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

PYTHON PROGRAMMING LAB

Prepared by:

Name of Student: Prem Thatikonda

Roll No: 06

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 numbers & 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 a given number is an Armstrong Number.
	1.7 Write a program using a 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 defines the list of countries that are in BRICS.
	2.2 Write a program to traverse a list in reverse order. 1.By using the 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 a given list having numbers and add its cube in tuples. 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 a dictionary of cubes of odd numbers in the range.
	2.7 Write a program for various list slicing operations. a= [10,20,30,40,50,60,70,80,90,100] i. Print Complete list ii. Print 4th element of list iii. Print list from 0th to 4th index. iv. Print list -7th to 3rd element v. Appending an element to the list. vi. Sorting the element of the list. vii. Popping an element. viii. Removing Specific elements. 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 a 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 the 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 a bank account and import the module in the 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 the 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: Prem Thatikonda

Roll Number: 06

Experiment No: 1

Title:

1.1 - Compute Simple Interest

1.2 - Perform arithmetic, Relational operators.

1.3 - Find whether a given no is even & odd.

1.4 - Print first n natural numbers & their sum.

1.5 - Determine whether a character entered is a vowel or not.

1.6 - Find whether a number is an armstrong number or not

1.7 - Loop to calculate the factorial of a number

1.8 - Pattern Programs

Theory:

1.1 - Simply take inputs for principal amount, rate of interest, time duration and calculate the simple interest by using the formula (principal * rate of interest * time duration) / 100.

1.2 - Take number inputs from the user and perform arithmetic operations on them, then use relational operators to find out the relationship between the numbers/values.

1.3 - Check if the remainder you get on dividing the entered number with 2 is 0.

1.4 - Run a for loop from 1 to 'n' (number entered by the user) and just print every value in that range, while adding that number to a sum variable. Print the 'sum' after the loop.

1.5 - Check if the entered character is 'a', 'e', 'i', 'o' or 'u'.

1.6 - Find out the number of digits in the number entered, then raise every digit by that number of digits in the number. If the sum of all exponentiated digits is the same as the entered number, it's an armstrong number.

1.7 - Initialize the factorial to 1. Run a loop from 1 to the entered number, and multiply factorial by the current iteration value (Factorial *= value).

1.8 - Print starts or numbers according to the pattern required using nested for loops and conditional statements.

Code:

1.1

```
principal = float(input("Enter principal amount: "))

rate_of_interest = float(input("Enter rate of interest: "))

time_period = float(input("Enter time period in years: "))



interest = (principal * rate_of_interest * time_period ) // 100
```

```
print(f"Simple interest: {interest}")
```

1.2

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

b = int(input("Enter b: "))



addition_result = a + b

subtraction_result = a - b

multiplication_result = a * b

division_result = a / b

floor_div_result = a // b

modulus_result = a % b

exponentiation_result = a ** b



print(f"Addition: {addition_result}")

print(f"Subtraction: {subtraction_result}")

print(f"Multiplication: {multiplication_result}")

print(f"Division: {division_result}")

print(f"Floor division result: {floor_div_result}")

print(f"Modulus: {modulus_result}")

print(f"Exponentiation: {exponentiation_result}")



x = float(input("\n\nEnter x: "))

y = float(input("Enter y: "))



equality_result = (x == y)

inequality_result = (x != y)

greater_than_result = (x > y)

less_than_result = (x < y)

greater_than_equal_result = (x >= y)

less_than_equal_result = (x <= y)
```

```
print(f"Equality: {equality_result}")

print(f"Inequality: {inequality_result}")

print(f"Greater Than: {greater_than_result}")

print(f"Less Than: {less_than_result}")

print(f"Greater Than or Equal To: {greater_than_equal_result}")

print(f"Less Than or Equal To: {less_than_equal_result}")
```

1.3

```
num = int(input("Enter number to check \nif it's odd or even: "))



if num % 2 ==0:

   print("Even")

else:

   print("Odd")
```

1.4

```
n = int(input("Enter n: "))

sum = 0



for num in range(1, n+1):

   sum+=num

   print(num, end=" ")



print(f"\nSum of first {n} numbers: {sum}")
```

1.5

```
char = input("Enter a character: ")
```

```
vowels = ['a', 'e', 'i', 'o', 'u']



if char.lower() in vowels:

   print(f"'{char}' is a vowel")

else:

   print(f"'{char}' is a consonant")
```

1.6

```
number_to_check = int(input("Enter number to check: "))

num_str = str(number_to_check)

num_digits = len(num_str)



armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)



if armstrong_sum == number_to_check:

   print(f"{number_to_check} is an Armstrong number!")

else:

   print(f"{number_to_check} is not an Armstrong number.")
```

1.7

```
number = int(input("Enter number to get its factorial: "))

fact = 1



for num in range(2, number+1):

   fact *= num



print(f"Factorial of {number} = {fact}")
```

1.8

i)

```
for i in range(5):
```

```
    for j in range(i+1):
        print("*", end="")
    print()
```

ii)

```
for i in range(1, 6):
    for j in range(1,i+1):
        print(i, end="")
    print()
```

iii)

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

Output: (screenshot)

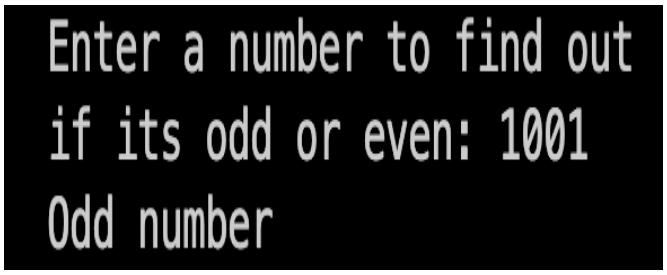
1.1

```
/usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter principal amount: 5000
Enter rate of interest: 10
Enter time period in years: 5
Simple interest: 2500.0
```

1.2

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter a: 5
Enter b: 10
Addition: 15.0
Subtraction: -5.0
Multiplication: 50.0
Division: 0.5
Floor division result: 0.0
Modulus: 5.0
Exponentiation: 9765625.0


Enter x: 50
Enter y: 5
Equality: False
Inequality: True
Greater Than: True
Less Than: False
Greater Than or Equal To: True
Less Than or Equal To: False
```

1.3



1.4

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter n: 50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50
Sum of first 50 numbers: 1275
```

1.5

```
Enter a character: a
'a' is a vowel
```

1.6

```
Enter number to check: 153
153 is an Armstrong number!
```

1.7

```
/usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter number to get its factorial: 5
Factorial of 5 = 120
```

1.8

```
/usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
*
**
***
****
*****

1
22
333
4444
55555

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

Test Case: Any two (screenshot)

1.1

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter principal amount: 1000
Enter rate of interest: 5
Enter time period in years: 2
Simple interest: 100.0
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter principal amount: 1200
Enter rate of interest: 10
Enter time period in years: 3
Simple interest: 360.0
```

1.2

```
Enter a: 10
Enter b: 5
Addition: 15.0
Subtraction: 5.0
Multiplication: 50.0
Division: 2.0
Floor division result: 2.0
Modulus: 0.0
Exponentiation: 100000.0


Enter x: 50
Enter y: 50
Equality: True
Inequality: False
Greater Than: False
Less Than: False
Greater Than or Equal To: True
Less Than or Equal To: True
```

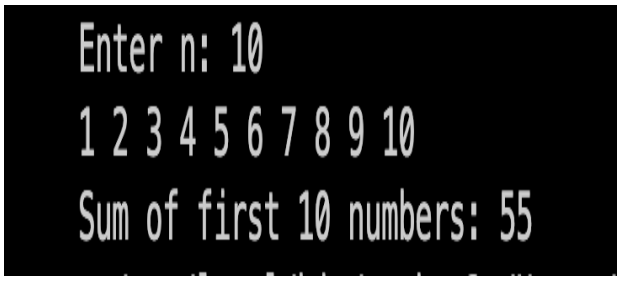
```
Enter a: 5
Enter b: 10
Addition: 15.0
Subtraction: -5.0
Multiplication: 50.0
Division: 0.5
Floor division result: 0.0
Modulus: 5.0
Exponentiation: 9765625.0


Enter x: 50
Enter y: 5
Equality: False
Inequality: True
Greater Than: True
Less Than: False
Greater Than or Equal To: True
Less Than or Equal To: False
```

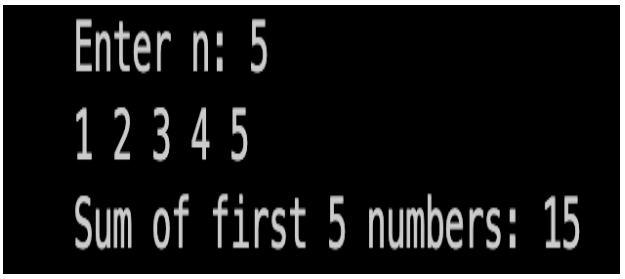
1.3

```
Enter a number to find out
if its odd or even: 1000
Even number
```

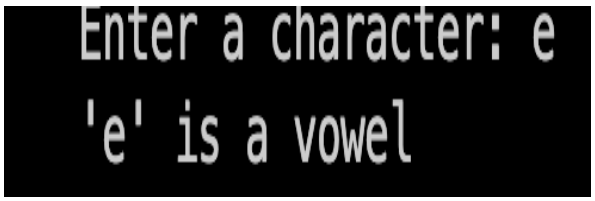
```
Enter a number to find out
if its odd or even: 99990
Even number
```

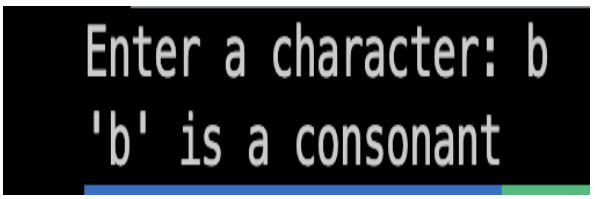
1.4



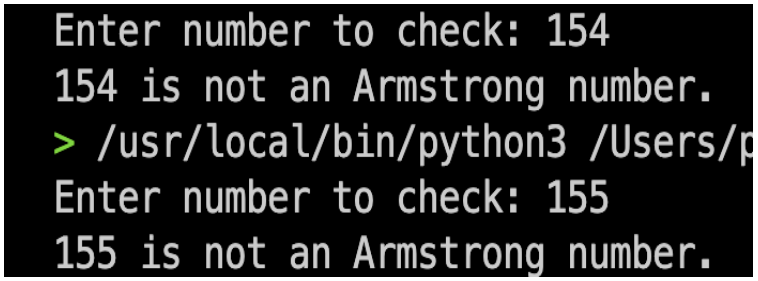


1.5

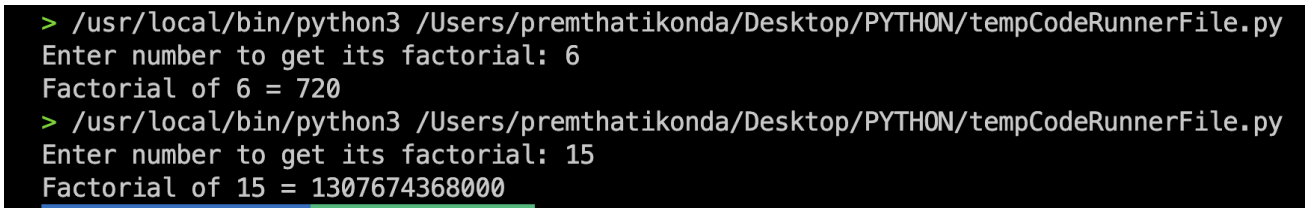




1.6



1.7



Conclusion:

Hence, using for loops and conditional statements, all the above programs have been executed.

Experiment No: 2

Title:

2.1 - Write a program that defines the list of countries in BRICS.

2.2 - Traverse a list (i) using reverse method (ii) using slicing

2.3 - Scan email and make a tuple of username and domain.

2.4 - Create a list of tuples with cubes of numbers in another list.

2.5 - Compare 2 dictionaries.

2.6 - Dictionary of cubes of odd numbers in a range

2.7 - List operations

Theory:

Code:

2.1

```
BRICS = ['brazil', 'russia', 'india', 'china', 'south africa']

for country in BRICS:

   print(country.capitalize())
```

2.2

```
numbers = [1,2,3,4,5,6,7,8,9,10]

numbers.sort(reverse=True)

print(numbers)
```

```
numbers = [1,2,3,4,5,6,7,8,9,10]

print(numbers[::-1])
```

2.3

```
email = input("Enter email: ")

tup = email.split('@')

print(tuple(tup))



print(f"Username: {tup[0]}")

print(f"Domain   : {tup[1]}")
```

2.4

```
num_list = [1,2,3,4,5,6,7,8,9,10]

cubes = [(num, num**3) for num in num_list]

print(cubes)
```

2.5

```
dict_a = {'a': 1, 'b': 2, 'c': 3}

dict_b = {'a': 1, 'b': 2, 'd': 3}

print(f"{dict_a}\n{dict_b}")

result = dict_a == dict_b

if result:

   print("They're same")

else:

   print("They're not the same")
```

2.6

```
limit = int(input("Enter range: "))

dictionary = {}

for value in range(1, limit+1):

   if value % 2 != 0:

       dictionary[value] = value**3
```

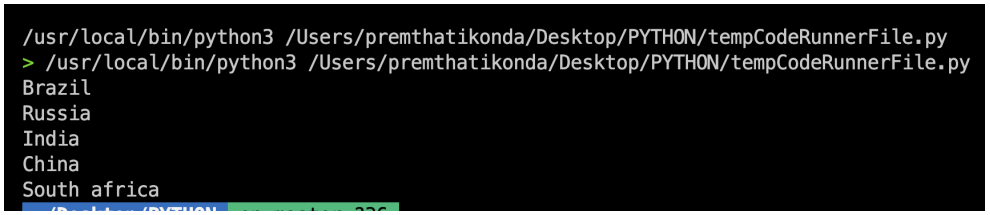
```
print(dictionary)
```

2.7

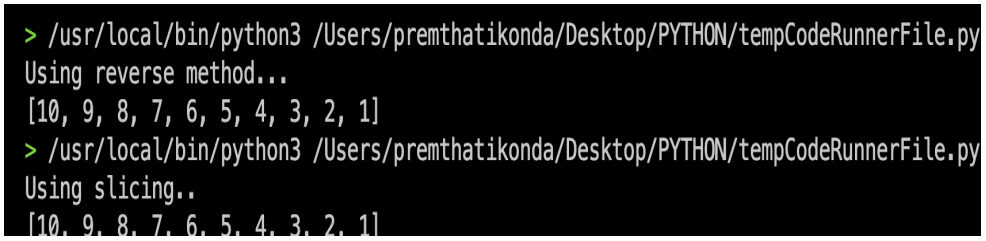
```
a= [10,20,30,40,50,60,70,80,90,100]

print(f"Complete list : {a}")

print(f"4th element   : {a[3]}")

print(f"0th-4th index : {a[:4]}")

print(f"-7th-3rd index: {a[-7:3]}")

a.append(11)

print(f"Appended list : {a}")

a.sort()

print(f"Sorted list: {a}")

print(f"Element popped: {a.pop()}\nUpdated list: {a}")

a.remove(11)

print(f"Removed 2nd element\nUpdated list: {a}")

a.insert(0, 0)

print(f"0 inserted at 0th index : {a}")

a.append(11)

print(f"11 appended into list: {a}")

print(f"Count of 11 in list: {a.count(11)}")

a.extend([100, 110])

print(f"Extended list: {a}")

a.reverse()

print(f"Reversed list: {a}")
```

Output: (screenshot)

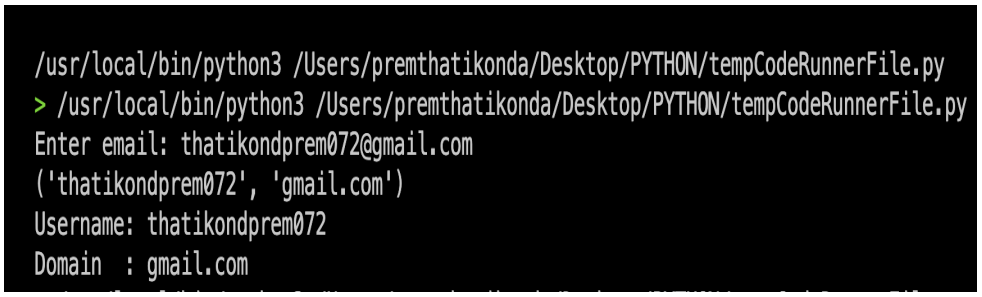
2.1



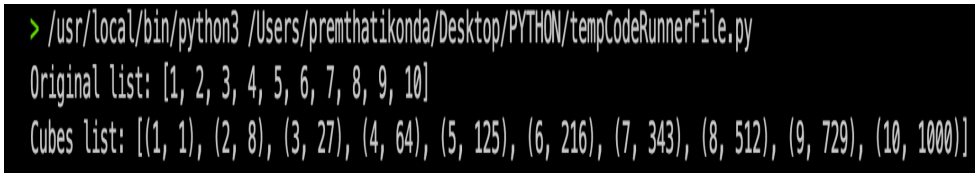
2.2



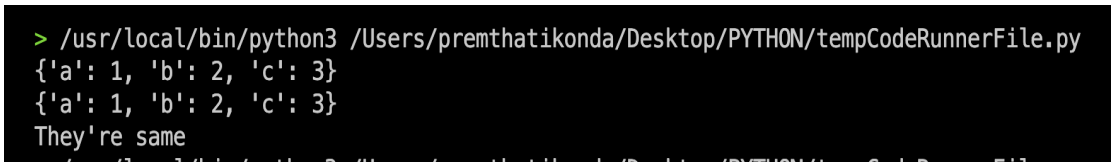
2.3



2.4



2.5



2.6

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter range: 20
{1: 1, 3: 27, 5: 125, 7: 343, 9: 729, 11: 1331, 13: 2197, 15: 3375, 17: 4913, 19: 6859}
```

2.7

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Complete list : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
4th element   : 40
0th-4th index : [10, 20, 30, 40]
-7th-3rd index: []
Appended list : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 11]
Sorted list: [10, 11, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Element popped: 100
Updated list: [10, 11, 20, 30, 40, 50, 60, 70, 80, 90]
Removed 2nd element
Updated list: [10, 20, 30, 40, 50, 60, 70, 80, 90]
0 inserted at 0th index : [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
11 appended into list: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 11]
Count of 11 in list: 1
Extended list: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 11, 100, 110]
Reversed list: [110, 100, 11, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0]
```

Test Case: Any two (screenshot)

2.3

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter email: 2023.tprem@isu.ac.in
('2023.tprem', 'isu.ac.in')
Username: 2023.tprem
Domain  : isu.ac.in
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter email: 373premthatikonda@gmail.com
('373premthatikonda', 'gmail.com')
Username: 373premthatikonda
Domain  : gmail.com
```

2.4

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Cubes list: [(1, 1), (2, 8), (3, 27), (4, 64), (5, 125), (6, 216), (7, 343), (8, 512), (9, 729), (10, 1000)]
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Original list: [11, 12, 3, 14, 15]
Cubes list: [(11, 1331), (12, 1728), (3, 27), (14, 2744), (15, 3375)]
```

2.5

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
{'a': 1, 'b': 2, 'c': 3}
{'a': 1, 'b': 2, 'c': 4}
They're not the same
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
{'a': 1, 'b': 2, 'c': 3}
{'a': 1, 'b': 2, 'd': 3}
They're not the same
```

2.6

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter range: 50
{1: 1, 3: 27, 5: 125, 7: 343, 9: 729, 11: 1331, 13: 2197, 15: 3375, 17: 4913, 19: 6859, 21: 9261, 23: 12167, 25: 15625, 27: 19683, 29:
 24389, 31: 29791, 33: 35937, 35: 42875, 37: 50653, 39: 59319, 41: 68921, 43: 79507, 45: 91125, 47: 103823, 49: 117649}
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter range: 10
{1: 1, 3: 27, 5: 125, 7: 343, 9: 729}
```

Conclusion:

Hence, using loops and other operations, the programs have been solved.

Experiment No: 3

Title:

3.1 - Extend a list

3.2 - Add 2 matrices

3.3 - Takes a list, returns a new list with distinct elements from the 1st list.

3.4 - Check whether a number is perfect or not

3.5 - Count the number of uppercase and lowercase letters in a string.

Theory:

Code:

3.1

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

b = [60,70,80,90,100]



print(f"Using '+' operator: {a+b}")

a.append(b)

print(f"Using append method: {a}")

a.remove(b)

a.extend(b)

print(f"Using extend method: {a}")
```

3.2

```
a = [

   [0,0,0],

   [0,0,0],

   [0,0,0]

]
```

```python
b = [
   [0,0,0],
   [0,0,0],
   [0,0,0]
]

c = [
   [0,0,0],
   [0,0,0],
   [0,0,0]
]

for i in range(3):
   for j in range(3):
       a[i][j] = int(input(f"Enter a[{i}][{j}]: "))
   print()

for i in range(3):
   for j in range(3):
       b[i][j] = int(input(f"Enter b[{i}][{j}]: "))
   print()



for i in range(3):
   for j in range(3):
       c[i][j] = a[i][j]+b[i][j]

print(f"{c[0]}\n{c[1]}\n{c[2]}")
```

3.3

```
def returnList(list1):
    if len(list1) == 0:
        return None
    else:
        return list(set(list1))


size = int(input("Enter size of the list: "))
list1 = []


for i in range(size):
    num = int(input(f"Enter number {i}: "))
    list1.append(num)


print(f"Distinct elements in the list: {returnList(list1)}")
```

3.4

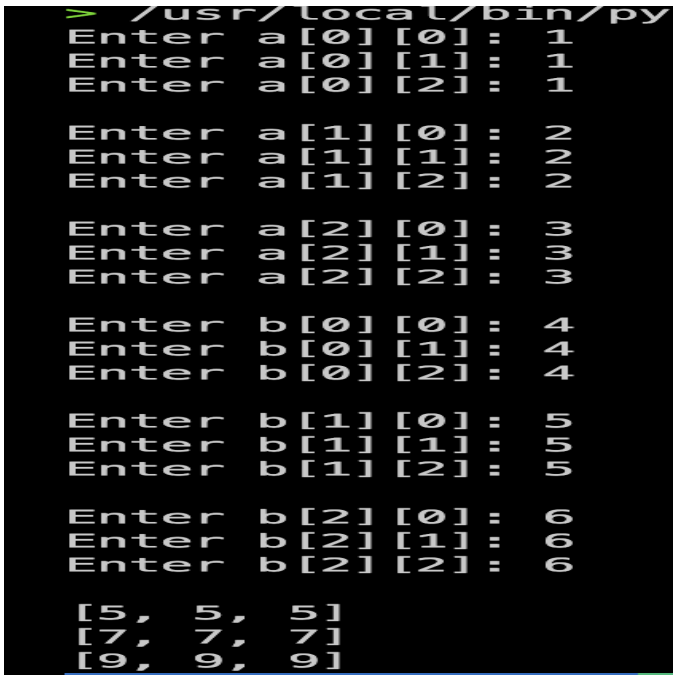
```
def is_perfect_number(num):
    divisors_sum = sum(i for i in range(1, num // 2 + 1) if num % i == 0)
    return divisors_sum == num




num = int(input("Enter a number: "))
if is_perfect_number(num):
    print(f"{num} is a perfect number.")
else:
    print(f"{num} is not a perfect number.")
```

Output(screenshot):

3.1

```
/usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Using '+' operator: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Using append method: [10, 20, 30, 40, 50, [60, 70, 80, 90, 100]]
Using extend method: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

3.2



3.3

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter size of the list: 5
Enter number 0: 0
Enter number 1: 1
Enter number 2: 0
Enter number 3: 2
Enter number 4: 0
Distinct elements in the list: [0, 1, 2]
```

3.4

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter a number: 496
496 is a perfect number.
```

3.5

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter string: WHAT IS UP man
Number of upper-case letters: 8
Number of lower-case letters: 3
```

Test Cases (any two screenshot) :

3.2

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/P
Enter a[0][0]: 1
Enter a[0][1]: 1
Enter a[0][2]: 1

Enter a[1][0]: 2
Enter a[1][1]: 2
Enter a[1][2]: 2

Enter a[2][0]: 3
Enter a[2][1]: 3
Enter a[2][2]: 3

Enter b[0][0]: 4
Enter b[0][1]: 4
Enter b[0][2]: 4

Enter b[1][0]: 5
Enter b[1][1]: 5
Enter b[1][2]: 5

Enter b[2][0]: 6
Enter b[2][1]: 6
Enter b[2][2]: 6

[5, 5, 5]
[7, 7, 7]
[9, 9, 9]
```

```
Enter a[0][0]: 1
Enter a[0][1]: 2
Enter a[0][2]: 3

Enter a[1][0]: 4
Enter a[1][1]: 5
Enter a[1][2]: 6

Enter a[2][0]: 7
Enter a[2][1]: 8
Enter a[2][2]: 9

Enter b[0][0]: 1
Enter b[0][1]: 2
Enter b[0][2]: 3

Enter b[1][0]: 4
Enter b[1][1]: 5
Enter b[1][2]: 6

Enter b[2][0]: 7
Enter b[2][1]: 8
Enter b[2][2]: 9

[2, 4, 6]
[8, 10, 12]
[14, 16, 18]
```

3.3

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter size of the list: 10
Enter number 0: 1
Enter number 1: 1
Enter number 2: 1
Enter number 3: 1
Enter number 4: 3
Enter number 5: 4
Enter number 6: 5
Enter number 7: 6
Enter number 8: 2
Enter number 9: 4
Distinct elements in the list: [1, 2, 3, 4, 5, 6]
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter size of the list: 5
Enter number 0: 1
Enter number 1: 2
Enter number 2: 3
Enter number 3: 4
Enter number 4: 1
Distinct elements in the list: [1, 2, 3, 4]
```

3.4

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter a number: 28
28 is a perfect number.
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter a number: 8
8 is not a perfect number.
```

3.5

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter string: Today is my Best Day
Number of upper-case letters: 3
Number of lower-case letters: 13
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter string: Hello from the other Side
Number of upper-case letters: 2
Number of lower-case letters: 19
```

Conclusion:

Experiment No: 4

Title:

4.1 - Create Employee Class & add methods to get employee details & print.

4.2 - Use *args and **kwargs

4.3 - Admit students in different departments and count number of students

4.4 - Store

4.5 - Addition of 2 numbers (using inheritance)

4.6 - LU and ITM

4.7 - Implement multiple inheritance

4.8 - Library Catalog

Theory:

Code:

4.1

```
class Employee:

    def __init__(self, name, emp_id, department):

        self.name = name

        self.emp_id = emp_id

        self.department = department



    def get_employee_details(self):

        return f"\nEmployee Details:\nName: {self.name}\nEmployee ID:
{self.emp_id}\nDepartment: {self.department}"



    def print_employee_details(self):

        print(self.get_employee_details())



name = input("Enter name: ")

emp_id = int(input("Enter employee id: "))

department = input("Enter department: ")
```

```
employee1 = Employee(name, emp_id, department)

employee1.print_employee_details()
```

4.2

```
def get_user_details_args(*args):
    name = args[0] if len(args) > 0 else None
    email = args[1] if len(args) > 1 else None
    age = args[2] if len(args) > 2 else None

    return f"User Details (Using *args):\nName: {name}\nEmail: {email}\nAge: {age}"

def get_user_details_kwargs(**kwargs):
    name = kwargs.get('name', None)
    email = kwargs.get('email', None)
    age = kwargs.get('age', None)

    return f"User Details (Using **kwargs):\nName: {name}\nEmail: {email}\nAge: {age}"

name_input = input("Enter your name: ")
email_input = input("Enter your email: ")
age_input = input("Enter your age: ")

details_args = get_user_details_args(name_input, email_input, age_input)
print(details_args)

details_kwargs = get_user_details_kwargs(name=name_input, email=email_input,
age=age_input)
print(details_kwargs)
```

4.3

```
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 get_student_count(self):
        return len(self.students)

def get_user_input():
    name = input("Enter student name (type 'done' to finish): ").capitalize()

    if name.lower() == 'done':
        return None, None

    department = input("Enter department (PGDM/BTech): ")
    return name, department
def main():
  pgdm_department = Department("PGDM Department")
  btech_department = Department("BTech Department")

  while True:
```

```
    name, department = get_user_input()


    if name is None and department is None:
        break


    new_student = Student(name, department)


    if department.upper() == "PGDM":
        pgdm_department.admit_student(new_student)
    elif department.upper() == "BTECH":
        btech_department.admit_student(new_student)
    else:
        print(f"Invalid department entered for {name}. Please enter PGDM or BTech.")


  print(f"Student count in {pgdm_department.name}: {pgdm_department.get_student_count()}")
  print(f"Student count in {btech_department.name}:
{btech_department.get_student_count()}")


if __name__=="__main__":
   main()
```

4.4

```
class Store:
    def __init__(self):
        # Dictionary to store product codes and their prices
        self.products = {
            '001': {'name': 'Books', 'price': 8.50},
            '002': {'name': 'Pencils', 'price': 3.50},
            '003': {'name': 'Pens', 'price': 4.50},
            # Add more products as needed
        }
```

```
    def display_menu(self):
        print("Product Code    Product Name    Price")
        print("-----------------------------------")
        for code, details in self.products.items():
            print(f"{code}              {details['name']}         ${details['price']:.2f}")
        print("-----------------------------------")

    def generate_bill(self, items):
        total_amount = 0.0

        print("\nYour Bill:")
        print("Product Code    Product Name    Quantity    Price")
        print("-------------------------------------------------")

        for code, quantity in items.items():
            if code in self.products:
                product_name = self.products[code]['name']
                price = self.products[code]['price']
                amount = price * quantity
                total_amount += amount

                print(f"{code}              {product_name}         {quantity}
${amount:.2f}")

        print("-------------------------------------------------")
        print(f"Total Amount: ${total_amount:.2f}")

def get_quantity(product_code):
    while True:
```

```
        try:
            quantity = int(input(f"Enter the quantity for {product_code}: "))
            if quantity >= 0:
                return quantity
            else:
                print("Please enter a non-negative quantity.")
        except ValueError:
            print("Invalid input. Please enter a valid quantity.")


# Example usage:
store = Store()


# Display the menu
store.display_menu()


# Get user input for product quantities
user_items = {}
while True:
    code = input("Enter product code: ")
    if code == "q":
        break
    quantity = get_quantity(code)
    user_items[code] = quantity


# Generate and display the bill
store.generate_bill(user_items)
```

Output:

4.1

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter name: Prem
Enter employee id: 1234
Enter department: IT

Employee Details:
Name: Prem
Employee ID: 1234
Department: IT
```

4.2

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter your name: Prem
Enter your email: thati@gmail.com
Enter your age: 18
User Details (Using *args):
Name: Prem
Email: thati@gmail.com
Age: 18
User Details (Using **kwargs):
Name: Prem
Email: thati@gmail.com
Age: 18
```

4.3

```
Enter student name (type 'done' to finish): prem
Enter department (PGDM/BTech): btech
Enter student name (type 'done' to finish): done
Student count in PGDM Department: 0
Student count in BTech Department: 1
```

Test Cases (any two):

4.1

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter name: Sumit
Enter employee id: 5678
Enter department: CS

Employee Details:
Name: Sumit
Employee ID: 5678
Department: CS
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter name: XYZ
Enter employee id: 0987
Enter department: PGDM

Employee Details:
Name: XYZ
Employee ID: 987
Department: PGDM
```

4.2

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter your name: ABC
Enter your email: abc@gmail.com
Enter your age: 18
User Details (Using *args):
Name: ABC
Email: abc@gmail.com
Age: 18
User Details (Using **kwargs):
Name: ABC
Email: abc@gmail.com
Age: 18
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter your name: dean
Enter your email: dean@gmail.com
Enter your age: 20
User Details (Using *args):
Name: dean
Email: dean@gmail.com
Age: 20
User Details (Using **kwargs):
Name: dean
Email: dean@gmail.com
Age: 20
```

4.3

```
> /usr/local/bin/python3 /Users/premthatikonda/Desktop/PYTHON/tempCodeRunnerFile.py
Enter student name (type 'done' to finish): ved
Enter department (PGDM/BTech): pgdm
Enter student name (type 'done' to finish): vishaka
Enter department (PGDM/BTech): pgdm
Enter student name (type 'done' to finish): prem
Enter department (PGDM/BTech): btech
Enter student name (type 'done' to finish): done
Student count in PGDM Department: 2
Student count in BTech Department: 1
```

Conclusion:

Experiment 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:

The Python code defines a Store class to simulate a store's inventory system. It allows adding products, displaying the menu, and generating a bill based on user input for product codes and quantities.

CODE:

```
# 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.


class Store:

   def __init__(self):

       self.products = {}


   def add_product(self, code, name, price):

       self.products[code] = {'name': name, 'price': price}


   def display_menu(self):

       print("Menu:")

       print("Code\tName\tPrice")

       for code, product in self.products.items():

           print(f"{code}\t{product['name']}\t₹{product['price']}")


   def generate_bill(self, order):
```

```
 total_amount = 0

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

       print("Code\tName\tPrice\tQuantity\tTotal")

       for code, quantity in order.items():

           product = self.products[code]

           item_total = quantity * product['price']

           total_amount += item_total


print(f"{code}\t{product['name']}\t₹{product['price']}\t{quantity}\t\t₹{item_total}")



       print("\nTotal Amount: ₹{:.2f}\n".format(total_amount))




def main():

   store = Store()


   store.add_product('001', 'Bread', 25.00)

   store.add_product('002', 'Chips', 15.00)

   store.add_product('003', 'KitKat', 10.00)

   store.add_product('004', 'Coke', 20.00)

   store.add_product('005', 'Biscuit', 12.00)

   store.add_product('006', 'Butter', 35.00)

   store.add_product('007', 'Rice', 30.00)

   store.add_product('008', 'Lentils', 35.00)

   store.add_product('009', 'Suji', 40.00)

   store.add_product('010', 'Spice', 20.00)


   store.display_menu()


   order = {}
```

```
 while True:

       code = input("Enter the product code (or 'done' to finish): ")

       if code.lower() == 'done':

           break

       elif code in store.products:

           quantity = int(input(f"Enter the quantity for {store.products[code]['name']}:
"))

           order[code] = quantity

       else:

           print("Invalid product code. Please enter a valid code.")


   store.generate_bill(order)



if __name__ == "__main__":

   main()
```

OUTPUT: