INSTITUTE OF TECHNOLOGY AND MANAGEMENT
SKILLS UNIVERSITY,
KHARGHAR, NAVI MUMBAI

PYTHON PROGRAMMING LAB

Prepared by:

Name of Student:Riya Singh

Roll No: 26

Batch: 2023-27

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Exp. No	List of Experiment
1	1.1 Write a program to compute Simple Interest.

	1.2 Write a program to perform arithmetic, Relational operators.
	1.3 Write a program to find whether a given no is even & odd.
	1.4 Write a program to print first n natural number & their sum.
	1.5 Write a program to determine whether the character entered is a Vowel or not .
	1.6 Write a program to find whether given number is an Armstrong Number.
	1.7 Write a program using for loop to calculate factorial of a No.
	1.8 Write a program to print the following pattern
	i) * * * * * * * * * * * * * * *
	ii) 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
	iii) *

2	2.1 Write a program that define the list of defines the list of define countries that are in BRICS.
	2.2 Write a program to traverse a list in reverse order. 1.By using Reverse method. 2.By using slicing
	2.3 Write a program that scans the email address and forms a tuple of username and domain.
	2.4 Write a program to create a list of tuples from given list having number and add its cube in tuple. i/p: c= [2,3,4,5,6,7,8,9]
	2.5 Write a program to compare two dictionaries in Python? (By using == operator)
	2.6 Write a program that creates dictionary of cube of odd numbers in the range.
	2.7 Write a program for various list slicing operation. a= [10,20,30,40,50,60,70,80,90,100] i. Print Complete list ii. Print 4th element of list iii. Print list from0th to 4th index. iv. Print list -7th to 3rd element v. Appending an element to list. vi. Sorting the element of list. vii. Popping an element. viii. Removing Specified element. ix. Entering an element at specified index. x. Counting the occurrence of a specified element. xi. Extending list. xii. Reversing the list.
3	3.1 Write a program to extend a list in python by using given approach. i. By using + operator. ii. By using Append () iii. By using extend ()

	3.2 Write a program to add two matrices.
	3.3 Write a Python function that takes a list and returns a new list with distinct elements from the first list.
	3.4 Write a program to Check whether a number is perfect or not.
	3.5 Write a Python function that accepts a string and counts the number of upper and lower-case letters. string_test= 'Today is My Best Day'
4	4.1 Write a program to Create Employee Class & add methods to get employee details & print.
	4.2 Write a program to take input as name, email & age from user using combination of keywords argument and positional arguments (*args and**kwargs) using function,
	4.3 Write a program to admit the students in the different Departments(pgdm/btech)and count the students. (Class, Object and Constructor).
	4.4 Write a program that has a class store which keeps the record of code and price of product display the menu of all product and prompt to enter the quantity of each item required and finally generate the bill and display the total amount.
	4.5 Write a program to take input from user for addition of two numbers using (single inheritance).
	4.6 Write a program to create two base classes LU and ITM and one derived class. (Multiple inheritance).
	4.7 Write a program to implement Multilevel inheritance, Grandfather→Father-→Child to show property inheritance from grandfather to child.

	4.8 Write a program Design the Library catalogue system using inheritance take base class (library item) and derived class (Book, DVD & Journal) Each derived class should have unique attribute and methods and system should support Check in and check out the system. (Using Inheritance and Method overriding)

5	5.1 Write a program to create my_module for addition of two numbers and import it in main script.
	5.2 Write a program to create the Bank Module to perform the operations such as Check the Balance, withdraw and deposit the money in bank account and import the module in main file.
	5.3 Write a program to create a package with name cars and add different modules (such as BMW, AUDI, NISSAN) having classes and functionality and import them in main file cars.
6	6.1 Write a program to implement Multithreading. Printing “Hello” with one thread & printing “Hi” with another thread.
7.	7.1 Write a program to use ‘whether API’ and print temperature of any city, also print the sunrise and sunset times for the same humidity of that area.
	7.2 Write a program to use the ‘API’ of crypto currency.

Name of Student:Riya Singh

Roll Number: 26

Experiment No: 1.1

Title: Write a program to compute Simple Interest.

Theory: Simple Interest(SI)=$P \times R \times T$

Where : P is the principal amount (the initial amount of money.)

R is the rate of interest per time period .

T is the time the money is invested .

The formula can be written more explicitly as:

***SI*=P*R*T/100**

Code: `#Write a program to compute Simple Interest.`

```
def calculate_simpleinterest (principal,rate,time):

   simpleinterest = (principal * rate * time / 100)

   return simpleinterest



principal = float(input("Enter the Principal Amount:"))

rate = float(input("Enter the Rate of Interest:"))

time = float(input("Enter the TimePeriod:"))



interest_result = calculate_simpleinterest(principal, rate, time)

print(f"simple interest: {interest_result} ")
```

Output: (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp1.py"
Enter the Principal Amount:5000
Enter the Rate of Interest:5
Enter the TimePeriod:2
simple interest: 500.0
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot)

:1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp1.py"
Enter the Principal Amount:400000
Enter the Rate of Interest:5
Enter the TimePeriod:4
simple interest: 80000.0
riyasingh@riyas-MacBook-Air submission python %
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp1.py"
Enter the Principal Amount:600
Enter the Rate of Interest:2
Enter the TimePeriod:3
simple interest: 36.0
riyasingh@riyas-MacBook-Air submission python %
```

Conclusion: **In conclusion, the simple interest formula provides a straightforward method for calculating the interest accrued on a principal amount over a specified period .**

Experiment No : 1.2

Title : Write a program to perform arithmetic,

Relational operators

Theory : The program takes two numbers as input, performs various arithmetic operations (addition, subtraction, multiplication, division) and relational operations (greater than, less than, equal to), and then prints the results.

Code :

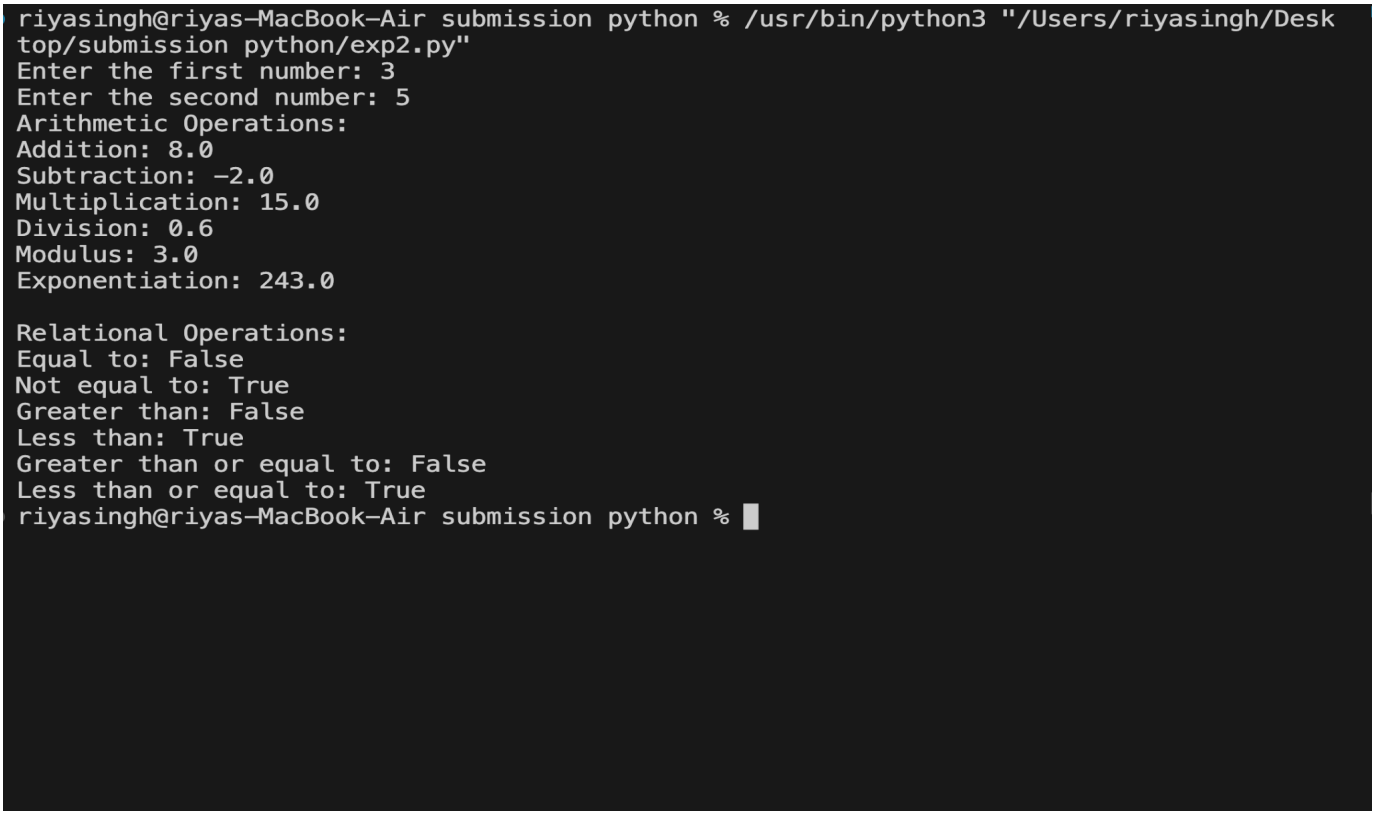
```
#Write a program to perform arithmetic, Relational operators.
def perform_operations(num1, num2):
    # Arithmetic Operations
    addition = num1 + num2
    subtraction = num1 - num2
    multiplication = num1 * num2
```

```
    division = num1 / num2
    modulus = num1 % num2
    exponentiation = num1 ** num2

    # Relational Operations
    equal_to = num1 == num2
    not_equal_to = num1 != num2
    greater_than = num1 > num2
    less_than = num1 < num2
    greater_than_or_equal_to = num1 >= num2
    less_than_or_equal_to = num1 <= num2

    print(f"Arithmetic Operations:")
    print(f"Addition: {addition}")
    print(f"Subtraction: {subtraction}")
                        print(f"Multiplication:
{multiplication}")
    print(f"Division: {division}")
    print(f"Modulus: {modulus}")
                        print(f"Exponentiation:
{exponentiation}")

    print("\nRelational Operations:")
    print(f"Equal to: {equal_to}")
    print(f"Not equal to: {not_equal_to}")
    print(f"Greater than: {greater_than}")
    print(f"Less than: {less_than}")
        print(f"Greater  than  or  equal  to:
{greater_than_or_equal_to}")
         print(f"Less  than  or  equal  to:
{less_than_or_equal_to}")

number1  =  float(input("Enter  the  first
number: "))
number2  =  float(input("Enter  the  second
number: "))

perform_operations(number1, number2)
```

Output (Screenshot) :

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.py"
Enter the first number: 2
Enter the second number: 2
Arithmetic Operations:
Addition: 4.0
Subtraction: 0.0
Multiplication: 4.0
Division: 1.0
Modulus: 0.0
Exponentiation: 4.0

Relational Operations:
Equal to: True
Not equal to: False
Greater than: False
Less than: False
Greater than or equal to: True
Less than or equal to: True
riyasingh@riyas-MacBook-Air submission python %
```

Test Case (Any 2 Screenshot) : 1.


riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.py"
Enter the first number: 3
Enter the second number: 5
Arithmetic Operations:
Addition: 8.0
Subtraction: -2.0
Multiplication: 15.0
Division: 0.6
Modulus: 3.0
Exponentiation: 243.0
Relational Operations:
Equal to: False
Not equal to: True
Greater than: False
Less than: True
Greater than or equal to: False
Less than or equal to: True
riyasingh@riyas-MacBook-Air submission python %

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.py"
Enter the first number: 7
Enter the second number: 5
Arithmetic Operations:
Addition: 12.0
Subtraction: 2.0
Multiplication: 35.0
Division: 1.4
Modulus: 2.0
Exponentiation: 16807.0

Relational Operations:
Equal to: False
Not equal to: True
Greater than: True
Less than: False
Greater than or equal to: True
Less than or equal to: False
riyasingh@riyas-MacBook-Air submission python %
```

Conclusion :This program illustrates the use of arithmetic and relational operators in Python. It demonstrates how to perform basic mathematical operations and make simple comparisons between two numbers.

Experiment No : 1.3

Title :Write a program to find whether a given no is even & odd.

Theory : In mathematics, even numbers are integers that are exactly divisible by 2, leaving no remainder. Odd numbers, on the other hand, are integers that are not divisible by 2 without leaving a remainder.

Code :

```python
#Write a program to find whether a given no is even & odd.
n = int(input("Enter a Number: "))
if n % 2 == 0:
   print(n , "is a even number")
else:
   print(n , "is a odd number")
```

Output(Screenshot) :

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp3.py"
Enter a Number: 2
2 is a even number
```

Test Case(Any 2 Screenshot) :1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp3.py"
Enter a Number: 10
10 is a even number
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp3.py"
Enter a Number: 22
22 is a even number
```

Conclusion : This program demonstrates a simple use of conditional statements in Python to determine whether a given number is even or odd.

Experiment No : 1.4

Title :Write a program to print first n natural number & their sum.

Theory : Natural numbers are the set of positive integers starting from 1 and extending infinitely (1, 2, 3, 4, ...).

Code :

```python
#Write a program to print first n natural number & their sum.
def natural_numbers_and_sum_numbers(n):
    natural_numbers = list(range(1 , n + 1))
    sum_numbers = sum(natural_numbers)
    print(f"first {n} natural numbers are : {natural_numbers}")
    print(f"Sum of {n} natural numbers is : {sum_numbers}")

n = int(input("Enter the range of number:"))
natural_numbers_and_sum_numbers(n)
```

Output(screenshot):

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.py"
Enter the range of number:10
first 10 natural numbers are : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of 10 natural numbers is : 55
```

Test Case (Ant 2 Screenshot):1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.py"
Enter the range of number:5
first 5 natural numbers are : [1, 2, 3, 4, 5]
Sum of 5 natural numbers is : 15
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.py"
Enter the range of number:15
first 15 natural numbers are : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Sum of 15 natural numbers is : 120
```

Conclusion : This program demonstrates how to use a loop to print the first *n* natural numbers and calculate their sum.

Experiment No: 1.5

Title: Write a program to determine whether the character entered is a Vowel or not .

Theory: Vowels are a set of letters in the alphabet that include 'a', 'e', 'i', 'o', and 'u' (both uppercase and lowercase).

Code:

```
#Write a program to determine whether the character entered is a Vowel or not
character = input("Enter a character: ")
if character.lower() in ['a', 'e', 'i', 'o', 'u','A','E','I','O','U']:
   print(f"{character} is a vowel.")
else:
   print(f"{character} is not a vowel.")
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp5.py"
Enter a character: Z
Z is not a vowel.
```

Test Case: Any two (screenshot) 1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp5.py"
Enter a character: a
a is a vowel.
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp5.py"
Enter a character: f
f is not a vowel.
```

Conclusion: This program demonstrates the use of a simple function to determine whether a given character is a vowel. The function checks if the input character is present in the set of vowels and returns True if it is a vowel, and False otherwise.

Experiment No: 1.6

Title: Write a program to find whether given number is an Armstrong Number.

Theory: An Armstrong number (also known as a narcissistic number, pluperfect digit, or pluperfect number) is a number that is the sum of its own digits each raised to the

power of the number of digits in the number.

Code:

```
#Write a program to find whether given number is an Armstrong Number.
'''number =str(input("ENTER THE NUMBER"))
digi=len(str(number))
if number=='''
num = int(input("Enter a number: "))

num_str = str(num)
sum_of_powers = sum(int(digit) ** len(num_str) for digit in num_str)
if sum_of_powers == num:
   print(f"{num} is an Armstrong number.")
else:
      print(f"{num} is not an Armstrong number.")
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp6.py"
Enter the Numbers:153
153 is  an Armstrong Number.
```

Test Case: Any two (screenshot) 1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp6.py"
Enter the Numbers:44
44 is not  an Armstrong Number.
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp6.py"
Enter a number: 66
66 is not an Armstrong number.
```

Conclusion:This program demonstrates the process of determining whether a given

number is an Armstrong number. The function `is_armstrong_number` calculates the sum of each digit raised to the power of the number of digits and checks if it equals the original number.

Experiment No:1.7

Title: Write a program using for loop to calculate factorial of a No.

Theory: The factorial of a non-negative integer n denoted by n!

Code:

```
#Write a program using for loop to calculate factorial of a No.
number = int(input("Enter a Number:"))
factorial = 1
for i in range (1 , number + 1):
   factorial *= i
print(f"factorial of {number} is {factorial}")
```

Output: (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp7.py"
Enter a Number:2
factorial of 2 is 2
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp7.py"
Enter a Number:10
factorial of 10 is 3628800
```

2.

```
/usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp7.py"
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp7.py"
Enter a Number:5
factorial of 5 is 120
```

Conclusion:This program illustrates the use of a `for` loop to calculate the factorial of a given number. The function `calculate_factorial` initializes a variable to 1 and multiplies it by each integer from 1 to the given number. The result is then printed.

Experiment No:1.8.i

Title: Write a program to print the following pattern

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

Theory: Using nested for loops to print ascending half pyramid pattern. One for loop for the rows and another for loop to print stars in each row. Number of stars in each row should be equal to the row they are in. Eg- In 4th row, there should be 4 stars, in 5th row, there should be 5 stars, etc.

Code:

```
#Write a program to print the following pattern
# *
# * *
# * * *
# * * * *
# * * * * *
num_rows = 5
for i in range (1 , num_rows+1):
   for j in range(i):
       print("*" , end = " ")
   print()
```

Output: (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp8.py"
*
* *
* * *
* * * *
* * * * *
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp8.py"
*
* *
* * *
* * * *
* * * * *
```

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp8.py"
*
* *
* * *
* * * *
* * * * *
```

Conclusion: Hence, printing a half pyramid ascending star pattern using nested for

loops.

Experiment No:1.8.ii

Title: Write a program to print the following pattern.

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

Theory:The outer loop (`for i in range(1, rows + 1)`) iterates through each row from 1 to the specified number of rows.The inner loop (`for j in range(i)`) iterates through each column in the current row, and it prints the row number (`i`) that many times.

Code:

```
#1
# 2 2
# 3 3 3
# 4 4 4 4
# 5 5 5 5 5
num_rows = 5
for i in range(1, num_rows + 1):
   for j in range(i):
       print(i, end=" ")
   print()
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp9.py"
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Test Case: Any two (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp9.py"
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp9.py"
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Conclusion: This program demonstrates the use of nested loops to print a triangular pattern. Understanding the structure of nested loops is crucial for creating patterns and matrices in programming.

Experiment No:1.8.iii

Title: Write a program to print the following pattern.

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

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

Theory: The outer loop (`for i in range(1, rows + 1)`) iterates through each row from 1 to the specified number of rows.The inner loop (`for j in range(i)`) iterates through each column in the current row, and it prints an asterisk (`*`) that many times.

Code:

```
#          *
#        * * *
#      * * * * *
#    * * * * * * *
# * * * * * * * * *
num_rows = 5
for i in range(1, num_rows + 1):
    for _ in range(num_rows - i):
        print(" ", end=" ")
    for j in range(2 * i - 1):
        print("*", end=" ")
    print()
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp10.py"
          *
        * * *
      * * * * *
    * * * * * * *
  * * * * * * * * *
```

Test Case: Any two (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp10.py"
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp10.py"
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

Conclusion: This program demonstrates the use of nested loops to print a triangular pattern of asterisks. The outer loop controls the number of rows, and the inner loop controls the number of asterisks printed in each row.

Experiment No:2.1

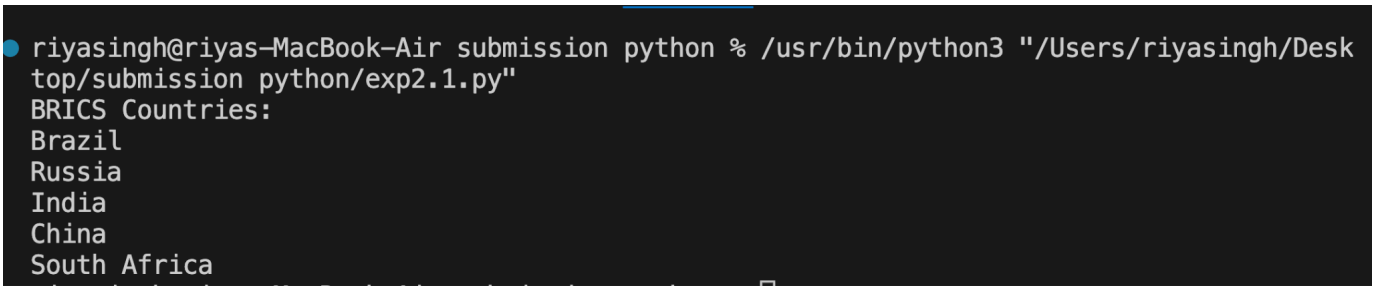
Title: Write a program that define the list of defines the list of define countries that are in BRICS.

Theory:BRICS countries= Brazil, Russia, India, China, South Africa. Using a membership operator(in), can check whether country given by user is a BRICS member or not. Can use in operator to check whether the value of variable is inside BRICS list or not.

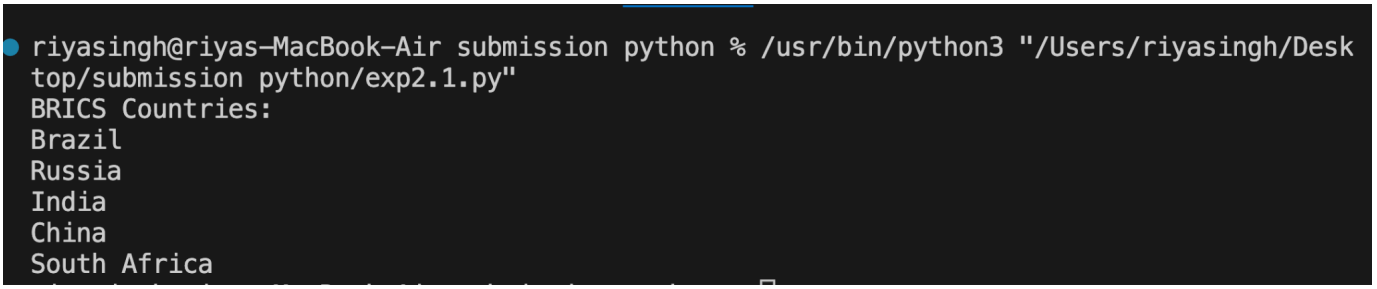
Code:

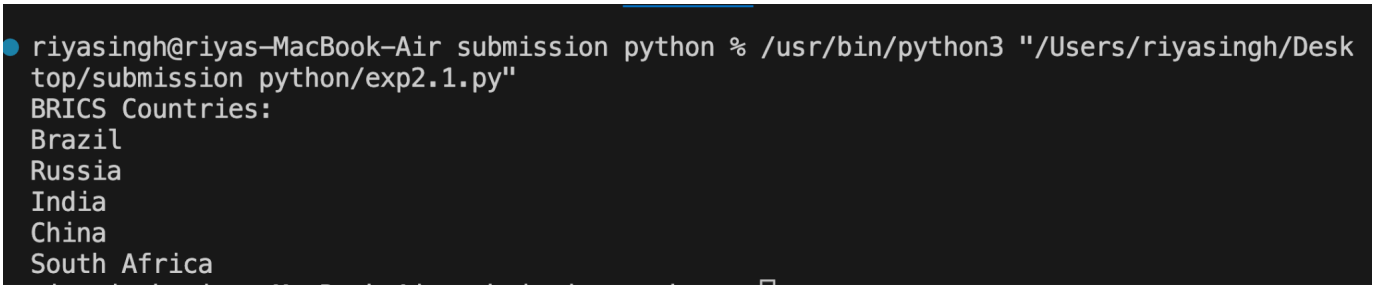
```
#Write a program that define the list of defines the list of define countries that are in BRICS
brics_countries = ["Brazil", "Russia", "India", "China", "South Africa"]
print("BRICS Countries:")
for country in brics_countries:
   print(country)
```

Output: **(screenshot)**



Test Case: Any two (screenshot)





Conclusion: Hence, using a membership operator(in), check user given country is in BRICS or not and printing the appropriate

message using if else statement.

Experiment No:2.2.1,2

Title: Write a program to traverse a list in reverse order.

1.By using Reverse method.
2.By using slicing

Theory: 1.By using predefined function reverse(),
we can reverse a list and print it.
2.By using index slicing, we can print the list in reverse using step size as -1.

Code:

```
#1.By using Reverse method.
list = [1, 2, 3, 4, 5]
list.reverse()
print("Reversed List:")
for item in list:
   print(item)
#2.By using slicing Define a sample list
list = [1, 2, 3, 4, 5]
reversed_list = list[::-1]
print("Reversed List (using slicing):")
for item in reversed_list:
   print(item)
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.2.py"
Reversed List:
5
4
3
2
1
Reversed List (using slicing):
5
4
3
2
1
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.2.py"
Reversed List:
5
4
3
2
1
Reversed List (using slicing):
5
4
3
2
1
riyasingh@riyas-MacBook-Air submission python %
```

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.2.py"
Reversed List:
5
4
3
2
1
Reversed List (using slicing):
5
4
3
2
1
riyasingh@riyas-MacBook-Air submission python %
```

Conclusion:Using Reverse Method: The `reverse` method modifies the original list in place by reversing its elements. It is important to note that this method alters the original list.

Using Slicing: Slicing creates a new list with elements in reverse order without modifying the original list. The syntax `my_list[::-1]` creates a new list starting from the end and going backward.

Experiment No:2.3

Title: Write a program that scans the email address and forms a tuple of username and domain.

Theory: Using split() function with @ as a parameter to split the email address given by user and store it in a tuple. Then print the first

element of the tuple as username and the second element from the same tuple as domain.

Code:

```
#Write a program that scans the email address and forms a tuple of username and domain
email_address = input("Enter your email address: ")
username, domain = email_address.split("@")
email_tuple = (username, domain)
print("Username:", email_tuple[0])
print("Domain:", email_tuple[1])
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.3.py"
Enter your email address: riya23@gmail.com
Username: riya23
Domain: gmail.com
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.3.py"
Enter your email address: bhagyashree@gmail.com
Username: bhagyashree
Domain: gmail.com
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.3.py"
Enter your email address: akriti@gmail.com
Username: akriti
Domain: gmail.com
```

Conclusion: Hence, using split() function to split the email address in username

and domain and printing them after storing them in a tuple.

Experiment No:2.4

Title: Write a program to create a list of tuples from given list having number and add its cube in tuple.

i/p: c= [2,3,4,5,6,7,8,9]

Theory: Using list comprehension, make another list with tuples as its elements. Each tuple would have the number from original list and it's cube using arithmetic operator(**).

Code:`#Write a program to create a list of tuples from given list having number and add its cube in tupple.`

```
# i/p: c= [2,3,4,5,6,7,8,9]
c = [2,3,4,5,6,7,8,9]
tuple_list = [(num, num**3) for num in c]
print("List of Tuples :")
for item in tuple_list:
   print(item)
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.4.py"
List of Tuples :
(2, 8)
(3, 27)
(4, 64)
(5, 125)
(6, 216)
(7, 343)
(8, 512)
(9, 729)
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using list comprehension to make a list of tuples containing the values and their cube using ** operator from the original list.

Experiment No:2.5

Title: Write a program to compare two dictionaries in Python?

(By using == operator)

Theory: Dictionary is a datatype in Python which stores information as key-value pairs, where keys are unique and are used to distinguish between values. Using relational operator(==) to check whether two dictionaries have same key-value pairs or not.

Code:
```
#Write a program to compare two dictionaries in Python?
```

```
# (By using == operator)
dict1 = {'a':1 , 'b':2 , 'c':3}
dict2 = {'a':2 , 'b':1 , 'c':4}
if dict1 == dict2:
   print("Dictionaries are equal")
else:
   print("Dictionaries are not  equal")
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp2.5.py"
Dictionaries are not  equal
```

Test Case: Any two (screenshot):Same as Above

Conclusion:Hence, using == operator to check whether two dictionaries have same key-value pairsor not and printing appropriate messages using if else statements.

Experiment No:2.6

Title: Write a program that creates dictionary of cube of odd numbers in the range.

Theory: Using for loop to iterate over every number from 1 till range given by user and using if statement to check whether a number is odd or even using modulus operator(%) and making the

number as key and its cube as a value, and printing the dictionary in the end.

Code:

```
#Write a program that creates dictionary of cube of odd numbers in the range.
start_range = int(input("Enter the start of the range: "))
end_range = int(input("Enter the end of the range: "))
odd_cubes_dict = {num: num**3 for num in range(start_range, end_range + 1) if num % 2 != 0}
print("Dictionary of Cubes of Odd Numbers:")
print(odd_cubes_dict)
```

Output: (screenshot)

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.6.py"
Enter the start of the range: 5
Enter the end of the range: 10
Dictionary of Cubes of Odd Numbers:
{5: 125, 7: 343, 9: 729}
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.6.py"
Enter the start of the range: 1
Enter the end of the range: 10
Dictionary of Cubes of Odd Numbers:
{1: 1, 3: 27, 5: 125, 7: 343, 9: 729}
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp2.6.py"
Enter the start of the range: 12
Enter the end of the range: 25
Dictionary of Cubes of Odd Numbers:
{13: 2197, 15: 3375, 17: 4913, 19: 6859, 21: 9261, 23: 12167, 25: 15625}
```

Conclusion: Hence, using for loop and if statement to create a dictionary of odd numbers as keys and their cube as values and printing the dictionary.

Experiment No:2.7

Title: Write a program for various list slicing operation.

a= [10,20,30,40,50,60,70,80,90,100]

i. Print Complete list
ii. Print 4th element of list
iii. Print list from0th to 4th index.
iv. Print list -7th to 3rd element
v. Appending an element to list.
vi. Sorting the element of list.
vii. Popping an element.
viii. Removing Specified element.
ix. Entering an element at specified index.
x. Counting the occurrence of a specified element.
xi. Extending list.
xii. Reversing the list.

Theory: Using index slicing to print specific elements, predefined functions such as append() to add an element in the list, sort() to sort the list in ascending or descending order, pop() to remove and element in the list, insert() to add an element in the list at a specific index, remove() to remove a specific element from the list, for loop to check the occurrence of an element, extend()

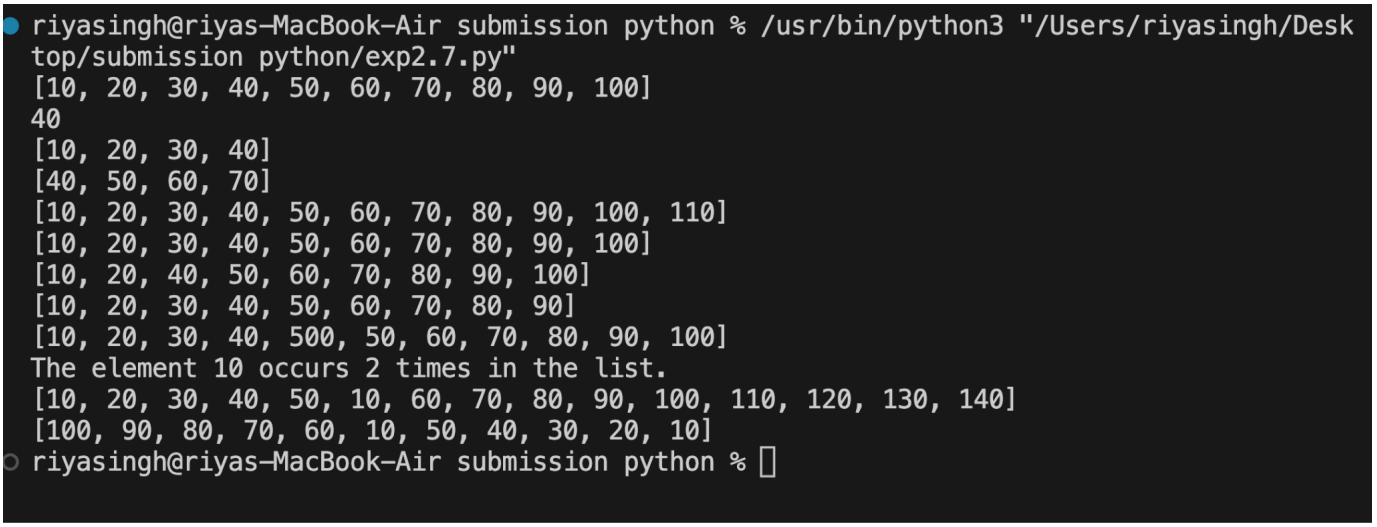
to add multiple elements in the list, reverse() to reverse the list.

Code:

```
#Write a program for various list slicing operation.
# a= [10,20,30,40,50,60,70,80,90,100]
# i. Print Complete list
a = [10,20,30,40,50,60,70,80,90,100]
print(a)

#ii. Print 4th element of list
a = [10,20,30,40,50,60,70,80,90,100]
print(a[3])

#iii. Print list from0th to 4th index.
a = [10,20,30,40,50,60,70,80,90,100]
print(a[0:4])

#iv. Print list -7th to 3rd element
a = [10,20,30,40,50,60,70,80,90,100]
print(a[-7:-3])

#v.Appending an element to list.
a = [10,20,30,40,50,60,70,80,90,100]
a.append(110)
print(a)

#vi.Sorting the element of list.
a = [20,50,10,80,60,100,30,40,70,90]
a.sort()
print(a)

#vii.Popping an element.
a = [10,20,30,40,50,60,70,80,90,100]
a.pop(2)
print(a)

#viii.Removing Specified element.
a = [10,20,30,40,50,60,70,80,90,100]
```

```
a.remove(100)
print(a)

#ix.Entering an element at specified index.
a = [10,20,30,40,50,60,70,80,90,100]
a.insert(4,500)
print(a)

#x.Counting the occurrence of a specified
element.
a = [10,20,30,40,50,10,60,70,80,90,100]
element_to_count = 10
occurrences = a.count(element_to_count)
print(f"The element {element_to_count}
occurs {occurrences} times in the list.")

#xi.Extending list.
a = [10,20,30,40,50,10,60,70,80,90,100]
b = [110,120,130,140]
a.extend(b)
print(a)

#xii.Reversing the list.
a = [10,20,30,40,50,10,60,70,80,90,100]
a.reverse()
print(a)
```

Output: **(screenshot)**



Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using for loop, slicing, and various predefined list functions to add, remove, sort and reverse a list.

Experiment No:3.1

Title: Write a program to extend a list in python by using given approach.

i. By using + operator.
ii. By using Append ()
iii. By using extend ()

Theory:+ operator is used to add a list to another list. Append() is used to add an element at the end of the list. Extend() is used to add multiple elements at the end of the list.

Code:

```
#3.1 Write a program to extend a list in python by using given approach.
# i. By using + operator.
a = [ 1,2,3,4,5]
b = [6,7,8,9,10]
c = a + b
print(c)

# ii. By using Append ()
a = [ 1,2,3,4,5]
a.append(6)
print(a)

# iii. By using extend ()
a = [ 1,2,3,4,5]
b = [6,7,8,9,10]
a.extend(b)
print(a)
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp3.1.py"
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using for loop and if statement to create a dictionary of odd numbers as keys and their cube as values and printing the dictionary.

Experiment No:3.2

Title: Write a program to add two matrices.

Theory: A matrix is a nested list in Python(a list of lists). It consists of two lists which have three lists inside each of them. Using nested for loop, one for each of the two lists, add elements from the two matrices and store the sum in another matrix of same dimension and print it.

Code:

```
#Write a program to add two matrices.
A = [ [1,5,8],
     [4,6,7],
     [7,2,3]]
B = [ [4,5,6],
     [8,9,1],
     [3,5,6]]
result = [ [0,0,0],
          [0,0,0],
          [0,0,0]]
for i in range (len(A)):
   for j in range(len(A[0])):
       result[i][j] = A[i][j] + B[i][j]
for r in result:
   print(r)
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.3.2.py"
[5, 10, 14]
[12, 15, 8]
[10, 7, 9]
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using nested for loop, add the elements of two nested lists and store the sum in another nested list

Experiment No:3.3

Title: Write a Python function that takes a list and returns a new list with distinct elements from the first list.

Theory: Making a user defined function distin() taking a user list as a parameter and then making an empty list and using a for loop to iterate through the user list and using membership operator(in) to check whether the element is present in the empty list too. If it is not present, append it in the list and check the next element for the same.

Code:

```
#Write a Python function that takes a list and returns a new list with distinct elements from the first list
def get_unique_elements(input_list):
    unique_elements = []
    for element in input_list:
        if element not in unique_elements:
            unique_elements.append(element)
    return unique_elements
original_list = [1, 2, 2, 3, 4, 4, 5, 6, 6]
result_list =
get_unique_elements(original_list)
print("Original List:", original_list)
```

```
print("List with Distinct Elements:",
result_list)
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp3.3.py"
Original List: [1, 2, 2, 3, 4, 4, 5, 6, 6]
List with Distinct Elements: [1, 2, 3, 4, 5, 6]
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, by making a function, using nested for loop to check whether each element in the list is only present once and appending it to another list and then printing the other list at the end.

Experiment No:3.4

Title: Write a program to Check whether a number is perfect or not.

Theory: A number is a perfect number if the sum of its divisors(excluding the number itself) is equal to the number itself. Eg- 6 is a perfect number as divisors of 6 are 1,2,3. Sum- 1+2+3=6 which is equal to the number itself.

Code:

```
#Write a program to Check whether a
number is perfect or not.
number = int(input("Enter a number: "))
```

```
if number <= 0:
   print("Please enter a positive integer.")
else:
   divisor_sum = 0
   for i in range(1, number):
       if number % i == 0:
           divisor_sum += i
   if divisor_sum == number:
            print(f"{number} is a perfect
number.")
   else:
          print(f"{number} is not a perfect
number.")
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.3.4.py"
Enter a number: 4
4 is not a perfect number.
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.3.4.py"
Enter a number: 22
22 is not a perfect number.
riyasingh@riyas-MacBook-Air submission python %
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.3.4.py"
Enter a number: 6
6 is a perfect number.
riyasingh@riyas-MacBook-Air submission python %
```

Conclusion: Hence, using a for loop to iterate over each number from 1 to number itself(exclusive) and checking whether it is a divisor of the number using if else statement

and adding it in sum variable and checking afterwards if the sum is equal to the number and printing the appropriate message using if else statement.

Experiment No:3.5

Title: Write a Python function that accepts a string and counts the number of upper and lower-case letters.

string_test= 'Today is My Best Day'

Theory:

Using a for loop to iterate each character in the string and if elif statement and predefined functions is upper() and is lower() to check whether a character is lowercase or uppercase and incrementing the value of the respective uppercase or lowercase counter by 1.

Code:

```
#Write a Python function that accepts a string and counts the number of upper and lower-case letters.
# string_test= 'Today is My Best Day'
def count_upper_lower(string):
   upper_count = 0
   lower_count = 0
   for char in string:
       if char.isupper():
           upper_count += 1
       elif char.islower():
           lower_count += 1

   return upper_count, lower_count
string_test = 'Today is My Best Day'
upper,                lower                =
count_upper_lower(string_test)
```

```
print("Original String:", string_test)
print("Number of Uppercase Letters:", upper)
print("Number of Lowercase Letters:", lower)
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.3.5.py"
Original String: Today is My Best Day
Number of Uppercase Letters: 4
Number of Lowercase Letters: 12
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using a for loop to iterate over each character in the string and using is upper() and is lower() to check the characters and increment the counter variables value by 1 using if elif statement.

Experiment No:4.1

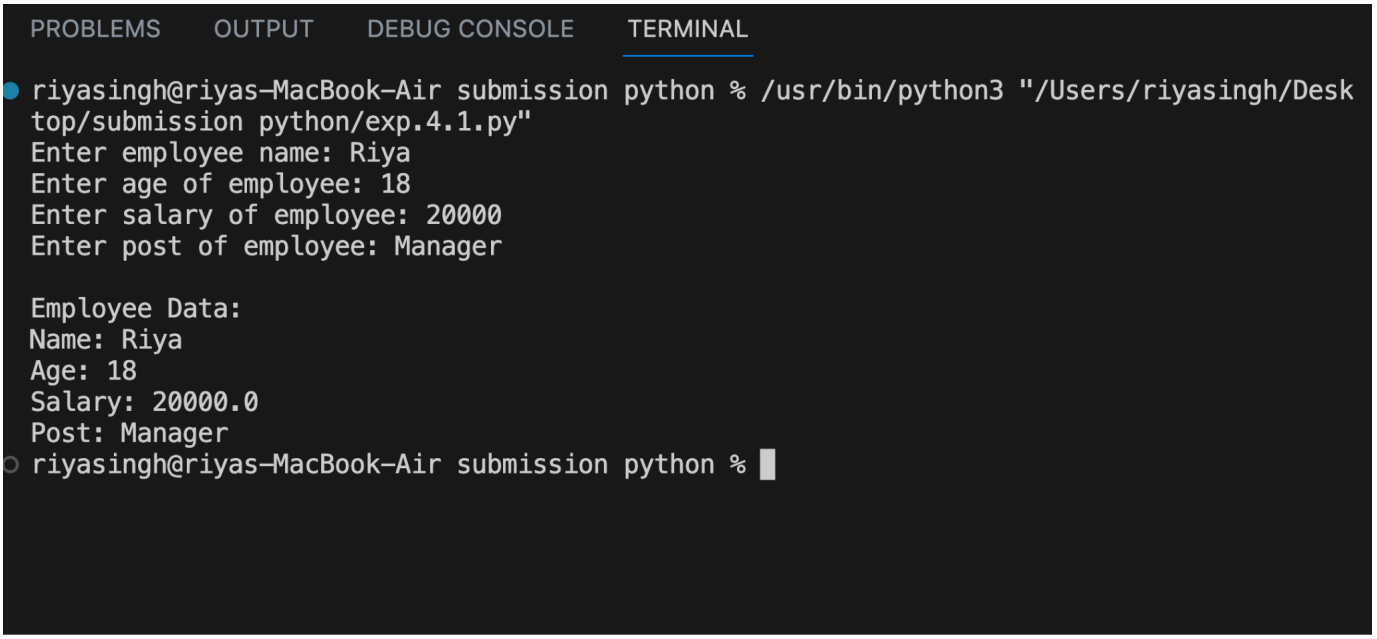
Title: Write a program to Create Employee Class & add methods to get employee details & print.

Theory:A class is a blueprint for objects. It contains attributes and methods which the objects can access and use. An object is an instance of a class. Each class has its own copy of attributes and share the methods of the class.

Code:

```
#Write a program to Create Employee Class & add methods to get employee details and print
class Employee:
   __name=""
   __age=0
   __salary=0
   _post=""
   def setData(self):
      name=input("Enter employee name: ")
      self.__name=name
       age=int(input("Enter age of employee: "))
      self.__age=age
         salary=float(input("Enter salary of employee: "))
      self.__salary=salary
      post=input("Enter post of employee: ")
      self.__post=post
   def getData(self):
       print("\nEmployee Data:")
       print("Name:", self.__name)
       print("Age:", self.__age)
       print("Salary:", self.__salary)
       print("Post:", self.__post)
a=Employee()
a.setData()
a.getData()
```

Output: (screenshot)



Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using get and set data functions to take values from the user and then assigning the attributes of the object these values and printing them to the user.

Experiment No:4.2

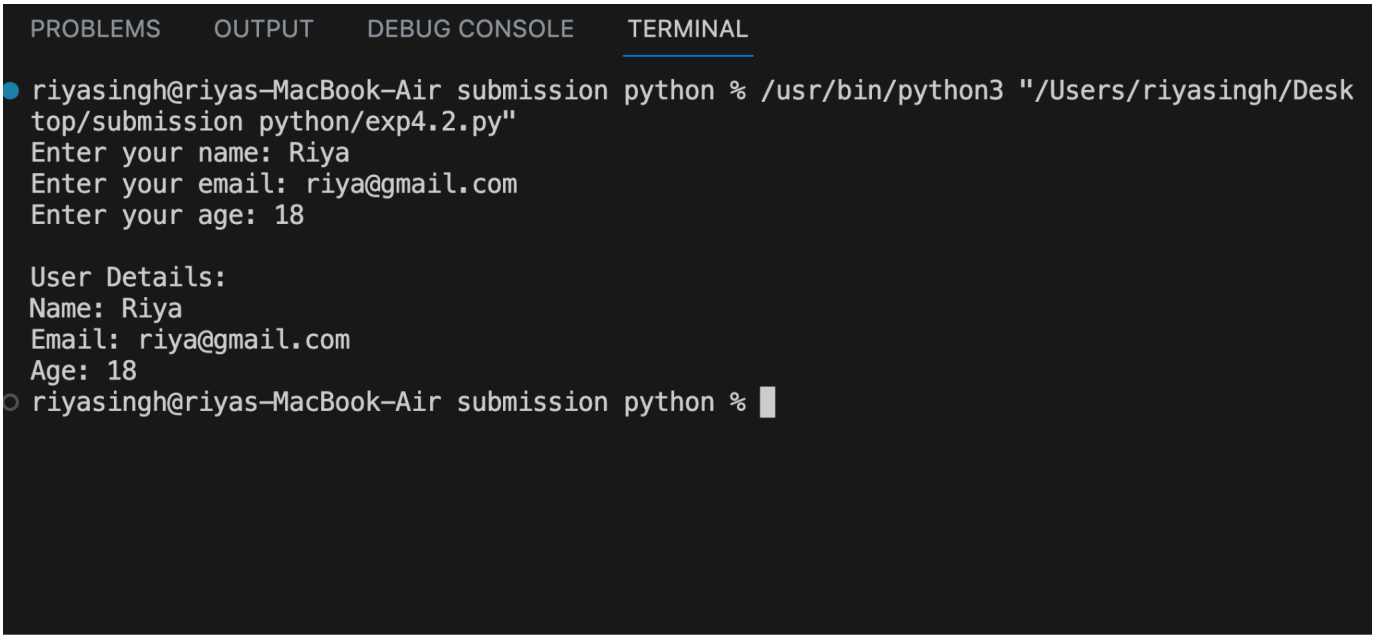
Title: Write a program to take input as name, email & age from user using combination of keywords argument and positional arguments (*args and**kwargs) using function,

Theory: While defining a function, we use *args and **kwargs as parameters when we don't know how many arguments the user will pass during function call. *args will take a tuple of positional arguments as parameter and work on it. **kwargs will take a dictionary of keyword arguments as parameter and assign the keys to keywords in arguments and the values as the values passed by the user in the arguments.

Code:

```
#Write a program to take input as name, email & age from user usingcombination of keywords argument and positional arguments (*argsand**kwargs) using function
def get_user_details(*args, **kwargs):
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    age = int(input("Enter your age: "))
    if args:
        name = args[0]
    if kwargs:
        email = kwargs.get('email', email)
        age = kwargs.get('age', age)
    return name, email, age
name, email, age = get_user_details()
print("\nUser Details:")
print("Name:", name)
print("Email:", email)
print("Age:", age)
```

Output: **(screenshot)**



Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using dynamic argument passing(*args and **kwargs) to take multiple arguments from the user in a combination of positional and keyword arguments and printing them using a user defined function.

Experiment No:4.3

Title: Write a program to admit the students in the different Departments(pgdm/btech)and count the students. (Class, Object and Constructor).

Theory: A constructor is automatically called when an object is created for a class. It is defined by __init__(self). It is used to initialise all the

attributes of a class and self is a pointer which is used to tell the program to make changes only in a particular object and not access data or modify other objects.

Code: `#Write a program to admit the students in the differentDepartments(pgdm/btech)and count the students. (Class, Object and Constructor).`

```
class Student:
    def __init__(self, name, department):
        self.name = name
        self.department = department

class Department:
    def __init__(self, name):
        self.name = name
        self.students = []

    def admit_student(self, student):
        self.students.append(student)

    def count_students(self):
        return len(self.students)
pgdm_department = Department("PGDM")
btech_department = Department("B.Tech")

pgdm_department.admit_student(Student("John Doe", "PGDM"))
pgdm_department.admit_student(Student("Jane Smith", "PGDM"))

btech_department.admit_student(Student("Bob Johnson", "B.Tech"))
btech_department.admit_student(Student("Alice Brown", "B.Tech"))

print(f"Number of students in {pgdm_department.name} department:
```

```
{pgdm_department.count_students()}")
print(f"Number of students in
{btech_department.name} department:
{btech_department.count_students()}")
```

Output: (screenshot)

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using constructors and get and set functions and using while loop to print a menu to the user and call functions as per the user's need and making a list of objects to store information of multiple students of different branches.

Experiment No:4.4

Title: Write a program that has a class store which keeps the record of code and price of product display the menu of all product and prompt to enter the quantity of each

item required and finally generate the bill and display the total amount.

Theory: Using constructor to print a menu of items to the user and getting quantity of the product and printing the bill with total amount at the end.

Code:

```
#Write a program that has a class store which keeps the record of code andprice of product display the menu of all product and prompt to enter the quantity ofeach item required and finally generate the bill and display the total amount.
class Store:
   def __init__(self):
          self.products = {"Product1": 100, "Product2": 50, "Product3": 30}
       self.bill = {}

   def display_menu(self):
       print("Product Menu:")
                   for product, price in self.products.items():
           print(f"{product}: Rs.{price}")

   def generate_bill(self):
       print("Your Bill:")
       total_amount = 0
                 for product, quantity in self.bill.items():
           price = self.products[product]
           total_price = price * quantity
             print(f"{product} x {quantity}: Rs.{total_price}")
           total_amount += total_price
```

```
                    print("Total   Amount:
Rs.{}".format(total_amount))


   def take_order(self, product, quantity):
           if product in self.products and
quantity > 0:
           self.bill[product] = quantity
             print(f"{quantity} {product}(s)
added to your bill.")
       else:
                  print("Invalid product or
quantity.")

store = Store()
store.display_menu()
store.take_order("Product1", 2)
store.take_order("Product3", 1)
store.generate_bill()
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.4.4.py"
Product Menu:
Product1: Rs.100
Product2: Rs.50
Product3: Rs.30
2 Product1(s) added to your bill.
1 Product3(s) added to your bill.
Your Bill:
Product1 x 2: Rs.200
Product3 x 1: Rs.30
Total Amount: Rs.230
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot):Same as Above

Conclusion:

Experiment No:4.5

Title: Write a program to take input from user for addition of two numbers using (single inheritance).

Theory: Single inheritance is when there is a single parent class and a single child class which inherits the attributes and methods of the parent class. It reduces lines of code as instead of writing the same attributes and methods again in a new class, we can simply inherit them.

Code:

```
#Write a program to take input from user for addition of two numbers using(single inheritance).
class Addition:
   def add(a,b):
       print("Sum:",a+b)
class Numbers(Addition):
   def getNumbers(self):
       c=float(input("Enter a number: "))
        d=float(input("Enter another number: "))
       Addition.add(c,d)
e=Numbers()
e.getNumbers()
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.5.py"
Enter a number: 5
Enter another number: 5
Sum: 10.0
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.5.py"
Enter a number: 2
Enter another number: 12
Sum: 14.0
riyasingh@riyas-MacBook-Air submission python %
```

2.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.5.py"
Enter a number: 34
Enter another number: 2
Sum: 36.0
riyasingh@riyas-MacBook-Air submission python %
```

Conclusion: Hence, using single inheritance to take values from the user from method of child class and adding and printing the sum from method of parent class.

Experiment No:4.6

Title: Write a program to create two base classes LU and ITM and one derived class. (Multiple inheritance).

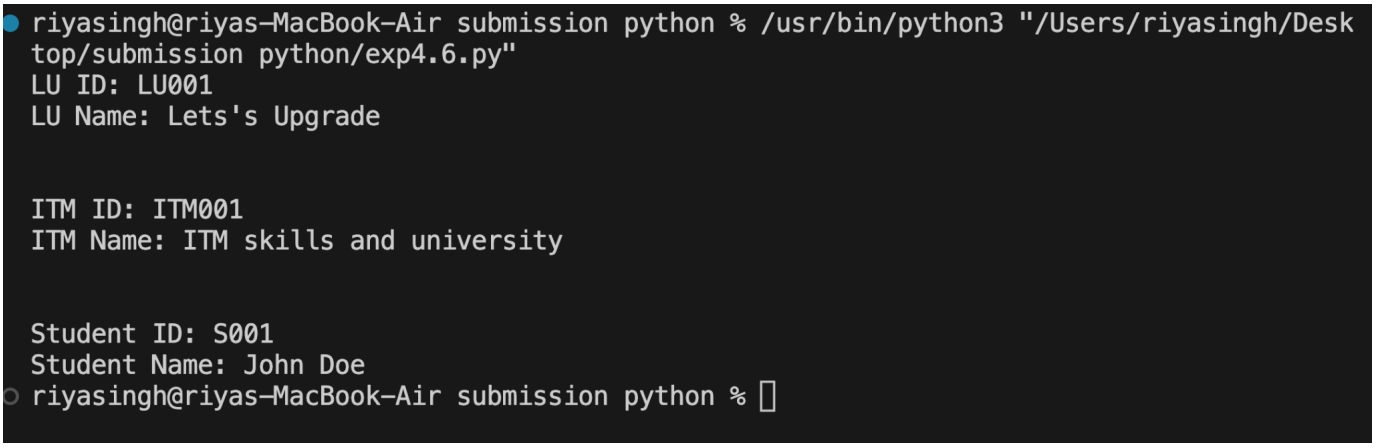
Theory: Multiple inheritance is when a single child class inherits methods and attributes from multiple parent classes.

Code:

```
#Write a program to create two base classes LU and ITM and one derived class.(Multiple inheritance).
class LU:
   def __init__(self, lu_id, lu_name):
       self.lu_id = lu_id
       self.lu_name = lu_name

   def display_lu_info(self):
        print(f"LU ID: {self.lu_id}\nLU Name: {self.lu_name}")


class ITM:
   def __init__(self, itm_id, itm_name):
       self.itm_id = itm_id
       self.itm_name = itm_name

   def display_itm_info(self):
          print(f"ITM ID: {self.itm_id}\nITM Name: {self.itm_name}")


class Student(LU, ITM):
      def __init__(self, lu_id, lu_name, itm_id, itm_name, student_id, student_name):
```

```
        LU.__init__(self, lu_id, lu_name)
        ITM.__init__(self, itm_id, itm_name)

        self.student_id = student_id
        self.student_name = student_name

    def display_student_info(self):
                        print(f"Student    ID:
{self.student_id}\nStudent              Name:
{self.student_name}")


# Example usage
lu_id = "LU001"
lu_name = "Lets's Upgrade"
itm_id = "ITM001"
itm_name = "ITM skills and university"
student_id = "S001"
student_name = "John Doe"

student  =  Student(lu_id,  lu_name,  itm_id,
itm_name, student_id, student_name)

student.display_lu_info()
print("\n")
student.display_itm_info()
print("\n")
student.display_student_info()
```

Output: **(screenshot)**



Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using multiple inheritance to show courses from multiple classes(LU and ITM).

Experiment No:4.7

Title: Write a program to implement Multilevel inheritance,

Grandfather → Father- → Child to show property inheritance from grandfather to child.

Theory: Multilevel inheritance is when there is a parent class which has a child class which in turn has a child class of its own. The last child class inherits all the methods and attributes from its parent class as well as from the grandfather class.

Code:#Write a program to implement Multilevel inheritance,Grandfather→Father-→Child to show property inheritance from grandfather tochild.

```
class Grandfather:
   def __init__(self, grandfather_property):
                self.grandfather_property  =
grandfather_property

   def display_grandfather_property(self):
              print(f"Grandfather  Property:
{self.grandfather_property}")



class Father(Grandfather):
    def __init__(self, grandfather_property,
father_property):

super().__init__(grandfather_property)
                    self.father_property   =
father_property

   def display_father_property(self):
                  print(f"Father  Property:
{self.father_property}")



class Child(Father):
    def __init__(self, grandfather_property,
father_property, child_property):

super().__init__(grandfather_property,
father_property)
       self.child_property = child_property

   def display_child_property(self):
                   print(f"Child  Property:
{self.child_property}")
```

```
# Example usage
grandfather_property = "Grandfather's Land"
father_property = "Father's Business"
child_property = "Child's Toy"

child_instance = Child(grandfather_property,
father_property, child_property)

child_instance.display_grandfather_property(
)
print("\n")
child_instance.display_father_property()
print("\n")
child_instance.display_child_property()
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp4.7.py"
Grandfather Property: Grandfather's Land


Father Property: Father's Business


Child Property: Child's Toy
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using multilevel inheritance to take name from the user and calculate inherited and purchased property from grandfather and father for the child and their full name using

methods and attributes from the grandfather and father class.

Experiment No:4.8

Title: Write a program Design the Library catalogue system using inheritance take base class (library item) and derived class (Book, DVD & Journal) Each derived class should have unique attribute and methods and system should support Check in and check out the system. (Using Inheritance and Method overriding)

Theory: Method overriding is when there is a method of same name in both base class and derived class and when an object is created of derived class and method is called, derived class object is called and base class method is overridden.

Code:

```
#Write a program Design the Library catalogue system using inheritance takebase class (library item) and derived class (Book, DVD & Journal) Each derivedclass should have unique attribute and methods and system should support Checkin and check out the system. (Using Inheritance and Method overriding)
class LibraryItem:
```

```
      def  __init__(self,  title,  item_id,
available=True):
      self.title = title
      self.item_id = item_id
      self.available = available

   def display_info(self):
                          print(f"Item    ID:
{self.item_id}\nTitle:
{self.title}\nAvailable: {self.available}")

   def check_out(self):
      if self.available:
           print(f"{self.title} checked out
successfully.")
          self.available = False
      else:
               print(f"{self.title}  is  not
available for check-out.")

   def check_in(self):
      if not self.available:
            print(f"{self.title} checked in
successfully.")
          self.available = True
      else:
            print(f"{self.title} is already
available.")

class Book(LibraryItem):
      def  __init__(self,  title,  item_id,
author, num_pages):
      super().__init__(title, item_id)
      self.author = author
      self.num_pages = num_pages

   def display_info(self):
      super().display_info()
       print(f"Author: {self.author}\nNumber
of Pages: {self.num_pages}")

class DVD(LibraryItem):
```

```
        def __init__(self, title, item_id,
director, duration):
        super().__init__(title, item_id)
        self.director = director
        self.duration = duration

    def display_info(self):
        super().display_info()
                            print(f"Director:
{self.director}\nDuration:    {self.duration}
minutes")

class Journal(LibraryItem):
        def __init__(self, title, item_id,
issue_number):
        super().__init__(title, item_id)
        self.issue_number = issue_number

    def display_info(self):
        super().display_info()
                        print(f"Issue   Number:
{self.issue_number}")

# Example usage
book = Book("The Catcher in the Rye",
"B001", "J.D. Salinger", 224)
dvd = DVD("The Shawshank Redemption",
"D001", "Frank Darabont", 142)
journal = Journal("Science Journal", "J001",
25)

book.display_info()
print("\n")
book.check_out()
print("\n")
book.check_in()

print("\n----------------------\n")

dvd.display_info()
print("\n")
dvd.check_out()
```

```
print("\n")
dvd.check_in()

print("\n-----------------------\n")

journal.display_info()
print("\n")
journal.check_out()
print("\n")
journal.check_in()
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/exp.4.8.py"
Item ID: B001
Title: The Catcher in the Rye
Available: True
Author: J.D. Salinger
Number of Pages: 224


The Catcher in the Rye checked out successfully.


The Catcher in the Rye checked in successfully.

-----------------------

Item ID: D001
Title: The Shawshank Redemption
Available: True
Director: Frank Darabont
Duration: 142 minutes


The Shawshank Redemption checked out successfully.


The Shawshank Redemption checked in successfully.

-----------------------

Item ID: J001
Title: Science Journal
Available: True
Issue Number: 25
```

```
Science Journal checked out successfully.


Science Journal checked in successfully.
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using single inheritance and method overriding, made a library catalogue system having checkout system for movies, and journals.

Experiment No:5.1

Title: Write a program to create my_module for addition of two numbers and import it in main script.

Theory: Module is a collection of functions. Its functions can be used in another program by importing the module.

Code:

```python
# main_script.py
import my_module


# Taking user input for two numbers
num1 = float(input("Enter the first number:
"))
```

```
num2 = float(input("Enter the second number:
"))

# Using the add_numbers function from the
imported module
result = my_module.add_numbers(num1, num2)

# Displaying the result
print(f"The sum of {num1} and {num2} is:
{result}")
```

```
#Write a program to create my_module for
addition of two numbers and import  it in
main script.
def add_numbers(num1, num2):
   return num1 + num2
```

Output: **(screenshot)**

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desk
top/submission python/51.py/main.py"
Enter the first number: 3
Enter the second number: 4
The sum of 3.0 and 4.0 is: 7.0
Enter the first number:
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, creating a module for addition of two values given by the user and printing it by importing the module in main program.

Experiment No:5.2

Title: Write a program to create the Bank Module to perform the operations such as Check the Balance, withdraw and deposit the money in bank account and import the module in main file.

Theory: Module is a collection of functions. Its functions can be used in another program by importing the module.

Code:

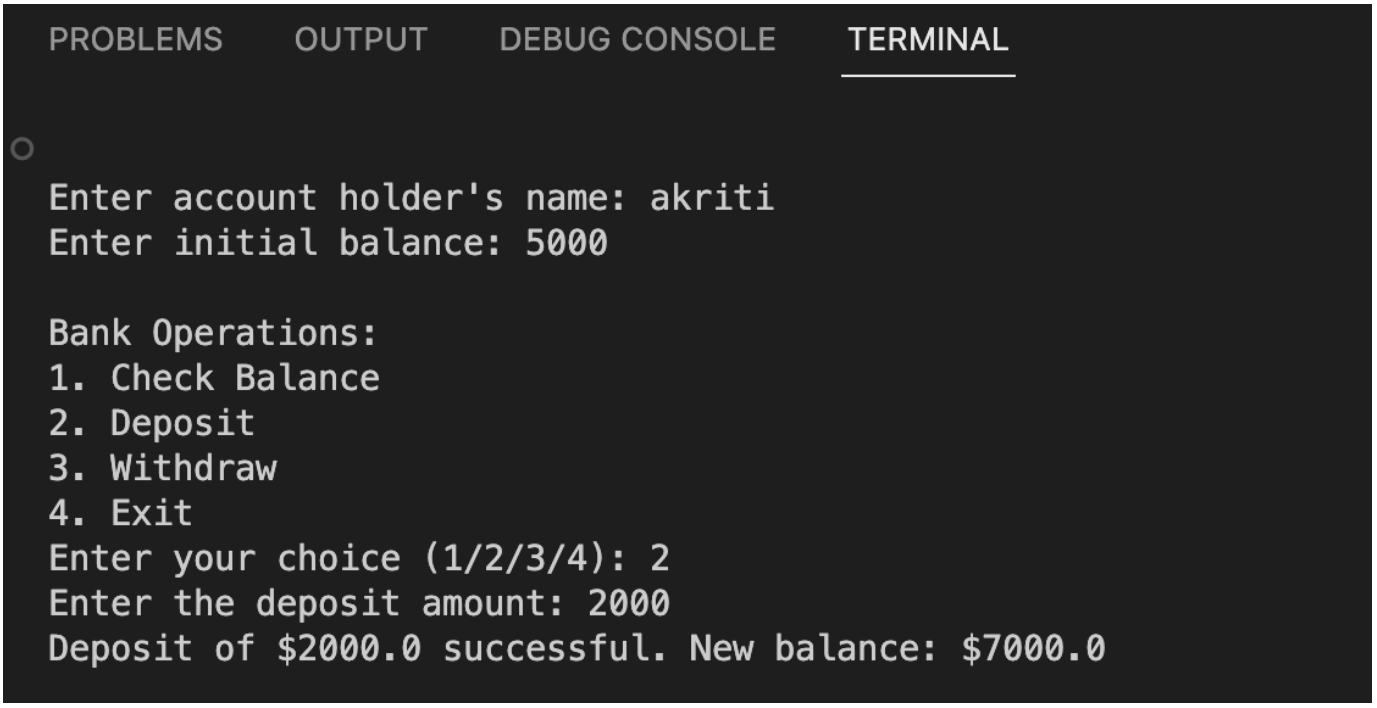
```
# bank_module.py
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        self.account_holder = account_holder
        self.balance = balance
    def check_balance(self):
        return self.balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposit of ${amount} successful. New balance: ${self.balance}")
        else:
            print("Invalid deposit amount. Please enter a positive amount.")
    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawal of ${amount} successful. New balance: ${self.balance}")
        else:
            print("Invalid withdrawal amount or insufficient funds.")
# main_script.py
from bank_module import BankAccount
def main():
    account_holder_name = input("Enter account holder's name: ")
    initial_balance = float(input("Enter initial balance: "))
    account = BankAccount(account_holder_name, initial_balance)
    while True:
```

```
    print("\nBank Operations:")
    print("1. Check Balance")
    print("2. Deposit")
    print("3. Withdraw")
    print("4. Exit")
    choice = input("Enter your choice (1/2/3/4): ")
    if choice == '1':
        print(f"Current Balance: ${account.check_balance()}")
    elif choice == '2':
        deposit_amount = float(input("Enter the deposit amount: "))
        account.deposit(deposit_amount)
    elif choice == '3':
        withdrawal_amount = float(input("Enter the withdrawal amount: "))
        account.withdraw(withdrawal_amount)
    elif choice == '4':
        print("Exiting program. Goodbye!")
        break
    else:
        print("Invalid choice. Please enter 1, 2, 3, or 4.")
if __name__ == "__main__":
  main()
```

Output: **(screenshot):**

```
kAccount/main_script.py"
Enter account holder's name: akriti
Enter initial balance: 2000

Bank Operations:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice (1/2/3/4): 1
Current Balance: $2000.0

Bank Operations:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice (1/2/3/4): 2
Enter the deposit amount: 3
Deposit of $3.0 successful. New balance: $2003.0

Bank Operations:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice (1/2/3/4): 4
Exiting program. Goodbye!
```

Test Case: Any two (screenshot):



Same as Above

Conclusion: Hence, creating a module for bank ATM with functions for deposit, withdraw, and checking bank balance and asking the user for deposit, withdraw or check bank balance and calling the user specified function using while loop.

Experiment No:5.3

Title: Write a program to create a package with name cars and add different modules (such as BMW, AUDI, NISSAN) having classes and functionality and import them in main file cars.

Theory:A package is a collection of modules. It is a folder containing modules and __init__.py file to let python treat the folder as a package. Each module has methods for drive and start engine.

Code:

```
# audi.py
class Audi:
    def __init__(self, model):
        self.model = model
    def start_engine(self):
        print(f"Audi {self.model}'s engine is started.")
    def drive(self):
        print(f"Driving the Audi {self.model}.")
# bmw.py
class BMW:
    def __init__(self, model):
        self.model = model
    def start_engine(self):
        print(f"BMW {self.model}'s engine is started.")
    def drive(self):
        print(f"Driving the BMW {self.model}.")
class Nissan:
    def __init__(self, model):
        self.model = model
    def start_engine(self):
        print(f"Nissan {self.model}'s engine is started.")
    def drive(self):
        print(f"Driving the Nissan {self.model}.")
# main_script.py
from cars import bmw, audi, nissan
def main():
    bmw_car = bmw.BMW(model="X5")
    audi_car = audi.Audi(model="A3")
    nissan_car = nissan.Nissan(model="Altima")
    print("BMW Functionality:")
    bmw_car.start_engine()
    bmw_car.drive()
    print("\nAudi Functionality:")
    audi_car.start_engine()
    audi_car.drive()
    print("\nNissan Functionality:")
    nissan_car.start_engine()
    nissan_car.drive()
if __name__ == "__main__":
    main()
```

Output: **(screenshot)**

```
ncars/main_script.py"
BMW Functionality:
BMW X5's engine is started.
Driving the BMW X5.

Audi Functionality:
Audi A3's engine is started.
Driving the Audi A3.

Nissan Functionality:
Nissan Altima's engine is started.
Driving the Nissan Altima.
```

Test Case: Any two (screenshot)

Conclusion: Hence, creating a package containing modules for different car models, with each module containing methods for each car model.

Experiment No:6

Title: Write a program to implement Multithreading. Printing “Hello” with one thread & printing “Hi” with another thread.

Theory: Multithreading is used for multitasking as multiple processes are run in parallel at the same time. Each process is given a thread to run on and all the threads are joined together to create a main thread at the end of execution.

Sleep is a function in time module which is used to suspend the execution of a thread for a specified number of seconds.

Code:

```
#Write a program to implement Multithreading. Printing "Hello" with onethread & printing "Hi" with another thread.
import threading

def print_hello():
   for _ in range(5):
       print("Hello")

def print_hi():
   for _ in range(5):
       print("Hi")

# Create two thread objects
thread_hello =
threading.Thread(target=print_hello)
thread_hi =
threading.Thread(target=print_hi)

thread_hello.start()
thread_hi.start()

thread_hello.join()
thread_hi.join()

print("Threads have finished.")
```

Output: **(screenshot)**

```
/usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp6.1.py"
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop
hon/exp6.1.py"
Hello
Hello
Hello
Hello
Hi
Hello
Hi
Hi
Hi
Hi
Threads have finished.
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, by using multithreading module in python to run multiple processes at once in parallel with each process getting their own threads to run on and joining them all at the end of execution of program. Also using sleep function from time module to let the processes on the threads run after a specified interval of time.

Experiment No:7.1

Title: Write a program to use ‘whether API’ and print temperature of any city, also print the sunrise and sunset times for the same humidity of that area.

Theory: Requests module is used to connect a website with our program and an API key is required to get information from any website. API key is used for authentication of a user for getting any information from a website. Datetime module is used to convert Unix Epoch time to IST time for user convenience.

Code:

```
#Write a program to use 'whether API' and print temperature of any city, alsoprint the sunrise and sunset times for the same humidity of that area.
import datetime as dt
import requests
baseurl="http://api.openweathermap.org/data/2.5/weather?"
api_key="6c253d1a79548fc3cb1314dc0c95ceb5"
city=input('Enter City: ')

def kel_to_cel_fahren(kelvin):
    celsius=kelvin-273
    fahrenheit=celsius*(9/5)+32
    return celsius,fahrenheit

url= baseurl + 'appid=' + api_key + '&q=' + city
response=requests.get(url).json()
print(response)

temp_kelvin=response['main']['temp']
temp_celsius,
temp_fahrenheit=kel_to_cel_fahren(temp_kelvin)
max_temp=response['main']['temp_max']
tempc,tempf=kel_to_cel_fahren(max_temp)
min_temp=response['main']['temp_min']
temc,temf=kel_to_cel_fahren(min_temp)
```

```
humidity=response['main']['humidity']
description=response['weather'][0]['descript
ion']
sunrise=dt.datetime.utcfromtimestamp(respons
e['sys']['sunrise']+response['timezone'])
sunset=dt.datetime.utcfromtimestamp(response
['sys']['sunset']+response['timezone'])

print(f"Tempertature in {city}:
{temp_celsius:.2f}'C or
{temp_fahrenheit}'F")
print(f"Maximum Tempertature in {city}:
{tempc:.2f}'C or {tempf}'F")
print(f"Minimum Tempertature in {city}:
{temc:.2f}'C or {temf}'F")
print(f"Humidity in {city}: {humidity}%")
print(f"General Weather in {city}:
{description}")
print(f"Sunrises in {city} at {sunrise}.")
print(f"Sunsets in {city} at {sunset}.")
```

Output: (screenshot)

```
/usr/bin/python3 "/Users/riyasingh/Desktop/submission python/exp7.1.py"
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission pyt
hon/exp7.1.py"
/Users/riyasingh/Library/Python/3.9/lib/python/site-packages/urllib3/__init__.py:34: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3
'. See: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
Enter City: Thane
{'coord': {'lon': 72.9667, 'lat': 19.2}, 'weather': [{'id': 711, 'main': 'Smoke', 'description': 'smoke', 'icon': '50d'}], 'base': 'stations', 'main': {'temp': 303.1, 'feels_like': 304.78, 'temp_min': 301
.05, 'temp_max': 303.1, 'pressure': 1010, 'humidity': 54}, 'visibility': 1800, 'wind': {'speed': 5.14, 'deg': 320}, 'clouds': {'all': 0}, 'dt': 1704194283, 'sys': {'type': 1, 'id': 9052, 'country': 'IN',
'sunrise': 1704159703, 'sunset': 1704199289}, 'timezone': 19800, 'id': 1254661, 'name': 'Thāne', 'cod': 200}
Tempertature in Thane: 30.10'C or 86.18000000000004'F
Maximum Tempertature in Thane: 30.10'C or 86.18000000000004'F
Minimum Tempertature in Thane: 28.05'C or 82.49000000000002'F
Humidity in Thane: 54%
General Weather in Thane: smoke
Sunrises in Thane at 2024-01-02 07:11:43.
Sunsets in Thane at 2024-01-02 18:11:29.
riyasingh@riyas-MacBook-Air submission python %
```

Test Case: Any two (screenshot)1.

```
riyasingh@riyas-MacBook-Air submission python % /usr/bin/python3 "/Users/riyasingh/Desktop/submission pyt
hon/exp7.1.py"
/Users/riyasingh/Library/Python/3.9/lib/python/site-packages/urllib3/__init__.py:34: NotOpenSSLWarning: u
rllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See
: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
Enter City: Vashi
{'coord': {'lon': 73.0005, 'lat': 19.0758}, 'weather': [{'id': 711, 'main': 'Smoke', 'description': 'smok
e', 'icon': '50d'}], 'base': 'stations', 'main': {'temp': 303.16, 'feels_like': 304.88, 'temp_min': 301.1
1, 'temp_max': 303.16, 'pressure': 1010, 'humidity': 54}, 'visibility': 1800, 'wind': {'speed': 5.14, 'de
g': 320}, 'clouds': {'all': 0}, 'dt': 1704194349, 'sys': {'type': 1, 'id': 9052, 'country': 'IN', 'sunris
e': 1704159681, 'sunset': 1704199295}, 'timezone': 19800, 'id': 6619347, 'name': 'Vashi', 'cod': 200}
Tempertature in Vashi: 30.16'C or 86.28800000000004'F
Maximum Tempertature in Vashi: 30.16'C or 86.28800000000004'F
Minimum Tempertature in Vashi: 28.11'C or 82.59800000000003'F
Humidity in Vashi: 54%
General Weather in Vashi: smoke
Sunrises in Vashi at 2024-01-02 07:11:21.
Sunsets in Vashi at 2024-01-02 18:11:35.
riyasingh@riyas-MacBook-Air submission python %
```

2.

```
/usr/bin/python3 "/Users/riyasingh/Desktop/submiss%
ion python/exp7.1.py"
gh/Desktop/submission python/exp7.1.py"
/Users/riyasingh/Library/Python/3.9/lib/python/site-packages/urllib3/__init__.py:34: NotOpenSSLWarning: u
rllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See
: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
Enter City: Seawoods
{'coord': {'lon': 73.0165, 'lat': 19.0134}, 'weather': [{'id': 711, 'main': 'Smoke', 'description': 'smok
e', 'icon': '50d'}], 'base': 'stations', 'main': {'temp': 303.16, 'feels_like': 304.88, 'temp_min': 301.1
1, 'temp_max': 303.16, 'pressure': 1010, 'humidity': 54}, 'visibility': 1800, 'wind': {'speed': 4.63, 'de
g': 290}, 'clouds': {'all': 0}, 'dt': 1704194379, 'sys': {'type': 1, 'id': 9052, 'country': 'IN', 'sunris
e': 1704159670, 'sunset': 1704199298}, 'timezone': 19800, 'id': 6619347, 'name': 'Seawoods', 'cod': 200}
Tempertature in Seawoods: 30.16'C or 86.28800000000004'F
Maximum Tempertature in Seawoods: 30.16'C or 86.28800000000004'F
Minimum Tempertature in Seawoods: 28.11'C or 82.59800000000003'F
Humidity in Seawoods: 54%
General Weather in Seawoods: smoke
Sunrises in Seawoods at 2024-01-02 07:11:10.
Sunsets in Seawoods at 2024-01-02 18:11:38.
riyasingh@riyas-MacBook-Air submission python %
```

Conclusion: Hence, using API of openweather to get weather details of user given city and printing the weather details with the help of requests module and date time module to convert Unix epoch time to IST time.

Experiment No:7.2

Title: Write a program to use the 'API' of crypto currency.

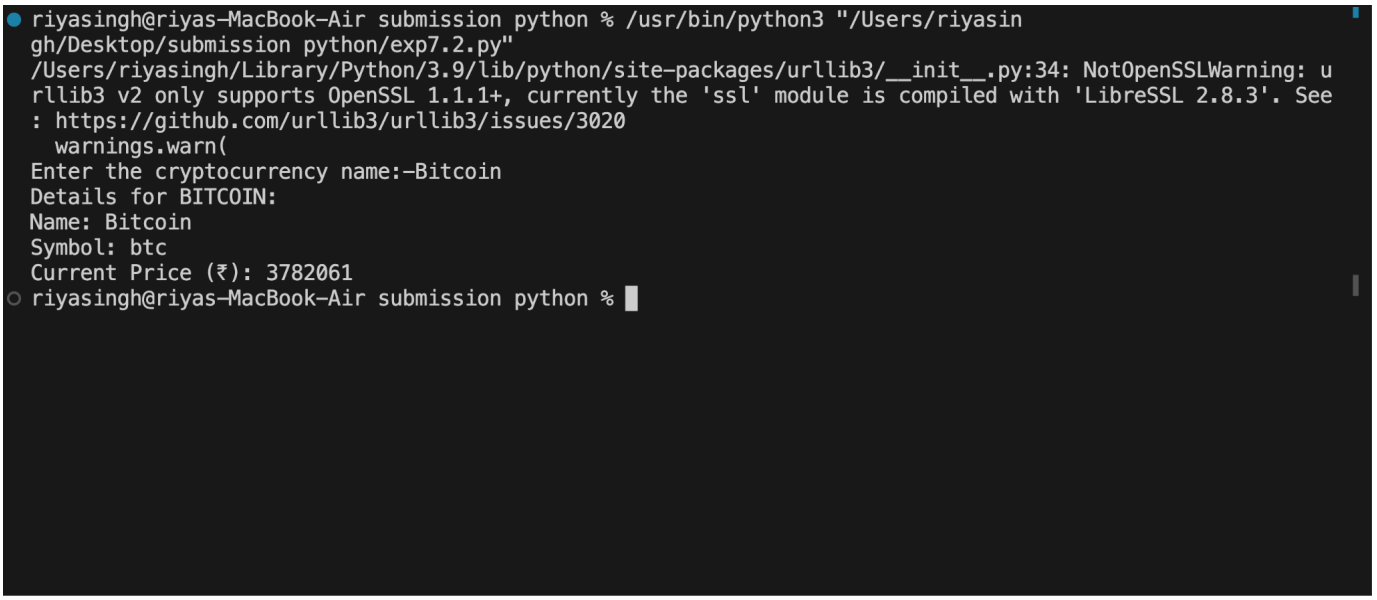
Theory: Requests module is used to connect a website with our program and an API key is required to get information from any website. API key is used for authentication of a user for getting any information from a website.

Code:

```
#Write a program to use the 'API' of crypto currency.
import requests

api_key="CG-hw5JqqeSPWRHKa2AdtbLLU3b"
base_url="https://api.coingecko.com/api/v3/coins/"


cryptocurrency = input("Enter the cryptocurrency name:-").lower()

url = f"{base_url}{cryptocurrency}"

headers = {
   "Content-Type": "application/json",
   "Authorization": f"Bearer {api_key}"
}

response = requests.get(url, headers=headers)

if response.status_code == 200:
   data = response.json()
   print(f"Details for {cryptocurrency.upper()}:")
   print("Name:", data['name'])
   print("Symbol:", data['symbol'])
```

```
            print("Current    Price    (₹):",
data['market_data']['current_price']['usd']*
83)

else:
       print(f"Failed   to   fetch   data   for
{cryptocurrency}.        Status        code:",
response.status_code)
```

Output: **(screenshot)**



Test Case: Any two (screenshot):Same as Above

Conclusion: Hence, using API of CoinGecko to get price of cryptocurrency given by the user and printing INR and USD price and asking for more crypto till the user chooses the option to terminate the program

using while loop.