Practical 3 - Write simple Python program using operators: a) Arithmetic Operators b) Logical Operators c) Bitwise Operators

1. Write a program to convert U.S. dollars to Indian rupees.

```
print("US dollar to Indian Rupees $1 = ₹82.77")
u = int(input("Enter U.S. Dollar Amount = "))
i = u*82.77
print("Indian Rupees = ",i)
```

```
US dollar to Indian Rupees $1 = ₹82.77
Enter U.S. Dollar Amount = 500
Indian Rupees =  41385.0
```

2. Write a program to convert bits to Megabytes, Gigabytes and Terabytes

```
print("Conversion of bites to MB, GB, TB")
bits = int(input("Enter no of bits = "))
byte = bits/8
kb = byte/1024
mb = kb/1024
gb = mb/1024
tb = gb/1024
print("Megabytes = ",mb)
print("Gigabytes = ",gb)
print("Terabytes = ",tb)
```

```
Conversion of bites to MB, GB, TB
Enter no of bits = 12
Megabytes = 1.430511474609375e-06
Gigabytes = 1.3969838619232178e-09
Terabytes = 1.3642420526593924e-12
```

3. Write a program to find the square root of a number

```
import math
number = float(input("Enter a number: "))
```

```
sqrt = math.sqrt(number)
print("The square root of", number, "is", sqrt)
```

```
Enter a number: 4
The square root of 4.0 is 2.0
```

4. Write a program to find the area of Rectangle

```
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

area = length * width

print("The area of the rectangle is:", area)
```

```
Enter the length of the rectangle: 5
Enter the width of the rectangle: 5
The area of the rectangle is: 25.0
```

5. Write a program to calculate area and perimeter of the square

```
side = float(input("Enter the length of the side of the square: "))

area = side * side

perimeter = 4 * side

print("The area of the square is:", area)
print("The perimeter of the square is:", perimeter)
```

```
Enter the length of the side of the square: 4
The area of the square is: 16.0
The perimeter of the square is: 16.0
```

6. Write a program to calculate surface volume and area of a cylinder.

```
pi = 22/7
height = float(input('Height of cylinder: '))
```

```
radius = float(input('Radius of cylinder: '))

volume = pi * radius * radius * height
surface_area = ((2*pi*radius) * height) + ((pi*radius**2)*2)

print("Volume is: {:.2f}".format(volume))
print("Surface Area is: {:.2f}".format(surface_area))
```

```
Height of cylinder: 6
Radius of cylinder: 6
Volume is: 678.86
Surface Area is: 452.57
```

7. Write a program to swap the value of two variables

```
a = input("Enter the value of a: ")
b = input("Enter the value of b: ")

print("Before swapping, a =", a, "and b =", b)

temp = a
a = b
b = temp

print("After swapping, a =", a, "and b =", b)
```

```
Enter the value of a: 21
Enter the value of b: 45
Before swapping, a = 21 and b = 45
After swapping, a = 45 and b = 21
```

Practical 4 - Write simple Python program to demonstrate use of conditional statements: a) 'if' statement b) 'if … else' statement c) Nested 'if' statement

1. Write a program to check whether a number is even or odd

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

if number % 2 == 0:
   print(number, "is even")
else:
   print(number, "is odd")
```

```
Enter a number: 2
2 is even
```

2. Write a program to find out absolute value of an input number

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

absolute_value = abs(number)

print("The absolute value of", number, "is", absolute_value)
```

```
Enter a number: -8
The absolute value of -8 is 8
```

3. Write a program to check the largest number among the three numbers

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
num3 = int(input("Enter third number: "))

if num1 >= num2 and num1 >= num3:
   print(num1, "is the largest number.")
```

```
elif num2 >= num1 and num2 >= num3:
   print(num2, "is the largest number.")
else:
   print(num3, "is the largest number.")
```

```
Enter first number: 234
Enter second number: 54
Enter third number: 66
234 is the largest number.
```

4. Write a program to check if the input year is a leap year of not

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

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

```
Enter a year: 2004
2004 is a leap year.
```

5. Write a program to check if a Number is Positive, Negative or Zero

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

if number > 0:
   print(number, "is positive.")
elif number < 0:
   print(number, "is negative.")
else:
   print(number, "is zero.")
```

```
Enter a number: 59
59.0 is positive.
```

6. Write a program that takes the marks of 5 subjects and displays the grade.

```
subject1 = float(input("Enter marks for subject 1: "))
subject2 = float(input("Enter marks for subject 2: "))
subject3 = float(input("Enter marks for subject 3: "))
subject4 = float(input("Enter marks for subject 4: "))
subject5 = float(input("Enter marks for subject 5: "))

total_marks = subject1 + subject2 + subject3 + subject4 + subject5
average_marks = total_marks / 5

if average_marks >= 90:
    grade = "A+"
elif average_marks >= 80:
    grade = "A"
elif average_marks >= 70:
    grade = "B"
elif average_marks >= 60:
    grade = "C"
elif average_marks >= 50:
    grade = "D"
else:
    grade = "F"

print("Total marks:", total_marks)
print("Average marks:", average_marks)
print("Grade:", grade)
```

```
Enter marks for subject 1: 100
Enter marks for subject 2: 100
Enter marks for subject 3: 70
Enter marks for subject 4: 80
Enter marks for subject 5: 90
Total marks: 440.0
Average marks: 88.0
Grade: A
```

5 Write Python program to demonstrate use of looping statements: a) 'while' loop b) 'for' loop c) Nested loops

1. Print the following patterns using loop:

a.

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

b.

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

c.

```
1010101
 10101
  101
   1
```

a.

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

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

b.

```
n = 3  # number of rows in the diamond (excluding the middle row)
for i in range(1, n+1):
    print(" "*(n-i) + "*"*(2*i-1))  # print the top half of the diamond
```

```
for i in range(n-1, 0, -1):
    print(" "*(n-i) + "*"*(2*i-1))  # print the bottom half of the diamond
```

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

c.

```
n = 4  # number of rows in the pattern
for i in range(n):
    for j in range(i):
        print(" ", end="")
    print("1" + "01" * (n-i-1))
```

```
1010101
 10101
  101
   1
```

2. Write a Python program to print all even numbers between 1 to 100 using while loop.

```
i = 1
while i <= 100:
    if i % 2 == 0:
        print(i)
    i += 1
```

```
2
4
6
8
.
.
.
98
100
```

3. Write a Python program to find the sum of first 10 natural numbers using for loop.

```
sum = 0
for i in range(1, 11):
    sum += i
print("The sum of first 10 natural numbers is:", sum)
```

```
The sum of first 10 natural numbers is: 55
```

4. Write a Python program to print Fibonacci series.

```
n = int(input("Enter the number of terms: "))
a, b = 0, 1
count = 0
if n <= 0:
    print("Invalid input")
elif n == 1:
    print(a)
else:
    print("Fibonacci series:")
    while count < n:
        print(a, end=" ")
        c = a + b
        a = b
        b = c
        count += 1
```

```
Enter the number of terms: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
```

5. Write a Python program to calculate factorial of a number

```
num = int(input("Enter a number: "))
factorial = 1

if num < 0:
    print("Factorial is not defined for negative numbers")
elif num == 0:
```

```
    print("Factorial of 0 is 1")
else:
    for i in range(1, num+1):
        factorial *= i
    print("Factorial of", num, "is", factorial)
```

```
Enter a number: 5
Factorial of 5 is 120
```

6. Write a Python Program to Reverse a Given Number

```
num = int(input("Enter a number: "))
original_num = num
rev = 0

while num > 0:
    rem = num % 10
    rev = rev * 10 + rem
    num //= 10

print("The reverse of", original_num, "is", rev)
```

```
Enter a number: 25
The reverse of 25 is 52
```

7. Write a Python program takes in a number and finds the sum of digits in a number.

```
num = int(input("Enter a number: "))
sum = 0

while num > 0:
    digit = num % 10
    sum += digit
    num //= 10

print("The sum of digits in the number is:", sum)
```

```
Enter a number: 15
The sum of digits in the number is: 6
```

8. Write a Python program that takes a number and checks whether it is a palindrome or not

```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
if(temp==rev):
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
```

```
Enter number: 26
The number isn't a palindrome!

Enter number:121
The number is a palindrome!
```

Practical 6 - Write Python program to perform following operations on Lists: a) Create list b) Access list c) Update list (Add item, Remove item) d) Delete list

1. Write a Python program to sum all the items in a list.

```
my_list = [1, 2, 3, 4, 5]
list_sum = sum(my_list)
print("The sum of the items in the list is:", list_sum)
```

```
The sum of the items in the list is: 15
```

2. Write a Python program to multiplies all the items in a list.

```
my_list = [1, 2, 3, 4, 5]
product = 1
for item in my_list:
    product *= item
print("The product of the items in the list is:", product)
```

```
The product of the items in the list is: 120
```

3. Write a Python program to get the largest number from a list.

```
my_list = [1, 2, 3, 4, 5]
largest = max(my_list)
print("The largest number in the list is:", largest)
```

```
The largest number in the list is: 5
```

4. Write a Python program to get the smallest number from a list.

```
my_list = [1, 2, 3, 4, 5]
smallest = min(my_list)
print("The smallest number in the list is:", smallest)
```

```
The smallest number in the list is: 1
```

5. Write a Python program to reverse a list.

```
list = [1, 2, 3]
print("Existing list",list)
# reversing
list.reverse()
print("After reversing all elements")
print(list)
```

```
Existing list [1, 2, 3]
After reversing all elements
[3, 2, 1]
```

6. Write a Python program to find common items from two lists.

```
list1 = [1, 2, 3, 4, 5]
list2 = [3, 4, 5, 6, 7]
common_items = set(list1) & set(list2)
print("The common items in the two lists are:", list(common_items))
```

```
The common items in the two lists are: [3, 4, 5]
```

7. Write a Python program to select the even items of a list.

```
my_list = [1, 2, 3, 4, 5]
even_items = [item for item in my_list if item % 2 == 0]
print("The even items in the list are:", even_items)
```

```
The even items in the list are: [2, 4]
```

Practical 7 - Write Python program to perform following operations on Tuples: a) Create Tuple b) Access Tuple c) Update Tuple d) Delete Tuple

1. Create a tuple and find the minimum and maximum number from it.

```
my_tuple = (1, 5, 2, 8, 3)
min_num = min(my_tuple)
max_num = max(my_tuple)
print("Minimum number in the tuple:", min_num)
print("Maximum number in the tuple:", max_num)
```

```
Minimum number in the tuple: 1
Maximum number in the tuple: 8
```

2. Write a Python program to find the repeated items of a tuple.

```
t = (2,34,45,6,7,2,4,5,78,34,2)
print(t)
count = t.count(2)
print(count)
```

```
(2, 34, 45, 6, 7, 2, 4, 5, 78, 34, 2)
3
```

3. Print the number in words for Example: 1234 => One Two Three Four

```
num_dict = {"0": "Zero", "1": "One", "2": "Two", "3": "Three", "4": "Four",
            "5": "Five", "6": "Six", "7": "Seven", "8": "Eight", "9": "Nine"}

num = input("Enter a number: ")
num_str = str(num)
words = [num_dict[digit] for digit in num_str]
print("Number in words:", " ".join(words))
```

```
Enter a number: 1234
Number in words: One Two Three Four
```

Practical 8 - Write Python program to perform following operations on Tuples: a) Create Set b) Access Set elements c) Update Set d) Delete Set

1. Write a Python program to create a set, add member(s) in a set and remove one item from set.

```
my_set = set()  # create an empty set

my_set.add(1)  # add an item to the set
my_set.add(2)
my_set.add(3)

print(my_set)  # prints {1, 2, 3}

my_set.remove(2)  # remove an item from the set
print(my_set)  # prints {1, 3}
```

```
{1, 2, 3}
{1, 3}
```

2. Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.

```
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

print("Set 1 = ",set1)
print("Set 2 = ",set2)

# intersection of sets
print("Intersection of sets = ",set1.intersection(set2))  # prints {3, 4}

# union of sets
print("Union of sets = ",set1.union(set2))  # prints {1, 2, 3, 4, 5, 6}

# set difference
print("Set difference = ", set1.difference(set2))  # prints {1, 2}
```

```
# symmetric difference
print("Symmetric difference = ", set1.symmetric_difference(set2))  # prints {1, 2, 5, 6}

# clear a set
set1.clear()
print(set1)  # prints set()
```

```
Set 1 =  {1, 2, 3, 4}
Set 2 =  {3, 4, 5, 6}
Intersection of sets =  {3, 4}
Union of sets =  {1, 2, 3, 4, 5, 6}
Set difference =  {1, 2}
Symmetric difference =  {1, 2, 5, 6}
set()
```

3. Write a Python program to find maximum and the minimum value in a set.

```
my_set = {1, 2, 3, 4, 5}

print("Set = ",my_set)
print("Maximum value in set = ",max(my_set))  # prints 5
print("Minimum value in set = ",min(my_set))  # prints 1
```

```
Set =  {1, 2, 3, 4, 5}
Maximum value in set =  5
Minimum value in set =  1
```

4. Write a Python program to find the length of a set.

```
my_set = {1, 2, 3, 4, 5}
print("Set = ",my_set)
print("Length of set = ",len(my_set))  # prints 5
```

```
Set =  {1, 2, 3, 4, 5}
Length of set =  5
```

Practical 9 - Write Python program to perform following operations on Dictionaries: a) Create Dictionary b) Access Dictionary elements c) Update Dictionary d) Delete Set e) Looping through Dictionary

1. Write a Python script to sort (ascending and descending) a dictionary by value.

```
def sort_dict_by_value(d, reverse = False):
  return dict(sorted(d.items(), key = lambda x: x[1], reverse = reverse))
print("Original dictionary elements:")
colors = {'Red': 1, 'Green': 3, 'Black': 5, 'White': 2, 'Pink': 4}
print(colors)
print("\nSort (ascending) the dictionary elements by value:")
print(sort_dict_by_value(colors))
print("\nSort (descending) the dictionary elements by value:")
print(sort_dict_by_value(colors, True))
```

```
Original dictionary elements:
{'Red': 1, 'Green': 3, 'Black': 5, 'White': 2, 'Pink': 4}

Sort (ascending) the dictionary elements by value:
{'Red': 1, 'White': 2, 'Green': 3, 'Pink': 4, 'Black': 5}

Sort (descending) the dictionary elements by value:
{'Black': 5, 'Pink': 4, 'Green': 3, 'White': 2, 'Red': 1}
```

2. Write a Python script to concatenate following dictionaries to create a new one.

a. Sample Dictionary:

b. dic1 = {1:10, 2:20}

c. dic2 = {3:30, 4:40}

d. dic3 = {5:50,6:60}

```
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}
dic4 = {}
# Using a for loop to iterate over the dictionaries and update dic4
```

```
for d in (dic1, dic2, dic3):
   dic4.update(d)
# Printing the concatenated dictionary
print(dic4)
```

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

3. Write a Python program to combine two dictionary adding values for common keys.

a. d1 = {'a': 100, 'b': 200, 'c':300}

b. d2 = {'a': 300, 'b': 200, 'd':400}

a.

```
d1 = {'a': 100, 'b': 200, 'c': 300}
d2 = {'a': 300, 'b': 200, 'd': 400}

for key in d1:
   if key in d2:
      d2[key] += d1[key]
   else:
      d2[key] = d1[key]

print('Combined dictionary:', d2)
```

```
Combined dictionary: {'a': 400, 'b': 400, 'd': 400, 'c': 300}
```

b.

```
from collections import Counter
dict1 = {'a': 100, 'b': 200, 'c':300}
dict2 = {'a': 300, 'b': 200, 'd':400}
new_dict = Counter(dict1) + Counter(dict2)
print("The new  dict is:", new_dict)
```

```
The new  dict is: Counter({'a': 400, 'b': 400, 'd': 400, 'c': 300})
```

4. Write a Python program to print all unique values in a dictionary.

a. Sample Data: [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

```
L = [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"},{"VIII":"S007"}]
print("Original List: ",L)
u_value = set( val for dic in L for val in dic.values())
print("Unique Values: ",u_value)
```

```
Original List:  [{'V': 'S001'}, {'V': 'S002'}, {'VI': 'S001'}, {'VI': 'S005'}, {'VII': 'S005'}, {'V': 'S009'}, {'VIII': 'S007'}]
Unique Values:  {'S002', 'S007', 'S009', 'S001', 'S005'}
```

5. Write a Python program to find the highest 3 values in a dictionary.

```
from heapq import nlargest

# Initial Dictionary
my_dict = {'A': 67, 'B': 23, 'C': 45, 'D': 56, 'E': 12, 'F': 69}

print("Initial Dictionary:")
print(my_dict, "\n")

ThreeHighest = nlargest(3, my_dict, key = my_dict.get)

print("Dictionary with 3 highest values:")
print("Keys: Values")

for val in ThreeHighest:
        print(val, ":", my_dict.get(val))
```

```
Initial Dictionary:
{'A': 67, 'B': 23, 'C': 45, 'D': 56, 'E': 12, 'F': 69}

Dictionary with 3 highest values:
Keys: Values
F : 69
A : 67
D : 56
```

Practical 10 - a) Write Python program to demonstrate math built- in functions (Any 2 programs) b) Write Python program to demonstrate string built – in functions (Any 2 programs)

1. Write a Python function that accepts a string and calculate the number of upper-case letters and lower-case letters.

```
x=input("Enter the string:- ")
def cal(x):
  u=0
  l=0
  for i in x:
     if i.islower():
      l+=1

     if i.isupper():
      u+=1

  print("LowerCase letter in the String",l)
  print("UpperCase letter in the String",u)
cal(x)
```

```
Enter the string:- Ritik
LowerCase letter in the String 4
UpperCase letter in the String 1
```

2. Write a Python program to generate a random float where the value is between 5 and 50 using Python math module.

```
import random

random_float = random.uniform(5, 50)
rounded_float = round(random_float, 2)

print(f"Random float between 5 and 50: {rounded_float}")
```

```
Random float between 5 and 50: 17.65
```

Practical 11 - Develop user defined Python function for given problem: a) Function with minimum 2 arguments b) Function returning values

1. Write a Python function that takes a number as a parameter and check the number is prime or not.

```
def is_prime(num):
    if num == 1:
        return False
    elif num > 1:
        # check for factors
        for i in range(2, num):
            if (num % i) == 0:
                return False
        else:
            return True
    else:
        return False

num = int(input("Enter a number: "))
if is_prime(num):
    print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

```
Enter a number: 5
5 is a prime number
```

2. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.

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

n = int(input("Input a number to compute the factiorial : "))
```

```
print(factorial(n))
```

```
Input a number to compute the factiorial : 3
6
```

3. Write a Python function that accepts a string and calculate the number of upper-case letters and lower case letters.

```
def count_upper_lower(x):
    u=0
    l=0
    for i in x:
        if i>='a' and i<='z':
            l+=1

        if i>='A' and i<='Z':
            u+=1

    print("LowerCase letter in the String:",l)
    print("UpperCase letter in the String:",u)

x = input("Enter the string: ")
count_upper_lower(x)
```

```
Enter the string: Ritik Prajapat
LowerCase letter in the String: 11
UpperCase letter in the String: 2
```

Practical 12 - Write a Python program to demonstrate use of: a) Built in module (e.g. keyword, math, number, operator) b) user defined module.

1. Write a Python program to create a user defined module that will ask your college name and will display the name of the college.

college.py

```
def display_college_name():
    college_name = input("Enter your college name: ")
    print("Your college name is:", college_name)
```

ex1.py

```
import college

college.display_college_name()
```

```
Enter your college name: VIVA Diploma
Your college name is: VIVA Diploma
```

2. Write a Python program that will calculate area and circumference of circle using inbuilt Math Module

```
import math

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

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

r = float(input(' Please Enter the radius of a circle: '))

circumference = find_Circumference(r)
area = find_Area(r)

print("Circumference Of a Circle = %.2f" %circumference)
print("Area Of a Circle = %.2f" %area)
```

```
Please Enter the radius of a circle: 5
```

```
Circumference Of a Circle = 31.42
Area Of a Circle = 78.54
```

3. Write a Python program that will display Calendar of given month using Calendar Module

```
import calendar

yy = int(input("Enter year: "))
mm = int(input("Enter month: "))

print(calendar.month(yy, mm))
```

```
Enter year: 2023
Enter month: 5
      May 2023
Mo Tu We Th Fr Sa Su
 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
```

Practical 13 - Write Python program to demonstrate use of: a) built-in packages (e.g. NumPy, Pandas) b) user defined packages

1. Write a Python program to create two matrices and perform addition, subtraction, multiplication and division operation on matrix.

```
import numpy as np

# Two matrices
mx1 = np.array([[5, 10], [15, 20]])
mx2 = np.array([[25, 30], [35, 40]])

print("Matrix1 = \n",mx1)
print("\nMatrix2 = \n",mx2)

# The addition() is used to add matrices
print ("\nAddition of two matrices: ")
print (np.add(mx1,mx2))

# The subtract() is used to subtract matrices
print ("\nSubtraction of two matrices: ")
print (np.subtract(mx1,mx2))

# The divide() is used to divide matrices
print ("\nMatrix Division: ")
print (np.divide(mx1,mx2))

# The multiply()is used to multiply matrices
print ("\nMultiplication of two matrices: ")
print (np.multiply(mx1,mx2))
```

```
Matrix1 =
[[ 5 10]
 [15 20]]

Matrix2 =
 [[25 30]
 [35 40]]

Addition of two matrices:
```

```
[[30 40]
 [50 60]]
Subtraction of two matrices:
[[-20 -20]
 [-20 -20]]

Matrix Division:
[[0.2        0.33333333]
 [0.42857143 0.5       ]]

Multiplication of two matrices:
[[125 300]
 [525 800]]
```

2. Write a Python program to concatenate two strings.

```
string1 = "Hello"
string2 = "world"
concatenated_string = string1 + " " + string2
print(concatenated_string)
```

```
Hello world
```

3. Write a NumPy program to generate six random integers between 10 and 30.

```
import numpy as np
x = np.random.randint(low=10, high=30, size=6)
print(x)
```

```
Random Output - [16 14 18 23 24 18]
```

Practical 14 - Write a program in Python to demonstrate following operations: a) Method overloading b) Method overriding

1. Write a Python program to create a class to print an integer and a character with two methods having the same name but different sequence of the integer and the character parameters. For example, if the parameters of the first method are of the form (int n, char c), then that of the second method will be of the form (char c, int n)

```
class MyClass:
    def print_int_char(self, n, c):
        print('Hello')

    def print_int_char(self, c, n):
        print('Hi')

obj = MyClass()
obj.print_int_char(10, 'a')
obj.print_int_char('b', 20)
```

```
Hi
Hi
```

2. Write a Python program to create a class to print the area of a square and a rectangle. The class has two methods with the same name but different number of parameters. The method for printing area of rectangle has two parameters which are length and breadth respectively while the other method for printing area of square has one parameter which is side of square.

```
class Shape:
    def area(self, a, b=None):
        if b is None:
            print("Area of square:", a * a)
        else:
            print("Area of rectangle:", a * b)
```

```
# Create an object of the Shape class
obj = Shape()

# Call the methods with different parameter values
obj.area(5)        # Area of square: 25
obj.area(10, 20)    # Area of rectangle: 200
```

```
Area of square: 25
Area of rectangle: 200
```

OR

```
class Area:
    def output(self, l, b):
        print("Area of Rectangle is =", l * b)

    def output(self, a):
        print("Area of Square is =", a ** 2)

obj = Area()
obj.output(5)
```

```
Area of Square is = 25
```

3. Write a Python program to create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

```
class Degree:
    def getDegree(self):
        print("I got a degree")

class Undergraduate(Degree):
    def getDegree(self):
        print("I am an Undergraduate")

class Postgraduate(Degree):
    def getDegree(self):
        print("I am a Postgraduate")
```

```
# create objects and call methods
degree = Degree()
degree.getDegree()

undergrad = Undergraduate()
undergrad.getDegree()

postgrad = Postgraduate()
postgrad.getDegree()
```

```
I got a degree
I am an Undergraduate
I am a Postgraduate
```

Practical 15 - Write a program in Python to demonstrate following operations: a) Simple inheritance b) Multiple inheritance

1. Create a class Employee with data members: name, department and salary. Create suitable methods for reading and printing employee information

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

    def read_employee_info(self):
        self.name = input("Enter name: ")
        self.department = input("Enter department: ")
        self.salary = float(input("Enter salary: "))

    def print_employee_info(self):
        print("Name:", self.name)
        print("Department:", self.department)
        print("Salary:", self.salary)

# create an employee object, read and print the info
employee = Employee("", "", 0.0)
employee.read_employee_info()
employee.print_employee_info()
```

```
Enter name: Chirag Yadav
Enter department: Flutter Developer
Enter salary: 5000000

Name: Chirag Yadav
Department: Flutter Developer
Salary: 5000000.0
```

2. Python program to read and print students information using two classes using simple inheritance.

```
class Person:
    def set_person(self):
        self.name = input("Enter student name: ")
        self.age = int(input("Enter student age: "))

class Student(Person):
    def set_student(self):
        self.roll_number = int(input("Enter student roll number: "))
        self.marks = int(input("Enter student marks: "))

    def display(self):
        print("Name:", self.name)
        print("Age:", self.age)
        print("Roll Number:", self.roll_number)
        print("Marks:", self.marks)

student1 = Student()

student1.set_person()
student1.set_student()

student1.display()
```

```
Enter student name: Shubham
Enter student age: 19
Enter student roll number: 58
Enter student marks: 91

Name: Shubham
Age: 19
Roll Number: 58
Marks: 91
```

3. Write a Python program to implement multiple inheritance

```
# define class A
class A:
    def method_A(self):
        print("Method A")
```

```
# define class B
class B:
    def method_B(self):
        print("Method B")

# define class C which inherits from both A and B
class C(A, B):
    def method_C(self):
        print("Method C")

# create an instance of class C and call its methods
c = C()
c.method_A() # Method A
c.method_B() # Method B
c.method_C() # Method C
```

```
Method A
Method B
Method C
```

Practical 16 - Write a program in Python to handle user defined exception for given problem

1. Write a Python program to Check for ZeroDivisionError Exception.

```
# divide two numbers entered by the user
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero")
```

```
Enter the first number: 4
Enter the second number: 0
ERROR!
Error: Cannot divide by zero
```

2. Write a Python program to create user defined exceptions that will check whether the password is correct or not?

```
class InvalidPassword(Exception):
    def __init__(self, message):
        super().__init__(message)

def check_password(password):
    correct_password = "password123" # set the correct password here
    if password != correct_password:
        raise InvalidPassword("Invalid password. Please try again.")
    else:
        print("Login successful.")

# test the function
try:
    password = input("Enter password: ")
```

```
    check_password(password)
except InvalidPassword as error:
    print(error)
```

```
Enter password: password123
Login successful.
```