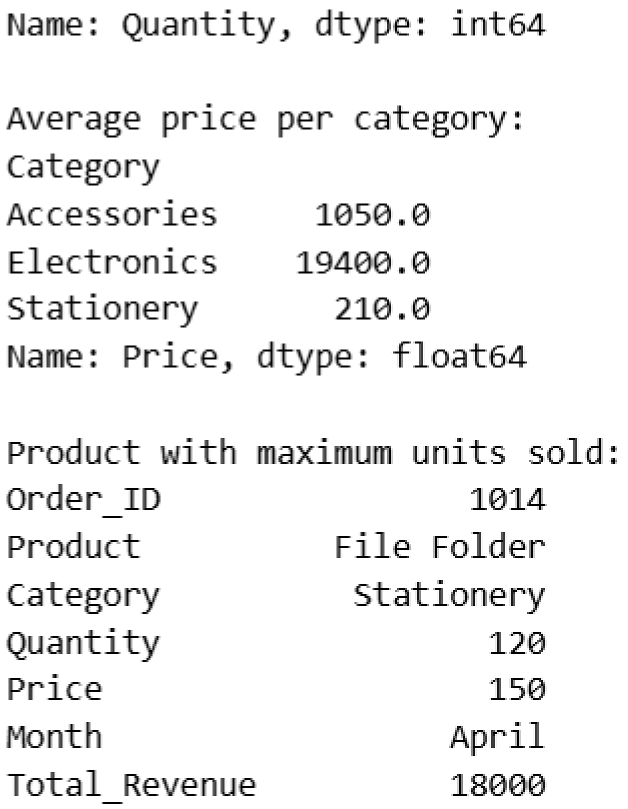
Output:-



```
Columns: ['Order_ID', 'Product', 'Category', 'Quantity', 'Price', 'Month']

Total revenue per product:
Product
Calculator       14000
File Folder      18000
Headphones       24000
Keyboard          9600
Laptop           90000
Marker Set       14400
Monitor          72000
Mouse             5250
Notebook          6000
Pen Set          25000
Printer          25500
Scanner          32500
Tablet          100000
USB Drive        14000
Webcam           20000
```

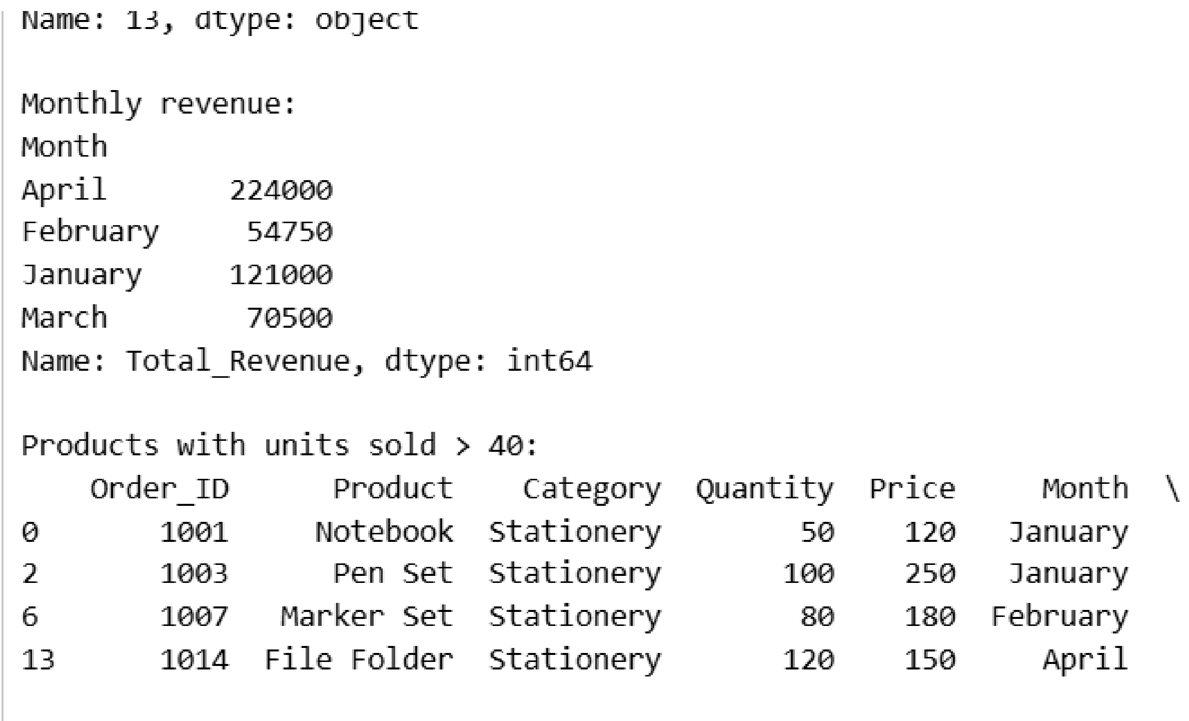


```
Name: Total_Revenue, dtype: int64

Total units sold per category:
Category
Accessories     90
Electronics     20
Stationery     390
Name: Quantity, dtype: int64

Average price per category:
Category
Accessories     1050.0
Electronics    19400.0
Stationery       210.0
```

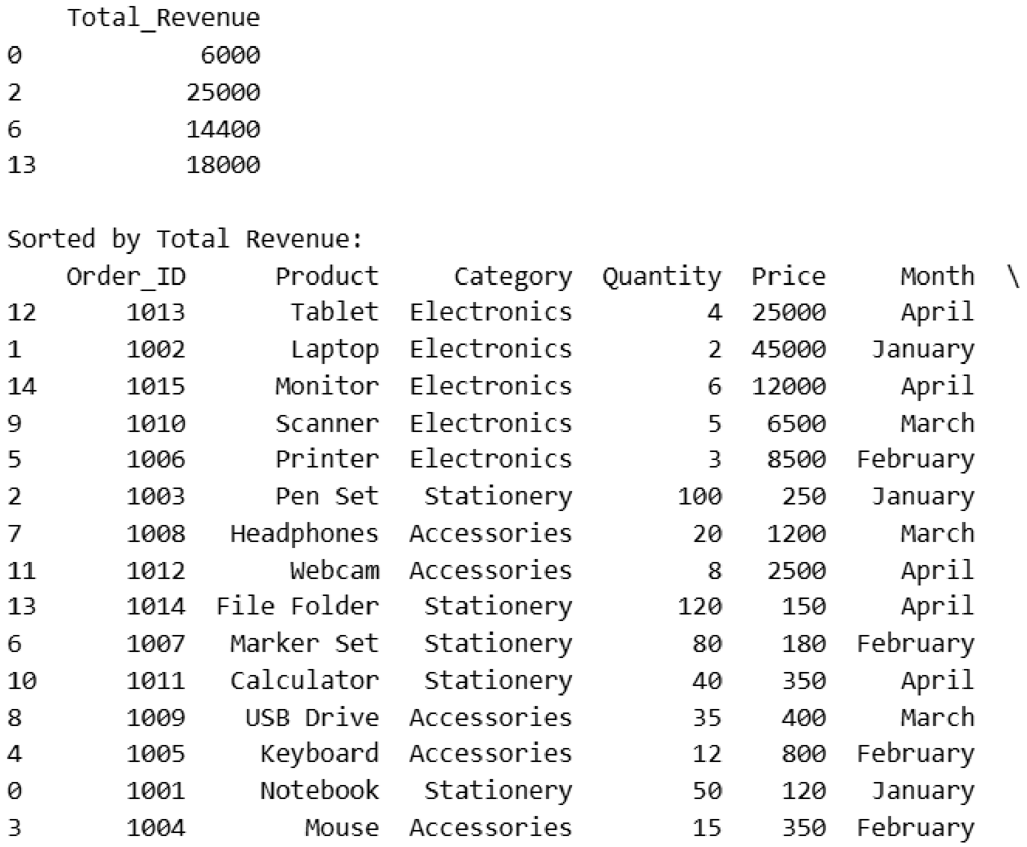


```
Name: Quantity, dtype: int64

Average price per category:
Category
Accessories     1050.0
Electronics    19400.0
Stationery       210.0
```

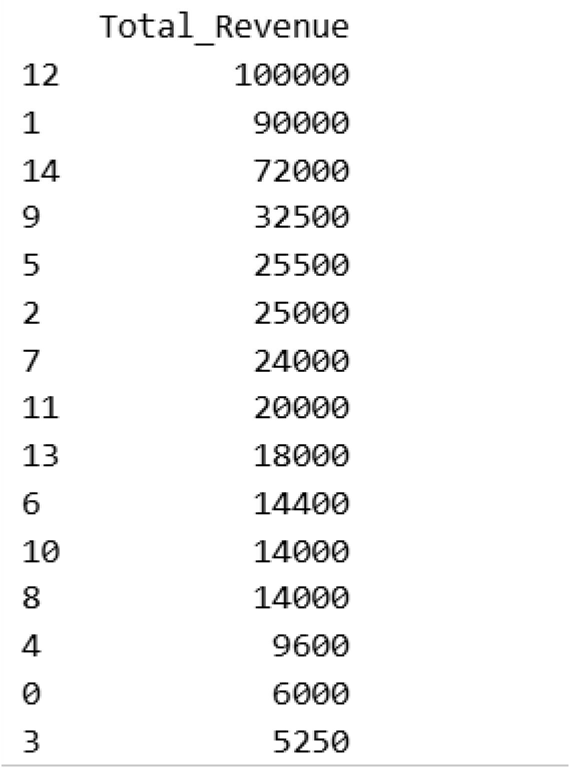
```
Name: Price, dtype: float64

Product with maximum units sold:
Order_ID                1014
Product          File Folder
Category          Stationery
Quantity                 120
Price                    150
Month                  April
Total_Revenue          18000
```



```
Name: 13, dtype: object

Monthly revenue:
Month
April       224000
February     54750
January     121000
March        70500
Name: Total_Revenue, dtype: int64

Products with units sold > 40:
    Order_ID      Product    Category  Quantity  Price     Month  \
0       1001     Notebook  Stationery        50    120   January
2       1003      Pen Set  Stationery       100    250   January
6       1007   Marker Set  Stationery        80    180  February
13      1014  File Folder  Stationery       120    150     April
```



```
    Total_Revenue
0            6000
2           25000
6           14400
13          18000

Sorted by Total Revenue:
    Order_ID      Product     Category  Quantity  Price     Month  \
12      1013       Tablet  Electronics         4  25000     April
1       1002       Laptop  Electronics         2  45000   January
14      1015      Monitor  Electronics         6  12000     April
9       1010      Scanner  Electronics         5   6500     March
5       1006      Printer  Electronics         3   8500  February
2       1003      Pen Set   Stationery       100    250   January
7       1008   Headphones  Accessories        20   1200     March
11      1012       Webcam  Accessories         8   2500     April
13      1014  File Folder   Stationery       120    150     April
6       1007   Marker Set   Stationery        80    180  February
10      1011   Calculator   Stationery        40    350     April
8       1009    USB Drive  Accessories        35    400     March
4       1005     Keyboard  Accessories        12    800  February
0       1001     Notebook   Stationery        50    120   January
3       1004        Mouse  Accessories        15    350  February
```



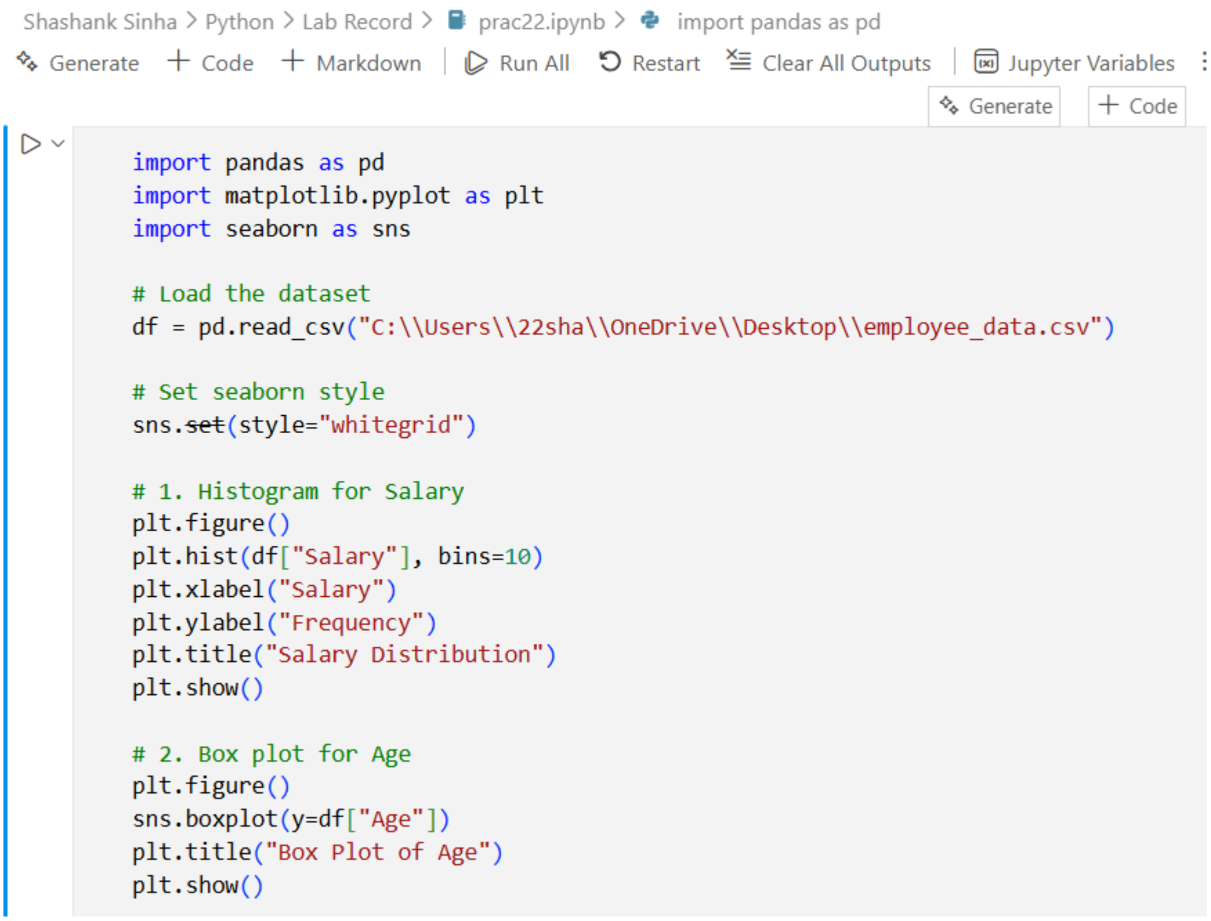
```
    Total_Revenue
12         100000
1           90000
14          72000
9           32500
5           25500
2           25000
7           24000
11          20000
13          18000
6           14400
10          14000
8           14000
4            9600
0            6000
3            5250
```

Practical-22

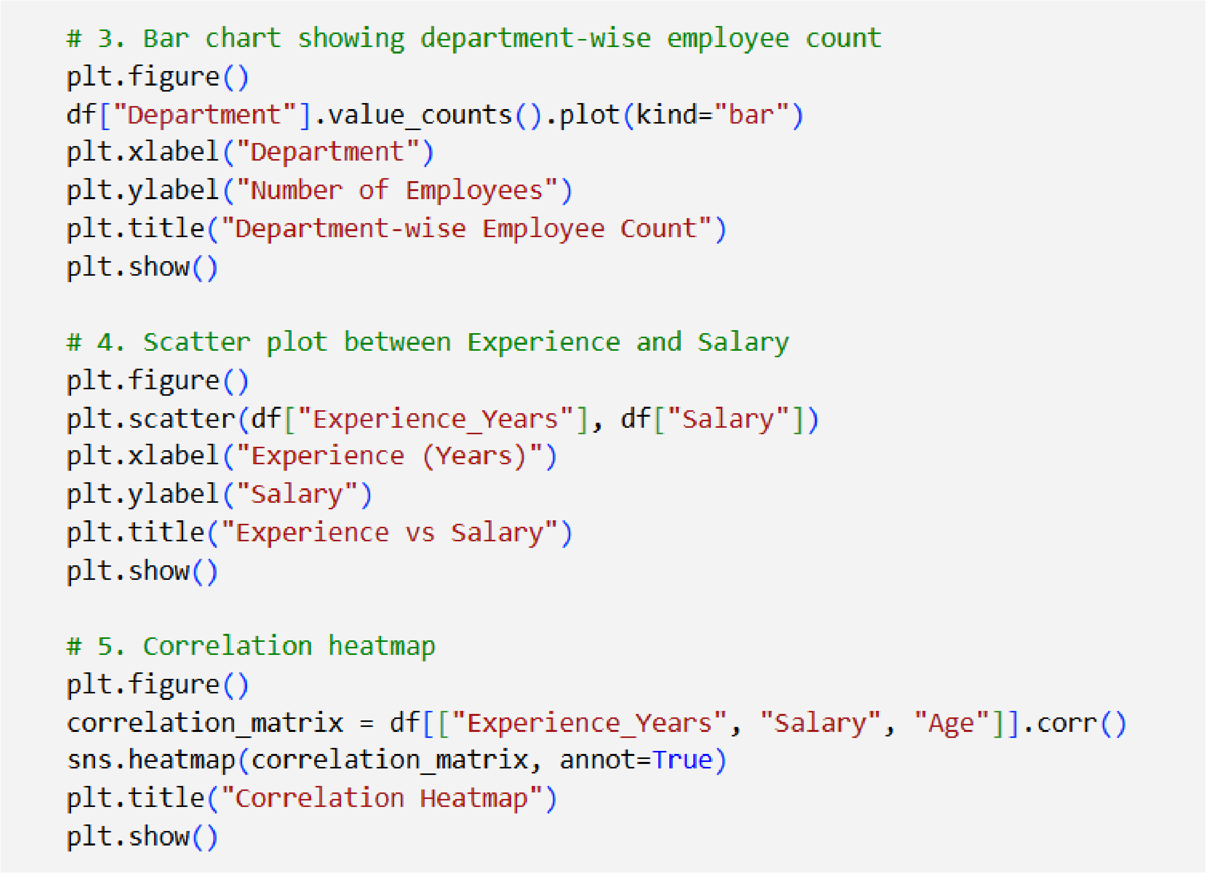
Aim:- Write a Python program to visualize an employee dataset using matplotlib and seaborn to understand data distribution and relationships.

- Plot a histogram for Salary
- Draw a box plot for Age
- Create a bar chart showing department-wise employee count
- Plot a scatter plot between Experience and Salary
- Display a correlation heatmap

Code:-

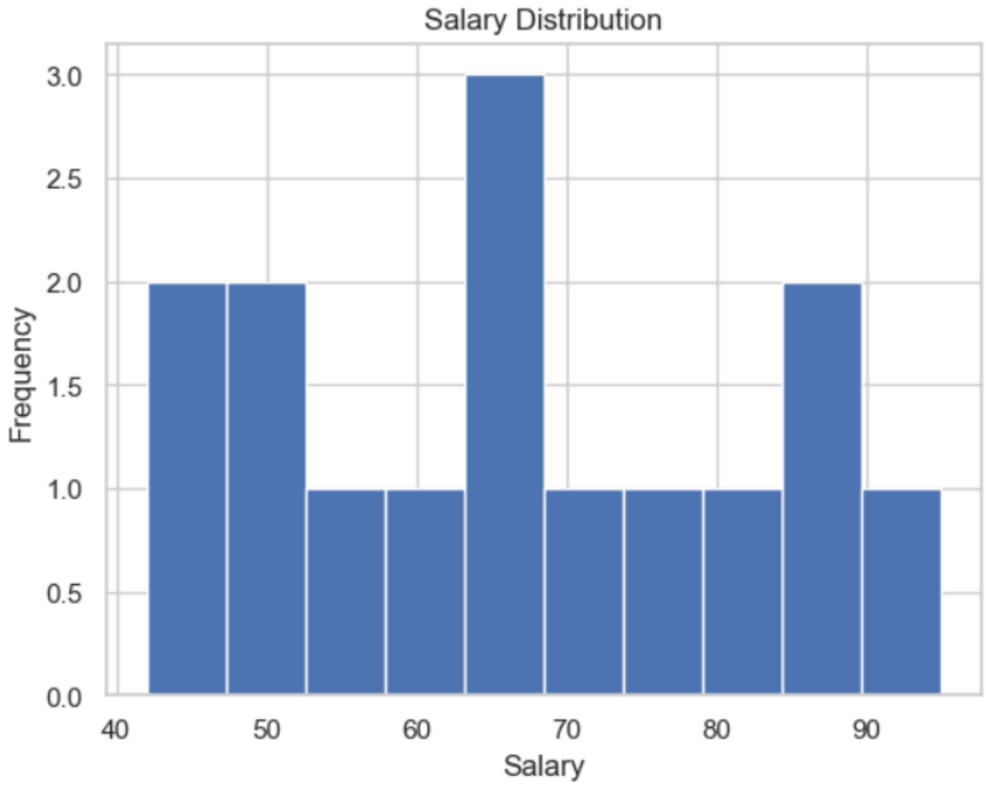


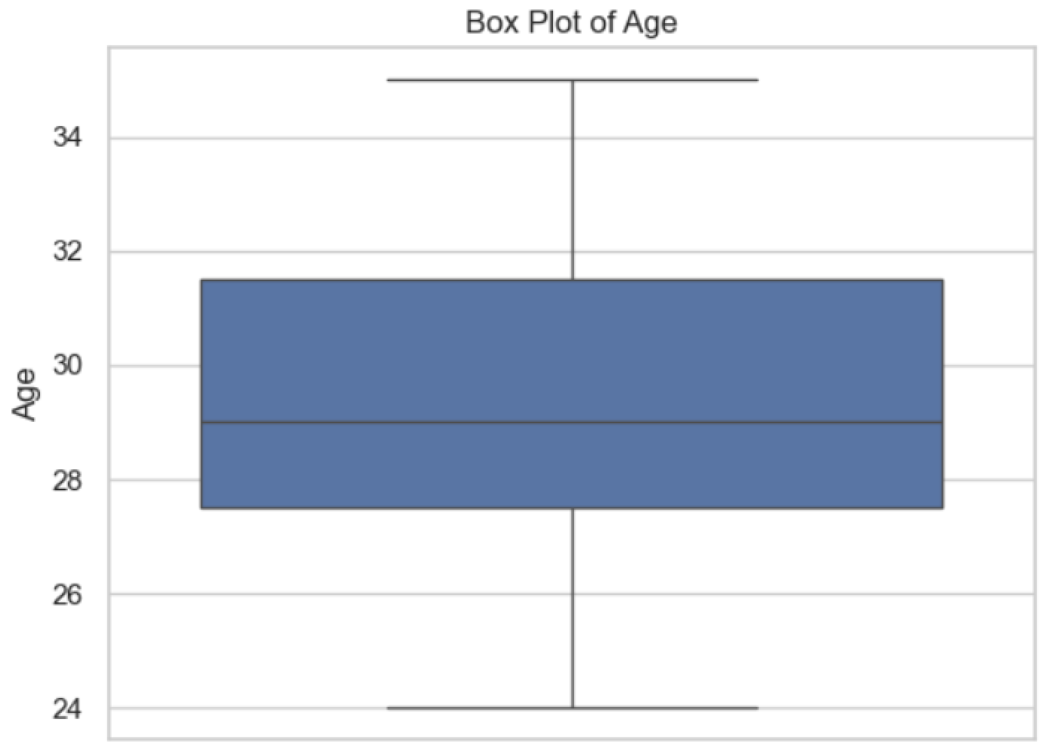
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("C:\\Users\\22sha\\OneDrive\\Desktop\\employee_data.csv")

# Set seaborn style
sns.set(style="whitegrid")

# 1. Histogram for Salary
plt.figure()
plt.hist(df["Salary"], bins=10)
plt.xlabel("Salary")
plt.ylabel("Frequency")
plt.title("Salary Distribution")
plt.show()

# 2. Box plot for Age
plt.figure()
sns.boxplot(y=df["Age"])
plt.title("Box Plot of Age")
plt.show()
```

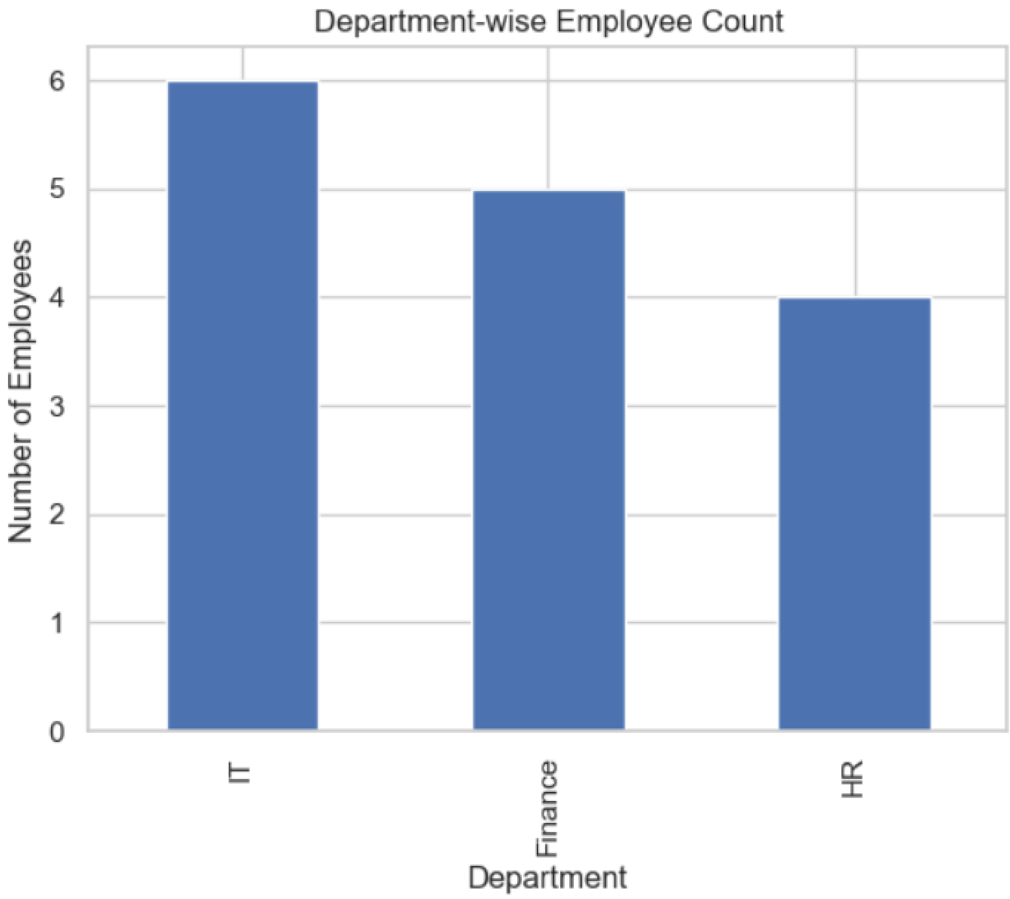


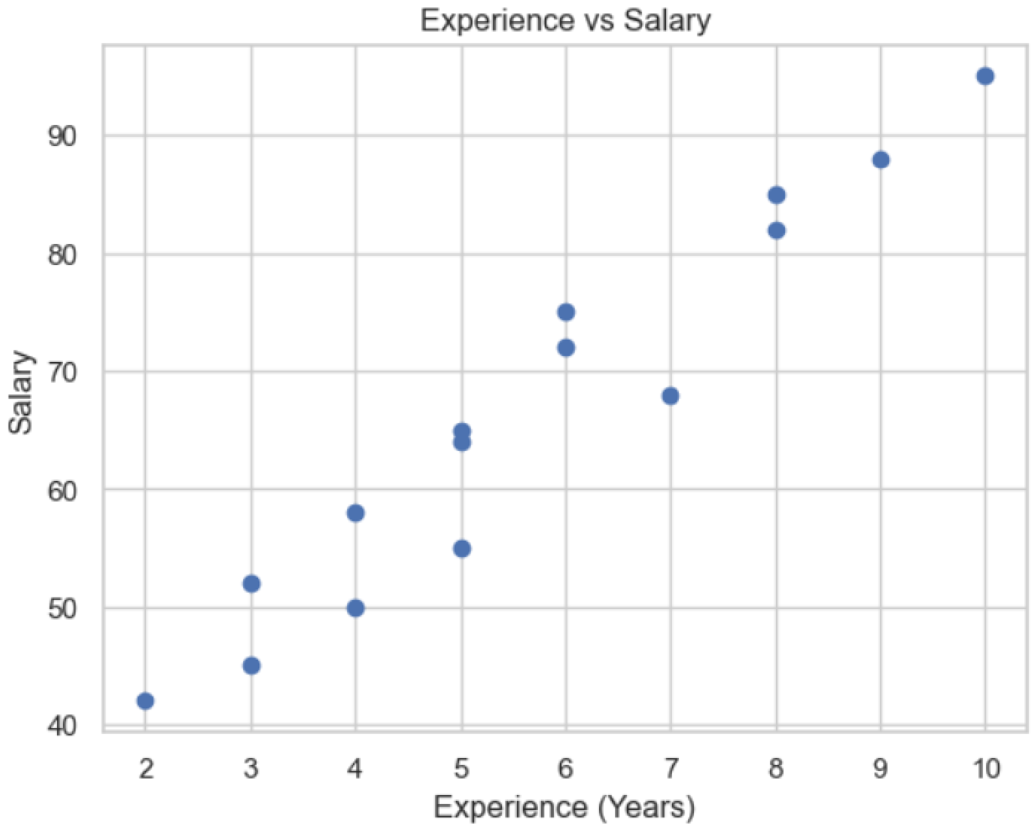
```
# 3. Bar chart showing department-wise employee count
plt.figure()
df["Department"].value_counts().plot(kind="bar")
plt.xlabel("Department")
plt.ylabel("Number of Employees")
plt.title("Department-wise Employee Count")
plt.show()

# 4. Scatter plot between Experience and Salary
plt.figure()
plt.scatter(df["Experience_Years"], df["Salary"])
plt.xlabel("Experience (Years)")
plt.ylabel("Salary")
plt.title("Experience vs Salary")
plt.show()

# 5. Correlation heatmap
plt.figure()
correlation_matrix = df[["Experience_Years", "Salary", "Age"]].corr()
sns.heatmap(correlation_matrix, annot=True)
plt.title("Correlation Heatmap")
plt.show()
```

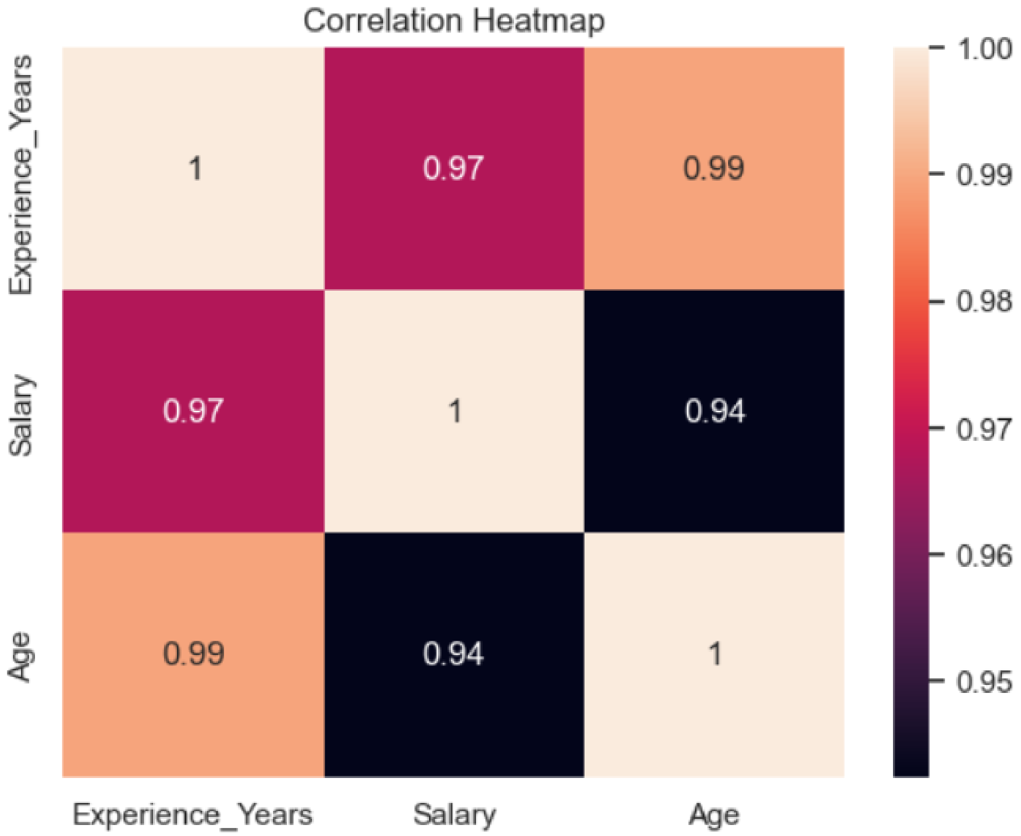
Output:-




Box Plot of Age
Age
34
32
30
28
26
24


Department-wise Employee Count
Number of Employees
6
5
4
3
2
1
0
IT
Finance
HR
Department


Experience vs Salary
Salary
90
80
70
60
50
40
2
3
4
5
6
7
8
9
10
Experience (Years)


Correlation Heatmap
Experience_Years
Salary
Age
1
0.97
0.99
0.97
1
0.94
0.99
0.94
1
Experience_Years
Salary
Age
1.00
0.99
0.98
0.97
0.96
0.95