A

PRACTICAL ASSIGNMENT

OF

"MCA 206 - Programming in Python"

Submitted in partial fulfillment of the requirements for the

Master of Computer Applications
MCA - IInd semester

From

PT. RAVISHANKAR SHUKLA UNIVERSITY SCHOOL OF STUDIES, RAIPUR(C.G)

Year: 2022-23

Guided By:-
Mr. Rahul Singh
(Ass. Professor ,C.S.&IT,PRSU)

Submitted by :-
Saumya Ranjan Nayak

Submitted to

(S. O. S. IN COMPUTER SCIENCE & IT, RAIPUR)
Pt.Ravishankar Shukla University Raipur (C.G.)

INDEX

S.No.	List of programs	Signature
25	Function to print prime numbers between I/O	
26	Function to take two integers and return sum & product	
27	Program to demonstrate positional arguments	
28	Program for keyword arguments of a function	
29	Program to demonstrate default arguments of function	
30	Program for variable length arguments	
31	Program to demonstrate keyword variable length	
32	Program to demonstrate global and local variables	
33	Program to find factorial using recursion	
34	Program to demonstrate lamda functions	
35	Program to demonstrate use of lambda functions and map	
36	Program to demonstrate the use of lambda function and reduce	
37	Program to demonstrate the various list procession methods	
38	Program to find biggest and smallest numbers in list	
39	Program to find common elements in two list	
40	Program to demonstrate the various tuple procession methods	
41	Program to demonstrate the use of dictionaries	
42	Program to find the number of occurrences of each letter in string using dictionaries	
43	Program to print the CWD and change the CWD	
44	Program that takes a list of words from the user and writes them into a file.	
45	Program that reads a file in text mode and counts the number of words that contain	
46	Programs to demonstrate the creation and use of “modules”	
47	Exception Handling Program that uses try and except	
48	Exception Handling Program that handles multiple types of exceptions	

S.No.	List of programs	Signature
49	Exception Handling Program that uses try, except and else.	
50	Exception Handling Program that uses finally with try.	
51	Program to create class	
52	Program that creates classes “Point” and “Rectangle”	
53	Program that creates a class Students which inherits the properties of the “Person”	
54	Program to demonstrate “Multiple Inheritance”	
55	Program to demonstrate “Method Overriding”	
56	Program to demonstrate “Method Overloading”	
57	Program to demonstrate “Operator Overloading” [+ and -] using a class “Book”	
58	Use the “turtle” module to draw concentric circles with different colours.	
59	Use the “turtle” module to print the multiplication table.	
60	Use the “turtle” module to draw (not write) your name.	

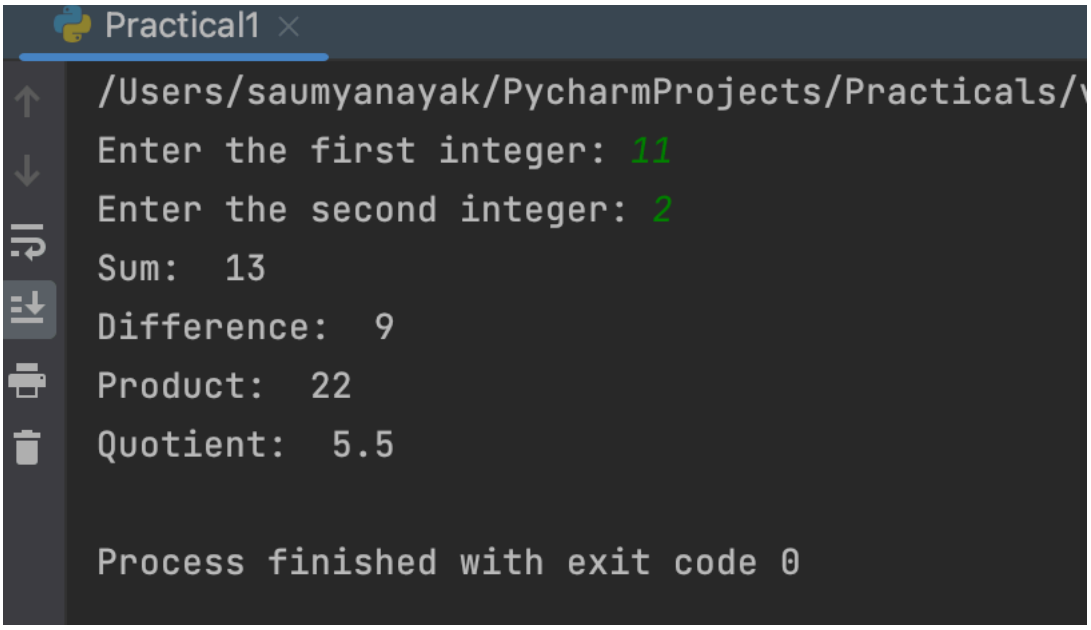
Practical No. 1

Program : Write a Python program to perform arithmetic operations on integers.

Code Section :

```
# Ask the user to input two integers
num1 = int(input("Enter the first integer: "))
num2 = int(input("Enter the second integer: "))

# Perform arithmetic operations
sum = num1 + num2
difference = num1 - num2
product = num1 * num2
quotient = num1 / num2

# Print the results
print("Sum: ", sum)
print("Difference: ", difference)
print("Product: ", product)
print("Quotient: ", quotient)
```

Output Section :



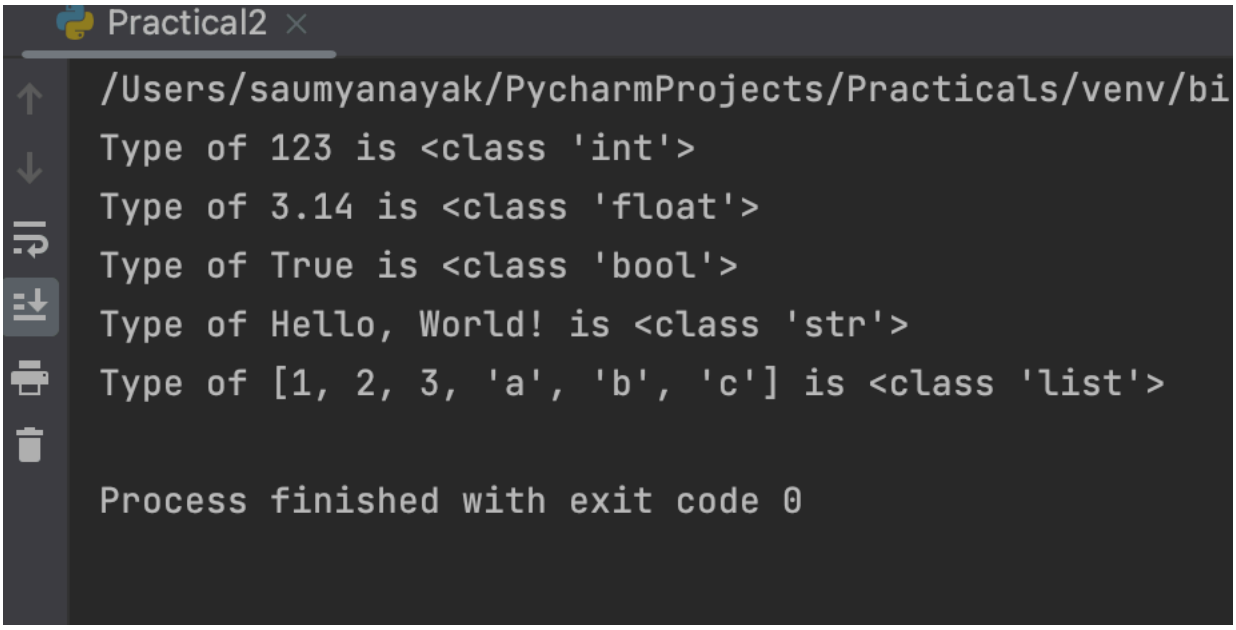
Practical No. 2

Program : Write a Python program to check and print the types of at least 5 different in- built objects.

Code Section :

```
# Check and print the types of different built-in objects


# Integer

num = 123

print("Type of", num, "is", type(num))


# Floating-point number

float_num = 3.14

print("Type of", float_num, "is", type(float_num))


# Boolean

is_true = True

print("Type of", is_true, "is", type(is_true))


# String

text = "Hello, World!"

print("Type of", text, "is", type(text))


# List

my_list = [1, 2, 3, "a", "b", "c"]

print("Type of", my_list, "is", type(my_list))
```

Output Section :



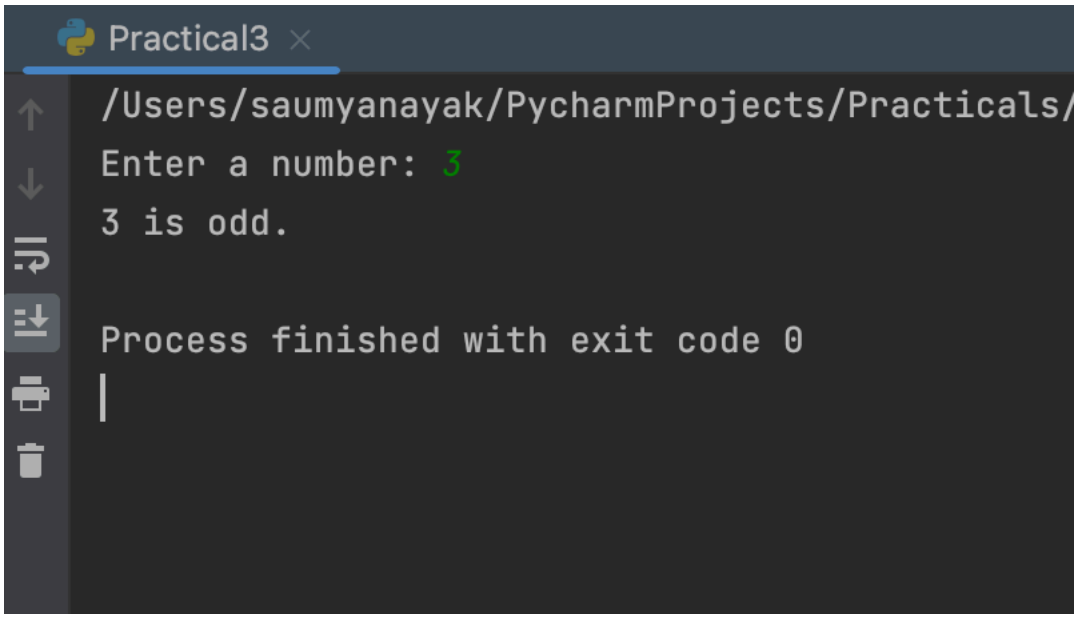
Practical No. 3

Program : Write a Python program to check if a number is EVEN or ODD.

Code Section :

```
# Ask the user to input a number
num = int(input("Enter a number: "))


# Check if the number is even or odd
if num % 2 == 0:
    print(num, "is even.")
else:
    print(num, "is odd.")
```

Output Section :



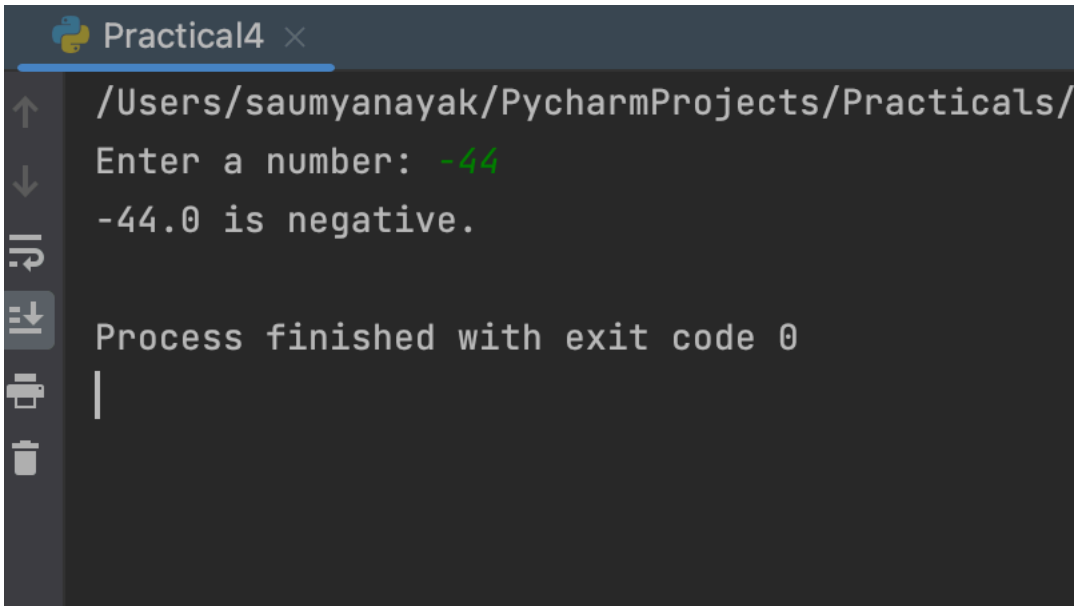
Practical No. 4

Program : Write a Python program to check if a number is Positive, Negative or Zero.

Code Section :

```
# Ask the user to input a number

num = float(input("Enter a number: "))


# Check if the number is positive, negative, or zero

if num > 0:

    print(num, "is positive.")

elif num < 0:

    print(num, "is negative.")

else:

    print(num, "is zero.")
```

Output Section :



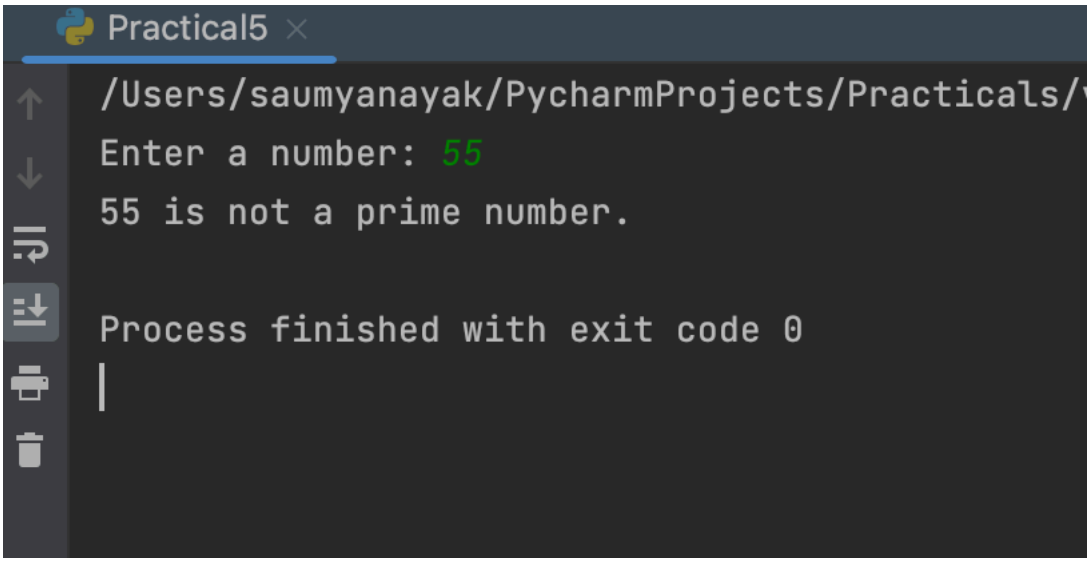
Practical No. 5

Program : Write a Python program to check if a number is PRIME or NOT.

Code Section :

```
# Ask the user to input a number
num = int(input("Enter a number: "))


# Check if the number is prime
if num > 1:
   # check for factors
   for i in range(2, num):
      if num % i == 0:
         print(num, "is not a prime number.")
         break
   else:
      print(num, "is a prime number.")
else:
   print(num, "is not a prime number.")
```

Output Section :



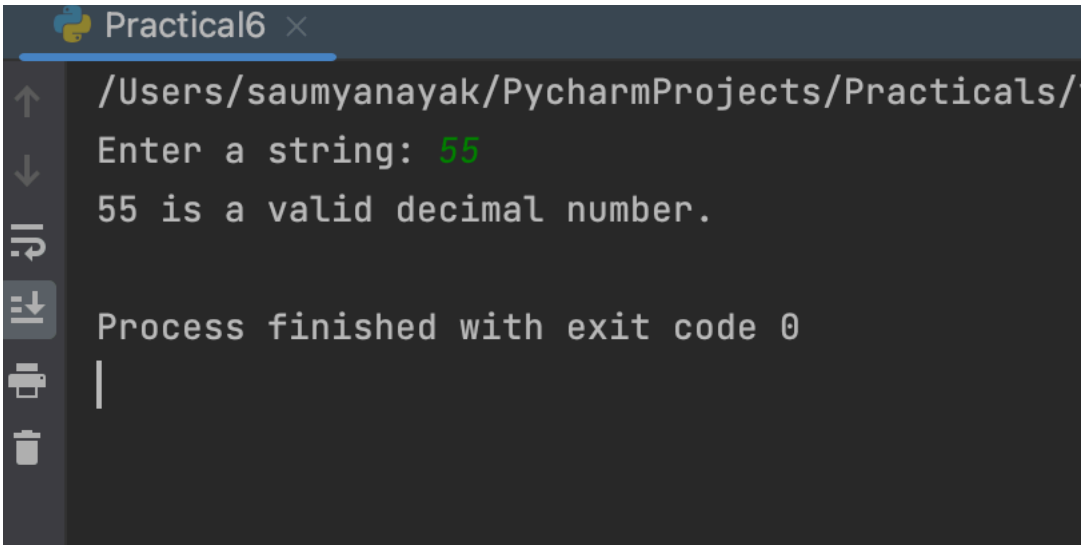
Practical No. 6

Program : Write a Python program to check whether a string entered by the user is a valid decimal number or not.

Code Section :

```
# Ask the user to input a string

str = input("Enter a string: ")


# Try to convert the string to a float

try:

    num = float(str)

    print(str, "is a valid decimal number.")

except ValueError:

    print(str, "is not a valid decimal number.")
```

Output Section :



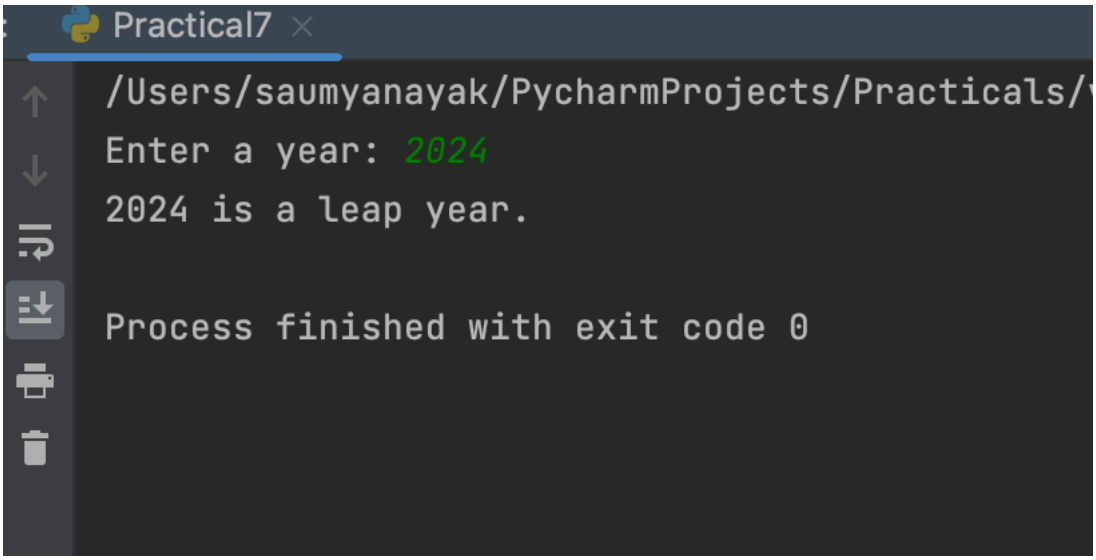
Practical No. 7

Program : Write a Python program to check if a year entered by the user is a Leap Year or NOT.

Code Section :

```
# Ask the user to input a year

year = int(input("Enter a year: "))


# Check if the year is a leap year

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

    print(year, "is a leap year.")

else:

    print(year, "is not a leap year.")
```

Output Section :



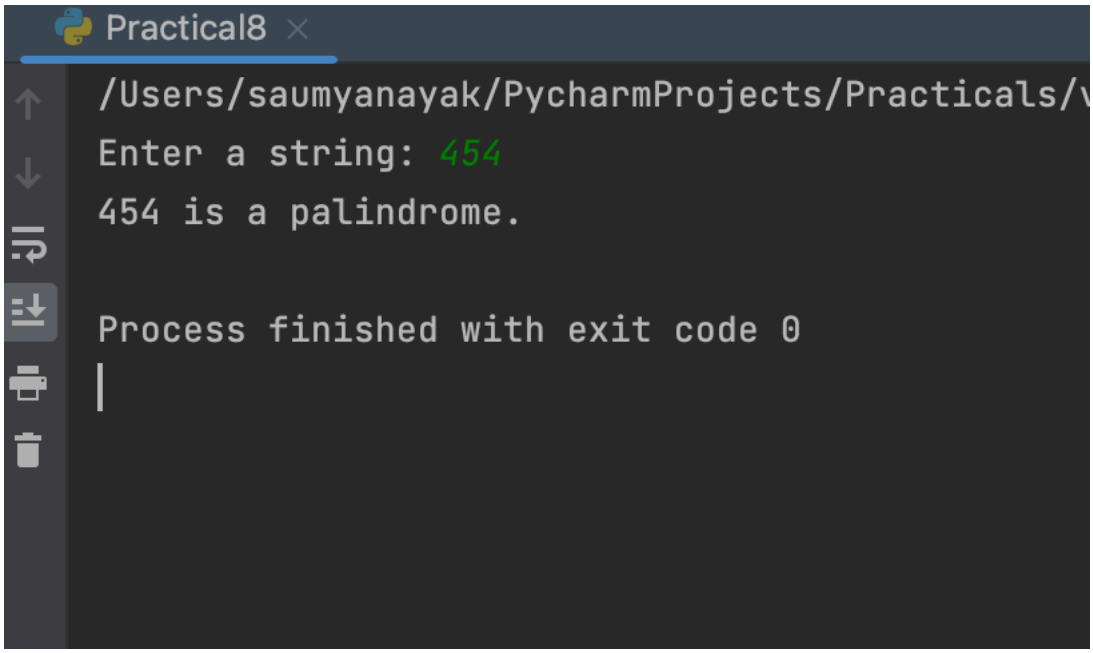
Practical No. 8

Program : Write a Python program to check whether a string entered by the user is a palindrome or not.

Code Section :

```
# Ask the user to input a string

str = input("Enter a string: ")


# Reverse the string

rev_str = str[::-1]


# Check if the string is a palindrome

if str == rev_str:

    print(str, "is a palindrome.")

else:

    print(str, "is not a palindrome.")
```

Output Section :



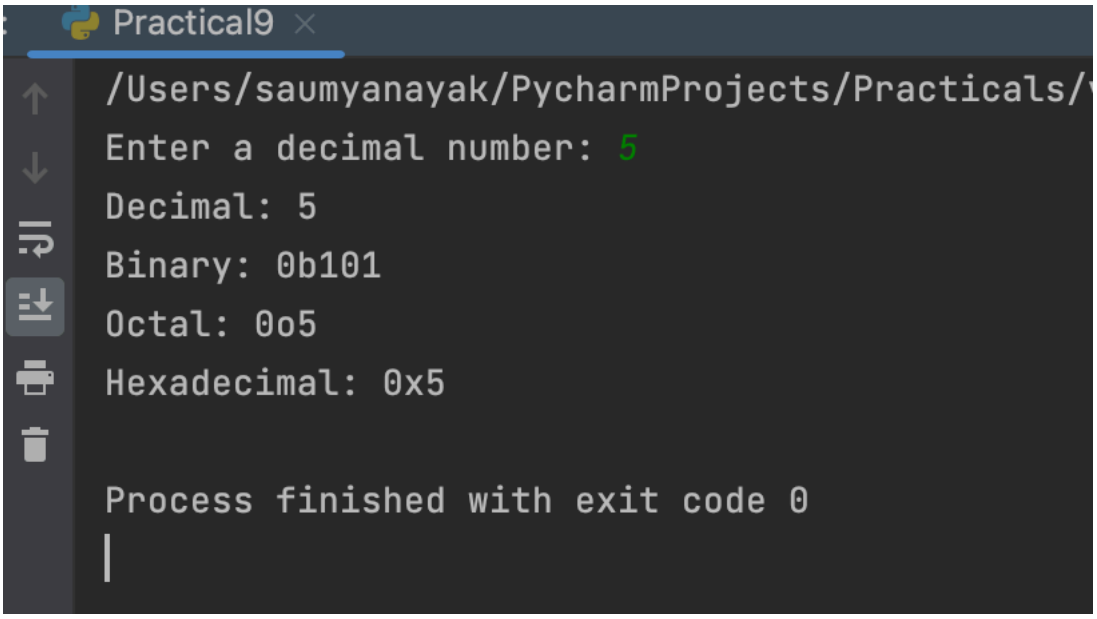
Practical No. 9

Program : Write a Python program to get a Decimal number from user and convert it into Binary, Octal and Hexadecimal.

Code Section :

```
# Ask the user to input a decimal number

decimal = int(input("Enter a decimal number: "))


# Convert the decimal number to binary, octal, and hexadecimal

binary = bin(decimal)

octal = oct(decimal)

hexadecimal = hex(decimal)


# Print the result

print("Decimal:", decimal)

print("Binary:", binary)

print("Octal:", octal)

print("Hexadecimal:", hexadecimal)
```

Output Section :



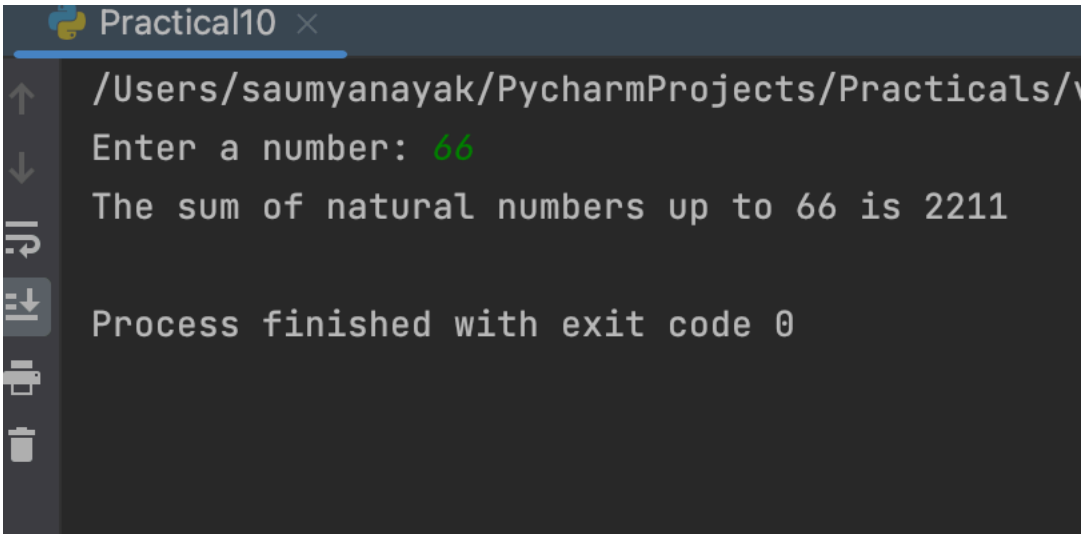
Practical No. 10

Program : Write a Python program to find sum of natural numbers, up to N.

Code Section :

```
# Ask the user to input a number
N = int(input("Enter a number: "))


# Initialize the sum to 0
sum = 0


# Use a loop to iterate from 1 to N and add each number to the sum
for i in range(1, N+1):
    sum += i


# Print the result
print("The sum of natural numbers up to", N, "is", sum)
```

Output Section :



Practical No. 11

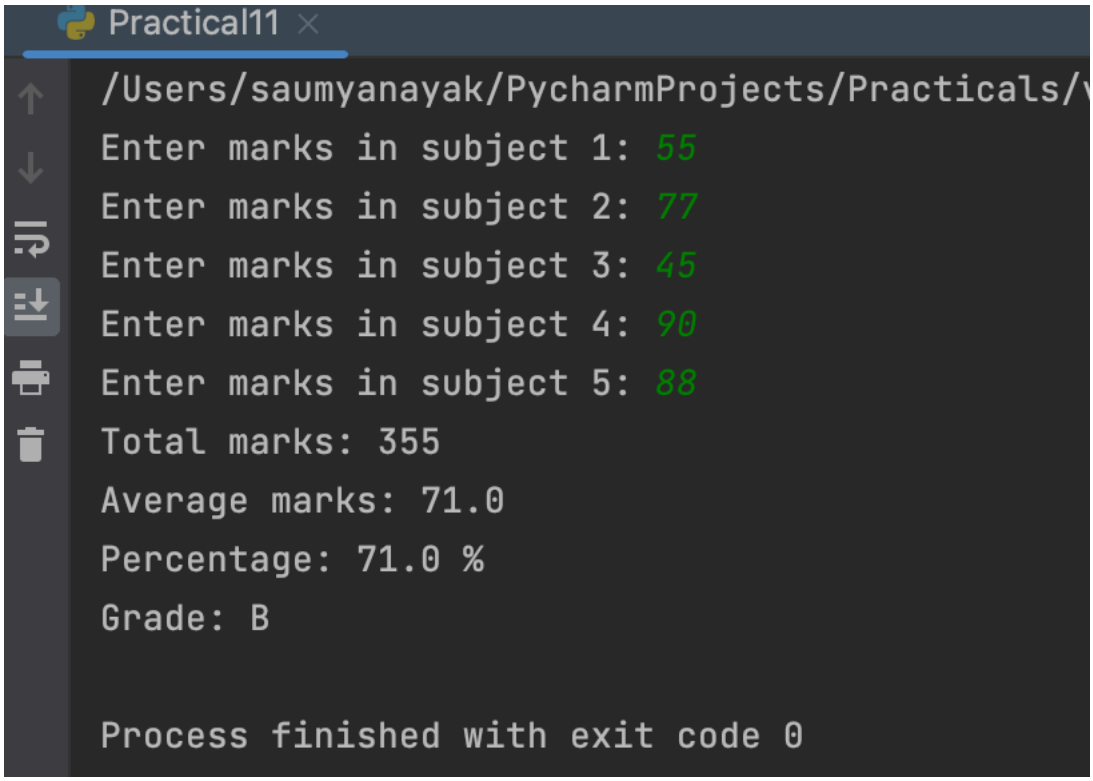
Program : Write a Python program to get marks in five subjects from user and calculate average marks, percentage and grade of a student.

Code Section :

```
# Ask the user to input marks in five subjects
subject1 = int(input("Enter marks in subject 1: "))
subject2 = int(input("Enter marks in subject 2: "))
subject3 = int(input("Enter marks in subject 3: "))
subject4 = int(input("Enter marks in subject 4: "))
subject5 = int(input("Enter marks in subject 5: "))


# Calculate the total marks and average marks
total_marks = subject1 + subject2 + subject3 + subject4 + subject5
average_marks = total_marks / 5


# Calculate the percentage
percentage = (total_marks / 500) * 100


# Determine the grade based on percentage
if percentage >= 90:
    grade = 'A+'
elif percentage >= 80:
    grade = 'A'
elif percentage >= 70:
    grade = 'B'
elif percentage >= 60:
```

```
   grade = 'C'
elif percentage >= 50:
    grade = 'D'
else:
    grade = 'Fail'

# Print the result
print("Total marks:", total_marks)
print("Average marks:", average_marks)
print("Percentage:", percentage, "%")
print("Grade:", grade)
```

Output Section :



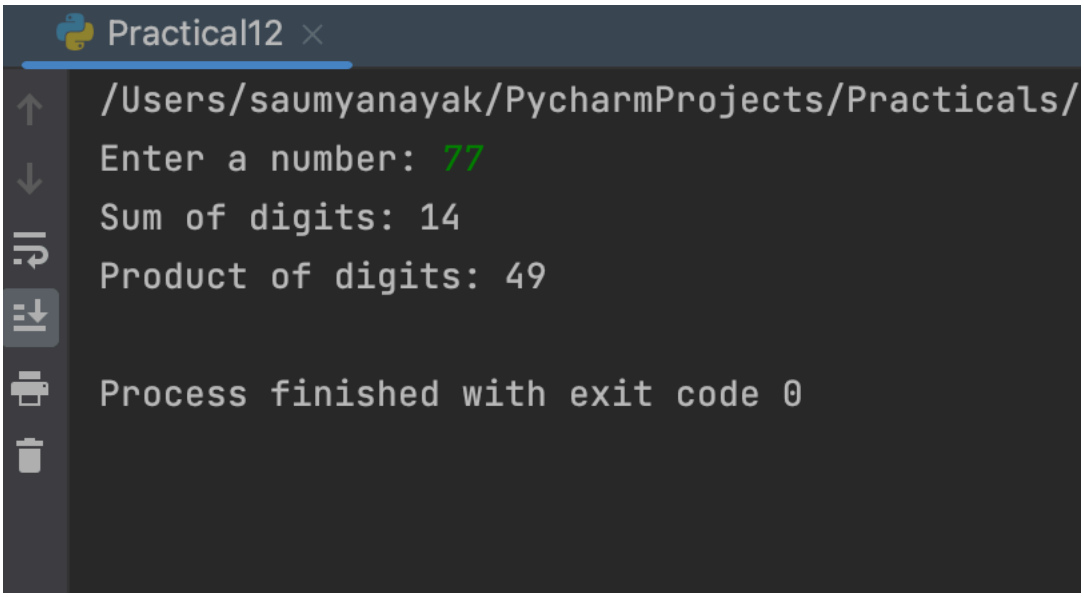
Practical No. 12

Program : Write a Python program to get a number and find the sum and product of its digits.

Code Section :

```
# Ask the user to input a number
number = int(input("Enter a number: "))


# Initialize the sum and product to 0 and 1 respectively
sum = 0
product = 1


# Use a loop to iterate over each digit of the number
while number > 0:
    # Get the last digit of the number
    digit = number % 10


    # Add the digit to the sum and multiply it to the product
    sum += digit
    product *= digit


    # Remove the last digit from the number
    number //= 10


# Print the result
print("Sum of digits:", sum)
print("Product of digits:", product)
```

Output Section :



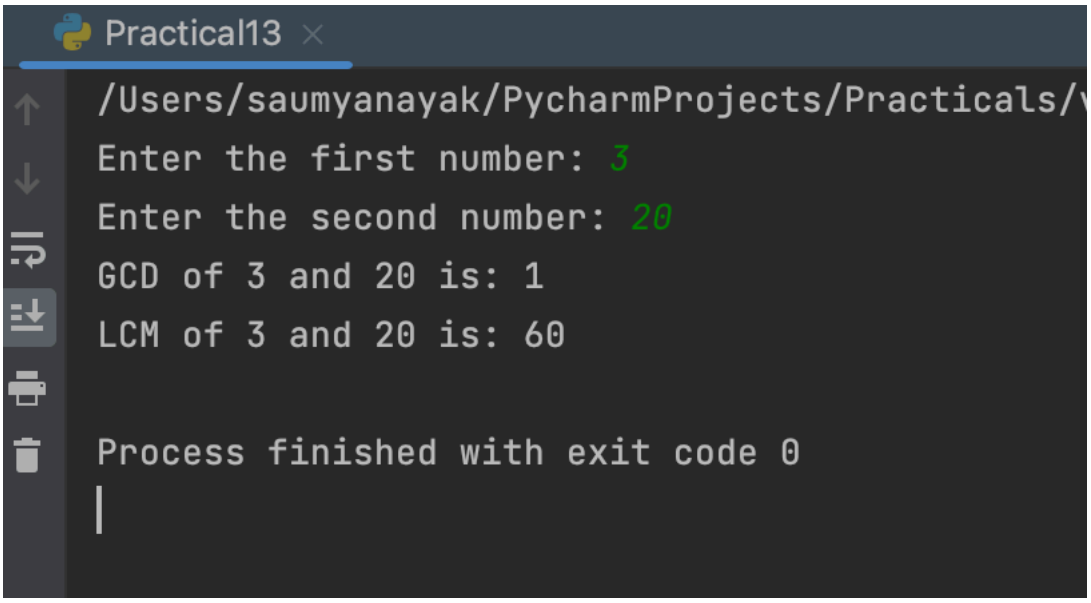
Practical No. 13

Program : Write a Python program to get two integers and find their GCD and LCM.

Code Section :

```
# Function to find GCD of two numbers
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)

# Function to find LCM of two numbers
def lcm(a, b):
    return (a * b) // gcd(a, b)

# Get two integers from the user
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

# Calculate the GCD and LCM
gcd_value = gcd(num1, num2)
lcm_value = lcm(num1, num2)

# Print the results
print("GCD of", num1, "and", num2, "is:", gcd_value)
print("LCM of", num1, "and", num2, "is:", lcm_value)
```

Output Section :



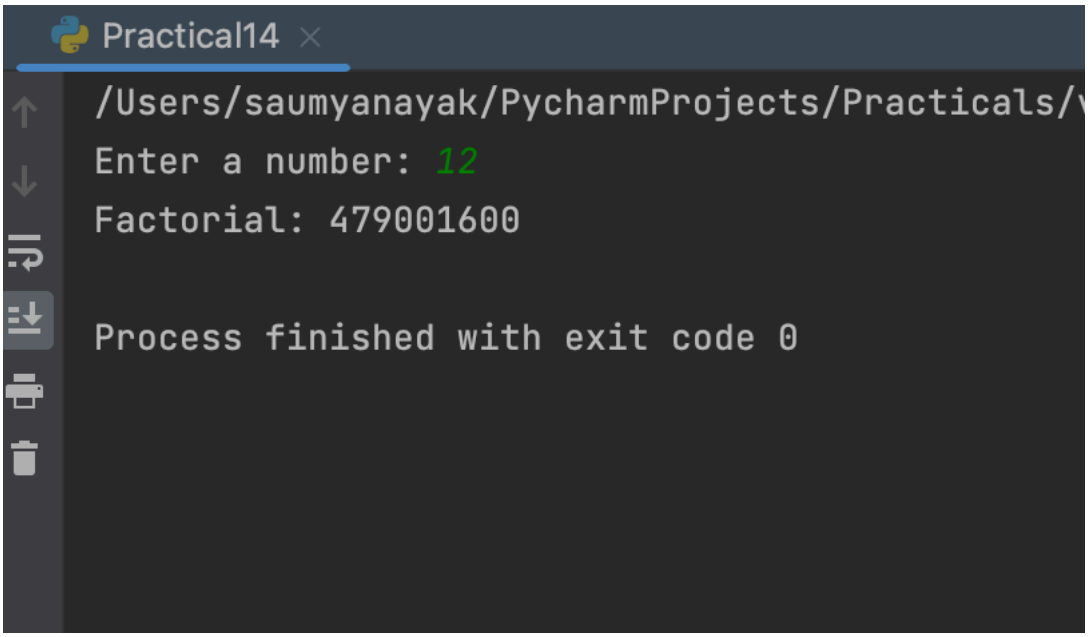
Practical No. 14

Program : Write a Python program to find factorial of a number using while loop.

Code Section :

```
# Get the input from the user
num = int(input("Enter a number: "))


# Initialize the factorial to 1
factorial = 1


# Use a while loop to calculate the factorial
while num > 0:
    factorial *= num
    num -= 1


# Print the result
print("Factorial:", factorial)
```

Output Section :



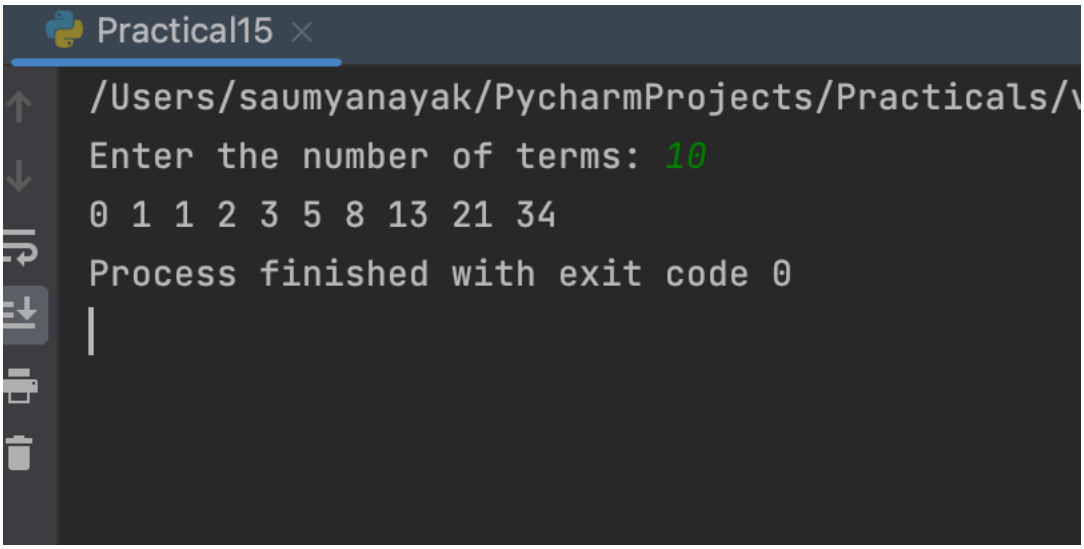
Practical No. 15

Program : Write a Python program to print Fibonacci series up to N terms.

Code Section :

```
# Get the input from the user
num_terms = int(input("Enter the number of terms: "))


# Initialize the first two terms of the series
a = 0
b = 1


# Use a loop to generate the series
for i in range(num_terms):
    print(a, end=" ")
    c = a + b
    a = b
    b = c
```

Output Section :



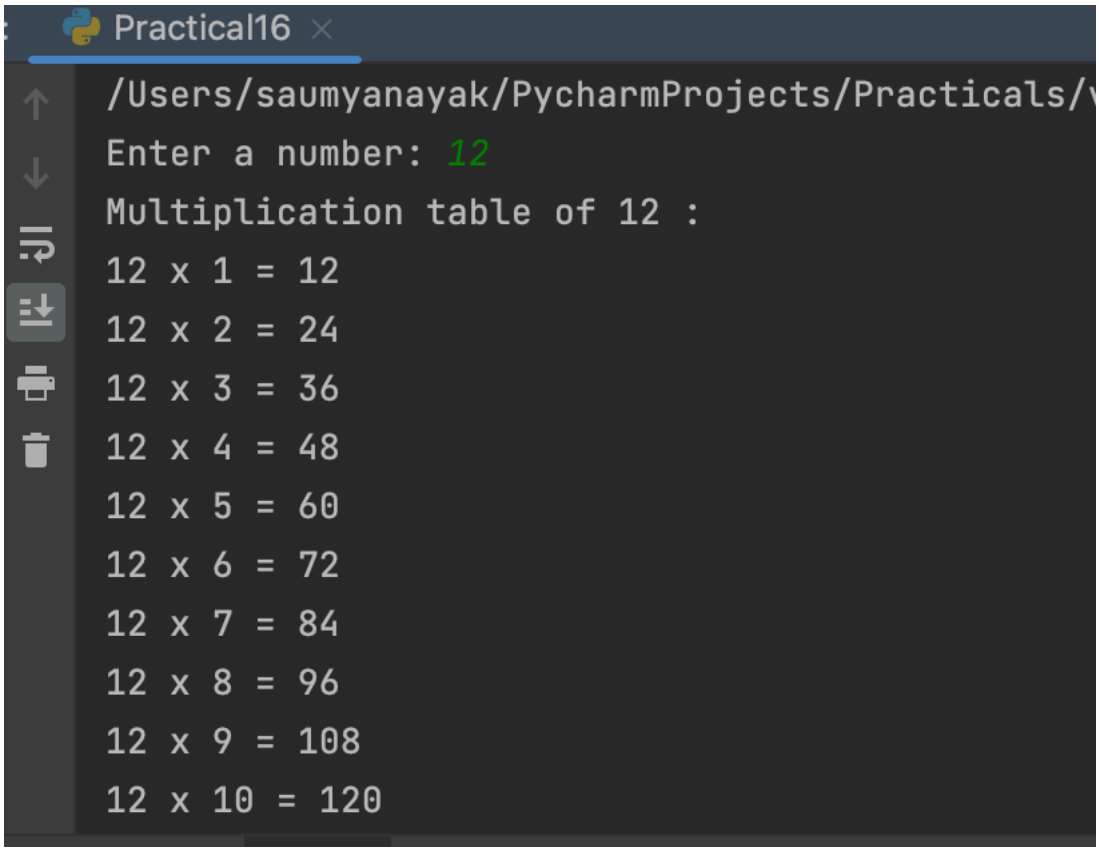
Practical No. 16

Program : Write a Python program to print multiplication table.

Code Section :

```
# Get the input from the user
num = int(input("Enter a number: "))


# Print the multiplication table
print("Multiplication table of", num, ":")


for i in range(1, 11):
    print(num, "x", i, "=", num * i)
```

Output Section :



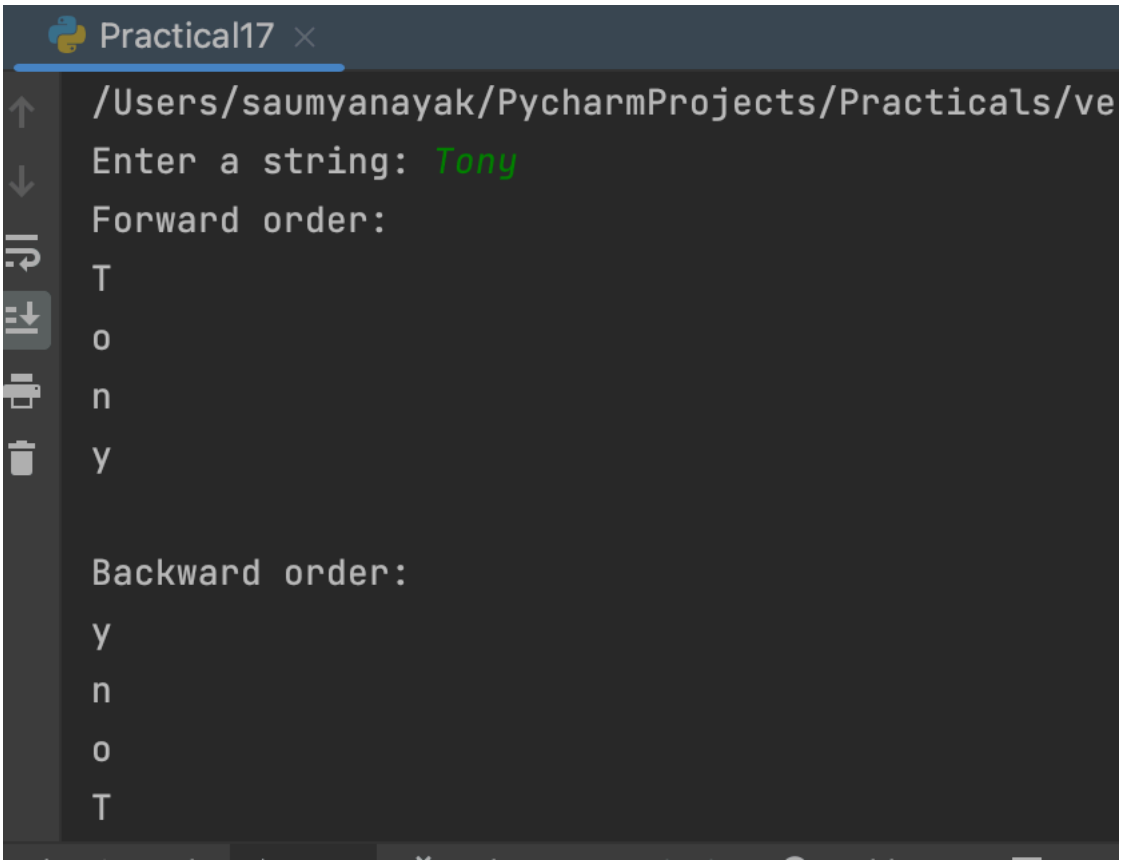
Practical No. 17

Program : Write a Python program to access each element of a string in forward and backward orders using the 'while' loop.

Code Section :

```
# Get the input from the user

string = input("Enter a string: ")


# Access each element of the string in forward order

print("Forward order:")


i = 0

while i < len(string):

    print(string[i])

    i += 1


# Access each element of the string in backward order

print("\nBackward order:")


i = len(string) - 1

while i >= 0:

    print(string[i])

    i -= 1
```

Output Section :



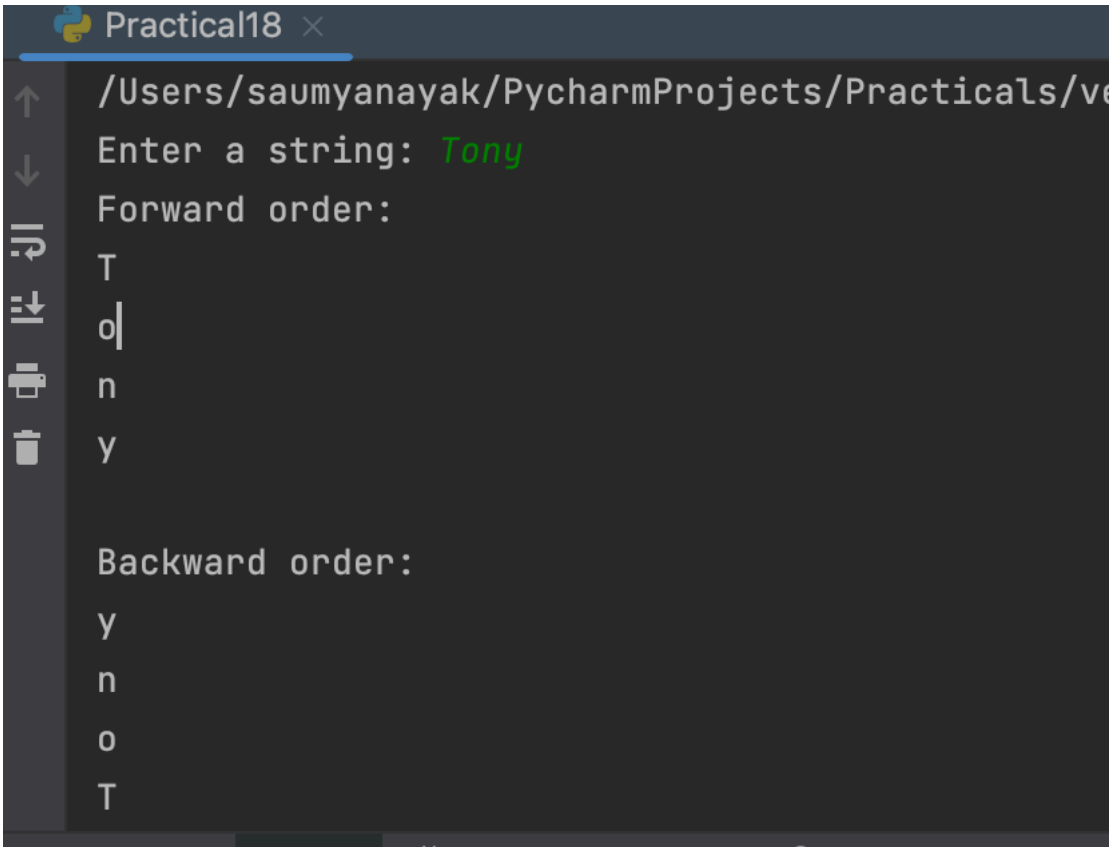
Practical No. 18

Program : Write a Python program to access each element of a string in forward and backward orders using the ‘for’ loop.

Code Section :

```
# Get the input from the user

string = input("Enter a string: ")


# Access each element of the string in forward order

print("Forward order:")


for char in string:

    print(char)


# Access each element of the string in backward order

print("\nBackward order:")


for char in reversed(string):

    print(char)
```

Output Section :



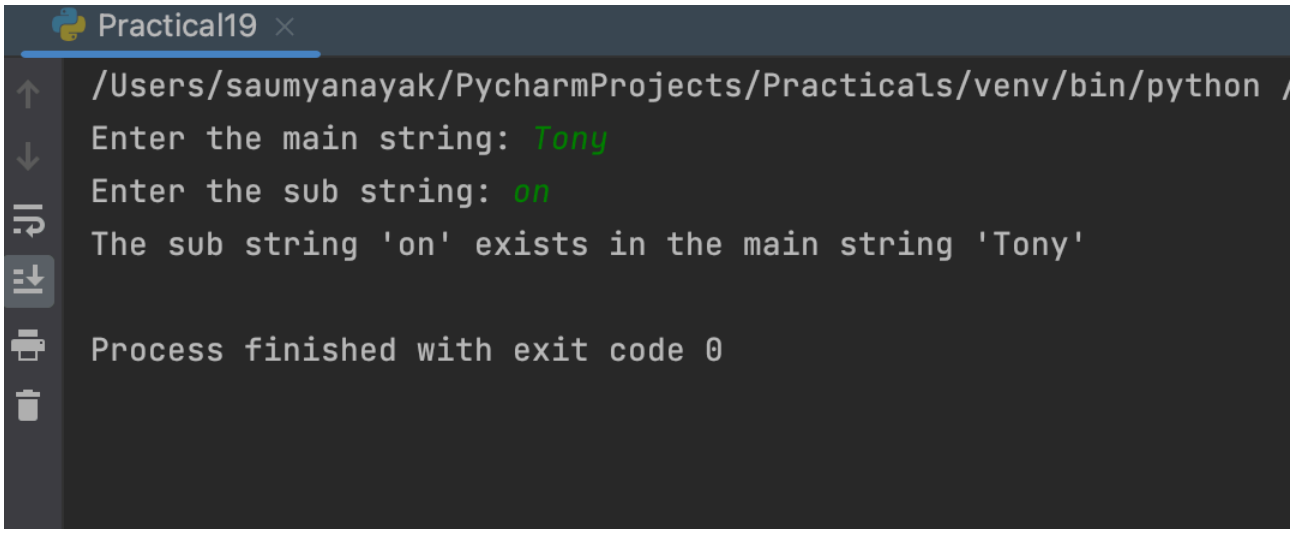
Practical No. 19

Program : Write a Python program to find whether a substring exists in main string or not.

Code Section :

```
# Get the input from the user

main_string = input("Enter the main string: ")

sub_string = input("Enter the sub string: ")


# Check if the sub string exists in the main string

if sub_string in main_string:

    print(f"The sub string '{sub_string}' exists in the main string '{main_string}'")

else:

    print(f"The sub string '{sub_string}' does not exist in the main string '{main_string}'")
```

Output Section :



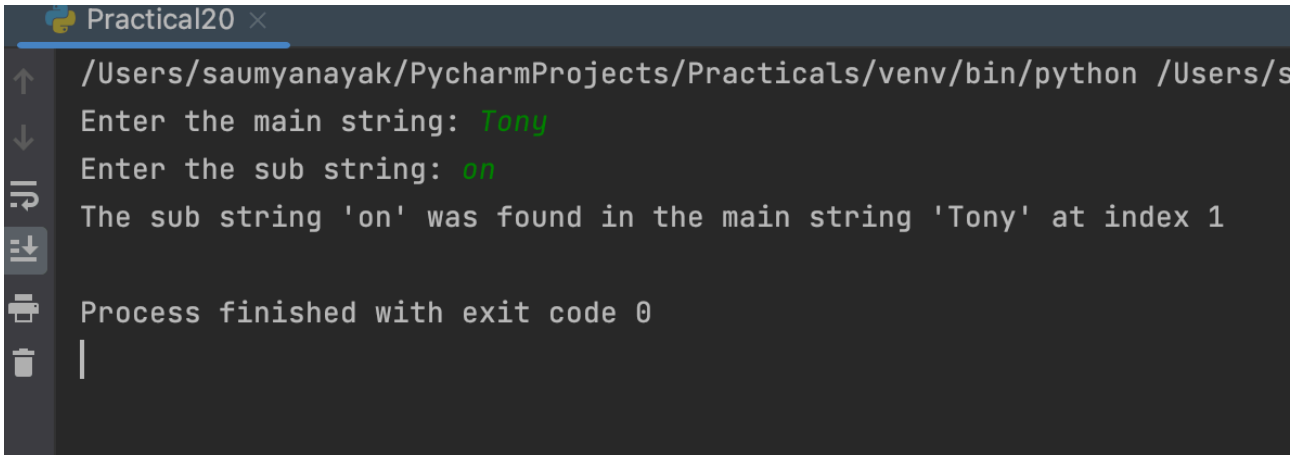
Practical No. 20

Program : Write a Python program to find the first occurrence of a substring in the main string.

Code Section :

```
# Get the input from the user

main_string = input("Enter the main string: ")

sub_string = input("Enter the sub string: ")


# Find the first occurrence of the sub string in the main string

index = main_string.find(sub_string)


# Check if the sub string was found in the main string

if index != -1:

    print(f"The sub string '{sub_string}' was found in the main string '{main_string}' at index {index}")

else:

    print(f"The sub string '{sub_string}' was not found in the main string '{main_string}'")
```

Output Section :



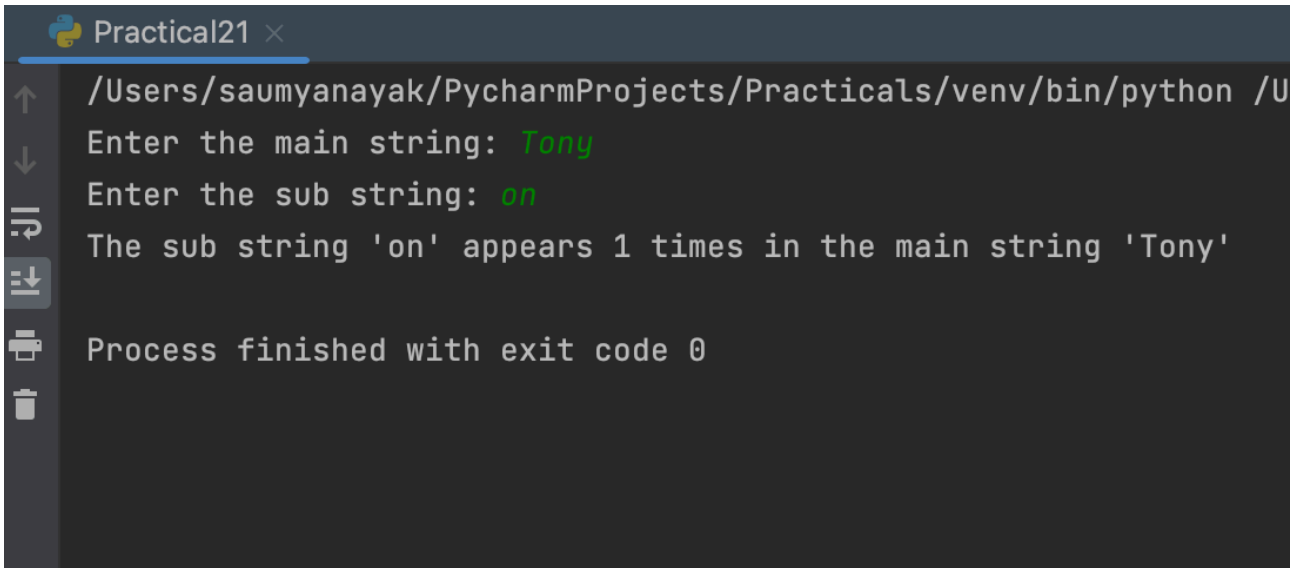
Practical No. 21

Program : Write a Python program to count the number of times a substring appears in the main string.

Code Section :

```
# Get the input from the user

main_string = input("Enter the main string: ")

sub_string = input("Enter the sub string: ")


# Count the number of occurrences of the sub string in the main string

count = main_string.count(sub_string)


# Print the result

print(f"The sub string '{sub_string}' appears {count} times in the main string '{main_string}'")
```

Output Section :



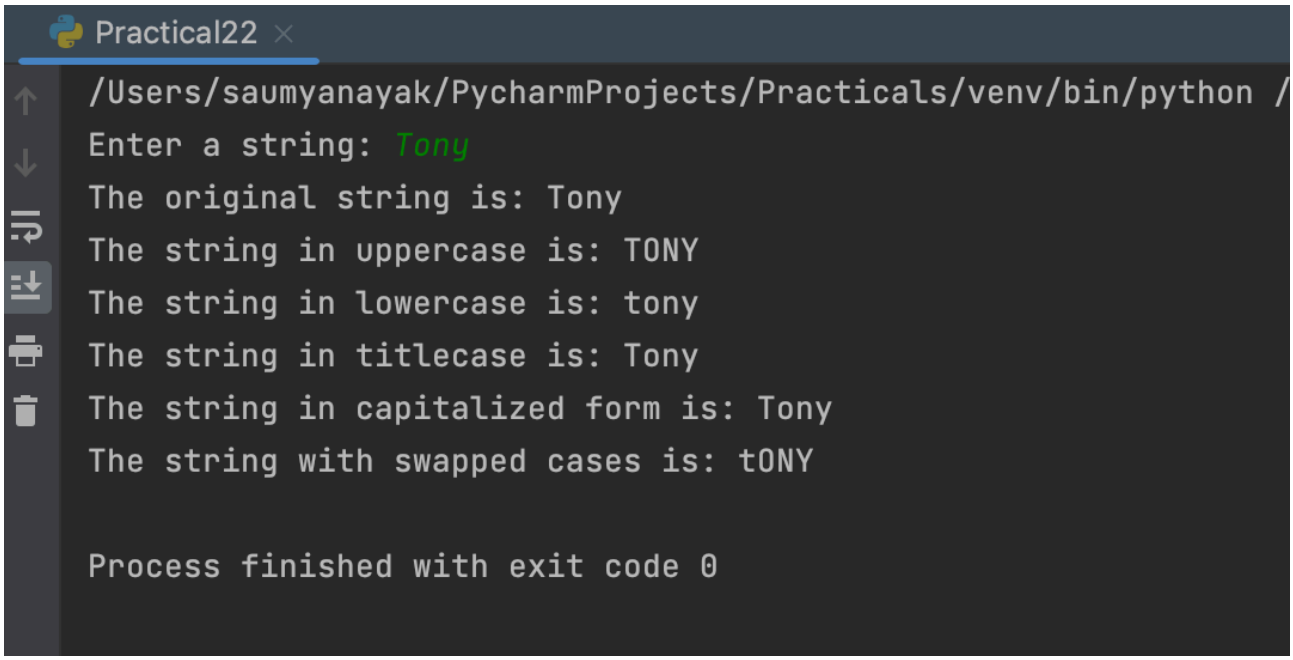
Practical No. 22

Program : Write a Python program to demonstrate the use of all “casing” methods and display a string in different cases.

Code Section :

```
# Get the input from the user

input_string = input("Enter a string: ")


# Convert the string to different cases using casing methods

uppercase_string = input_string.upper()

lowercase_string = input_string.lower()

titlecase_string = input_string.title()

capitalized_string = input_string.capitalize()

swapcase_string = input_string.swapcase()


# Print the string in different cases

print(f"The original string is: {input_string}")

print(f"The string in uppercase is: {uppercase_string}")

print(f"The string in lowercase is: {lowercase_string}")

print(f"The string in titlecase is: {titlecase_string}")

print(f"The string in capitalized form is: {capitalized_string}")

print(f"The string with swapped cases is: {swapcase_string}")
```

Output Section :



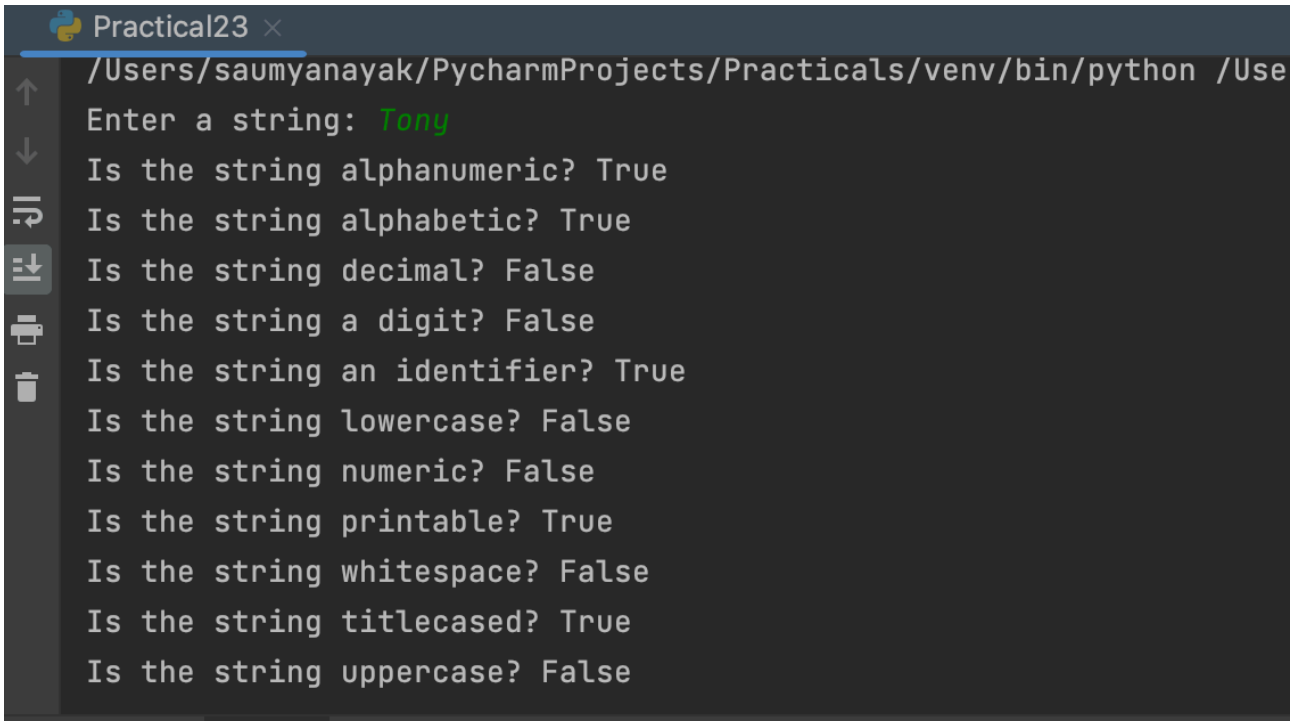
Practical No. 23

Program : Write a Python program to demonstrate the use of all string testing {isXXX()} methods.

Code Section :

```
# Get the input from the user

input_string = input("Enter a string: ")


# Use string testing methods to check different properties of the string

print(f"Is the string alphanumeric? {input_string.isalnum()}")

print(f"Is the string alphabetic? {input_string.isalpha()}")

print(f"Is the string decimal? {input_string.isdecimal()}")

print(f"Is the string a digit? {input_string.isdigit()}")

print(f"Is the string an identifier? {input_string.isidentifier()}")

print(f"Is the string lowercase? {input_string.islower()}")

print(f"Is the string numeric? {input_string.isnumeric()}")

print(f"Is the string printable? {input_string.isprintable()}")

print(f"Is the string whitespace? {input_string.isspace()}")

print(f"Is the string titlecased? {input_string.istitle()}")

print(f"Is the string uppercase? {input_string.isupper()}")
```

Output Section :



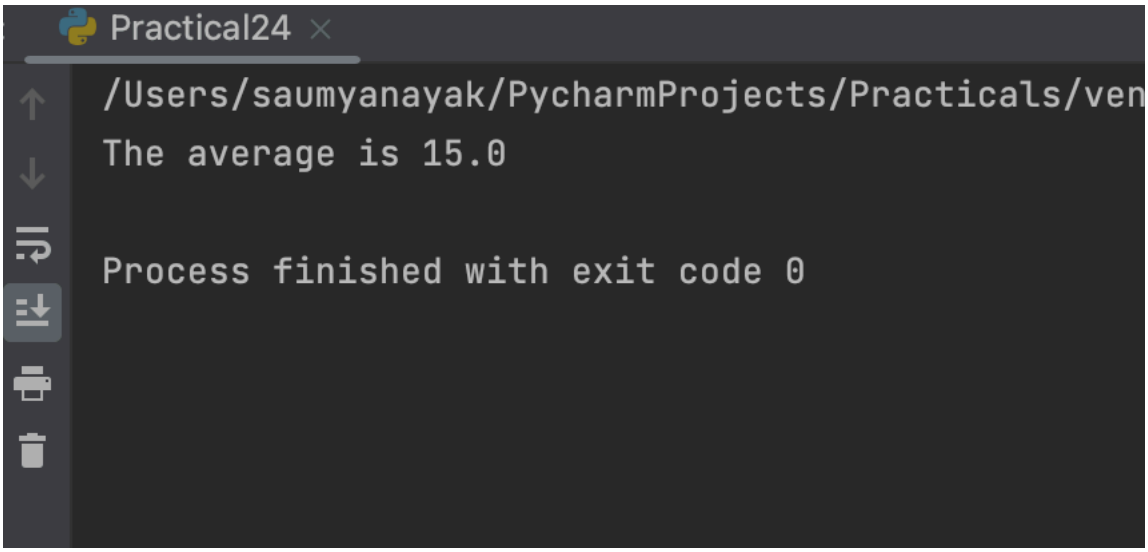
Practical No. 24

Program : Write a Python function to take a list of integers as input and return the average.

Code Section :

```
def find_average(lst):
    if len(lst) == 0:
        return 0
    else:
        return sum(lst) / len(lst)


lst = [5, 10, 15, 20, 25]
average = find_average(lst)
print(f"The average is {average}")
```

Output Section :



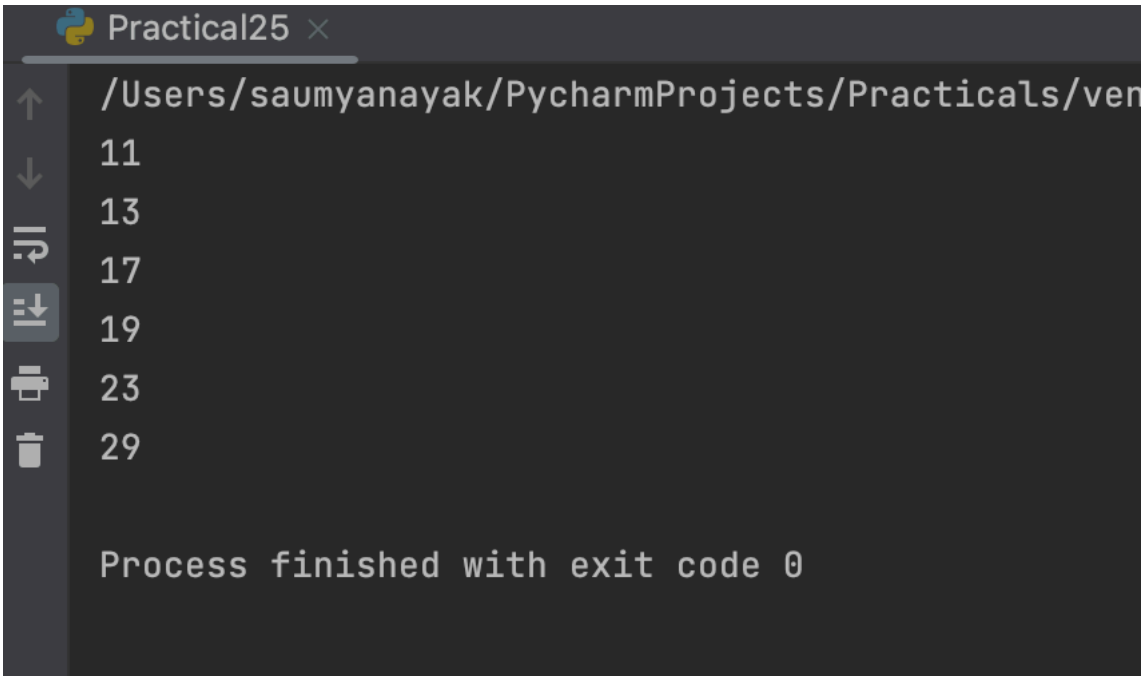
Practical No. 25

Program : Write a Python function to take two distinct integers as input and print all prime numbers between them.

Code Section :

```
def print_primes_between(num1, num2):
    for num in range(num1, num2 + 1):
        if num > 1:
            for i in range(2, num):
                if (num % i) == 0:
                    break
            else:
                print(num)

print_primes_between(10, 30)
```

Output Section :



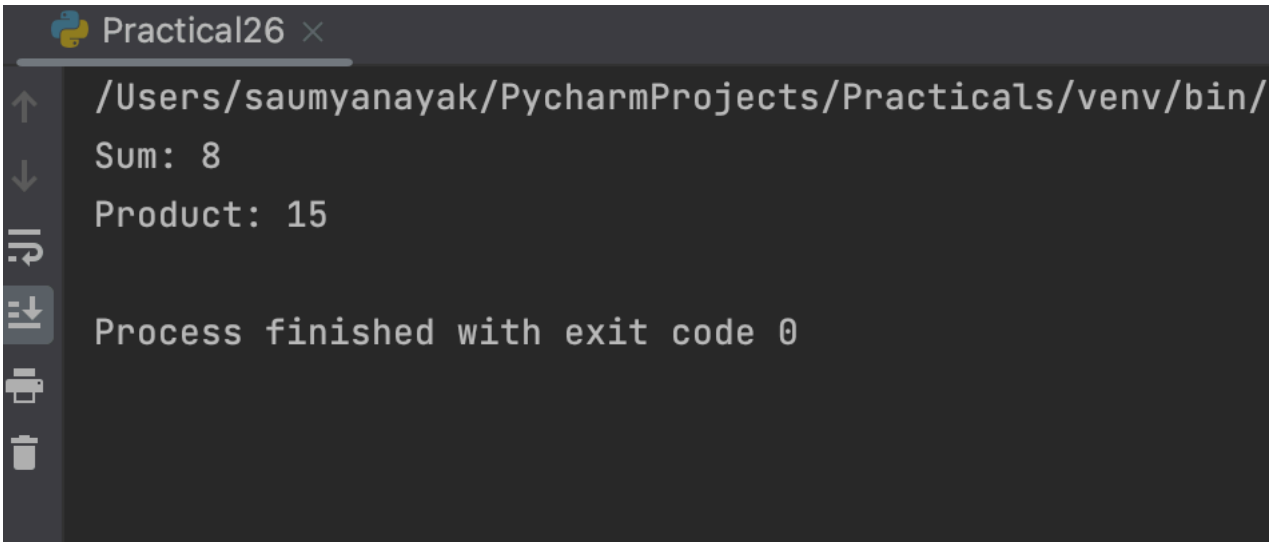
Practical No. 26

Program : Write a Python function to take two integers as input and return both their sum and product.

Code Section :

```
def sum_and_product(num1, num2):
    sum = num1 + num2
    product = num1 * num2
    return sum, product


result = sum_and_product(3, 5)
print("Sum:", result[0])
print("Product:", result[1])
```

Output Section :



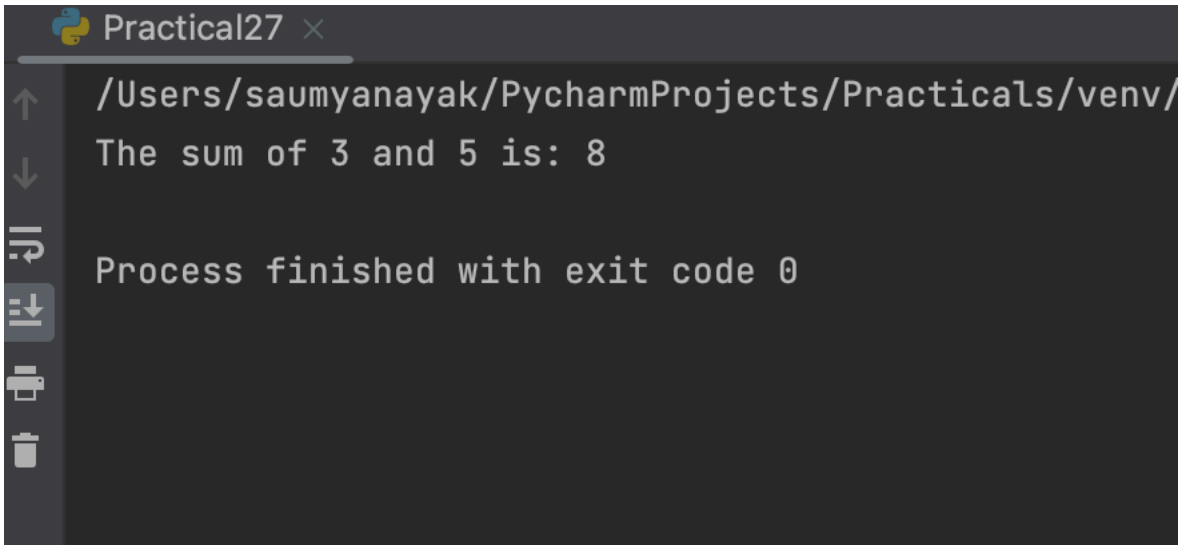
Practical No. 27

Program : Write a Python program to demonstrate the positional arguments of a function.

Code Section :

```
def add_numbers(a, b):
    sum = a + b
    return sum


result = add_numbers(3, 5)
print("The sum of 3 and 5 is:", result)
```

Output Section :



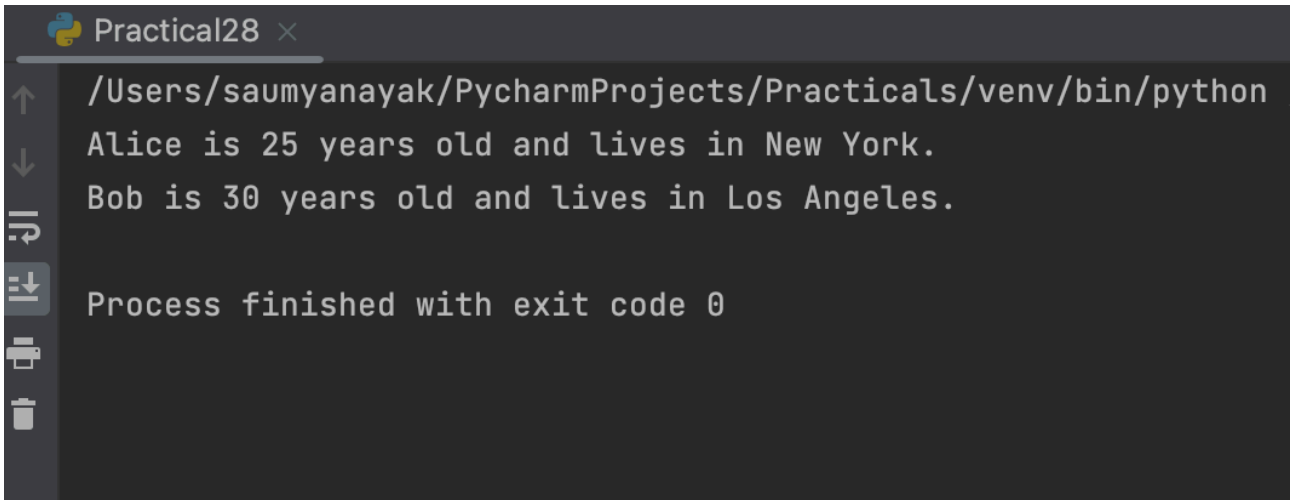
Practical No. 28

Program : Write a Python program to demonstrate the keyword arguments of a function.

Code Section :

```
def describe_person(name, age, city):
    print(f"{name} is {age} years old and lives in {city}.")


# calling the function using positional arguments
describe_person("Alice", 25, "New York")


# calling the function using keyword arguments
describe_person(name="Bob", age=30, city="Los Angeles")
```

Output Section :



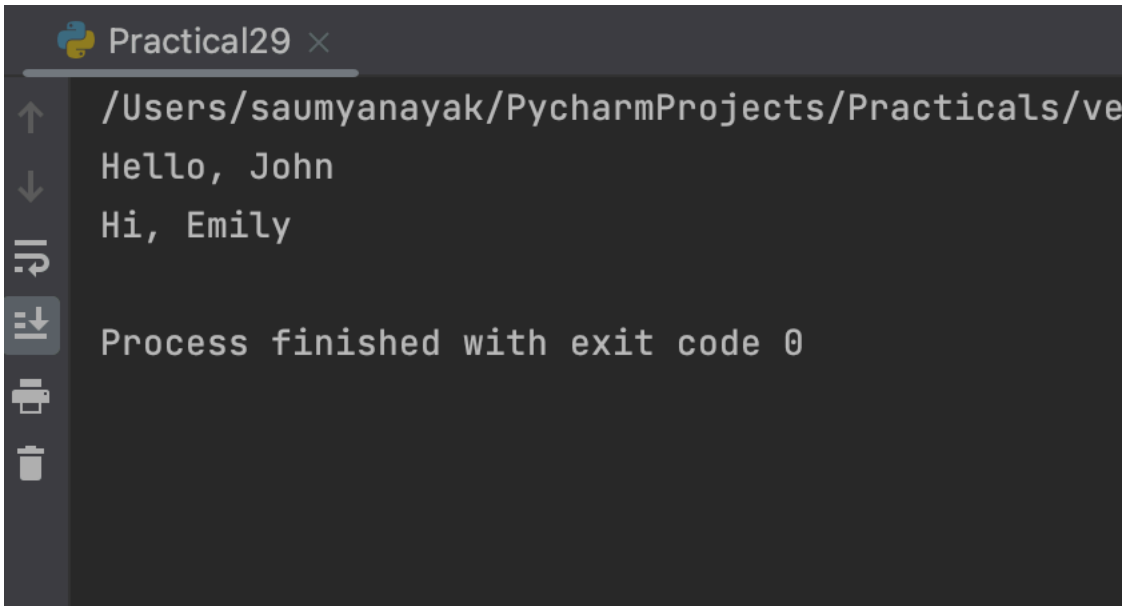
Practical No. 29

Program : Write a Python program to demonstrate the default arguments of a function.

Code Section :

```
def greet(name, greeting="Hello"):
    print(greeting + ", " + name)


# Call the function with one argument
greet("John") # Output: Hello, John


# Call the function with two arguments
greet("Emily", "Hi") # Output: Hi, Emily
```

Output Section :



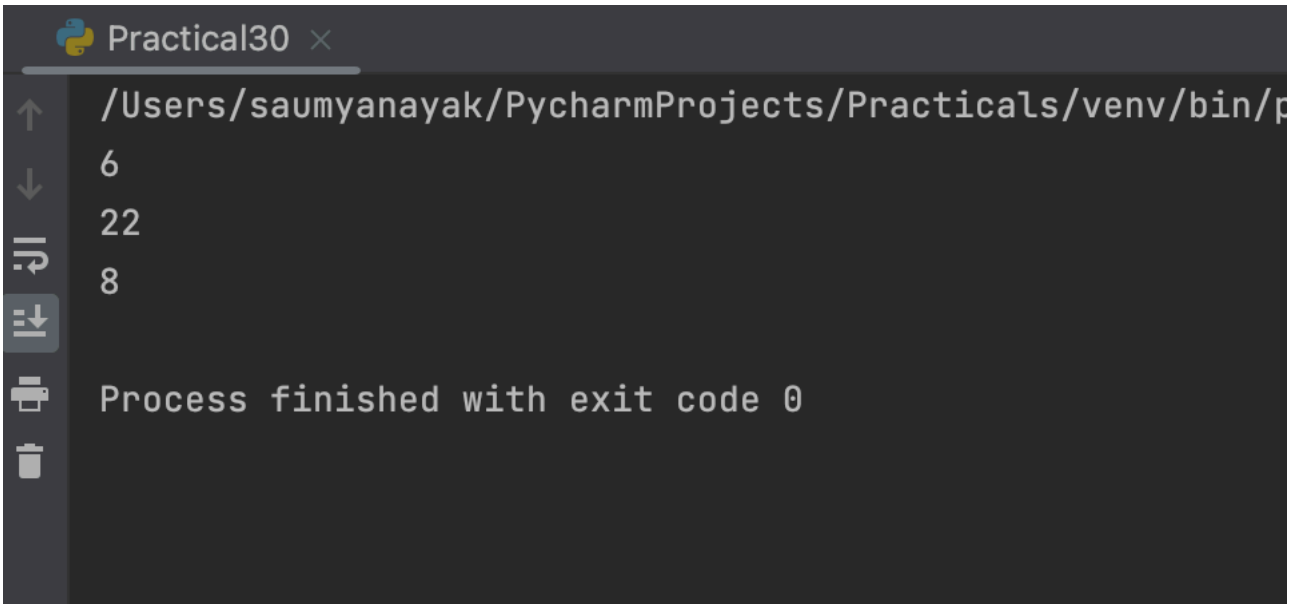
Practical No. 30

Program : Write a Python function to demonstrate variable length arguments.

Code Section :

```
def add_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total


# Call the function with different number of arguments
print(add_numbers(1, 2, 3)) # Output: 6
print(add_numbers(4, 5, 6, 7)) # Output: 22
print(add_numbers(8)) # Output: 8
```

Output Section :



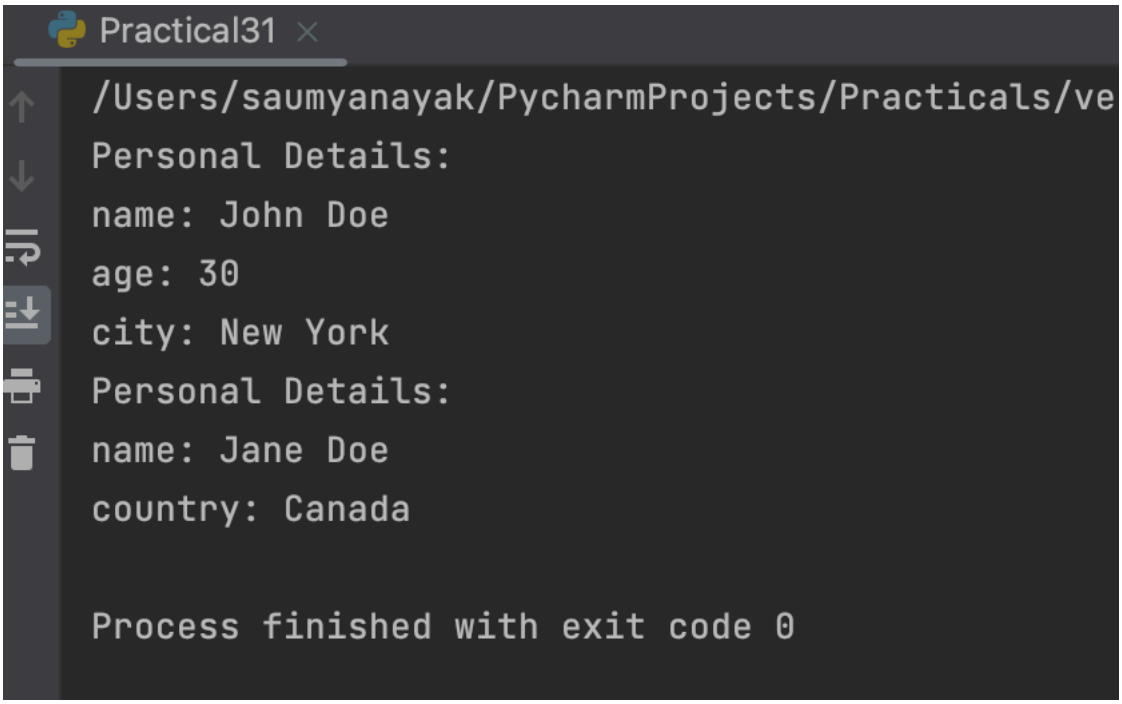
Practical No. 31

Program : Write a Python function to demonstrate keyword variable length arguments.

Code Section :

```
def person_details(**kwargs):
    print("Personal Details:")
    for key, value in kwargs.items():
        print(f"{key}: {value}")


# Call the function with different keyword arguments
person_details(name="John Doe", age=30, city="New York")
person_details(name="Jane Doe", country="Canada")
```

Output Section :



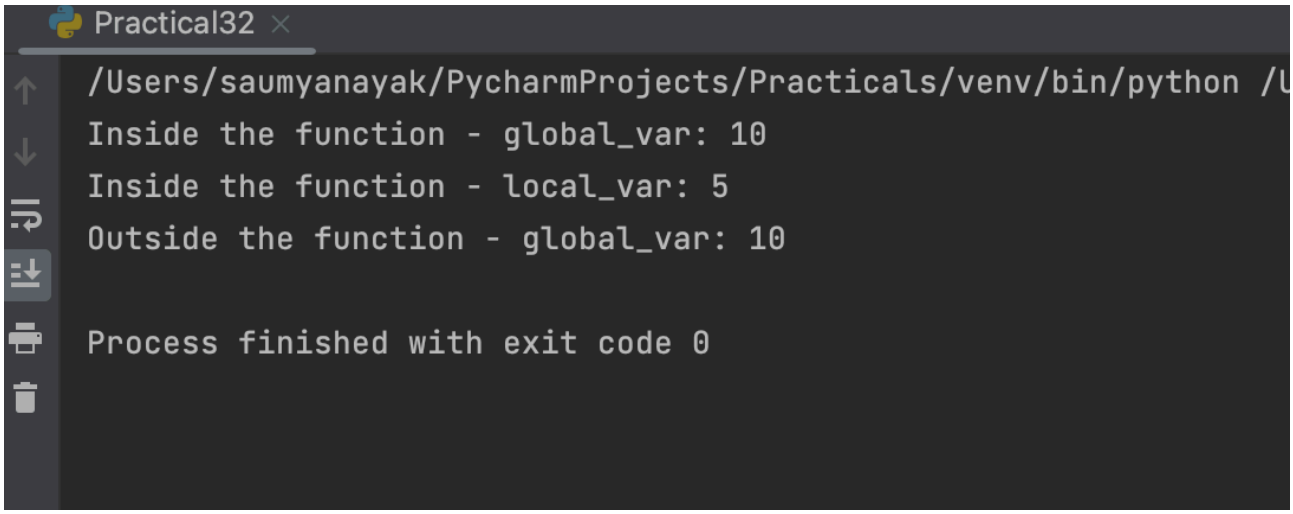
Practical No. 32

Program : Write a Python program to demonstrate global and local variables.

Code Section :

```
# Global variable
global_var = 10

def func():
    # Local variable
    local_var = 5
    # Accessing the global variable within the function
    print("Inside the function - global_var:", global_var)
    # Accessing the local variable within the function
    print("Inside the function - local_var:", local_var)

# Call the function
func()

# Accessing the global variable outside the function
print("Outside the function - global_var:", global_var)

# Trying to access the local variable outside the function raises an error
#print("Outside the function - local_var:", local_var)
```

Output Section :



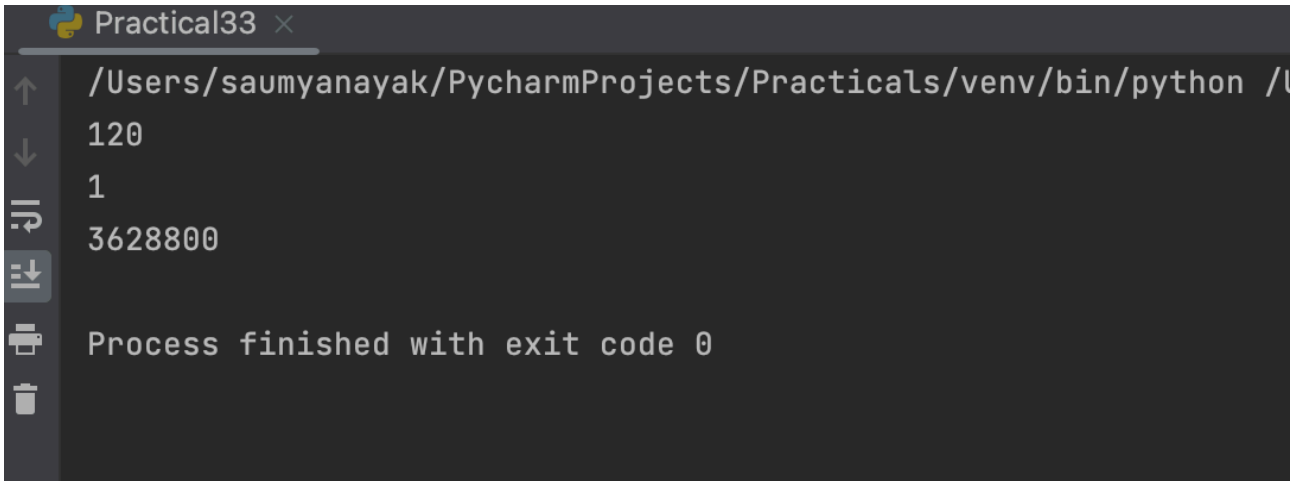
Practical No. 33

Program : Write a Python function that takes an integer as input and calculates its factorial using recursion.

Code Section :

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)


print(factorial(5))   # Output: 120
print(factorial(0))   # Output: 1
print(factorial(10))  # Output: 3628800
```

Output Section :



Practical No. 34

Program : Write a Python program to demonstrate the use of lambda functions.

Code Section :

```
# Example 1: Lambda function to add two numbers
add = lambda x, y: x + y
print(add(3, 5))  # Output: 8

# Example 2: Lambda function to check if a number is even
is_even = lambda x: x % 2 == 0
print(is_even(4))  # Output: True

# Example 3: Sorting a list of strings by their length using lambda function
fruits = ['apple', 'banana', 'orange', 'mango']
sorted_fruits = sorted(fruits, key=lambda x: len(x))
print(sorted_fruits)  # Output: ['mango', 'apple', 'banana', 'orange']
```

Output Section :

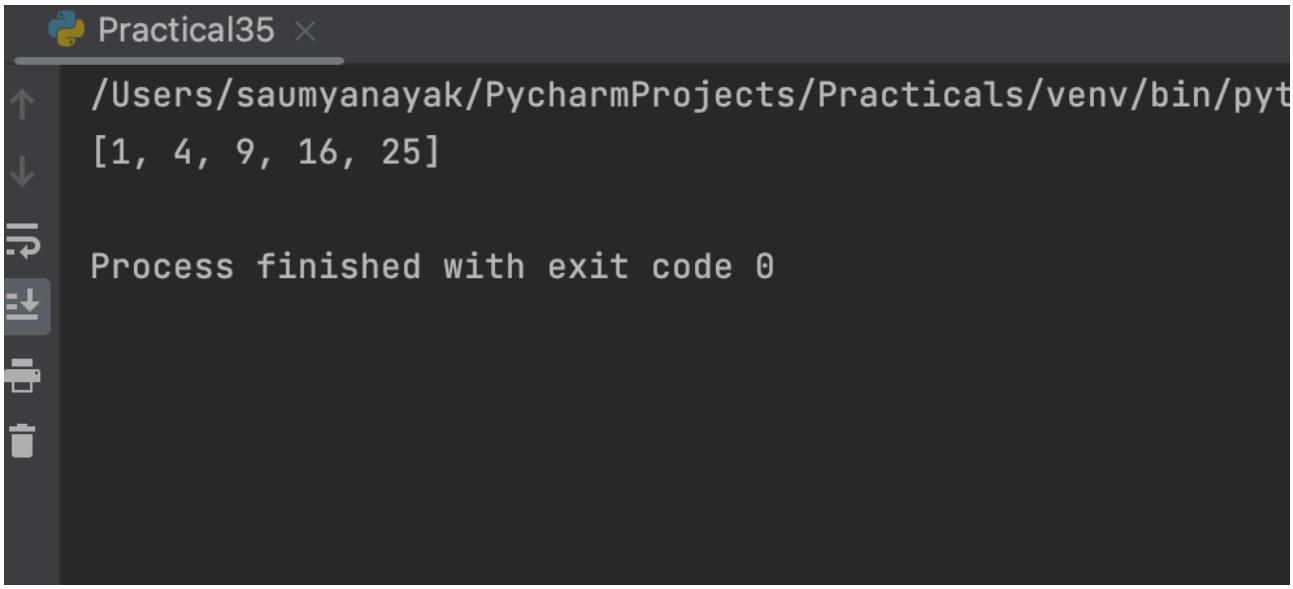
Practical No. 35

Program : Write a Python program to demonstrate the use of lambda functions and map.

Code Section :

```
# Define a list of numbers

numbers = [1, 2, 3, 4, 5]


# Use lambda function and map to square each number in the list

squared_numbers = list(map(lambda x: x**2, numbers))


# Print the squared numbers

print(squared_numbers)
```

Output Section :



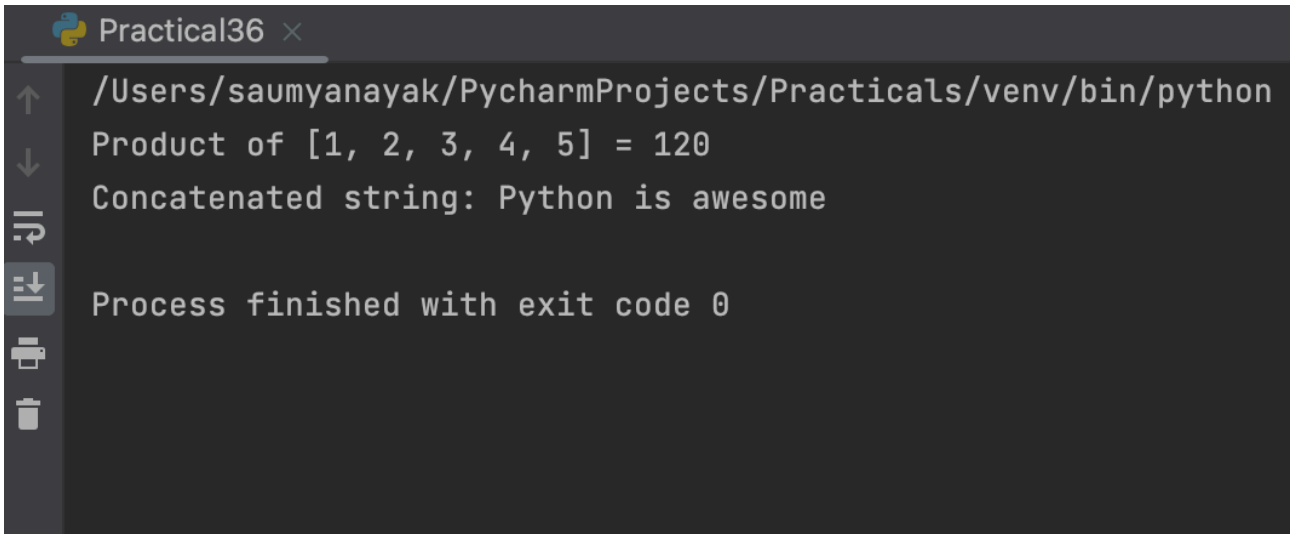
Practical No. 36

Program : Write a Python program to demonstrate the use of lambda functions and reduce.

Code Section :

```
from functools import reduce


# using lambda functions with reduce to calculate the product of a list of numbers
nums = [1, 2, 3, 4, 5]
product = reduce(lambda x, y: x * y, nums)
print(f"Product of {nums} = {product}")


# using lambda functions with reduce to concatenate a list of strings
strings = ["Python", "is", "awesome"]
result = reduce(lambda x, y: x + " " + y, strings)
print(f"Concatenated string: {result}")
```

Output Section :



Practical No. 37

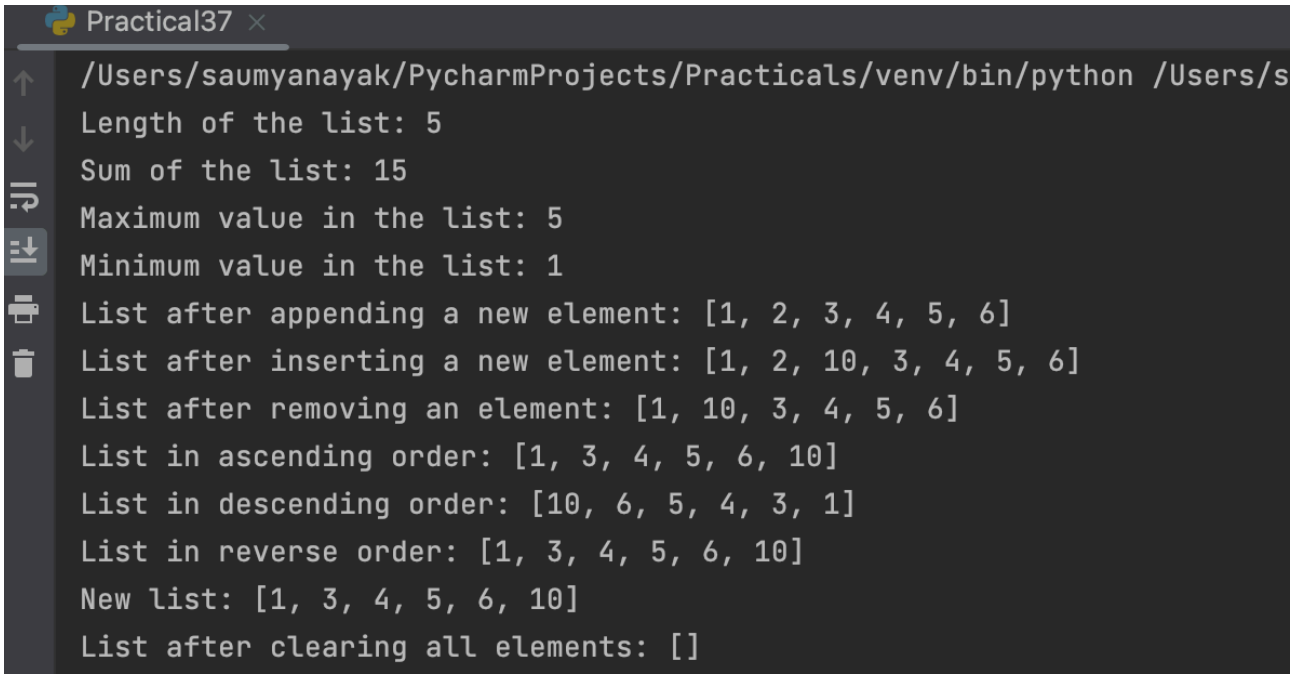
Program : Write a Python program to demonstrate the various list processing methods.

Code Section :

```
# create a list of integers

nums = [1, 2, 3, 4, 5]


# print the length of the list

print("Length of the list:", len(nums))


# print the sum of the list

print("Sum of the list:", sum(nums))


# print the maximum value in the list

print("Maximum value in the list:", max(nums))


# print the minimum value in the list

print("Minimum value in the list:", min(nums))


# add a new element to the end of the list

nums.append(6)

print("List after appending a new element:", nums)


# insert a new element at a specific position in the list

nums.insert(2, 10)

print("List after inserting a new element:", nums)
```

```
# remove the first occurrence of an element in the list
nums.remove(2)
print("List after removing an element:", nums)

# sort the list in ascending order
nums.sort()
print("List in ascending order:", nums)

# sort the list in descending order
nums.sort(reverse=True)
print("List in descending order:", nums)

# reverse the order of the list
nums.reverse()
print("List in reverse order:", nums)

# create a copy of the list
new_nums = nums.copy()
print("New list:", new_nums)

# clear all elements from the list
nums.clear()
print("List after clearing all elements:", nums)
```

Output Section :



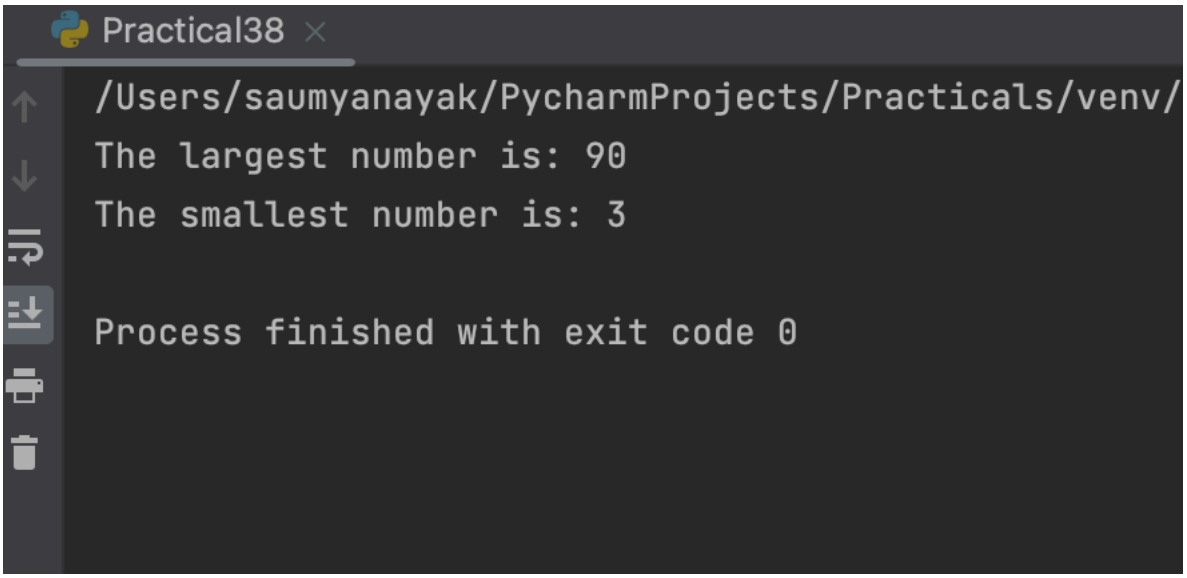
Practical No. 38

Program : Write a Python program to find the biggest and smallest numbers in a list of integers.

Code Section :

```
def find_largest_smallest(numbers):
    largest = max(numbers)
    smallest = min(numbers)
    return largest, smallest


numbers = [12, 3, 45, 6, 78, 90, 23]
largest, smallest = find_largest_smallest(numbers)


print("The largest number is:", largest)
print("The smallest number is:", smallest)
```

Output Section :



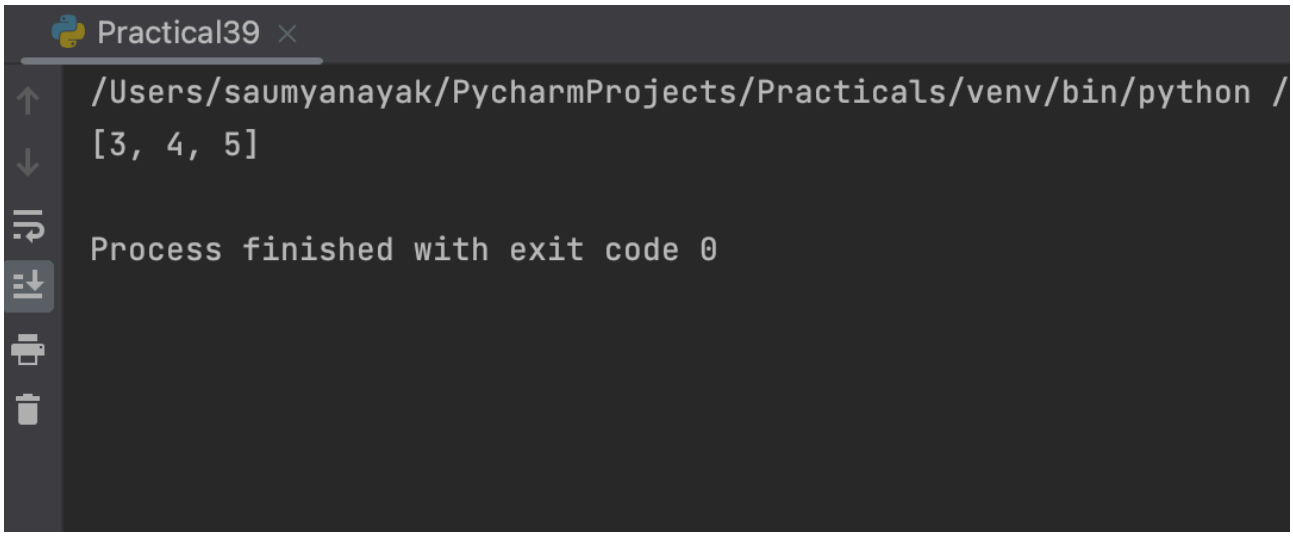
Practical No. 39

Program : Write a Python program to find common elements in two lists.

Code Section :

```
def common_elements(list1, list2):
    set1 = set(list1)
    set2 = set(list2)
    common = list(set1.intersection(set2))
    return common


# Example usage
list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]
print(common_elements(list1, list2)) # Output: [3, 4, 5]
```

Output Section :



Practical No. 40

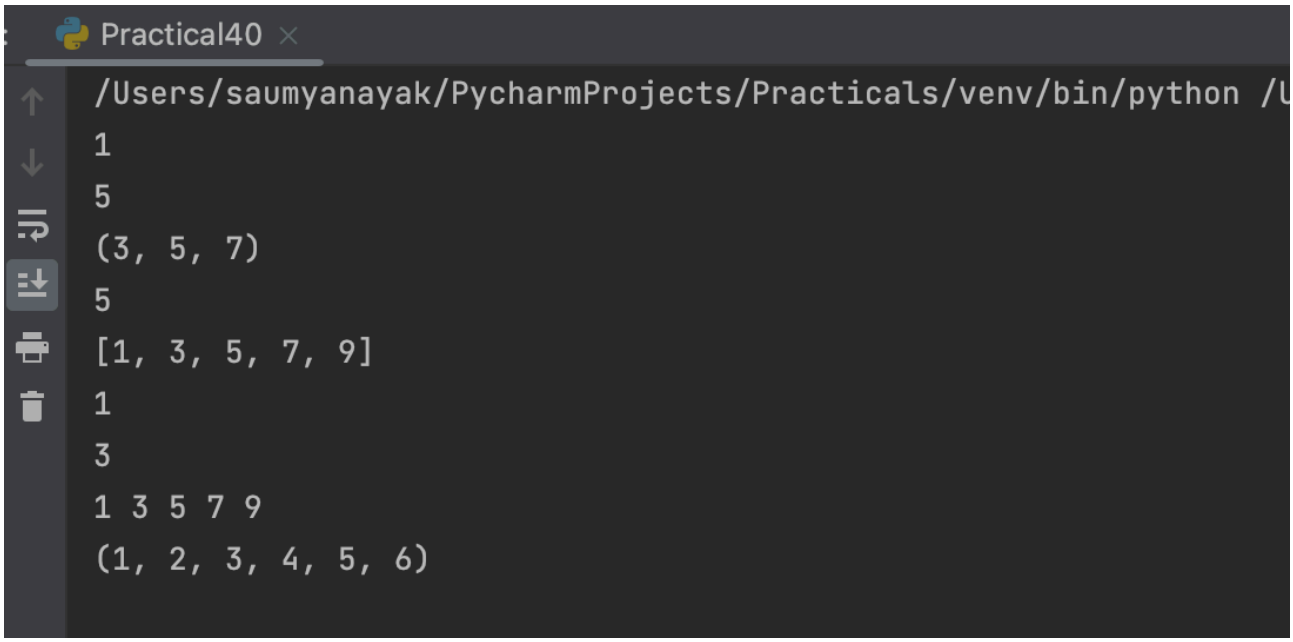
Program : Write a Python program to demonstrate the various tuple processing methods.

Code Section :

```
# create a tuple

my_tuple = (1, 3, 5, 7, 9)


# access elements of tuple using indexing

print(my_tuple[0])  # 1

print(my_tuple[2])  # 5


# slicing

print(my_tuple[1:4])  # (3, 5, 7)


# length of tuple

print(len(my_tuple))  # 5


# convert tuple to list

my_list = list(my_tuple)

print(my_list)  # [1, 3, 5, 7, 9]


# count number of occurrences of an element

print(my_tuple.count(5))  # 1


# index of the first occurrence of an element

print(my_tuple.index(7))  # 3
```

```
# unpacking a tuple
a, b, c, d, e = my_tuple
print(a, b, c, d, e)  # 1 3 5 7 9

# combining tuples
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
combined_tuple = tuple1 + tuple2
print(combined_tuple)  # (1, 2, 3, 4, 5, 6)
```

Output Section :



Practical No. 41

Program : Write a Python program to demonstrate the use of dictionaries.

Code Section :

```
# Define a dictionary with some key-value pairs

person = {'name': 'John', 'age': 30, 'city': 'New York'}


# Accessing dictionary values using keys

print(person['name'])  # Output: John

print(person['age'])  # Output: 30

print(person['city'])  # Output: New York

# Adding a new key-value pair to the dictionary

person['occupation'] = 'Software Engineer'

print(person)  # Output: {'name': 'John', 'age': 30, 'city': 'New York', 'occupation': 'Software Engineer'}

# Modifying an existing value in the dictionary

person['age'] = 35

print(person)  # Output: {'name': 'John', 'age': 35, 'city': 'New York', 'occupation': 'Software Engineer'}

# Removing a key-value pair from the dictionary

del person['occupation']

print(person)  # Output: {'name': 'John', 'age': 35, 'city': 'New York'}

# Looping through all the keys in a dictionary

for key in person:

    print(key, person[key])
```

Output Section :

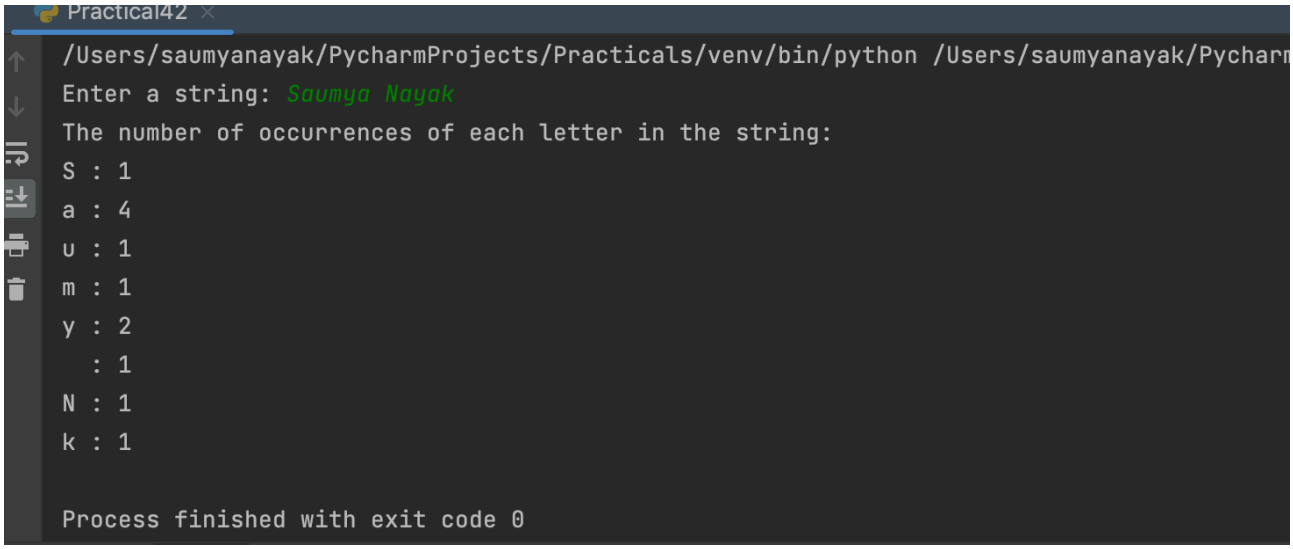
Practical No. 42

Program : Write a Python program to find the number of occurrences of each letter in a string using dictionaries.

Code Section :

```
string = input("Enter a string: ")

count = {}


for letter in string:
    if letter in count:
        count[letter] += 1
    else:
        count[letter] = 1


print("The number of occurrences of each letter in the string:")
for letter, num in count.items():
    print(letter, ":", num)
```

Output Section :



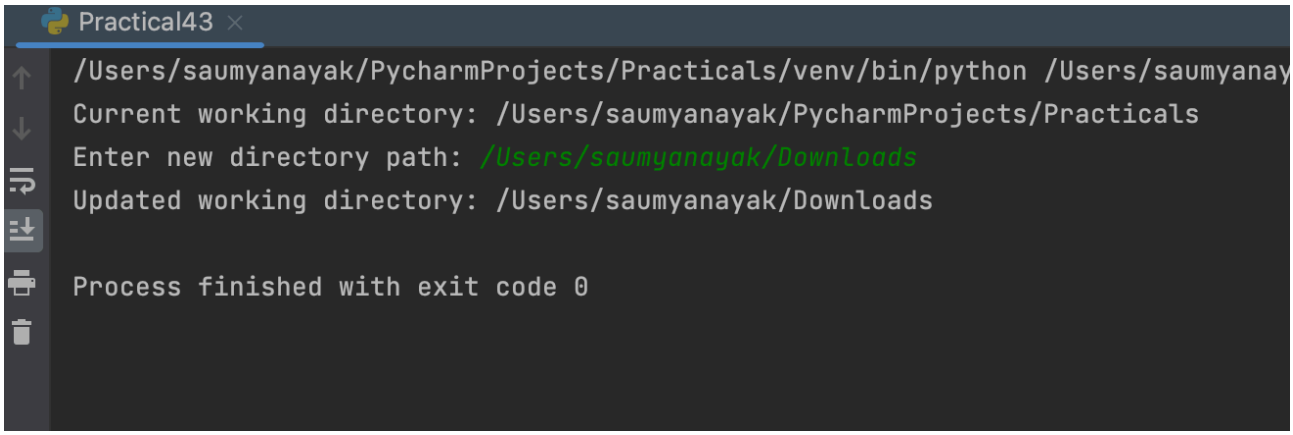
Practical No. 43

Program : Write a Python program to print the CWD and change the CWD.

Code Section :

```
import os


# Print the current working directory
print("Current working directory:", os.getcwd())


# Change the current working directory
new_dir = input("Enter new directory path: ")
os.chdir(new_dir)


# Print the updated current working directory
print("Updated working directory:", os.getcwd())
```

Output Section :



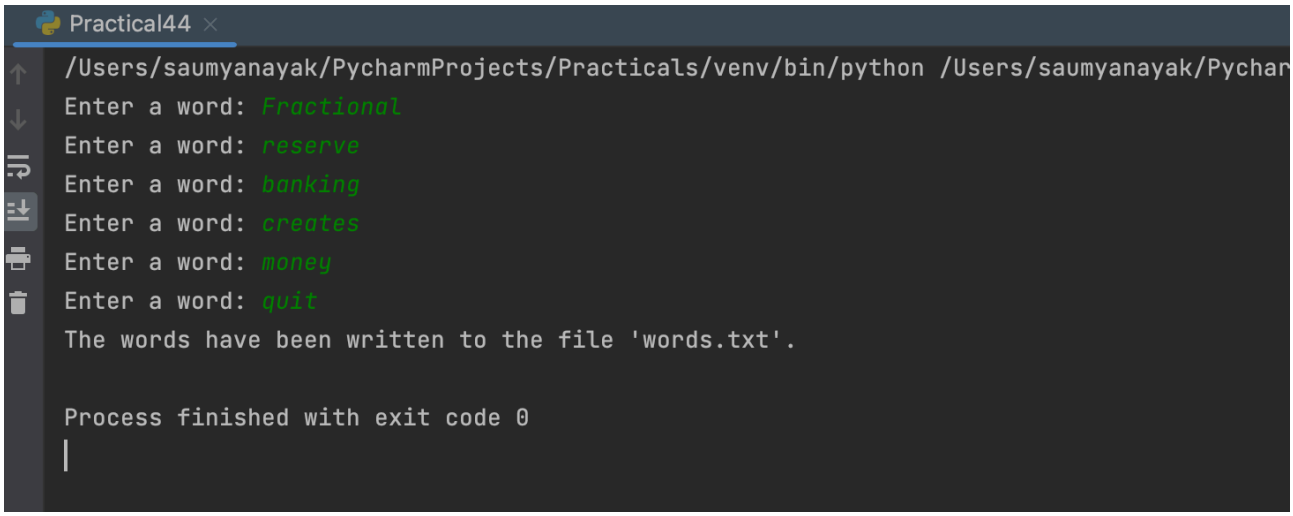
Practical No. 44

Program : Write a Python program that takes a list of words from the user and writes them into a file. The program should stop when the user enters the word ‘quit’.

Code Section :

```
filename = 'words.txt'


# Open the file in write mode
with open(filename, 'w') as file:


    # Loop until the user enters 'quit'
    while True:
        word = input("Enter a word: ")


        # Write the word to the file
        file.write(word + '\n')


        # If the user enters 'quit', break out of the loop
        if word == 'quit':
            break


print(f"The words have been written to the file '{filename}'.")
```

Output Section :



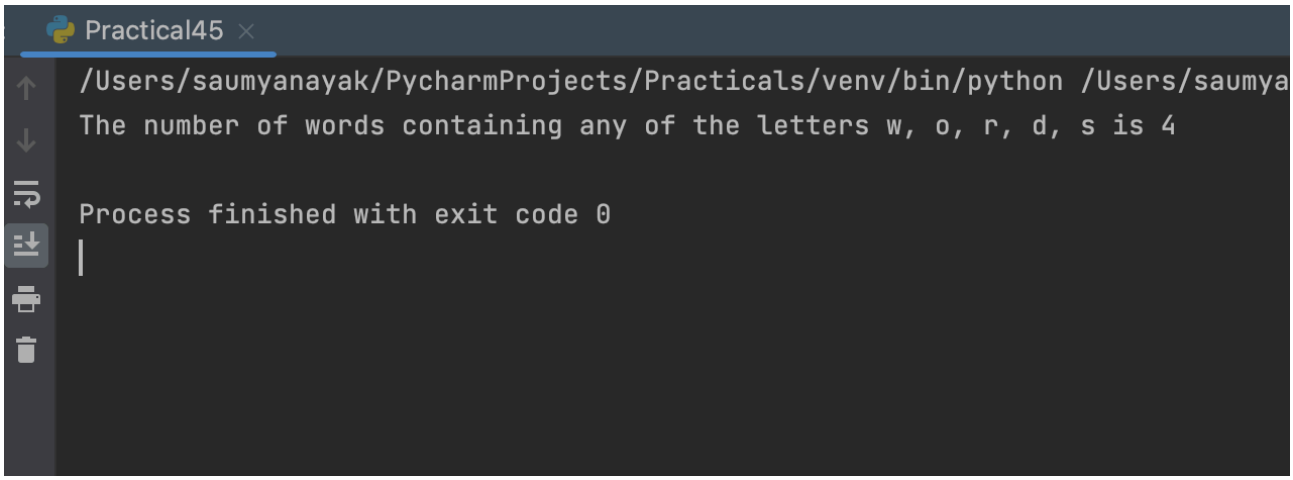
Practical No. 45

Program : Write a Python program that reads a file in text mode and counts the number of words that contain anyone of the letters [‘w’, ‘o’, ‘r’, ‘d’, ‘s’].

Code Section :

```
with open('words.txt', 'r') as file:
    word_count = 0
    for line in file:
        words = line.strip().split()
        for word in words:
            for letter in ['w', 'o', 'r', 'd', 's']:
                if letter in word:
                    word_count += 1
                    break

print(f'The number of words containing any of the letters w, o, r, d, s is {word_count}')
```

Output Section :



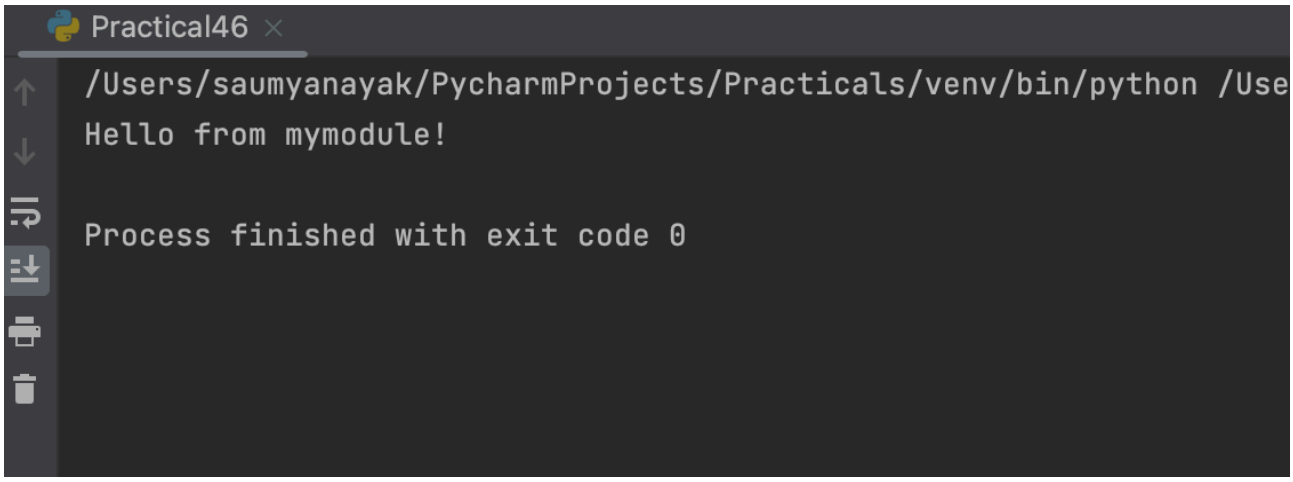
Practical No. 46

Program : Python programs to demonstrate the creation and use of “modules”.

Code Section :

```
//mymodule
def say_hello():
    print("Hello from mymodule!")


//importing module
import mymodule


mymodule.say_hello()
```

Output Section :



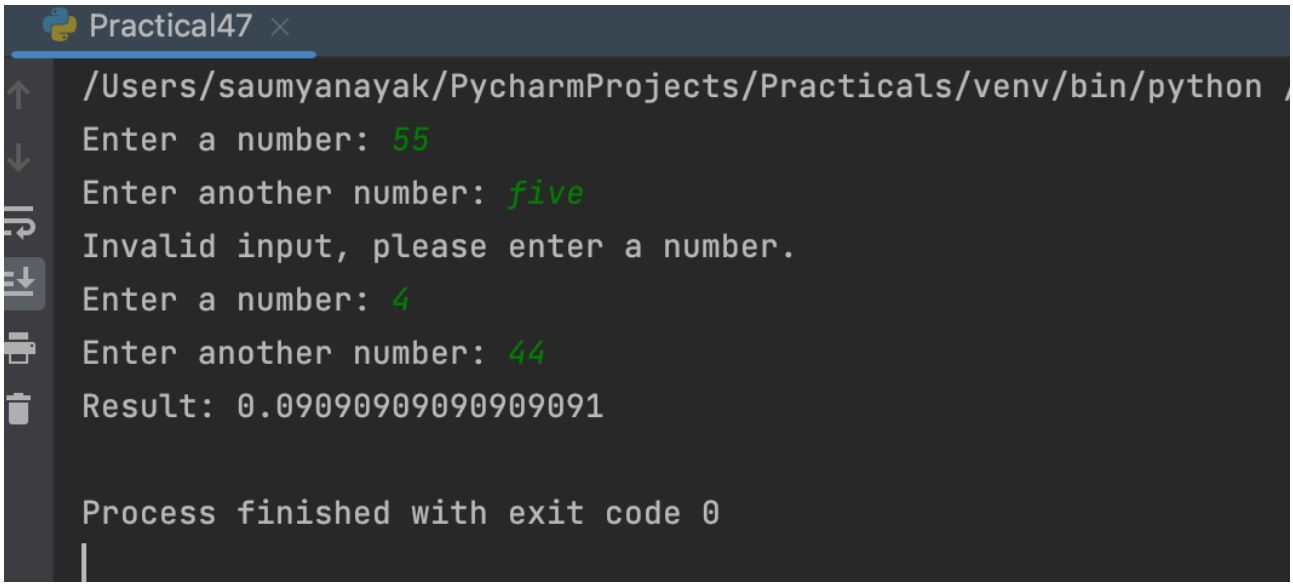
Practical No. 47

Program : Exception Handling Program that uses try and except.

Code Section :

```
while True:
    try:
        x = int(input("Enter a number: "))
        y = int(input("Enter another number: "))
        result = x / y
        print("Result:", result)
        break
    except ValueError:
        print("Invalid input, please enter a number.")
    except ZeroDivisionError:
        print("Cannot divide by zero, please enter a non-zero value for the second number.")
```

Output Section :



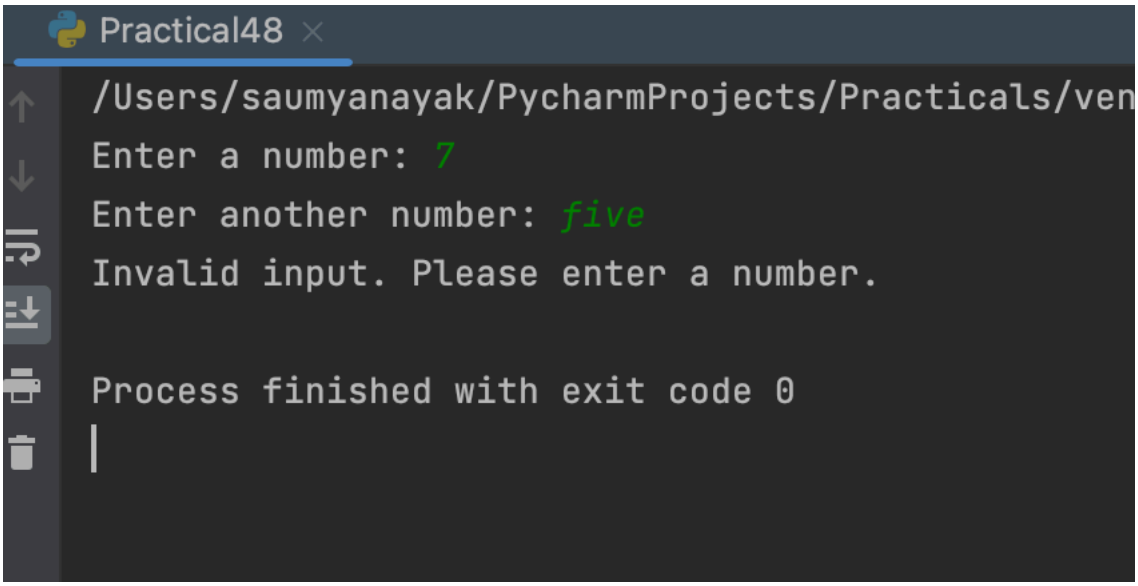
Practical No. 48

Program : Exception Handling Program that handles multiple types of exceptions.

Code Section :

```
try:
    # Code that may raise an exception
    num1 = int(input("Enter a number: "))
    num2 = int(input("Enter another number: "))
    result = num1 / num2
    print("The result is:", result)

except ValueError:
    print("Invalid input. Please enter a number.")

except ZeroDivisionError:
    print("Cannot divide by zero.")

except Exception as e:
    print("An error occurred:", e)
```

Output Section :



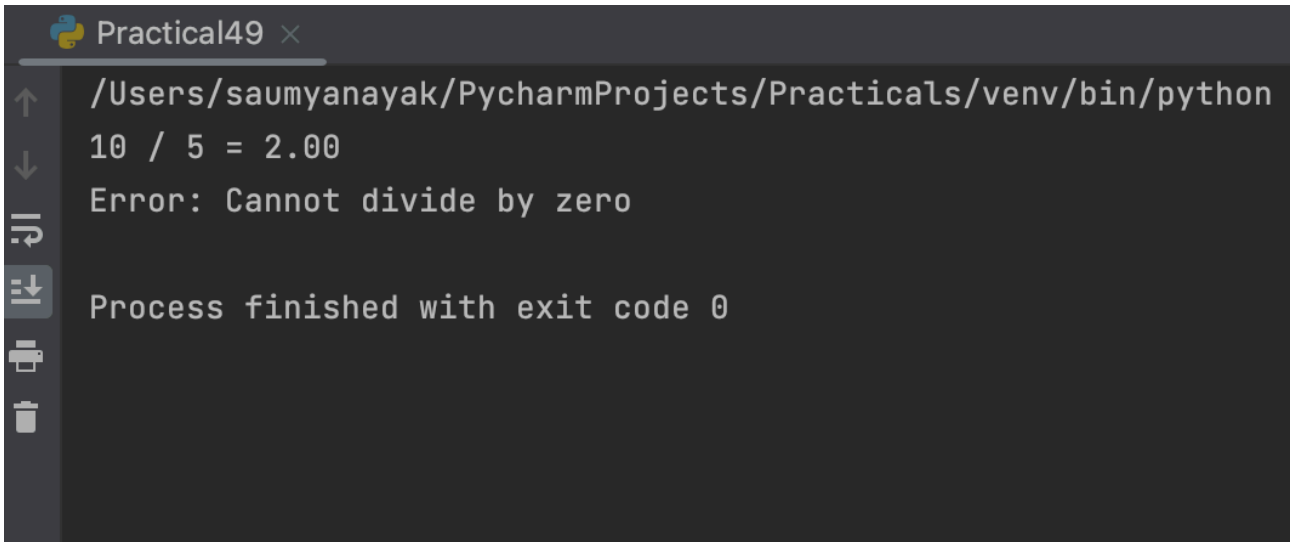
Practical No. 49

Program : Exception Handling Program that uses try, except and else.

Code Section :

```
def divide_numbers(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Error: Cannot divide by zero")
    else:
        print(f"{a} / {b} = {result:.2f}")


# Example usage:
divide_numbers(10, 5) # Output: 10 / 5 = 2.00
divide_numbers(10, 0) # Output: Error: Cannot divide by zero
```

Output Section :



Practical No. 50

Program : Exception Handling Program that uses finally with try.

Code Section :

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print("Result:", result)
except ValueError:
    print("Invalid input. Please enter a number.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
else:
    print("No exception occurred.")
finally:
    print("Program executed.")
```

Output Section :

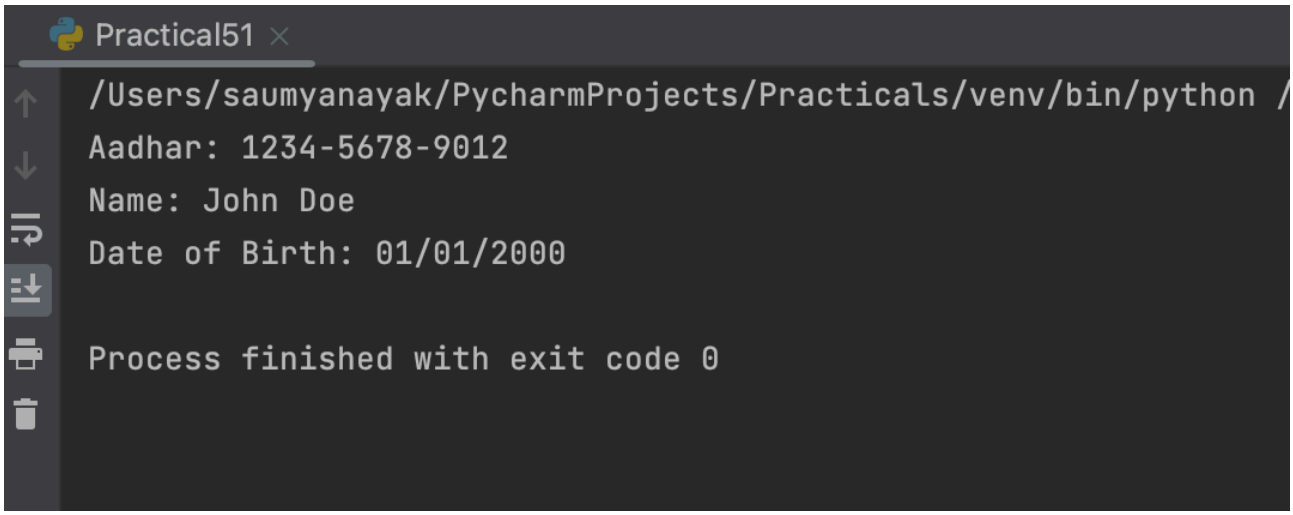
Practical No. 51

Program : Write a Python program that creates a class “Person”, with attributes [aadhar, name, DoB]

Code Section :

```
class Person:
    def __init__(self, aadhar, name, DoB):
        self.aadhar = aadhar
        self.name = name
        self.DoB = DoB

    def display(self):
        print("Aadhar:", self.aadhar)
        print("Name:", self.name)
        print("Date of Birth:", self.DoB)



# Create an instance of the Person class
p = Person("1234-5678-9012", "John Doe", "01/01/2000")


# Display the details of the person
p.display()
```

Output Section :



Practical No. 52

Program : Write a Python program that creates classes “Point” and “Rectangle” where the Rectangle class has a Point object as its attribute.

Code Section :

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


class Rectangle:
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2


    def area(self):
        width = abs(self.p2.x - self.p1.x)
        height = abs(self.p2.y - self.p1.y)
        return width * height


# example usage
p1 = Point(2, 3)
p2 = Point(5, 7)
rect = Rectangle(p1, p2)
print(rect.area()) # prints 12
```

Output Section :

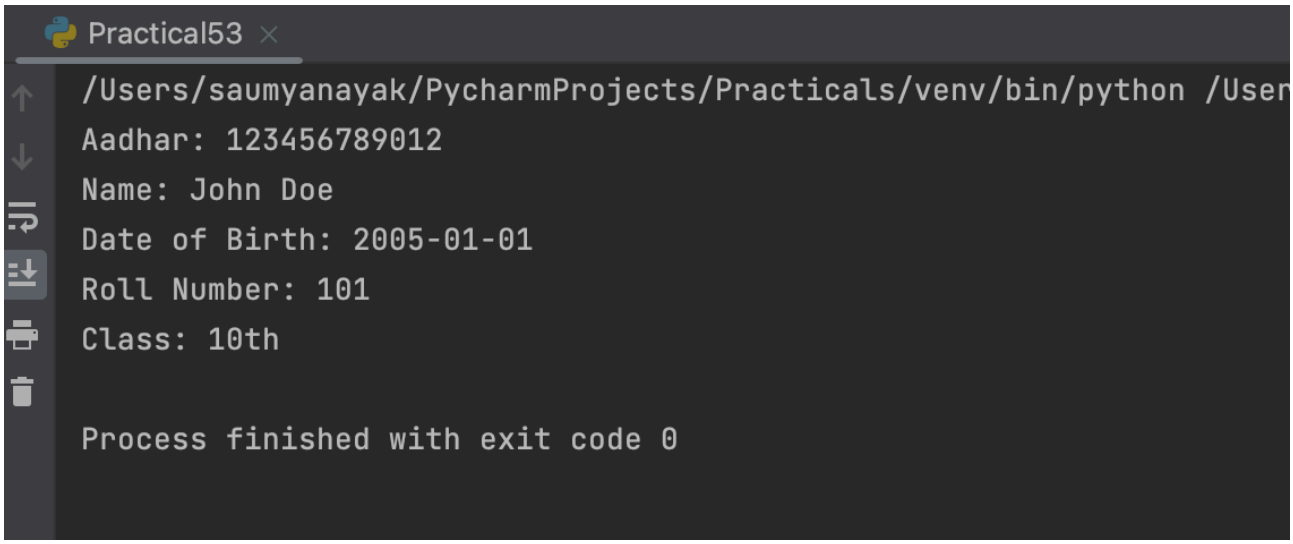
Practical No. 53

Program : Write a Python program that creates a class Students which inherits the properties of the “Person” class; add attributes [roll_no, class].

Code Section :

```
class Person:
    def __init__(self, aadhar, name, dob):
        self.aadhar = aadhar
        self.name = name
        self.dob = dob


class Students(Person):
    def __init__(self, aadhar, name, dob, roll_no, class_name):
        super().__init__(aadhar, name, dob)
        self.roll_no = roll_no
        self.class_name = class_name


    def display(self):
        print("Aadhar:", self.aadhar)
        print("Name:", self.name)
        print("Date of Birth:", self.dob)
        print("Roll Number:", self.roll_no)
        print("Class:", self.class_name)


# creating object of Students class and displaying its details
s = Students("123456789012", "John Doe", "2005-01-01", "101", "10th")
s.display()
```

Output Section :



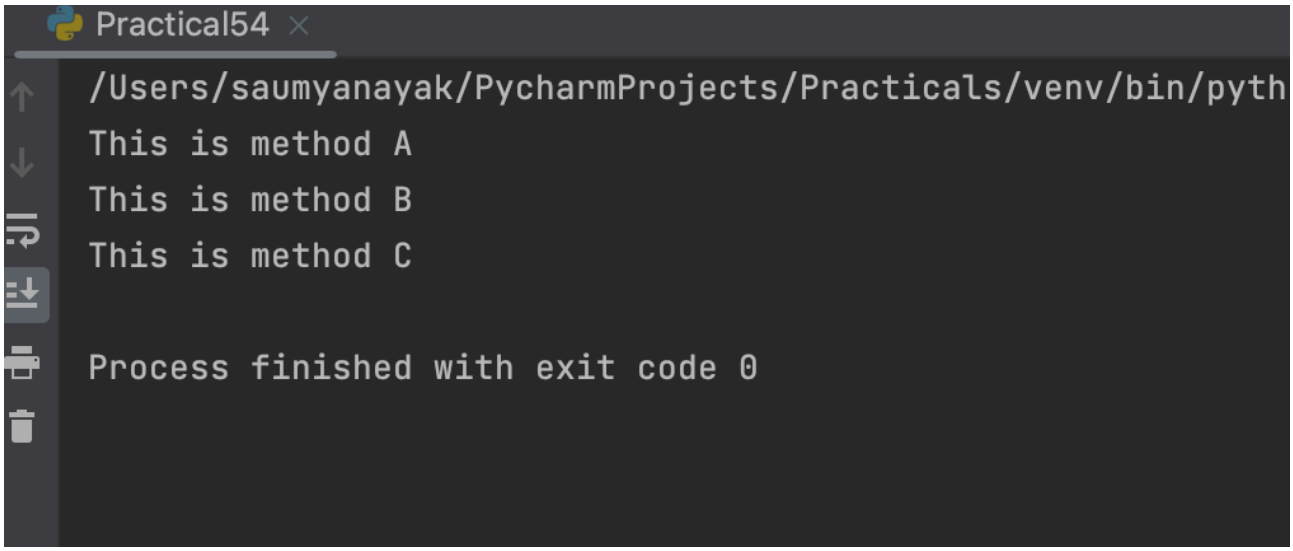
Practical No. 54

Program : Write a Python program to demonstrate “Multiple Inheritance”.

Code Section :

```
# Defining the parent classes
class A:
    def method_a(self):
        print("This is method A")

class B:
    def method_b(self):
        print("This is method B")

# Defining the child class which inherits from both A and B
class C(A, B):
    def method_c(self):
        print("This is method C")

# Creating an object of class C
obj = C()

# Calling the methods of A, B, and C using the object of C
obj.method_a()
obj.method_b()
obj.method_c()
```

Output Section :



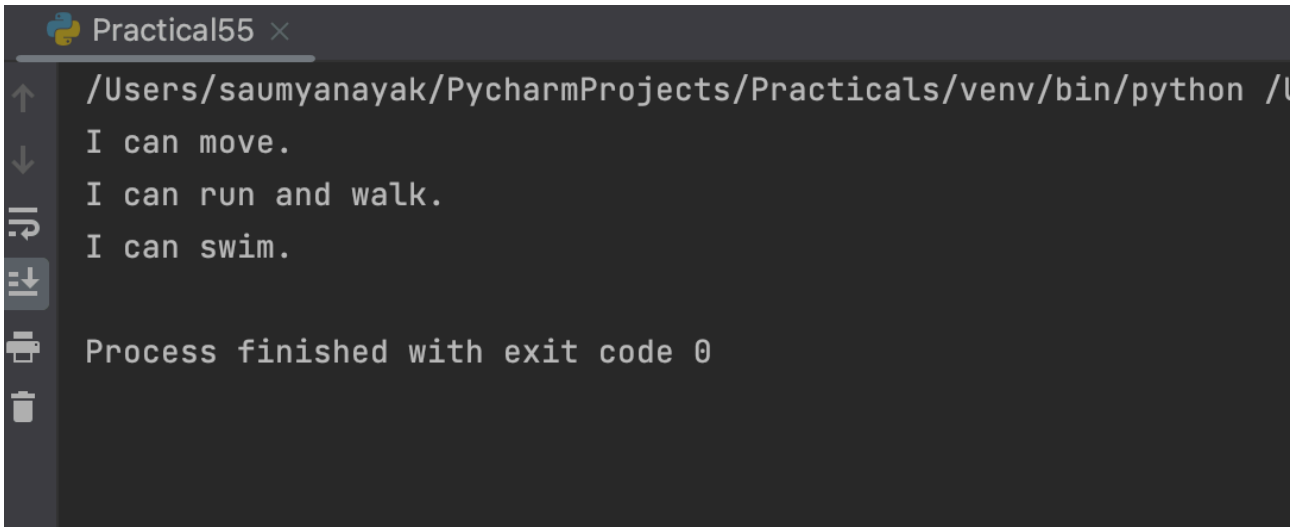
Practical No. 55

Program : Write a Python program to demonstrate “Method Overriding”.

Code Section :

```
class Animal:
    def move(self):
        print("I can move.")


class Dog(Animal):
    def move(self):
        print("I can run and walk.")


class Fish(Animal):
    def move(self):
        print("I can swim.")


# create objects of the classes
animal = Animal()
dog = Dog()
fish = Fish()


# call the move() method on each object
animal.move()   # prints "I can move."
dog.move()      # prints "I can run and walk."
fish.move()     # prints "I can swim."
```

Output Section :



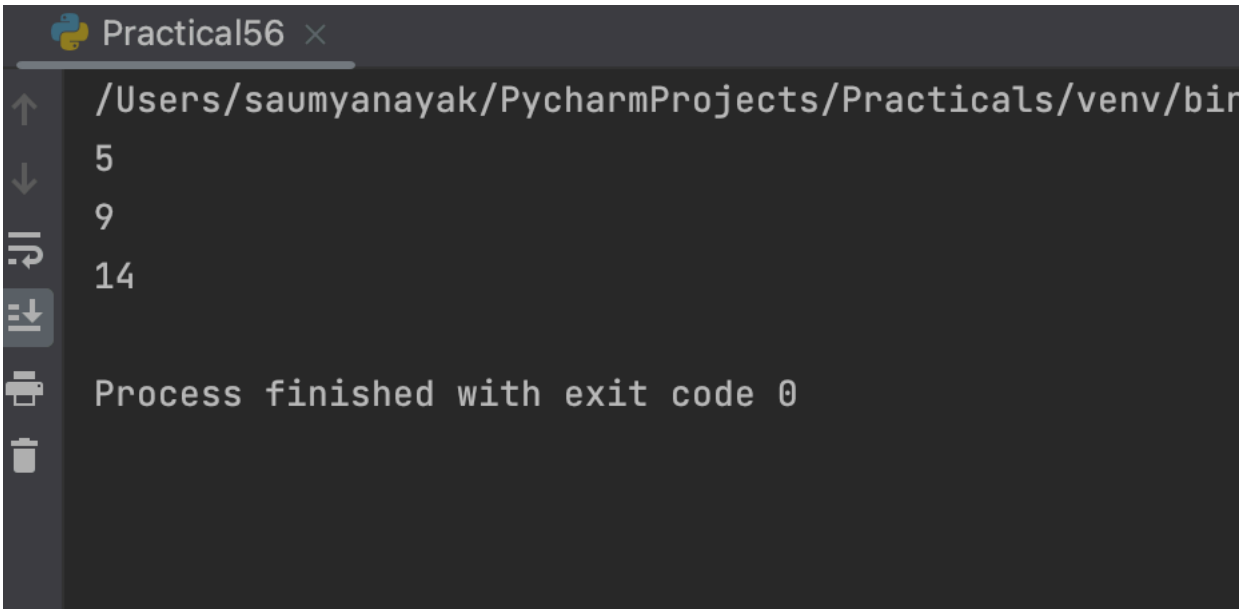
Practical No. 56

Program : Write a Python program to demonstrate “Method Overloading”.

Code Section :

```
class Calculator:
    def add(self, a, b, c=0, d=0):
        return a + b + c + d


calc = Calculator()


print(calc.add(2, 3))
print(calc.add(2, 3, 4))
print(calc.add(2, 3, 4, 5))
```

Output Section :



Practical No. 57

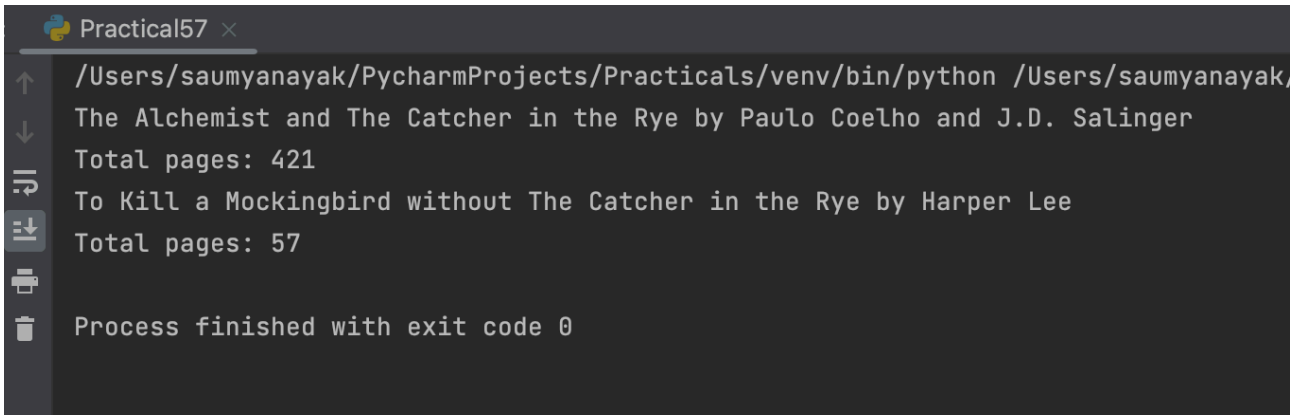
Program : Write a Python program to demonstrate “Operator Overloading” [+ and -] using a class “Book”.

Code Section :

```
class Book:
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):
        return f"{self.title} by {self.author}"

    def __add__(self, other):
        new_title = f"{self.title} and {other.title}"
        new_author = f"{self.author} and {other.author}"
        new_pages = self.pages + other.pages
        return Book(new_title, new_author, new_pages)

    def __sub__(self, other):
        new_title = f"{self.title} without {other.title}"
        new_author = self.author
        new_pages = self.pages - other.pages
        return Book(new_title, new_author, new_pages)

book1 = Book("The Alchemist", "Paulo Coelho", 197)
```

```
book2 = Book("The Catcher in the Rye", "J.D. Salinger", 224)
book3 = Book("To Kill a Mockingbird", "Harper Lee", 281)

# adding two books
new_book = book1 + book2
print(new_book)
print(f"Total pages: {new_book.pages}")

# subtracting one book from another
new_book = book3 - book2
print(new_book)
print(f"Total pages: {new_book.pages}")
```

Output Section :



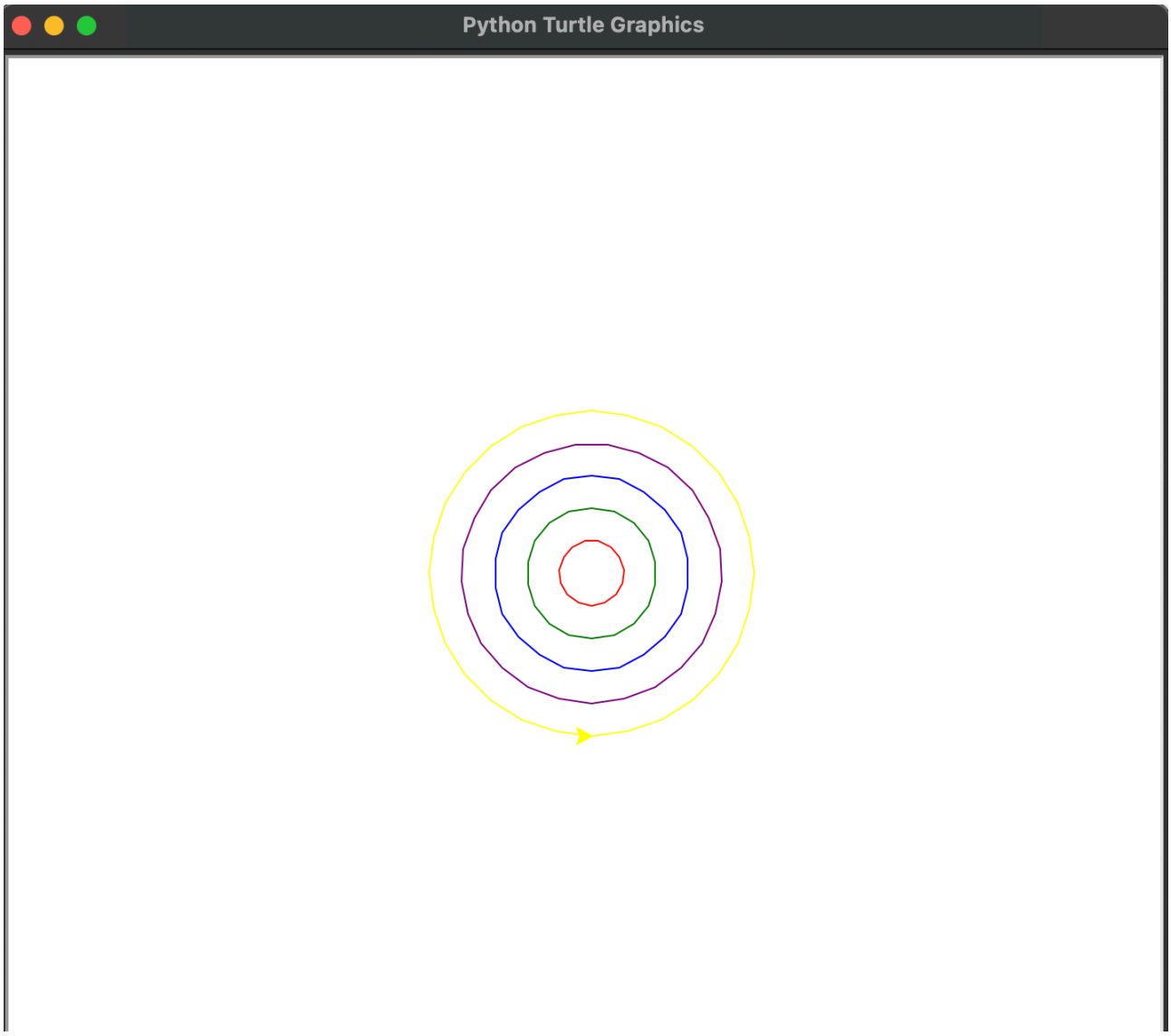
Practical No. 58

Program : Use the “turtle” module to draw concentric circles with different colours.

Code Section :

```
import turtle


colors = ["red", "green", "blue", "purple", "yellow"]
turtle.speed(0)


for i in range(5):
    turtle.color(colors[i])
    turtle.penup()
    turtle.goto(0, -i * 20)
    turtle.pendown()
    turtle.circle(20 * (i + 1))


turtle.done()
```

Output Section :



Practical No. 59

Program : Use the “turtle” module to print the multiplication table.

Code Section :

```
import turtle


# Set up the turtle
t = turtle.Turtle()
t.speed(0)


# Define a function to draw a single multiplication table
def draw_table(num, x, y):
    t.penup()
    t.goto(x, y)
    t.pendown()
    for i in range(1, 11):
        t.write("{} x {} = {}".format(num, i, num * i))
        t.penup()
        t.goto(x, y - i * 20)
        t.pendown()


# Draw the multiplication tables
for i in range(1, 11):
    draw_table(i, i * 50, 0)


# Keep the turtle window open
turtle.done()
```

Output Section :

1 x 1 = 1	2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5	6 x 1 = 6	7 x 1 = 7	8 x 1 = 8	9 x 1 = 9	10 x 1 = 10
1 x 2 = 2	2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10	6 x 2 = 12	7 x 2 = 14	8 x 2 = 16	9 x 2 = 18	10 x 2 = 20
1 x 3 = 3	2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15	6 x 3 = 18	7 x 3 = 21	8 x 3 = 24	9 x 3 = 27	10 x 3 = 30
1 x 4 = 4	2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20	6 x 4 = 24	7 x 4 = 28	8 x 4 = 32	9 x 4 = 36	10 x 4 = 40
1 x 5 = 5	2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25	6 x 5 = 30	7 x 5 = 35	8 x 5 = 40	9 x 5 = 45	10 x 5 = 50
1 x 6 = 6	2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30	6 x 6 = 36	7 x 6 = 42	8 x 6 = 48	9 x 6 = 54	10 x 6 = 60
1 x 7 = 7	2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35	6 x 7 = 42	7 x 7 = 49	8 x 7 = 56	9 x 7 = 63	10 x 7 = 70
1 x 8 = 8	2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40	6 x 8 = 48	7 x 8 = 56	8 x 8 = 64	9 x 8 = 72	10 x 8 = 80
1 x 9 = 9	2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45	6 x 9 = 54	7 x 9 = 63	8 x 9 = 72	9 x 9 = 81	10 x 9 = 90
1 x 10 = 10	2 x 10 = 20	3 x 10 = 30	4 x 10 = 40	5 x 10 = 50	6 x 10 = 60	7 x 10 = 70	8 x 10 = 80	9 x 10 = 90	10 x 10 = 100

Practical No. 60

Program : Use the “turtle” module to draw (not write) your name.

Code Section :

```
import turtle as t
t.pen
t.backward(200)
t.undo()
t.penup()
t.backward(250)
t.pendown()
#S
t.forward(50)
t.left(90)
t.forward(25)
t.left(90)
t.forward(50)
t.right(90)
t.forward(25)
t.right(90)
t.forward(50)
t.penup()
t.goto(150,0)
t.undo()
#A
t.goto(-150,0)
t.left(90)
```

```
t.pendown()
t.forward(50)
t.right(90)
t.forward(40)
t.right(90)
t.forward(50)
t.backward(50)
t.forward(25)
t.right(90)
t.forward(50)
t.undo()
t.forward(40)
t.penup()
#U
t.goto(-70,0)
t.right(90)
t.forward(50)
t.pendown()
t.backward(50)
t.right(90)
t.forward(40)
t.left(90)
t.forward(50)
#M
t.goto(20,0)
t.undo()
```

```
t.penup()
t.goto(20,0)
t.pendown()
t.forward(50)
t.right(90)
t.forward(20)
t.right(45)
t.forward(25)
t.left(90)
t.forward(25)
t.right(45)
t.forward(20)
t.right(90)
t.forward(50)
t.penup()
#Y
t.goto(120,0)
t.goto(140,0)
t.left(180)
t.pendown()
t.forward(25)
t.left(45)
t.forward(25)
t.backward(25)
t.right(90)
t.forward(25)
```

```
t.penup()
#A
t.goto(190,0)
t.left(45)
t.pendown()
t.forward(50)
t.right(90)
t.forward(40)
t.right(90)
t.forward(25)
t.right(90)
t.forward(40)
t.backward(40)
t.left(90)
t.forward(25)
```

Output Section :