Name: Rishant Kushwaha (B-46)
Sap Id: 500126797
Enrollment No: R2142232084

Python LAB Experiment

Experiment 1:

 # 2. Write Python programs to print strings in the given manner:

```
# a) Hello Everyone !!!
print("Hello Everyone !!!")

# b) Hello
#    World
print("Hello\nWorld")

# c) Hello
#           World
print("Hello\n         World")
```

 # d) 'Rohit's date of birth is 12\05\1999'

```
print("'Rohit's date of birth is 12\\05\\1999'")
```

 # 3. Declare a string variable called x and assign it the value "Hello".
 # Print out the value of x

```
x = "Hello"
print(x)
```

 # 4. Take different data types and print values using print function.

```
var_int = 10
var_float = 3.14
var_bool = True
var_str = "Hello World"
print("Integer:", var_int)
print("Float:", var_float)
print("Boolean:", var_bool)
print("String:", var_str)
```

 # 5. Take two variable a and b. Assign your first name and last name. Print your Name
 # after adding your First name and Last name together.

```
a = "John"
b = "Doe"
print("Name:", a + " " + b)
```

 # 6. Declare three variables, consisting of your first name, your last name, and Nickname.
 # Write a program that prints out your first name, then your nickname in parenthesis, and
 # then your last name.
 # Example output: George (woody) Washington.

```
first_name = "George"
last_name = "Washington"
nickname = "woody"
print(first_name + " (" + nickname + ") " + last_name)
```

 # 7. Declare and assign values to suitable variables and print in the following way:

```
name = input("Enter your name: ")
sap_id = input("Enter your SAP ID: ")
dob = input("Enter your date of birth (DD MMM YYYY): ")
address_line1 = input("Enter your address line 1: ")
address_line2 = input("Enter your address line 2: ")
pincode = input("Enter your pincode: ")
programme = input("Enter your programme: ")
semester = input("Enter your semester: ")

print("NAME :", name)
print("SAP ID :", sap_id)
print("DATE OF BIRTH :", dob)
print("ADDRESS :", address_line1)
print("          ", address_line2)
print("          Pincode :", pincode)
print("Programme :", programme)
print("Semester :", semester)
```

OUTPUT:

```
IDLE Shell 3.12.1
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Python\Exp 1\Q1..py
Hello Everyone !!!
Hello
World
Hello
          World
'Rohit's date of birth is 12\05\1999'
Hello
Integer: 10
Float: 3.14
Boolean: True
String: Hello World
Name: John Doe
George (woody) Washington
Enter your name: Rishant
Enter your SAP ID: 500126797
Enter your date of birth (DD MMM YYYY): 2004/07/19
Enter your address line 1: Kathmandu
Enter your address line 2: Pokhara
Enter your pincode: 248007
Enter your programme: B-Tech CSE
Enter your semester: 2nd
NAME : Rishant
SAP ID : 500126797
DATE OF BIRTH : 2004/07/19
ADDRESS : Kathmandu
            Pokhara
           Pincode : 248007
Programme : B-Tech CSE
Semester : 2nd
>>>
```

Experiment 2:

 # 1. Declare these variables (x, y and z) as integers. Assign a value of 9 to x, Assign a # value of 7 to y, perform addition, multiplication, division and subtraction on these # two variables and Print out the result.

```
x = 9
y = 7
Add = x + y
mul = x * y
div = x / y
sub = x - y
print("Addition:", Add)
print("Multiplication:", mul)
print("Division:", div)
print("Subtraction:", sub)
```

 # 2. Write a Program where the radius is taken as input to compute the area of a circle.

```
import math
radius = float(input("Enter the radius of the circle: "))
area = math.pi * radius**2
print("Area of the circle:", area)
```

 # 3. Write a Python program to solve (x+y)*(x+y) Test data : x = 4 , y = 3 Expected output: 49

```
x = 4
y = 3
result = (x + y) * (x + y)
print("Result:", result)
```

 # 4. Write a program to compute the length of the hypotenuse (c) of a right triangle

```
# using Pythagoras theorem.
a = float(input("Enter Perpendicular: "))
b = float(input("Enter Base: "))
c = math.sqrt(a**2 + b**2)
print("Hypotenuse =", c)
```

 # 5. Write a program to find simple interest.

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time (in years): "))
simple_interest = (principal * rate * time) / 100
print("Simple Interest:", simple_interest)
```

 # 6. Write a program to find area of triangle when length of sides are given.

```
a = float(input("Enter the length of side 'a': "))
b = float(input("Enter the length of side 'b': "))
c = float(input("Enter the length of side 'c': "))
s = (a + b + c) / 2
area = math.sqrt(s * (s - a) * (s - b) * (s - c))
print("Area of the triangle:", area)
```

 # 7. Write a program to convert given seconds into hours, minutes and remaining seconds.

```
seconds = int(input("Enter the number of seconds: "))
hours = seconds // 3600
minutes = (seconds % 3600) // 60
remaining_seconds = seconds % 60
print("Hours:", hours)
print("Minutes:", minutes)
print("Remaining Seconds:", remaining_seconds)
```

 # 8. Write a program to swap two numbers without taking additional variable.

```
a = int(input("Enter the first number (a): "))
b = int(input("Enter the second number (b): "))
a = a + b
b = a - b
a = a - b
print("After swapping, a =", a)
print("After swapping, b =", b)
```

 # 9. Write a program to find sum of first n natural numbers.

```
n = int(input("Enter a positive integer (n): "))
sum_natural_numbers = (n * (n + 1)) // 2
print("Sum of first", n, "natural numbers:", sum_natural_numbers)
```

 # 10. Write a program to print truth table for bitwise operators(& , | and ^ operators)

```
print("Truth Table for Bitwise AND (&):")
print("0 & 0 =", 0 & 0)
print("0 & 1 =", 0 & 1)
print("1 & 0 =", 1 & 0)
print("1 & 1 =", 1 & 1)

print("\nTruth Table for Bitwise OR (|):")
print("0 | 0 =", 0 | 0)
```

```
print("0 | 1 =", 0 | 1)
print("1 | 0 =", 1 | 0)
print("1 | 1 =", 1 | 1)

print("\nTruth Table for Bitwise XOR (^):")
print("0 ^ 0 =", 0 ^ 0)
print("0 ^ 1 =", 0 ^ 1)
print("1 ^ 0 =", 1 ^ 0)
print("1 ^ 1 =", 1 ^ 1)
```

 # 11. Write a program to find left shift and right shift values of a given number.

```
number = int(input("Enter a number: "))
shift_value = int(input("Enter the shift value: "))
left_shift_result = number << shift_value
right_shift_result = number >> shift_value
print("Left Shift Result:", left_shift_result)
print("Right Shift Result:", right_shift_result)
```

 # 12. Using membership operator find whether a given number is in sequence (10,20,56,78,89)

```
sequence = [10, 20, 56, 78, 89]
number_to_check = int(input("Enter a number to check: "))
if number_to_check in sequence:
    print(number_to_check, "is in the sequence.")
else:
    print(number_to_check, "is not in the sequence.")
```

 # 13. Using membership operator find whether a given character is in a string.

```
string_input = input("Enter a string: ")
character_to_check = input("Enter a character to check: ")
if character_to_check in string_input:
    print(character_to_check, "is present in the string.")
else:
    print(character_to_check, "is not present in the string.")
```

OUTPUT:

```
========================================================================== RESTART: D:\Python\Exp 1\Q2.py ==========================================================================
Enter a number:24
Enter a number:31
744
55
0.7741935483870968
Enter the radius of the circle: 4
Area of the circle: 50.26548245743669
Result: 49
Enter Perpendicular: 4
Enter Base: 5
Hypotenuse = 41.0
Enter the length of side 'a': 7
Enter the length of side 'b': 8
Enter the length of side 'c': 1
Area of the triangle: 0.0
Enter the principal amount: 5
Enter the rate of interest: 79
Enter the time (in years): 2
Simple Interest: 7.9
Enter the number of seconds: 6700
Hours: 1
Minutes: 51
Remaining Seconds: 40
Enter the first number (a): 2
Enter the second number (b): 3
After swapping, a = 3
After swapping, b = 2
Enter a positive integer (n): 2
Sum of first 2 natural numbers: 3
Truth Table for Bitwise AND (&):
0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1

Truth Table for Bitwise OR (|):
0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

Truth Table for Bitwise XOR (^):
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0
```

```
Enter a number: 46
Enter the shift value: 23
Left Shift Result: 385875968
Right Shift Result: 0
Enter a number to check: 153
153 is not in the sequence.
Enter a string: Rishant
Enter a character to check: @
@ is not present in the string.
```

Experiment 3:

```
import math
```

 # 1. Check whether given number is divisible by 3 and 5 both.

```
num = 15
if num % 3 == 0 and num % 5 == 0:
    print(f"{num} is divisible by both 3 and 5.")
else:
    print(f"{num} is not divisible by both 3 and 5.")
```

 # 2. Check whether a given number is a multiple of five or not.

```
num = 20
if num % 5 == 0:
    print(f"{num} is a multiple of five.")
else:
    print(f"{num} is not a multiple of five.")
```

 # 3. Find the greatest among two numbers. If numbers are equal, then print "numbers are equal".

```
num1, num2 = 10, 20
if num1 > num2:
    print(f"{num1} is greater than {num2}.")
elif num2 > num1:
    print(f"{num2} is greater than {num1}.")
else:
    print("Numbers are equal.")
```

 # 4. Find the greatest among three numbers assuming no two values are the same.

```
num1, num2, num3 = 30, 40, 50
if num1 > num2 and num2 > num3:
    print(num1,"is greater:")
elif num2 > num3 and num2 > num1:
    print(num2,"is greater:")
else:
    print(num3,"is greater:")
```

 # 5. Check whether the quadratic equation has real roots or imaginary roots. Display the roots.

```
a, b, c = 1, -3, 2
discriminant = b**2 - 4*a*c
if discriminant > 0:
    root1 = (-b + math.sqrt(discriminant)) / (2*a)
    root2 = (-b - math.sqrt(discriminant)) / (2*a)
    print(f"The quadratic equation has real roots: {root1}, {root2}")
elif discriminant == 0:
    root = -b / (2*a)
    print(f"The quadratic equation has real and equal roots: {root}")
else:
    real_part = -b / (2*a)
    imaginary_part = math.sqrt(abs(discriminant)) / (2*a)
    print(f"The quadratic equation has imaginary roots: {real_part} + {imaginary_part}i,
{real_part} - {imaginary_part}i")
```

 # 6. Find whether a given year is a leap year or not.

```
year = 2024
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

 # 7. Write a program which takes any date as input and display the next date of the calendar.

```
day, month, year = 29, 2, 2004
leap_year = (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
days_in_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
if leap_year:
    days_in_month[1] = 29
if day < days_in_month[month - 1]:
    next_day = day + 1
    next_month = month
    next_year = year
else:
    next_day = 1

    if month == 12:
        next_month = 1
```

```
        next_year = year + 1
    else:
        next_month = month + 1
        next_year = year
print(f"Next date: day={next_day} month={next_month} year={next_year}")
```

 # 8. Print the grade sheet of a student for the given range of CGPA.

```
name = "Rishant Kushwaha"
roll_number = "R2142232084"
semester = 1
marks = {'PDS': 80, 'Python': 80, 'Chemistry': 90, 'English': 78, 'Physics': 75}
total_marks = sum(marks.values())
percentage = total_marks / (len(marks) * 100) * 100
cgpa = percentage / 10

if cgpa <= 3.4:
    grade = 'F'
elif cgpa <= 5.0:
    grade = 'C+'
elif cgpa <= 6.0:
    grade = 'B'
elif cgpa <= 7.0:
    grade = 'B+'
elif cgpa <= 8.0:
    grade = 'A'
elif cgpa <= 9.0:
    grade = 'A+'
else:
    grade = 'O (Outstanding)'

print(f"Name: {name}\nRoll Number: {roll_number}\nSem: {semester}")
print("Subject name: Marks")
for subject, marks in marks.items():
    print(f"{subject}: {marks}")
print(f"Percentage: {percentage}%\nCGPA: {cgpa}\nGrade: {grade}")
```

OUTPUT:

```
========================================================================== RESTART: D:\Python\Exp 1\Q3.py ===============
15 is divisible by both 3 and 5.
20 is a multiple of five.
20 is greater than 10.
50 is greater:
The quadratic equation has real roots: 2.0, 1.0
2024 is a leap year.
Next date: day=1 month=3 year=2004
Name: Rishant Kushwaha
Roll Number: R2142232084
Sem: 1
Subject name: Marks
PDS: 80
Python: 80
Chemistry: 90
English: 78
Physics: 75
Percentage: 80.60000000000001%
CGPA: 8.06
Grade: A+
```

Experiment 4:

1. Find factorial of a given number.

```
number = int(input("Enter a number to find its factorial: "))
factorial = 1
if number < 0:
    print("Factorial does not exist for negative numbers.")
elif number == 0:
    print("Factorial of 0 is 1")
else:
    for i in range(1, number + 1):
        factorial *= i
    print("Factorial of", number, "is", factorial)
```

2. Find whether the given number is Armstrong number.

```
number = int(input("Enter a number to check if it's an Armstrong number: "))
order = len(str(number))
sum = 0
temp = number
while temp > 0:
    digit = temp % 10
    sum += digit ** order
    temp //= 10
if number == sum:
    print(number, "is an Armstrong number.")
else:
```

```
    print(number, "is not an Armstrong number.")
```

 # 3. Print Fibonacci series up to a given term.

```
terms = int(input("Enter the number of terms for Fibonacci series: "))
a, b = 0, 1
count = 0
if terms <= 0:
    print("Please enter a positive integer.")
elif terms == 1:
    print("Fibonacci sequence upto", terms, "term:", a)
else:
    print("Fibonacci sequence:")
    while count < terms:
        print(a, end=" ")
        nth = a + b
        a = b
        b = nth
        count += 1
```

 # 4. Write a program to find if a given number is a prime number or not.

```
number = int(input("Enter a number to check if it's prime: "))
if number > 1:
    for i in range(2, int(number / 2) + 1):
        if (number % i) == 0:
            print(number, "is not a prime number.")
            break
    else:
        print(number, "is a prime number.")
else:
    print(number, "is not a prime number.")
```

 # 5. Check whether the given number is palindrome or not.

```
number = input("Enter a number to check if it's a palindrome: ")
if number == number[::-1]:
    print(number, "is a palindrome.")
else:
    print(number, "is not a palindrome.")

# 6. Write a program to print the sum of digits.
number = input("Enter a number to find the sum of its digits: ")
```

```
sum = 0
for digit in number:
    sum += int(digit)
print("Sum of digits of", number, "is", sum)
```

 # 7. Count and print all numbers divisible by 5 or 7 between 1 to 100.

```
print("Numbers divisible by 5 or 7 between 1 to 100:")
for i in range(1, 101):
    if i % 5 == 0 or i % 7 == 0:
        print(i, end=" ")
```

```
# 8. Convert all lower cases to upper case in a string.
string_input = input("Enter a string to convert all lowercase to uppercase: ")
print("Uppercase:", string_input.upper())
```

```
# 9. Print all prime numbers between 1 and 100.
print("Prime numbers between 1 and 100:")
for num in range(1, 101):
    if num > 1:
        for i in range(2, int(num / 2) + 1):
            if (num % i) == 0:
                break
        else:
            print(num, end=" ")
```

 # 10. Print the table for a given number.

```
number = int(input("Enter a number to print its table: "))
print("Table for", number, ":")
for i in range(1, 11):
    print(number, "*", i, "=", number * i)
```

OUTPUT:

```
---------------------------------------------------------------------------- RESTART: D:\Python\Exp 4\Exp 4.py ----------------------------------------------------------------------------
Enter a number: 5
5 is a prime number
Enter a number: 7
7 is a palindrome
Enter a number: 345
Sum of digits in 345 is: 12
Numbers divisible by 5 or 7 between 1 and 100: [5, 7, 10, 14, 15, 20, 21, 25, 28, 30, 35, 40, 42, 45, 49, 50, 55, 56, 60, 63, 65, 70, 75, 77, 80, 84, 85, 90, 91, 95, 98, 100]
Prime numbers between 1 and 100:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Enter a number: 45
Multiplication table for 45
45 * 1 = 45
45 * 2 = 90
45 * 3 = 135
45 * 4 = 180
45 * 5 = 225
45 * 6 = 270
45 * 7 = 315
45 * 8 = 360
45 * 9 = 405
45 * 10 = 450
```

Experiment 5:

 # 1. Write a program to count and display the number of capital letters in a given string.

```
string_input = input("Enter a string: ")
count = 0
for char in string_input:
    if char.isupper():
        count += 1
print("Number of capital letters:", count)
```

 # 2. Count total number of vowels in a given string.

```
string_input = input("Enter a string: ")
vowels = 'aeiouAEIOU'
count = 0
for char in string_input:
    if char in vowels:
        count += 1
print("Total number of vowels:", count)
```

 # 3. Input a sentence and print words in separate lines.

```
sentence = input("Enter a sentence: ")
words = sentence.split()
for word in words:
    print(word)
```

 # 4. WAP to enter a string and a substring. You have to print the number of times that
 # the substring occurs in the given string. String traversal will take place from left to
 # right, not from right to left.

```
string_input = input("Enter a string: ")
substring = input("Enter a substring: ")
count = string_input.count(substring)
print("Number of times substring occurs in the string:", count)
```

 # 5. Given a string containing both upper and lower case alphabets. Write a Python
 # program to count the number of occurrences of each alphabet (case insensitive)
 # and display the same.

```
string_input = input("Enter a string: ")
frequency = {}
for char in string_input:
    if char.isalpha():
        char = char.upper()
        if char in frequency:
            frequency[char] += 1
        else:
            frequency[char] = 1
for char, count in frequency.items():
    print(count, char)
```

 # 6. Program to count number of unique words in a given sentence using sets.

```
sentence = input("Enter a sentence: ")
words = sentence.split()
unique_words = set(words)
print("Number of unique words:", len(unique_words))
```

 # 7. Create 2 sets s1 and s2 of n fruits each by taking input from user and find:
 # a) Fruits which are in both sets s1 and s2
 # b) Fruits only in s1 but not in s2
 # c) Count of all fruits from s1 and s2

```
n = int(input("Enter the number of fruits in each set: "))
s1 = set(input("Enter fruits for set 1 separated by space: ").split())
s2 = set(input("Enter fruits for set 2 separated by space: ").split())
print("Fruits in both sets:", s1.intersection(s2))
print("Fruits only in set 1 but not in set 2:", s1.difference(s2))
print("Count of all fruits from both sets:", len(s1.union(s2)))
```

```
# 8. Take two sets and apply various set operations on them:
S1 = {'Red', 'yellow', 'orange', 'blue'}
S2 = {'violet', 'blue', 'purple'}
print("Intersection:", S1.intersection(S2))
print("Union:", S1.union(S2))
print("Difference (S1 - S2):", S1.difference(S2))
print("Difference (S2 - S1):", S2.difference(S1))
print("Symmetric Difference:", S1.symmetric_difference(S2))
```

OUTPUT:

```
========================================================================== RESTART: D:\Python\Exp 5\exp5.py ====================
Enter a string: Rishant
Number of capital letters: 1
Enter a string: Kushwaha
Total number of vowels: 3
Enter a sentence: Rishant is a good student
Rishant
is
a
good
student
Enter a string: Nepal
Enter a substring: Pokhara
Number of times the substring occurs: 0
Enter a string: Tiring
1 t
2 i
1 r
1 n
1 g
Enter a sentence: Rishant will get more than 8 CGPA in 2nd sem
Number of unique words: 10
Enter the number of fruits: 4
Enter fruits for set 1: Banana
Enter fruits for set 2: Apple
Fruits in both sets: set()
Fruits only in s1: {'Banana'}
Total count of fruits: 2
Union of sets: {'Red', 'blue', 'violet', 'orange', 'purple', 'yellow'}
Intersection of sets: {'blue'}
Difference of sets (S1 - S2): {'orange', 'Red', 'yellow'}
Symmetric difference of sets: {'Red', 'orange', 'purple', 'violet', 'yellow'}
```

Experiment 6:

 # 1. Scan n values in range 0-3 and print the number of times each value has occurred.

```
n = int(input("Enter the number of values: "))
occurrences = {0: 0, 1: 0, 2: 0, 3: 0}
for _ in range(n):
    value = int(input("Enter a value (0-3): "))
    if value in occurrences:
        occurrences[value] += 1
print("Occurrences of each value:", occurrences)
```

 # 2. Create a tuple to store n numeric values and find the average of all values.

```
n = int(input("Enter the number of values: "))
values = tuple(float(input("Enter a numeric value: ")) for _ in range(n))
average = sum(values) / n
print("Average of all values:", average)
```

 # 3. WAP to input a list of scores for N students in a list data type. Find the score of the
 # runner-up and print the output.

```
N = int(input("Enter the number of students: "))
scores = list(map(int, input("Enter scores of N students separated by space: ").split()))
unique_scores = list(set(scores))
unique_scores.sort(reverse=True)
print("Runner-up score:", unique_scores[1])
```

 # 4. Create a dictionary of n persons where the key is the name and the value is the city.
 # a) Display all names
 # b) Display all city names
 # c) Display student name and city of all students.
 # d) Count the number of students in each city.

```
n = int(input("Enter the number of persons: "))
persons = {}
for _ in range(n):
    name = input("Enter person's name: ")
    city = input("Enter person's city: ")
    persons[name] = city
```

 # a) Display all names

```
print("Names:", list(persons.keys()))
```

 # b) Display all city names

```
print("City Names:", list(set(persons.values())))
```

 # c) Display student name and city of all students.

```
print("Student Name and City:")
for name, city in persons.items():
    print(name, "-", city)
```

 # d) Count the number of students in each city.

```
city_count = {}
for city in persons.values():
    if city in city_count:
        city_count[city] += 1
    else:
        city_count[city] = 1
print("Number of students in each city:", city_count)
```

 # 5. Store details of n movies in a dictionary by taking input from the user. Each movie
 # must store details like name, year, director name, production cost, collection made (earning).
 # Perform the following:
 # a) print all movie details
 # b) display names of movies released before 2015
 # c) print movies that made a profit.
 # d) print movies directed by a particular director.

```
n = int(input("Enter the number of movies: "))
movies = []
for i in range(n):
    movie_details = {}
    print(f"Enter details for movie {i+1}:")
    movie_details['name'] = input("Name: ")
    movie_details['year'] = int(input("Year: "))
    movie_details['director'] = input("Director Name: ")
    movie_details['production_cost'] = float(input("Production Cost: "))
    movie_details['collection'] = float(input("Collection: "))
    movies.append(movie_details)

# a) Print all movie details
print("All movie details:")
for movie in movies:
    print(movie)
```

 # b) Display names of movies released before 2015

```
print("Movies released before 2015:")
for movie in movies:
    if movie['year'] < 2015:
        print(movie['name'])
```

 # c) Print movies that made a profit

```
print("Movies that made a profit:")
for movie in movies:
    if movie['collection'] > movie['production_cost']:
        print(movie['name'])
```

 # d) Print movies directed by a particular director

```
director = input("Enter director's name to find their movies: ")
print(f"Movies directed by {director}:")
for movie in movies:
    if movie['director'] == director:
        print(movie['name'])
```

OUTPUT:

```
========================================================================== RESTART: D:\Python\Exp 6\exp 6.py ============
Enter the number of values: 4
Enter a value (0-3): 2
Enter a value (0-3): 2
Enter a value (0-3): 1
Enter a value (0-3): 0
Occurrences of each value: {0: 1, 1: 1, 2: 2, 3: 0}
Enter the number of values: 5
Enter a numeric value: 2
Enter a numeric value: 3
Enter a numeric value: 4
Enter a numeric value: 1
Enter a numeric value: 7
Average of all values: 3.4
Enter the number of students: 5
Enter scores of N students separated by space: 43 67 45 39 89
Runner-up score: 67
Enter the number of persons: 4
Enter person's name: Rishant
Enter person's city: Nepal
Enter person's name: Anadi
Enter person's city: Dehradun
Enter person's name: Akshat
Enter person's city: UP
Enter person's name: Priyanshu
Enter person's city: Uttrakhand
Names: ['Rishant', 'Anadi ', 'Akshat', 'Priyanshu']
City Names: ['Nepal', 'Uttrakhand', 'Dehradun', 'UP']
Student Name and City:
Rishant - Nepal
Anadi  - Dehradun
Akshat - UP
Priyanshu - Uttrakhand
Number of students in each city: {'Nepal': 1, 'Dehradun': 1, 'UP': 1, 'Uttrakhand': 1}
Enter the number of movies: 1
Enter details for movie 1:
Name: 3 idiot
Year: 2009
Director Name: Raj kumar hirani
Production Cost: 100crore
```

Experiment 7:

 # 1. Write a Python function to find the maximum and minimum numbers from a sequence of numbers.

```
def find_max_min(sequence):
    maximum = sequence[0]
    minimum = sequence[0]
    for num in sequence:
        if num > maximum:
            maximum = num
        if num < minimum:
            minimum = num
    return maximum, minimum
```

 # 2. Write a Python function that takes a positive integer and returns the sum of the cube of all the positive integers smaller than the specified number.

```
def sum_cube_positive_integers(n):
    return sum(i**3 for i in range(1, n))
```

 # 3. Write a Python function to print 1 to n using recursion.

```
def print_numbers(n):
    if n > 0:
        print_numbers(n - 1)
        print(n)
```

 # 4. Write a recursive function to print Fibonacci series upto n terms.

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

def print_fibonacci_series(n):
    for i in range(n):
        print(fibonacci(i))
```

 # 5. Write a lambda function to find the volume of a cone.

```
volume_of_cone = lambda radius, height: (1/3) * 3.14159 * radius**2 * height
```

 # 6. Write a lambda function which gives a tuple of max and min from a list.

```
max_min_tuple = lambda lst: (max(lst), min(lst))
```

 # 7. Write functions to explain mentioned concepts:
 # a. Keyword argument

```
def keyword_argument_example(name, age):
    print("Name:", name)
    print("Age:", age)
```

 # b. Default argument

```
def default_argument_example(name="John", age=30):
    print("Name:", name)
    print("Age:", age)
```

 # c. Variable length argument

```
def variable_length_argument_example(*args):
    print("Arguments:", args)
```

 # Test cases for the functions

```
sequence = [10, 6, 8, 90, 12, 56]
print("Max and Min:", find_max_min(sequence))
print("Sum of cubes up to 5:", sum_cube_positive_integers(5))
print("Numbers from 1 to 5:")
print_numbers(5)
print("Fibonacci series up to 6 terms:")
print_fibonacci_series(6)
print("Volume of cone with radius 3 and height 5:", volume_of_cone(3, 5))
print("Max and Min from list:", max_min_tuple(sequence))
```

 # Test cases for function explanation

```
keyword_argument_example(age=25, name="Alice")
default_argument_example()
variable_length_argument_example(1, 2, 3, 4, 5)
```

OUTPUT:

```
========================================================================== RESTART: D:/Python/exp 7/exp 7.py ==================================
Max and Min: (90, 6)
Sum of cubes up to 5: 100
Numbers from 1 to 5:
1
2
3
4
5
Fibonacci series up to 6 terms:
0
1
1
2
3
5
Volume of cone with radius 3 and height 5: 47.12384999999999
Max and Min from list: (90, 6)
Name: Alice
Age: 25
Name: John
Age: 30
Arguments: (1, 2, 3, 4, 5)
```

Experiment 8:

 # 1. Add few names, one name in each row, in “name.txt" file.

```
with open("name.txt", "w") as file:
    file.write("Alice\n")
    file.write("Bob\n")
    file.write("Charlie\n")
    file.write("David\n")
    file.write("Eva\n")

# a. Count the number of names
with open("name.txt", "r") as file:
    names = file.readlines()
num_of_names = len(names)
print("Number of names:", num_of_names)

# b. Count all names starting with a vowel
vowels = "aeiouAEIOU"
num_starting_with_vowel = sum(1 for name in names if name[0] in vowels)
print("Number of names starting with a vowel:", num_starting_with_vowel)

# c. Find the longest name
longest_name = max(names, key=len).strip()
print("Longest name:", longest_name)
```

 # 2. Store integers in a file.

```
with open("integers.txt", "w") as file:
    file.write("10\n")
    file.write("20\n")
    file.write("30\n")
    file.write("40\n")
    file.write("50\n")

# Read integers from file
with open("integers.txt", "r") as file:
    integers = [int(line.strip()) for line in file]

# a. Find the max number
max_number = max(integers)
print("Max number:", max_number)

# b. Find average of all numbers
average = sum(integers) / len(integers)
print("Average:", average)

# c. Count number of numbers greater than 100
num_greater_than_100 = sum(1 for num in integers if num > 100)
print("Number of numbers greater than 100:", num_greater_than_100)
```

 # 3. Create a file named "city.txt" with details of 5 cities.

```
with open("city.txt", "w") as file:
    file.write("Dehradun 5.78 308.20\n")
    file.write("Delhi 190 1484\n")
    file.write("Mumbai 120 603\n")
    file.write("Kolkata 70 185\n")
    file.write("Chennai 60 426\n")

# a. Display details of all cities
with open("city.txt", "r") as file:
    cities = [line.strip().split() for line in file]
print("Details of all cities:")
for city in cities:
    print(city)

# b. Display city names with population more than 10 Lakhs
```

```
population_more_than_10_lakhs = [city[0] for city in cities if float(city[1]) > 10]
print("City names with population more than 10 Lakhs:", population_more_than_10_lakhs)

# c. Display sum of areas of all cities
sum_of_areas = sum(float(city[2]) for city in cities)
print("Sum of areas of all cities:", sum_of_areas)
```

 # 4. Implement the integer division operation.

```
try:
    N = int(input("Enter the number of test cases: "))
    for _ in range(N):
        a, b = map(int, input().split())
        print(a // b)
except ZeroDivisionError:
    print("Error Code: integer division or modulo by zero")
except ValueError as e:
    print(f"Error Code: {e}")
```

 # 5. Create multiple suitable exceptions for a file handling program.

```
try:
    with open("non_existent_file.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("File not found.")
except PermissionError:
    print("Permission denied to access the file.")
except Exception as e:
    print(f"An error occurred: {e}")
```

OUTPUT:

```
========================================================================== RESTART: D:/Python/exp 8/exp 8.py ==========================================================================
Number of names: 5
Number of names starting with a vowel: 2
Longest name: Charlie
Max number: 50
Average: 30.0
Number of numbers greater than 100: 0
Details of all cities:
['Dehradun', '5.78', '308.20']
['Delhi', '190', '1484']
['Mumbai', '120', '603']
['Kolkata', '70', '185']
['Chennai', '60', '426']
City names with population more than 10 Lakhs: ['Delhi', 'Mumbai', 'Kolkata', 'Chennai']
Sum of areas of all cities: 3006.2
Enter the number of test cases:
```

Experiment 9:

 # 1. Create a class of student (name, sap id, marks[phy,chem,maths]). Create 3
 # objects by taking inputs from the user and display details of all students.

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

    def display_details(self):
        print("Name:", self.name)
        print("SAP ID:", self.sap_id)
        print("Marks (Physics, Chemistry, Maths):", self.marks)

# Create three Student objects
students = []
for i in range(3):
    name = input("Enter student name: ")
    sap_id = input("Enter SAP ID: ")
    phy_marks = int(input("Enter Physics marks: "))
    chem_marks = int(input("Enter Chemistry marks: "))
    maths_marks = int(input("Enter Maths marks: "))
    marks = [phy_marks, chem_marks, maths_marks]
    student = Student(name, sap_id, marks)
    students.append(student)

# Display details of all students
print("\nDetails of all students:")
for student in students:
    student.display_details()
    print()
```

 # 2. Add constructor in the above class to initialize student details of n students and
 # implement following methods:
 # a) Display() student details
 # b) Find Marks_percentage() of each student
 # c) Display result() [Note: if marks in each subject >40% than Pass else Fail]
 # Write a Function to find average of the class.

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

    def display_details(self):
        print("Name:", self.name)
        print("SAP ID:", self.sap_id)
        print("Marks (Physics, Chemistry, Maths):", self.marks)

    def marks_percentage(self):
        total_marks = sum(self.marks)
        percentage = (total_marks / (len(self.marks) * 100)) * 100
        return percentage

    def result(self):
        for mark in self.marks:
            if mark < 40:
                return "Fail"
        return "Pass"

def class_average(students):
    total_percentage = sum(student.marks_percentage() for student in students)
    average_percentage = total_percentage / len(students)
    return average_percentage

# Create a list of Student objects
students = []
n = int(input("Enter the number of students: "))
for i in range(n):
    name = input("Enter student name: ")
    sap_id = input("Enter SAP ID: ")
    phy_marks = int(input("Enter Physics marks: "))
    chem_marks = int(input("Enter Chemistry marks: "))
    maths_marks = int(input("Enter Maths marks: "))
```

```
    marks = [phy_marks, chem_marks, maths_marks]
    student = Student(name, sap_id, marks)
    students.append(student)

# Display details, marks percentage, and result of each student
print("\nDetails of all students:")
for student in students:
    student.display_details()
    print("Marks Percentage:", student.marks_percentage())
    print("Result:", student.result())
    print()

# Calculate and display the average percentage of the class
average = class_average(students)
print("Average Percentage of the class:", average)
```

OUTPUT:

```
========================================================================== RESTART: D:/Python/exp 8/exp 9.py =====
Enter student name: Rishant
Enter SAP ID: 500126797
Enter Physics marks: 57
Enter Chemistry marks: 65
Enter Maths marks: 49
Enter student name: Anadi
Enter SAP ID: 500126798
Enter Physics marks: 45
Enter Chemistry marks: 67
Enter Maths marks: 39
Enter student name: Akshat
Enter SAP ID: 500124379
Enter Physics marks: 88
Enter Chemistry marks: 67
Enter Maths marks: 45

Details of all students:
Name: Rishant
SAP ID: 500126797
Marks (Physics, Chemistry, Maths): [57, 65, 49]

Name: Anadi
SAP ID: 500126798
Marks (Physics, Chemistry, Maths): [45, 67, 39]

Name: Akshat
SAP ID: 500124379
Marks (Physics, Chemistry, Maths): [88, 67, 45]

Enter the number of students: 2
Enter student name: Rishant
Enter SAP ID: 500126797
Enter Physics marks: 57
Enter Chemistry marks: 65
Enter Maths marks: 49
Enter student name: Anadi
Enter SAP ID: 500126798
Enter Physics marks: 45
Enter Chemistry marks: 67
Enter Maths marks: 39

Details of all students:
Name: Rishant
SAP ID: 500126797
Marks (Physics, Chemistry, Maths): [57, 65, 49]
Marks Percentage: 56.99999999999999
```

Experiment 10:

```
import numpy as np
import pandas as pd
```

 # 1. Create numpy array to find the sum of all elements in an array.

```
arr1 = np.array([1, 2, 3, 4, 5])
sum_arr1 = np.sum(arr1)
print("Sum of all elements in arr1:", sum_arr1)
```

 # 2. Create numpy array of (3,3) dimension. Now find the sum of all rows & columns
 # individually. Also find the 2nd maximum element in the array.

```
arr2 = np.array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]])
row_sums = np.sum(arr2, axis=1)
col_sums = np.sum(arr2, axis=0)
second_max = np.partition(arr2.flatten(), -2)[-2]
print("Row sums:", row_sums)
print("Column sums:", col_sums)
print("Second maximum element:", second_max)
```

 # 3. Perform Matrix multiplication of any 2 n*n matrices.

```
matrix1 = np.array([[1, 2],
                    [3, 4]])
matrix2 = np.array([[5, 6],
                    [7, 8]])
result = np.dot(matrix1, matrix2)
print("Matrix multiplication result:")
print(result)
```

 # 4. Write a Pandas program to get the powers of array values element-wise.

```
data = {'X': [78, 85, 96, 80, 86], 'Y': [84, 94, 89, 83, 86], 'Z': [86, 97, 96, 72, 83]}
df = pd.DataFrame(data)

# Define a function to calculate powers element-wise
def calculate_power(value, exponent):
    return np.power(value, exponent)

# Apply the function to each element of the DataFrame
```

```
powers_df = df.apply(calculate_power, exponent=2)
print("Powers of array values element-wise:")
print(powers_df)
```

 # 5. Write a Pandas program to get the first 3 rows of a given DataFrame.

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew',
'Laura', 'Kevin', 'Jonas'],
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data, index=labels)
first_three_rows = df.head(3)
print("First three rows of the DataFrame:")
print(first_three_rows)
```

 # 6. Write a Pandas program to find and replace the missing values in a given DataFrame
 # which do not have any valuable information.

```
df.replace(np.nan, 0, inplace=True)
print("DataFrame after replacing missing values:")
print(df)
```

 # 7. Create a program to demonstrate different visual forms using Matplotlib.

```
import matplotlib.pyplot as plt
```

 # Example: Plotting a simple line graph

```
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title("Sine Wave")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

OUTPUT:

```
========================================================================== RESTART: D:/Python/exp 8/exp 10.py ==========================================
Sum of all elements in arr1: 15
Row sums: [ 6 15 24]
Column sums: [12 15 18]
Second maximum element: 8
Matrix multiplication result:
[[19 22]
 [43 50]]
Powers of array values element-wise:
      X     Y     Z
0  6084  7056  7396
1  7225  8836  9409
2  9216  7921  9216
3  6400  6889  5184
4  7396  7396  6889
First three rows of the DataFrame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
DataFrame after replacing missing values:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    0.0         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    0.0         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```