

**Name: Saral (B-45)**  
**EXPERIMENT**

**LAB-**

**Sap Id: 500126672**

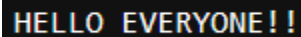
**Enrollment No: R2142232082**

## **EXPERIMENT-1**

Q.1 Write Python programs to print strings in the given manner:

a) Hello Everyone !!

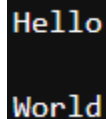
```
print("HELLO EVERYONE!!")
```

A terminal window with a black background and yellow text displaying "HELLO EVERYONE!!".

b) Hello

World

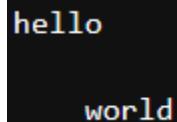
```
print("Hello\n\nWorld")
```

A terminal window with a black background and yellow text displaying "Hello" on the first line and "World" on the second line, separated by a blank line.

c) Hello

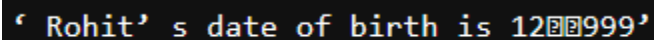
World

```
print("hello\n\n world")
```

A terminal window with a black background and yellow text displaying "hello" on the first line and "world" on the second line, separated by a blank line.

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

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

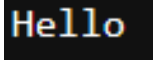
A terminal window with a black background and yellow text displaying "' Rohit' s date of birth is 12051999' " (with escaped backslashes).

Q.2 Declare a string variable called x and assign it the value "Hello".

Print out the value of x

```
x="Hello"
```

```
print(x)
```



Q.3 Take different data types and print values using print function.

```
x=5
```

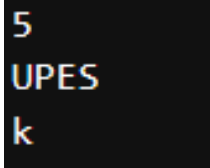
```
y="UPES"
```

```
z='k'
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

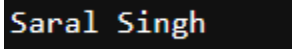


Q.4 Take two variable a and b. Assign your first name and last name. Print your Name after adding your First name and Last name together.

```
a="Saral "
```

```
b="Singh"
```

```
print(a+b)
```



Q.5 Declare three variables, consisting of your first name, your last name and Nickname.

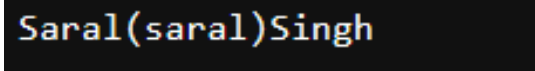
Write a program that prints out your first name, then your nickname in parenthesis and then your last name.

```
a="Saral"
```

```
b="Singh"
```

```
c="saral"
```

```
print(a+"("+c+")"+b)
```



Q.6 Declare and assign values to suitable variables and print in the following way :

NAME : NIKUNJ BANSAL

SAP ID : 500069944

DATE OF BIRTH : 13 Oct 1999

ADDRESS : UPES

Bidholi Campus

Pincode : 248007

Programme : AI & ML

Semester : 2

n="Saral"

sap="500126672"

dob="13 dec 2004"

add="UPES\n      bidholi campus\n      pincode:248007"

prog="AI & ML"

sem="2"

print("NAME : "+n)

print("SAP ID : "+sap)

print("DATE OF BIRTH : "+dob)

print("ADDRESS : "+add)

print("PROGRAMME : "+prog)

print("SEMESTER : "+sem)

```
NAME : Saral
SAP ID : 500126672
DATE OF BIRTH : 13 dec 2004
ADDRESS : UPES
          bidholi campus
          pincode:248007
PROGRAMME : AI & ML
SEMESTER : 2
```

## EXPERIMENT-2

Q.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
```

```
z=1
```

```
print(x+y)
```

```
print(y*z)
```

```
print(z-x)
```

```
16
7
-8
```

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

```
r=6
```

```
ar=3.14*r*r
```

```
print(ar)
```

```
113.03999999999999
```

Q.3 Write a Python program to solve  $(x+y)*(x+y)$

Test data :  $x = 4$  ,  $y = 3$

Expected output: 49"

```
x=4
```

```
y=3
```

```
print((x+y)*(x+y))
```

```
49
```

Q.4 Write a program to compute the length of the hypotenuse (c) of a right triangle using Pythagoras theorem

```
a=int(input())
b=int(input())
import math
c=a*a+b*b
print(c**0.5)
```

```
76
76
107.48023074035522
```

Q.5 Write a program to find simple interest.

```
p=100
r=5
t=6
si=(p*r*t)/100
print(si)
```

```
30.0
```

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

```
s1 = int(input("Enter the side 1: "))
s2 = int(input("Enter the side 2: "))
s3 = int(input("Enter the side 3: "))
s = (a + b + c) / 2
import math
area = math.sqrt((s)* (s-s1) * (s-s2)*(s-s3))
print("The area of the triangle is:",area)\
```

```
Enter the side 1: 4
Enter the side 2: 5
Enter the side 3: 6
The area of the triangle is: 34202022.8761217
```

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

```
t=int(input("Enter time in seconds :"))
hours=t//3600
t=t%3600
min=t//60
t=t%60
print(f"{hours}:{min}:{t}")
```

```
Enter time in seconds : 45678
12:41:18
```

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

```
x=int(input())
y=int(input())
print ("Before swapping: ")
print("Value of x:", x, " and y:", y)
x, y = y, x
print ("After swapping: ")
print("Value of x:", x, " and y:", y)
```

```
4
5
Before swapping:
Value of x: 4 and y: 5
After swapping:
Value of x: 5 and y: 4
```

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

```
n=int(input("Enter Till where you want to find the sum"))
```

```
sum=0
for x in range(n):
    sum=sum+x
print(f"Sum of {n} numbers is =",sum)
```

```
Enter Till where you want to find the sum 65
Sum of 65 numbers is = 2080
```

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

```
print("Truth Table for AND")
```

```
print("0 & 0 =", 0 & 0)
```

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

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

```
print("1 & 1 =", 1 & 1)
```

```
print("Truth Table for R")
```

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

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

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

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

```
print("Truth Table for XOR")
```

```
print("0 ^ 0 =", 0 ^ 0)
```

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

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

```
print("1 ^ 1 =", 1 ^ 1)
```

Truth Table for AND

0 & 0 = 0

0 & 1 = 0

1 & 0 = 0

1 & 1 = 1

Truth Table for R

0 | 0 = 0

0 | 1 = 1

1 | 0 = 1

1 | 1 = 1

Truth Table for XOR

0 ^ 0 = 0

0 ^ 1 = 1

1 ^ 0 = 1

1 ^ 1 = 0

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

```
num= int(input("Enter a number: "))
shifts = int(input("Enter the number of shifts: "))
ls= num << shifts
rs= num >> shifts
print("Left shifted value:", ls)
print("Right shifted value:", rs)
```

```
Enter a number: 4
Enter the number of shifts: 6
Left shifted value: 256
Right shifted value: 0
```

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

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



```
Enter a number to check: 01
1 is not in the sequence.
```

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

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

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

```
if char in str:
```

```
    print(char, "is present in the string.")
```

```
else:
```

```
    print(char, "is not present in the string.")
```

```
Enter a string: Bachhan Ji
Enter a character to check: p
p is not present in the string.
```

### EXPERIMENT-3

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

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

```
if n%3==0 and n%5==0:
```

```
    print(f'Yes, {n} is divisible by both 3 and 5')
```

```
else:
```

```
    print(f'No, {n} is not divisible by both 3 and 5')
```

```
Enter a number: 15
Yes, 15 is divisible by both 3 and 5
```

**'Q.2 Check whether a given number is multiple of five or not. '**

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

```
if n%5==0:
```

```
print(f'Yes {n} is a multiple of 5')
```

else:

```
print(f'{n} is not a multiple of 5')
```

```
Enter a number 5
Yes 5 is a multiple of 5
```

**Q.3 Find the greatest among two numbers. If numbers are equal than print “numbers are equal”.**

```
n1=int(input('Enter the first number'))
```

```
n2=int(input('Enter the second number'))
```

if  $n1 > n2$ :

```
print(f'{n1} is Greater than {n2}')
```

elif  $n1 < n2$ :

```
print(f'{n2} is Greater than {n1}')
```

else:

```
print(f'both are equal')
```

```
Enter the first number 4
Enter the second number 4
both are equal
```

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

```
n1=int(input('Enter the First number'))
```

```
n2=int(input('Enter the second number'))
```

```
n3=int(input('Enter the third number'))
```

if  $n1 > n2$  and  $n1 > n3$ :

```

    print(f"{n1} is the Greater than {n2} and {n3}")
elif n2>n1 and n2>n3:
    print(f"{n2} is the Greater than {n1} and {n3}")
else:
    print(f"{n3} is the Greater than {n1} and {n2}")

```

```

Enter the First number 5
Enter the second number 7
Enter the third number 8
8 is the Greater than 5 and 7

```

**'Q.5 Check whether the quadratic equation has real roots or imaginary roots. Display the roots''**

```
import math
```

```
def quadratic_roots(a, b, c):
```

```
    discriminant = b ** 2 - 4 * a * c
```

```
    if discriminant > 0:
```

```
        root1 = (-b + math.sqrt(discriminant)) / (2 * a)
```

```
        root2 = (-b - math.sqrt(discriminant)) / (2 * a)
```

```
        return "Two distinct real roots:", root1, root2
```

```
    elif discriminant == 0:
```

```
        root = -b / (2 * a)
```

```
        return "One real root (repeated):", root

```

**else:**

```
real_part = -b / (2 * a)  
imaginary_part = math.sqrt(abs(discriminant)) / (2 * a)  
root1 = complex(real_part, imaginary_part)  
root2 = complex(real_part, -imaginary_part)  
return "Two imaginary roots:", root1, root2
```

```
a = float(input('Enter the coefficient of x^2 (a): '))
```

```
b = float(input('Enter the coefficient of x (b): '))
```

```
c = float(input('Enter the constant term (c): '))
```

```
roots = quadratic_roots(a, b, c)
```

```
print("Roots of the quadratic equation:")
```

```
for root in roots:
```

```
    print(root)
```

```
Enter the coefficient of x^2 (a): 5  
Enter the coefficient of x (b): 6  
Enter the constant term (c): a
```

**Q.6 Find whether a given year is a leap year or not."**

```
def is_leap_year(year):
```

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

```
    return True
```

```
else:
```

```
    return False
```

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

```
if is_leap_year(year):
```

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

```
else:
```

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

```
Enter a year: 2037
2037 is not a leap year.
```

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

**calendar**

**e.g.**

**I/P: day=20 month=9 year=2005**

**O/P: day=21 month=9 year 2005'''**

```
def is_leap_year(year):
```

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

```
        return True
```

**else:**

**return False**

**def next\_date(day, month, year):**

**days\_in\_month = [31, 28 if not is\_leap\_year(year) else 29, 31, 30,  
31, 30, 31, 31, 30, 31, 30, 31]**

**if day < days\_in\_month[month - 1]:**

**day += 1**

**else:**

**day = 1**

**if month < 12:**

**month += 1**

**else:**

**month = 1**

**year += 1**

**return day, month, year**

**day = int(input("Enter the day: "))**

**month = int(input("Enter the month: "))**

**year = int(input("Enter the year: "))**

```
next_day, next_month, next_year = next_date(day, month, year)
```

```
print("Next date:", "day =", next_day, "month =", next_month,  
      "year =", next_year)
```

```
Enter the day: 5  
Enter the month: 5  
Enter the year: 2005  
Next date: day = 6 month = 5 year = 2005
```

**Q.8 Print the grade sheet of a student for the given range of cgpa.**

**Scan marks of five**

**subjects and calculate the percentage.**

**CGPA=percentage/10**

**CGPA range:**

**0 to 3.4 -> F**

**3.5 to 5.0->C+**

**5.1 to 6->B**

**6.1 to 7-> B+**

**7.1 to 8-> A**

**8.1 to 9->A+**

**9.1 to 10-> O (Outstanding)**

**Sample Gradesheet**

**Name: Rohit Sharma**

**Roll Number: R17234512      SAPID: 50005673**

**Sem: 1**

**Course: B.Tech. CSE AI&ML**

**Subject name: Marks**

**PDS: 70**

**Python: 80**

**Chemistry: 90**

**English: 60**

**Physics: 50**

**Percentage: 70%**

**CGPA:7.0**

**Grade: ''**

**maths=int(input("MATHS (100)="))**

**chem=int(input("CHEMISTRY (100)="))**

**phy=int(input("PHYSICS (100)="))**

**eng=int(input("ENGLISH (100)="))**

**bio=int(input("BIOLOGY (100)="))**

**per=float((((maths+chem+phy+eng+bio)/500)\*100))**

**cgpa=per/10**

**if (cgpa>=0) and (cgpa<=3.4):**

**g='F'**

**elif (cgpa>=3.5) and (cgpa<=5):**

**g='C+'**

**elif (cgpa>=5.1) and (cgpa<=6):**

**g='B'**



**elif (cgpa>=6.1) and (cgpa<=7):**

**g='B+'**

**elif (cgpa>=7.1) and (cgpa<=8):**

**g='A'**

**elif (cgpa>=8.1) and (cgpa<=9):**

**g='A+'**

**elif (cgpa>=9.1) and (cgpa<=10):**

**g='O'**

**else:**

**print("Entered Wrong information")**

**name=input("NAME: ")**

**roll,sap=input("Roll Number: "),input("SAP ID: ")**

**sem,cou=input("Sem: "),input("Course: ")**

**print(f"NAME: {name}")**

**print("Roll Number: "+roll+"\tSAP ID: "+sap)**

**print("Semester: " + sem + "\tCourse: " + cou)**

**print(f"Subject Name:\t Marks ")**

**print(f"CHEMISTRY:\t {chem}")**

**print(f"PHYSICS:\t {phy}")**

**print(f"ENGLISH:\t {eng}")**

**print(f"MATHS:\t\t {maths}")**

**print(f"BIOLOGY:\t {bio}")**

```
print(f"PERCENTAGE:\t {per:.2f}%")
```

```
print(f"CGPA:\t {cgpa}")
```

```
print(f"GRADE:\t {g}")
```

```
MATHS (100)= 2
CHEMISTRY (100)= 3
PHYSICS (100)= 88
ENGLISH (100)= 99
BIOLOGY (100)= 0
NAME: Mehendak Pratap Chanewala
Roll Number: R000000000001
SAP ID: 50000009
Sem: 1St
Course: b.tech cse
NAME: Mehendak Pratap Chanewala
Roll Number: R000000000001 SAP ID: 50000009
Semester: 1St Course: b.tech cse
Subject Name: Marks
CHEMISTRY: 3
PHYSICS: 88
ENGLISH: 99
MATHS: 2
BIOLOGY: 0
PERCENTAGE: 38.40%
CGPA: 3.84
GRADE: C+
```

## EXPERIMENT-4

Q.1 Find a factorial of given number ?"

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

```
fact = 1
```

```
if n >= 1:
```

```
    for i in range(1, n+1):
```

```
fact = fact * i
```

```
print("Factorial of the given number is: ", fact)
```

```
Enter a number: 7
Factorial of the given number is: 5040
```

Q.2 Find whether the given number is Armstrong number ?"

```
def is_armstrong(num):
```

```
    num_str = str(num)
```

```
    num_digits = len(num_str)
```

```
    sum_of_digits = sum(int(digit) ** num_digits for digit in num_str)
```

```
    return sum_of_digits == num
```

```
number = int(input("Enter a number to check if it's an Armstrong number: "))
```

```
if is_armstrong(number):
```

```
    print(number, "is an Armstrong number.")
```

```
else:
```

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

```
Enter a number to check if it's an Armstrong number: 3456
3456 is not an Armstrong number.
```

Q.3 Print Fibonacci series up to given term ?"

```
def fibonacci_series(n):
```

```
    fib_series = [0, 1]
```

```
    while len(fib_series) < n:
```

```

    next_term = fib_series[-1] + fib_series[-2]

    fib_series.append(next_term)

return fib_series

n = int(input("Enter the number of terms for the Fibonacci series: "))

print("Fibonacci series up to", n, "terms:")

print(fibonacci_series(n))

```

```

Enter the number of terms for the Fibonacci series: 56
Fibonacci series up to 56 terms:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817, 39088169, 63245986, 102334155, 165580141, 267914296, 433494437, 701408733, 1134903170, 1836311903, 2971215073, 4807526976, 7778742049, 12586269025, 20365011074, 32951280099, 53316291173, 86267571272, 139583862445]

```

Q.5 Check whether given number is palindrome or not ?"

```

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

temp=num

rev=0

while(num>0):

    dig=num%10

    rev=rev*10+dig

    num=num//10

if(temp==rev):

    print("It is a palindrome!")

else:

    print("It is not a palindrome!")

```

```

Enter a number: 76
It is a palindrome!

```

Q.6 Write a program to print sum of digits ?"

```

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

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

```

```
print(a+b)
```

```
Enter a:  2
Enter b:  4
6
```

Q.7 Count and print all numbers divisible by 5 or 7 between 1 to 100 ?"

```
count = 0
```

```
for i in range(1, 101):
```

```
    if i % 5 == 0 or i % 7 == 0:
```

```
        print(i)
```

```
        count += 1
```

```
print("Total count: ", count)
```

```
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
Total count: 32
```

Q.8 Convert all lower cases to upper case in a string ?"

```
s = "python classes"
```

```
s_upper = s.upper()
```

```
print(s_upper)
```

## PYTHON CLASSES

Q.9 Print all prime numbers between 1 and 100 ?"

```
for num in range(2, 101):
```

```
    is_prime = True
```

```
    for i in range(2, num):
```

```
        if num % i == 0:
```

```
            is_prime = False
```

```
            break
```

```
    if is_prime:
```

```
        print(num)
```

```
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
```

Q.10 Print the table for a given number:

$5 * 1 = 5$

$5 * 2 = 10.....$

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

```
for i in range(1, 11):
```

```
    product = num * i
```

```
    print(f"{num} x {i} = {product}")
```

```
Enter a number: 89
89 x 1 = 89
89 x 2 = 178
89 x 3 = 267
89 x 4 = 356
89 x 5 = 445
89 x 6 = 534
89 x 7 = 623
89 x 8 = 712
89 x 9 = 801
89 x 10 = 890
```

## EXPERIMENT-5



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

```
def count_and_display_capital_letters(input_string):  
    capital_count = sum(1 for char in input_string if char.isupper())  
    capital_letters = ''.join(char for char in input_string if  
char.isupper())  
    print('Number of capital letters in the string:', capital_count)  
    print('Capital letters in the string:', capital_letters if  
capital_letters else "None")  
input_string = input('Enter a string: ')  
count_and_display_capital_letters(input_string)
```

```
Enter a string: ADGYljiefnbsh  
Number of capital letters in the string: 4  
Capital letters in the string: ADGY
```

**Q.2 Count total number of vowels in a given string. '''**

```
def count_vowels(string):  
    vowels = "aeiouAEIOU"  
    count = 0  
    for char in string:  
        if char in vowels:  
            count += 1  
    return count  
string = "Hello World"  
print('Total number of vowels:', count_vowels(string))
```

```
Total number of vowels: 3
```

**Q.3 Input a sentence and print words in separate lines.'**

```
def print_words(sentence):  
    words = sentence.split()  
    for word in words:  
        print(word)  
  
sentence = input("Enter a sentence: ")  
print("Words in separate lines:")  
print_words(sentence)
```

**Q.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.**

**Sample Input**

**ABCD CDC**

**CDC**

**Sample Output**

**2**

```
def count_substring(string, substring):  
    count = 0  
    length = len(substring)
```

```
for i in range(len(string)):  
    if string[i:i+length] == substring:  
        count += 1  
  
return count
```

**'Q.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.**

**Sample Input**

**ABaBCbGc**

**Sample Output**

**2A**

**3B**

**2C**

**1G'''**

```
def count_alphabets(string):  
    counts = {}  
    for char in string:  
        if char.isalpha():  
            char = char.upper()  
            counts[char] = counts.get(char, 0) + 1
```

```
for char, count in sorted(counts.items()):  
    print(f'{count}{char}')  
string = "ABaBCbGc"  
count_alphabets(string)
```

```
2A  
3B  
2C  
1G
```

**Q.6 Program to count number of unique words in a given sentence using sets.'**

```
def count_unique_words(sentence):  
    words = sentence.split()  
    unique_words = set(words)  
    return len(unique_words)  
  
sentence = "This is a sample sentence with repeated words. This  
sentence has repeated words."  
  
print("Number of unique words:", count_unique_words(sentence))
```

```
Number of unique words: 9
```

**Q.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"**

```
def main():
```

```
n = int(input('Enter the number of fruits for each set: '))
```

```
print('Enter fruits for set s1:')
```

```
s1 = set(input_fruits(n))
```

```
print('Enter fruits for set s2:')
```

```
s2 = set(input_fruits(n))
```

```
fruits_in_both = s1.intersection(s2)
```

```
fruits_only_in_s1 = s1.difference(s2)
```

```
count_all_fruits = len(s1) + len(s2)
```

```
print('Fruits which are in both sets s1 and s2:', fruits_in_both)
```

```
print('Fruits only in s1 but not in s2:', fruits_only_in_s1)
```

```
print('Count of all fruits from s1 and s2:', count_all_fruits)
```

```
def input_fruits(n):
```

```
    fruits = []
```

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

```
        fruit = input(f'Enter fruit {i+1}: ')
```

```
        fruits.append(fruit)
```

```
    return fruits
```

```
if __name__ == "__main__":
```

```
    main()
```

**Q.8 Take two sets and apply various set operations on them :**

**S1 = {Red ,yellow, orange , blue }**

**S2 = {violet, blue , purple}'''**

**def main():**

**S1 = {"Red", "yellow", "orange", "blue"}**

**S2 = {"violet", "blue", "purple"}**

**union = S1.union(S2)**

**intersection = S1.intersection(S2)**

**difference\_S1\_S2 = S1.difference(S2)**

**difference\_S2\_S1 = S2.difference(S1)**

**print("Union of S1 and S2:", union)**

**print("Intersection of S1 and S2:", intersection)**

**print("Difference of S1 from S2:", difference\_S1\_S2)**

**print("Difference of S2 from S1:", difference\_S2\_S1)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

```
Union of S1 and S2: {'blue', 'purple', 'orange', 'violet', 'yellow', 'Red'}
Intersection of S1 and S2: {'blue'}
Difference of S1 from S2: {'yellow', 'Red', 'orange'}
Difference of S2 from S1: {'purple', 'violet'}
```

## **EXPERIMENT-6**

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

**def count\_occurrences(n):**

**counts = {0: 0, 1: 0, 2: 0, 3: 0}**

**for \_ in range(n):**

**value = int(input(f'Enter value ({0} - {3}): '))**

**if value in counts:**

**counts[value] += 1**

**for value, count in counts.items():**

**print(f'Value {value} occurred {count} times.')**

**n = int(input('Enter the number of values: '))**

**count\_occurrences(n)**

```
Enter the number of values: 5
Enter value (0 - 3): 2
Enter value (0 - 3): 1
Enter value (0 - 3): 3
Enter value (0 - 3): 4
Enter value (0 - 3): 5
Value 0 occurred 0 times.
Value 1 occurred 1 times.
Value 2 occurred 1 times.
Value 3 occurred 1 times.
```

**Q.2 Create a tuple to store n numeric values and find average of all values. '''**

**def calculate\_average(num\_values):**

**values = tuple(float(input(f'Enter value {i + 1}: ')) for i in  
range(num\_values))**

**total = sum(values)**

**average = total / num\_values**

**return average**

**n = int(input('Enter the number of values: '))**

**avg = calculate\_average(n)**

**print('Average of all values:', avg)**



```
Enter the number of values: 4
Enter value 1: 1
Enter value 2: 2
Enter value 3: 3
Enter value 4: 4
Average of all values: 2.5
```

**Q.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.**

**Sample Input**

**N = 5**

**Scores= 2 3 6 6 5**

**Sample output**

**5**

**Note: Given list is [2, 3, 6, 6, 5]. The maximum score is 6, second maximum is 5.**

**Hence, we print 5 as the runner-up score.'''**

**def find\_runner\_up\_score(scores):**

**sorted\_scores = sorted(scores, reverse=True)**

**max\_score = sorted\_scores[0]**

**runner\_up\_score = max(s for s in sorted\_scores if s < max\_score)**

**return runner\_up\_score**

**N = int(input("Enter the number of students: "))**

```
scores = list(map(int, input("Enter the scores separated by space: ").split()))
```

```
runner_up_score = find_runner_up_score(scores)
```

```
print("The runner-up score is:", runner_up_score)
```

```
Enter the number of students: 2
Enter the scores separated by space: 13
```

**Q.4 Create a dictionary of n persons where key is name and value is city.**

**a) Display all names**

**b) Display all city names**

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

**d) Count number of students in each city.'**

```
def create_persons_dictionary(n):
```

```
    persons = {}
```

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

```
        name = input(f"Enter name of person {i + 1}: ")
```

```
        city = input(f"Enter city of person {i + 1}: ")
```

```
        persons[name] = city
```

```
    return persons
```

```
def display_all_names(persons):
```

```
    print("All names:")
```

```
    for name in persons.keys():
```

```
print(name)
```

```
def display_all_cities(persons):
```

```
    print("All cities:")
```

```
    for city in set(persons.values()):
```

```
        print(city)
```

```
def display_persons_info(persons):
```

```
    print("Name and city of all persons:")
```

```
    for name, city in persons.items():
```

```
        print(f"Name: {name}, City: {city}")
```

```
def count_students_in_each_city(persons):
```

```
    city_counts = {}
```

```
    for city in persons.values():
```

```
        city_counts[city] = city_counts.get(city, 0) + 1
```

```
    print("Number of students in each city:")
```

```
    for city, count in city_counts.items():
```

```
        print(f"{city}: {count}")
```

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

```
persons = create_persons_dictionary(n)
```

```
display_all_names(persons)
```

**display\_all\_cities(persons)**

**display\_persons\_info(persons)**

**count\_students\_in\_each\_city(persons)**

```
Enter the number of persons: 3
Enter name of person 1: sdfgh
Enter city of person 1: asdf
Enter name of person 2: asdf
Enter city of person 2: asdf
Enter name of person 3: sdf
Enter city of person 3: asd
All names:
sdfgh
asdf
sdf
All cities:
asd
asdf
Name and city of all persons:
Name: sdfgh, City: asdf
Name: asdf, City: asdf
Name: sdf, City: asd
Number of students in each city:
asdf: 2
asd: 1
```

**Q.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 name of movies released before 2015**

**c) print movies that made a profit.**

**d) print movies directed by a particular director.'**

```
def create_movies_dictionary(n):
```

```
    movies = {}
```

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

```
        name = input(f'Enter name of movie {i + 1}: ')
```

```
        year = int(input(f'Enter year of release for movie {name}: '))
```

```
        director = input(f'Enter director name for movie {name}: ')
```

```
        production_cost = float(input(f'Enter production cost for  
movie {name}: '))
```

```
        collection = float(input(f'Enter collection made for movie  
{name}: '))
```

```
        movies[name] = {'Year': year, 'Director': director, 'Production  
Cost': production_cost, 'Collection': collection}
```

```
    return movies
```

```
def print_all_movie_details(movies):
```

```
    print("All movie details:")
```

```
    for name, details in movies.items():
```

```
        print(f'Name: {name}, Year: {details['Year']}, Director:  
{details['Director']}, Production Cost: {details['Production Cost']},  
Collection: {details['Collection']}')
```

```
def movies_released_before_2015(movies):
```

```
    print("Movies released before 2015:")
```

```
for name, details in movies.items():
```

```
    if details['Year'] < 2015:
```

```
        print(name)
```

```
def movies_with_profit(movies):
```

```
    print('Movies that made a profit:')
```

```
    for name, details in movies.items():
```

```
        if details['Collection'] > details['Production Cost']:
```

```
            print(name)
```

```
def movies_by_director(movies, director):
```

```
    print(f'Movies directed by {director}:')
```

```
    for name, details in movies.items():
```

```
        if details['Director'] == director:
```

```
            print(name)
```

```
n = int(input('Enter the number of movies: '))
```

```
movies = create_movies_dictionary(n)
```

```
print_all_movie_details(movies)
```

```
movies_released_before_2015(movies)
```

```
movies_with_profit(movies)
```

```
director = input('Enter the director name to filter movies: ')
```

**movies\_by\_director(movies, director)**

```
Enter the number of movies: 2
Enter name of movie 1:  godzilla
Enter year of release for movie godzilla:  2019
Enter director name for movie godzilla:  systum
Enter production cost for movie godzilla:  123 MILLIONS
```

## **Experiment-7**

**Q.1 Write a Python function to find the maximum and minimum numbers from a sequence of numbers. (Note: Do not use built-in functions.) '''**

**def find\_max\_min(sequence):**

**max\_num = sequence[0]**

**min\_num = sequence[0]**

**for num in sequence:**

**if num > max\_num:**

**max\_num = num**

**if num < min\_num:**

**min\_num = num**

**return max\_num, min\_num**

**sequence = [5, 3, 9, 1, 7]**

**max\_num, min\_num = find\_max\_min(sequence)**

```
print("Maximum number:", max_num)
```

```
print("Minimum number:", min_num)
```

```
Maximum number: 9  
Minimum number: 1
```

**Q.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_of_cube(n):
```

```
    sum_cubes = 0
```

```
    for i in range(1, n):
```

```
        sum_cubes += i ** 3
```

```
    return sum_cubes
```

```
n = int(input("Enter a positive integer: "))
```

```
result = sum_of_cube(n)
```

```
print("Sum of the cubes of positive integers smaller than", n, "is:",  
result)
```

```
Enter a positive integer: 2  
Sum of the cubes of positive integers smaller than 2 is: 1
```

**Q.3 Write a Python function to print 1 to n using recursion. (Note: Do not use loop)''**



```
def print_numbers(n):
```

```
    if n > 0:
```

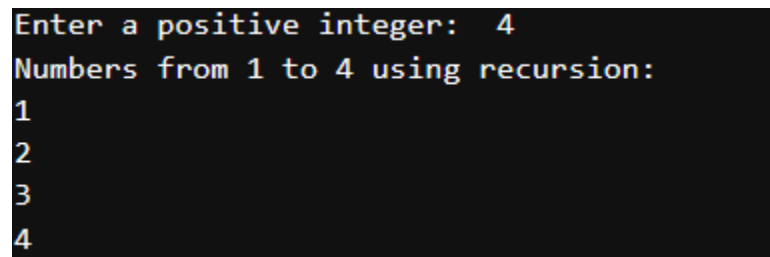
```
        print_numbers(n - 1)
```

```
        print(n)
```

```
n = int(input('Enter a positive integer: '))
```

```
print('Numbers from 1 to', n, 'using recursion:')
```

```
print_numbers(n)
```



```
Enter a positive integer: 4
Numbers from 1 to 4 using recursion:
1
2
3
4
```

**'Q.4 Write a recursive function to print Fibonacci series upto n terms.'**

```
def fibonacci(n, a=0, b=1):
```

```
    if n > 0:
```

```
        print(a, end=" ")
```

```
        fibonacci(n - 1, b, a + b)
```

```
n = int(input('Enter the number of terms for the Fibonacci series: '))
```

```
print('Fibonacci series up to', n, 'terms:')
```

```
fibonacci(n)
```

```
Enter the number of terms for the Fibonacci series: 2
Fibonacci series up to 2 terms:
0 1
```

**Q.5 Write a lambda function to find volume of cone. '''**

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

```
radius = float(input("Enter the radius of the cone: "))
```

```
height = float(input("Enter the height of the cone: "))
```

```
volume = cone_volume(radius, height)
```

```
print("Volume of the cone:", volume)
```

```
Enter the radius of the cone: 4
Enter the height of the cone: 5
Volume of the cone: 83.77573333333332
```

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

**Sample input: [10, 6, 8, 90, 12, 56]**

**Sample output: (90,6) '''**

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

```
input_list = [10, 6, 8, 90, 12, 56]
```

```
result = max_min_tuple(input_list)
```

```
print("Result:", result)
```

```
Result: (90, 6)
```

**'Q.7 Write functions to explain mentioned concepts:**

**a. Keyword argument**

## **b. Default argument**

## **c. Variable length argument ''**

```
def greet(name, message):
```

```
    print(f'{message}, {name}!')
```

```
greet(name="Alice", message="Hello")
```

```
Hello, Alice!
```

```
def greet(name, message="Hello"):
```

```
    print(f'{message}, {name}!')
```

```
greet("Alice")
```

```
Hello, Alice!
```

```
def sum_values(*args):
```

```
    return sum(args)
```

```
result = sum_values(1, 2, 3, 4, 5)
```

```
print("Result:", result)
```

```
Result: 15
```

## **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}")
```

```
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

**'Q.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("Physics Marks:", self.marks[0])**

**print("Chemistry Marks:", self.marks[1])**

**print("Maths Marks:", self.marks[2])**

**print()**

**students = []**

**for i in range(3):**

**print(f"Enter details for student {i + 1}:")**

**name = input("Enter student's name: ")**

**sap\_id = input("Enter student's SAP ID: ")**

**phy\_marks = float(input("Enter Physics marks: "))**

**chem\_marks = float(input("Enter Chemistry marks: "))**

```

math_marks = float(input("Enter Maths marks: "))
marks = [phy_marks, chem_marks, math_marks]
students.append(Student(name, sap_id, marks))

print("\nDetails of all students:")

for student in students:
    student.display_details()

```

**Q.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.**

**3. Create programs to implement different types of inheritances.**

**4. Create a class to implement method Overriding.**

**5. Create a class for operator overloading which adds two Point Objects where Point**

**has x & y values**

**e.g. if**

**P1(x=10,y=20)**

**P2(x=12,y=15)**

**P3=P1+P2 => P3(x=22,y=35)'''**

**class Student:**

**def \_\_init\_\_(self, name, roll\_no, marks):**

**self.name = name**

**self.roll\_no = roll\_no**

**self.marks = marks**

**def display(self):**

**print("Name:", self.name)**

**print("Roll No:", self.roll\_no)**

**print("Marks:", self.marks)**

**def marks\_percentage(self):**

**total\_marks = sum(self.marks)**

**percentage = (total\_marks / (len(self.marks) \* 100)) \* 100**

**return percentage**

**def display\_result(self):**

**percentage = self.marks\_percentage()**

**if all(mark >= 40 for mark in self.marks):**

**print("Result: Pass")**

**else:**



```
print("Result: Fail")
```

```
def class_average(students):
```

```
    total_percentage = sum(student.marks_percentage() for student  
    in students)
```

```
    average_percentage = total_percentage / len(students)
```

```
    return average_percentage
```

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

```
students = []
```

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

```
    name = input("Enter student's name: ")
```

```
    roll_no = input("Enter student's roll number: ")
```

```
    marks = [int(x) for x in input("Enter student's marks separated  
by space: ").split()]
```

```
    students.append(Student(name, roll_no, marks))
```

```
print("\nStudent Details:")
```

```
for student in students:
```

```
    student.display()
```

```
    print()
```

```
print("\nStudent Results:")
```

**for student in students:**

**student.display\_result()**

**print()**

**average = class\_average(students)**

**print("\nClass Average Marks Percentage:", average)**

```
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()
```

```
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
```

