COMPUTER PROGRAMMING – II LABORATORY

NAME – OM SHRIVASTAVA

ROLL NUMBER – 24BEE083

* LAB-1:

1. Aim- Write a program to take 2 variables and arithmetic operations on it.

print("Om Shrivastava")

print("24BEE083")

P = int(input("Enter the first number:"))

Q = int(input("Enter the second number:"))

A = P + Q

print("addition =",A)

B = P - Q

print("subtraction =",B)

C = P \* Q

print("multiplication =",C)

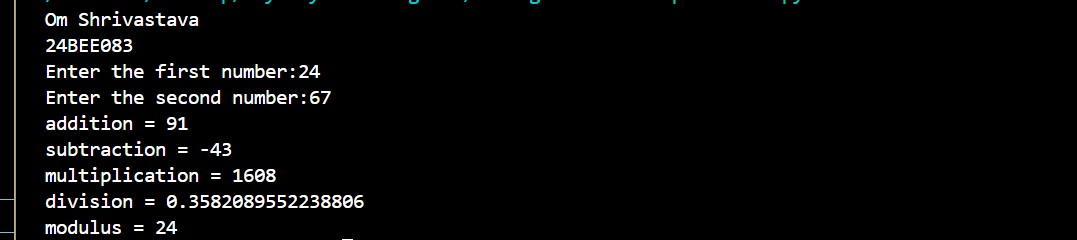
D = P / Q

print("division =",D)

E = P % Q

print("modulus =",E)

OUTPUT:



2. Aim- Write a program to take marks of a student in 5 subjects and compute average.

print("Om Shrivastava")

print("24BEE083")

A = int(input("Marks in Maths ="))

B = int(input("Marks in English ="))

C = int(input("Marks in Computer ="))

D = int(input("Marks in Science ="))

E = int(input("Marks in Economics ="))

average = ((A+B+C+D+E)/5)

print("Average marks of student is:", average)

OUTPUT 

3. Aim- Write a program to convert a temperature from degree celsius to degree fahrenheit.

**print("Om Shrivastava")**

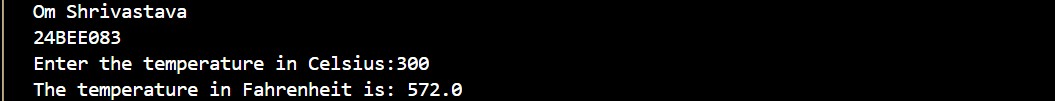
**print("24BEE083")**

**C = int(input("Enter the temperature in Celsius:"))**

**fahrenheit = ((9/5\*C)+32)**

**print("The temperature in Fahrenheit is:", fahrenheit)**

* OUTPUT:



4. Aim- Write a program to compute area and perimeter of a circle.

**print("Om Shrivastava")**

**print("24BEE083")**

**R = int(input("Enter radius of the circle:"))**

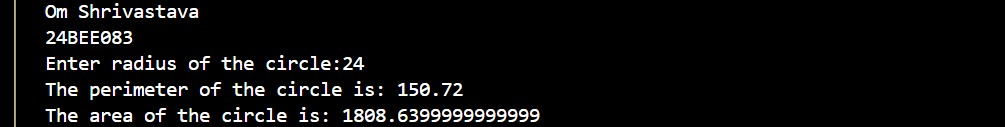
**P = 2\*3.14\*R**

**A = 3.14\*R\*R**

**print("The perimeter of the circle is:",P)**

**print("The area of the circle is:",A)**

* OUTPUT:



5. Aim- Write a program to convert an amount from one currency to another.

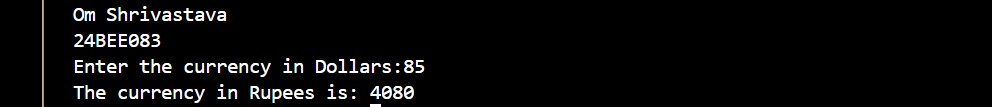
**print("Om Shrivastava")**

**print("24BEE083")**

**A = int(input("Enter the currency in Dollars:"))**

**B = A\*48**

**print("The currency in Rupees is:",B)**

OUTPUT: 

6. Aim- Write a program to swap 2 values.

**print(“Om Shrivastava”)**

**print(“24BEE083”)**

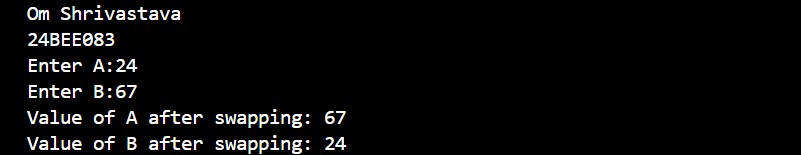
**A = int(input("Enter A:"))**

**B = int(input("Enter B:"))**

**A,B = B,A**

**print("Value of A after swapping:",A)**

**print("Value of B after swapping:",B)**

* OUTPUT: 

7. Aim- Print your Roll number, Name, E-Mail and Mobile Number in a certain format.

**print("Om Shrivastava")**

**print("24BEE083")**

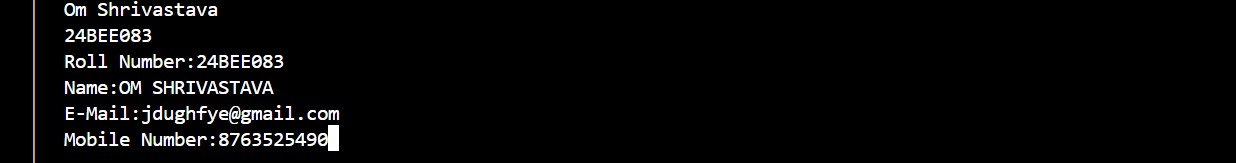
**input("Roll Number:")**

**input("Name:")**

**input("E-Mail:")**

**input("Mobile Number:")**

* OUTPUT:



* LAB-2:

1. Aim- Print largest and smallest values out of three.

**print("Om Shrivastava")**

**print("24BEE083")**

**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 largest", num1)**

**elif(num2>num1 and num2>num3):**

**print("num2 is largest", num2)**

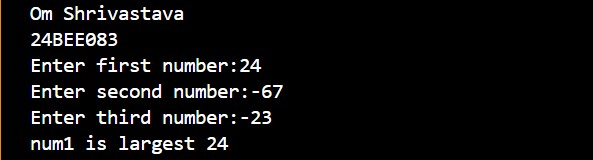
**elif(num1==num2 and num2==num3):**

**print("All are equal.")**

**else:**

**print("num3 is largest", num3)**

OUTPUT:

2. Aim- Check whether a given number is odd or even.

**print("Om Shrivastava")**

**print("24BEE083")**

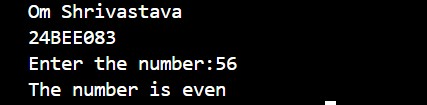
**A = int(input("Enter the number:"))**

**if(A % 2 == 0):**

**print("The number is even")**

**else:**

**print("The number is odd")**

* OUTPUT: 

3. Aim- Check whether a given number is divisible by 10 or not.

**print("Om Shrivastava")**

**print("24BEE083")**

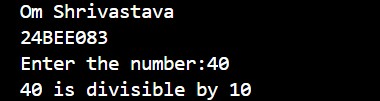
**num = int(input("Enter the number:"))**

**if(num%10 == 0):**

**print(num,"is divisible by 10")**

**else:**

**print(num,"is not divisible by 10")**

* OUTPUT: 

4. Aim- Accept age of a person. If age is less than 18, print minor otherwise Major.

**print("Om Shrivastava")**

**print("24BEE083")**

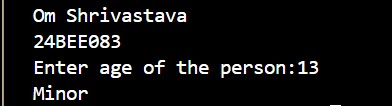
**A = int(input("Enter age of the person:"))**

**if(A<18):**

**print("Minor")**

**else:**

**print("Major")**

* OUTPUT: 

5. Aim- Accept a year value from the user. Check whether it is a leap year or not.

**print("Om Shrivastava")**

**print("24BEE083")**

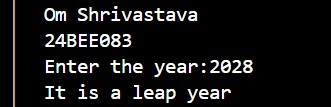
**A = int(input("Enter the year:"))**

**if(A % 4 == 0):**

**print("It is a leap year")**

**else:**

**print("It is not a leap year")**

* OUTPUT: 

6. Aim- Check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees.

**print("Om Shrivastava")**

**print("24BEE083")**

**A = int(input("Enter the first angle:"))**

**B = int(input("Enter the seconf angle:"))**

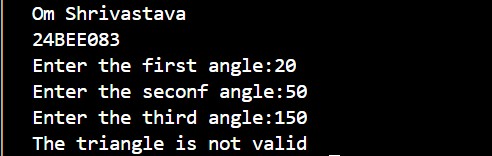
**C = int(input("Enter the third angle:"))**

**if(A + B + C == 180):**

**print("The triangle is valid")**

**else:**

**print("The triangle is not valid")**

OUTPUT: 

7. Aim- Given the length and breadth of a rectangle, write a program to find whether the area of the rectangle is greater than its perimeter.

**print("Om Shrivastava")**

**print("24BEE083")**

**L = int(input("Enter the length of the rectangle:"))**

**B = int(input("Enter the breadth of the rectangle:"))**

**A = L\*B #area of rectangle**

**P = 2\*(L+B) #perimeter of rectangle**

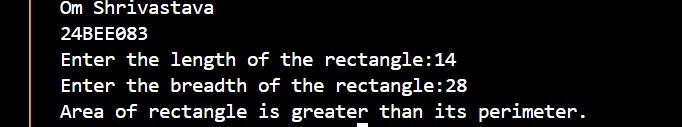
**if(A>P):**

**print("Area of rectangle is greater than its perimeter.")**

**else:**

**print("Area of rectangle is not greater than its perimeter.")**

* OUTPUT:



8. Aim- Given three points (x1,y1), (x2,y2) and (x3,y3), check if all the three points fall on one straight line.

**print("Om Shrivastava")**

**print("24BEE083")**

**x1 = int(input("Enter the x1 coordinate"))**

**y1 = int(input("Enter the y1 coordinate"))**

**x2 = int(input("Enter the x2 coordinate"))**

**y2 = int(input("Enter the y2 coordinate"))**

**x3 = int(input("Enter the x3 coordinate"))**

**y3 = int(input("Enter the y3 coordinate"))**

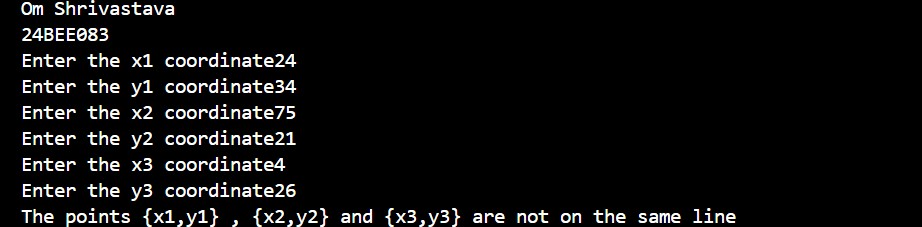
**if(x1+x2 == x3):**

**print("The points {x1,y1} , {x2,y2} and {x3,y3} are on the same line")**

**else:**

**print("The points {x1,y1} , {x2,y2} and {x3,y3} are not on the same line")**

* OUTPUT:



9. Aim- Given the coordinates (x,y) of center of a circle and its radius, determine whether a point lies inside the circle, on the circle or outside the circle.

**print("Om Shrivastava")**

**print("24BEE083")**

**import math as m**

**x = int(input("Enter the x coordinate:"))**

**y = int(input("Enter the y coordinate:"))**

**print("Enter the coordinates of the origin:")**

**x1 = int(input("Enter the x1 coordinate:"))**

**y1 = int(input("Enter the y1 coordinate:"))**

**rad = int(input("Enter the radius of the circle:"))**

**c = x-x1**

**d = y-y1**

**f = m.pow(c,2)**

**g = m.pow(d,2)**

**l = f+g**

**dis = m.sqrt(l)**

**if(dis == rad):**

**print("the point is on the circle")**

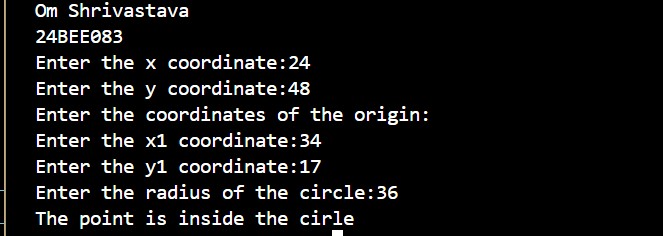
**elif(dis>rad):**

**print("the point is outside the circle")**

**else:**

**print("The point is inside the cirle")**

* OUTPUT:



10. Aim- Convert number 0 to 19 to its equivalent words. E.g. 0 is zero, 19 is nineteen.

**print("Om Shrivastava")**

**print("24BEE083")**

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

**if n == 0:**

**print("zero")**

**elif n == 1:**

**print("one")**

**elif n == 2:**

**print ("two")**

**elif n == 3:**

**print ("three")**

**elif n == 4:**

**print ("four")**

**elif n == 5:**

**print ("five")**

**elif n == 6:**

**print("six")**

**elif n == 7:**

**print("seven")**

**elif n == 8:**

**print ("eight")**

**elif n == 9:**

**print ("nine")**

**elif n == 10:**

**print("ten")**

**elif n == 11:**

**print("eleven")**

**elif n == 12:**

**print("twelve")**

**elif n == 13:**

**print("thirteen")**

**elif n == 14:**

**print("fourteen")**

**elif n == 15:**

**print("fifteen")**

**elif n == 16:**

**print("sixteen")**

**elif n == 17:**

**print("seventeen")**

**elif n == 18:**

**print("eighteen")**

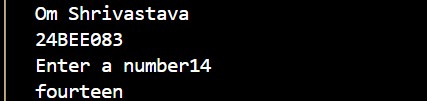
**elif n == 19:**

**print("nineteen")**

**else:**

**print("Number out of range (0-19 only).")**

* OUTPUT:



11. Aim- Accept marks of three subjects. Print total and average along with whether a candidate has passed or fail. If student secures <= 39 marks in any subject, consider him as fail. Also assigned a subject wise grade based on the following table:

Marks Range Grade

Absent NA

0 – 39 F

40 – 44 P

45 – 49 C

50 – 54 B

55 – 59 B+

60 – 69 A

70 – 79 A+

80 – 100 O

**print("Om Shrivastava")**

**print("24BEE083")**

**m1=int(input("Enter marks of Subject1: "))**

**m2=int(input("Enter marks of Subject2:"))**

**m3=int(input("Enter marks of Subject3: "))**

**print("The total marks is",(m1+m2+m3))**

**print("The average marks is",(m1+m2+m3)/3)**

**if (m1<=39):**

**print("Grade in Subject1: Fail")**

**elif (m1>=40 and m1<=44):**

**print("Grade in Subject1: Pass")**

**elif (m1>=45 and m1<=49):**

**print("Grade in Subject1: C")**

**elif (m1>=50 and m1<=54):**

**print("Grade in Subject1: B")**

**elif (m1>=55 and m1<=59):**

**print("Grade in Subject1: B+")**

**elif (m1>=60 and m1<=69):**

**print("Grade in Subject1: A")**

**elif (m1>=70 and m1<=79):**

**print("Grade in Subject1: A+")**

**elif (m1>=80):**

**print("Grade in Subject1: O")**

**if (m2<=39):**

**print("Grade in Subject2: Fail")**

**elif (m2>=40 and m2<=44):**

**print("Grade in Subject2: Pass")**

**elif (m2>=45 and m2<=49):**

**print("Grade in Subject2: C")**

**elif (m2>=50 and m2<=54):**

**print("Grade in Subject2: B")**

**elif (m2>=55 and m2<=59):**

**print("Grade in Subject2: B+")**

**elif (m2>=60 and m2<=69):**

**print("Grade in Subject2: A")**

**elif (m2>=70 and m2<=79):**

**print("Grade in Subject2: A+")**

**elif (m2>=80):**

**print("Grade in Subject2: O")**

**if (m3<=39):**

**print("Grade in Subject3: Fail")**

**elif (m3>=40 and m3<=44):**

**print("Grade in Subject3: Pass")**

**elif (m3>=45 and m3<=49):**

**print("Grade in Subject3: C")**

**elif (m3>=50 and m3<=54):**

**print("Grade in Subject3: B")**

**elif (m3>=55 and m3<=59):**

**print("Grade in Subject3: B+")**

**elif (m3>=60 and m3<=69):**

**print("Grade in Subject3: A")**

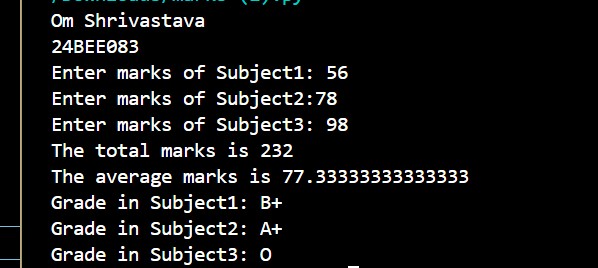
**elif (m3>=70 and m3<=79):**

**print("Grade in Subject3: A+")**

**elif (m3>=80):**

**print("Grade in Subject3: O")**

* OUTPUT:



LAB 3 PROGRAMS

Q1 COUNT HOW MANY VOWELS ARE THERE IN A STRING ACCEPT THE STRING FROM THE USER

INPUT:

print("om shrivastav")

print("24BEE083")

a = input("Enter a string")

a.lower()

b=len(a)

c=0

for i in range (b):

if(a[i]=='a' or a[i]=='e' or a[i]=='o' or a[i]=='u' or a[i]=='i' ):

c+=1

print(f"The number of vowels are : {c}")

OUTPUT:

om shrivastav

24BEE083

Enter a stringom

The number of vowels are : 1

PS C:\Users\OM>

Q2 WRITE YOUR OWN FUNCTION ( WITHOUT USING BUILT IN FUNCTIONS) TO CONVERT ALL CHARACTERS INTO LOWER CASE/ UPPER CASE/ TOGGLE CASE

INPUT:

print("om shrivastav")

print("24BEE083")

a = input("Enter a string")

def lower(b):

result = ""

for i in b:

if('A'<=i<='Z'):

result+= chr(ord(i)+32)

else:

result+=i

print(f"Lowercase ={result}")

lower(a)

def upper(b):

result = ""

for i in b:

if('a'<=i<='z'):

result+= chr(ord(i)-32)

else:

result+=i

print(f"Uppercase ={result}")

upper(a)

def toggle(b):

result = ""

for i in b:

if('a'<=i<='z'):

result+= chr(ord(i)-32)

elif('A'<=i<='Z'):

result+=chr(ord(i)+32)

else:

result+=i

print(f"Toggle case ={result}")

toggle(a)

OUTPUT:

om shrivastav

24BEE083

Enter a stringABcde

Lowercase =abcde

Uppercase =ABCDE

Toggle case =abCDE

Q3 ACCEPT TWO STRINGS CHECK WHEATHER ONE STRING IS THERE IN ANOTHER STRING

INPUT:

print("om shrivastav")

print("24BEE083")

a = input("Enter the first string")

b= input("Enter the second string")

isthere= False

for i in a:

if i in b :

isthere=True

if isthere:

print("The string is present in other")

else:

print("The string is not present in another")

OUTPUT:

om shrivastav

24BEE083

Enter the first stringpdeu

Enter the second stringsoet

The string is present in other

LAB 4: LOOPS

Q1 PRINT ALL ALPHABETS IN UPPER AND LOWER CASE

INPUT:

print("om shrivastav")

print("24BEE083")

for i in range (65,91):

print(chr(i),end =" ")

print("")

for i in range (97,123):

print(chr(i),end=" ")

OUTPUT:

om shrivastav

24BEE083

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

Q2. PRINT A MULTIPLICATION TABLE OF A GIVEN NUMBER

INPUT:

print("om shrivastav")

print("24BEE083")

a = int(input("Enter the number of which the table you want"))

print(f"The table of {a} is")

for i in range (1,11):

print(a\*i)

OUTPT:  
The table of 5 is

5

10

15

20

25

30

35

40

45

50

Q3. COUNT NO OF ALPHABETS AND NO OF DIGITS IN A STRING

INPUT:

print("om shrivastav")

print("24BEE083")

a = input("Enter a string")

b = len(a)

a.lower()

c=0

d=0

for i in a:

if('a'<= i <= 'z'):

c+=1

else:

d+=1

print(f"The number of alphabets is {c}")

print(f"The number of digits is {d}")

OUTPUT:

om shrivastav

24BEE083

Enter a stringalp23

The number of alphabets is 3

The number of digits is 2

Q4. CHECK WHETHER A GIVEN NUMBER IS A PRIME , IS A PALLINDROME, IS A ARMSTRONG, IS A AUTROMORPHIC

INPUT:

print("om shrivastav")

print("24BEE083")

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

def armstrong(a):

n=0

sum=0

f=a

while(a>0):

n=a%10

sum=(n\*n\*n)+sum

a=a//10

if f==sum:

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

else:

print(f"{f} is not an armstrong number")

armstrong(num)

def prime(a):

c=0

for i in range (1,a+1):

if(a%i==0):

c+=1

if(c==2):

print(f"{a} is a prime number")

else:

print(f"{a} is not a prime number")

prime(num)

def palindrome(a):

f=a

rev=0

n=0

while(a>0):

n=a%10

rev=rev\*10+n

a//=10

if(f==rev):

print(f"{f} is a palindrome number")

else:

print(f"{f} is not a palindrome number")

palindrome(num)

def perfect(a):

sum=0

for i in range (1,a):

if(a%i==0):

sum= sum + i

if(sum==a):

print(f"{a} is a perfect number")

else:

print(f"{a} is not a perfect number")

perfect(num)

def automorphic(a):

square = a\*a

lasttwo = square%100

if(a==lasttwo):

print(f"{a} is an automorphic number")

else:

print(f"{a} is not an automorphic number")

automorphic (num)

OUTPUT:

om shrivastav

24BEE083

Enter a number89

89 is not an armstrong number

89 is a prime number

89 is not a palindrome number

89 is not a perfect number

89 is not an automorphic number

Q5. GENERATE ALL PYTHOGORAS TRIPLETS WITH <=30

INPUT:

print("om shrivastav")

print("24BEE083")

for a in range (1,31):

for b in range (a+1,31):

for c in range (b+1,31):

if((a\*a)+(b\*b)==(c\*c)):

print(f"The pythagorean triplet is {a},{b},{c}")

OUTPUT:  
om shrivastav

24BEE083

The pythagorean triplet is 3,4,5

The pythagorean triplet is 5,12,13

The pythagorean triplet is 6,8,10

The pythagorean triplet is 7,24,25

The pythagorean triplet is 8,15,17

The pythagorean triplet is 9,12,15

The pythagorean triplet is 10,24,26

The pythagorean triplet is 12,16,20

The pythagorean triplet is 15,20,25

The pythagorean triplet is 18,24,30

The pythagorean triplet is 20,21,29

Q6. PRINT 24 HOURS OF A DAY WITH SUITABLE SUFFIXES LIKE AM, PM , MIDNIGHT AND NOON

INPUT:

print("om shrivastav")

print("24BEE083")

def print\_day\_with\_suffixes():

for hour in range(24):

if hour == 0:

print(f"12:00 AM - Midnight")

elif hour == 12:

print(f"12:00 PM - Noon")

elif hour < 12:

print(f"{hour}:00 AM")

else:

print(f"{hour - 12}:00 PM")

print\_day\_with\_suffixes()

OUTPUT:

om shrivastav

24BEE083

12:00 AM - Midnight

1:00 AM

2:00 AM

3:00 AM

4:00 AM

5:00 AM

6:00 AM

7:00 AM

8:00 AM

9:00 AM

10:00 AM

11:00 AM

12:00 PM - Noon

1:00 PM

2:00 PM

3:00 PM

4:00 PM

5:00 PM

6:00 PM

7:00 PM

8:00 PM

9:00 PM

10:00 PM

11:00 PM

Q7. PRINT NCR AND NPR

INPUT:

print("om shrivastav")

print("24BEE083")

n = int(input("Enter the value of n"))

r =int(input("Enter the value of r"))

def nCr(a,b):

fact =1

ract=1

tact=1

for i in range (1,a+1):

fact = fact\*i

for j in range (1,b+1):

ract=ract\*j

for k in range(1,(a-b)+1):

tact=tact\*k

l = ract \*tact

k=fact//l

print(f"The nCr value is {k} ")

nCr(n,r)

def nPr (a,b):

fact =1

tact =1

for i in range (1,a+1):

fact = fact \*i

for j in range (1,(a-b)+1):

tact=tact\*j

f = fact//tact

print(f"The nPr value is {f}")

nPr(n,r)

OUTPUT:

om shrivastav

24BEE083

Enter the value of n3

Enter the value of r2

The nCr value is 3

The nPr value is 6

Q8. PRINT FACTORIAL OF A GIVEN NUMBER

INPUT:

print("om shrivastav")

print("24BEE083")

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

fact =1

for i in range (1,a+1):

fact = fact\*i

print(f"The factorial of number {a} is {fact}")

OUTPUT:

om shrivastav

24BEE083

Enter a number4

The factorial of number 4 is 24

Q9. PRINT N NATURAL NUMBERS IN REVERSE

INPUT:

print("om shrivastav")

print("24BEE083")

a = int(input("Enter the terms upto which you want the series"))

print("The series in reverse is :")

for i in range (a,0,-1):

print(i,end =" ")

OUTPUT:

om shrivastav

24BEE083

Enter the terms upto which you want the series5

The series in reverse is :

5 4 3 2 1

Q10. GENERATE N NUMBERS OF FIBONNACCI SERIES

INPUT:

print("om shrivastav")

print("24BEE083")

a = int (input("Enter the first number"))

b=int(input("Enter the second number"))

n = int (input("Enter the range of the series"))

c=0

while(c<=n):

c=a+b

print(c ,end=" ")

a=b

b=c

c+=1

OUTPUT:

om shrivastav

24BEE083

Enter the first number5

Enter the second number4

Enter the range of the series5

9

Q11. Calculate sin(x); x is a radian value. The formula is as under:  
  
sin 𝑥 = 𝑥 − 𝑥3  
  
3! + 𝑥5  
  
5! − 𝑥7  
  
7! …  
  
(hint: degrees can be converted into radians by 3.14159 / 180.)

INPUT:

print("om shrivastav")

print("24BEE083")

import math

angle = int(input("Enter the value of angle in degrees"))

rad = angle\*(3.14//180)

result=0

n = int(input("Enter the range of the series"))

for i in range (0,n+1):

power = 2\*i+1

term = (((-1)\*\*i)\*(rad\*\*power))/math.factorial(power)

result+=term

print(result)

OUTPUT:

om shrivastav

24BEE083

Enter the value of angle in degrees60

Enter the range of the series10

0.0

0.0

0.0

LAB5 : LISTS

Q1. Create a list of 5 odd integers using random nos. Similarly create a list of 4 even integers using

random nos. Replace the third element of odd integers with a list of 4 even integers. Flattern, sort

and print the list. Provide appropriate message at each stage

INPUT :

print("om shrivastav")

print("24BEE083")

list=[]

def num (l):

n = int(input("Enter the range"))

l= []

c=0

d=0

for i in range (n):

l.append(int(input("Enter the values of the list")))

if(l[i]%2==0):

c+=1

else:

d+=1

print(f"The number of odd numbers is {d}")

print(f"The number of even numbers is{c}")

num(list)

OUTPUT:

om shrivastav

24BEE083

Enter the range5

Enter the values of the list5

Enter the values of the list6

Enter the values of the list7

Enter the values of the list8

Enter the values of the list9

The number of odd numbers is 3

The number of even numbers is2

Q2. Generate 20 random integers and store them in a list. Accept a number from the user and print

position of all occurrences of that number in the list.

INPUT:

print("om shrivastav")

print("24BEE083")

list =[]

def square(l):

l =[]

j=[]

n = int(input("Enter the range"))

for i in range (n):

l.append((int(input("Enter the values of the list"))))

a=l[i]

f=a\*a

#print("The list of square of numbers is")

j.append(f)

print(j)

square(list)

OUTPUT:

om shrivastav

24BEE083

Enter the range5

Enter the values of the list1

Enter the values of the list2

Enter the values of the list3

Enter the values of the list4

Enter the values of the list5

[1, 4, 9, 16, 25]

Q3. Generate 50 random numbers in the range 1 and 30. Remove all duplicate values from the list

INPUT:

print("om shrivastav")

print("24BEE083")

list =[]

def prime(l):

l=[]

n=int(input("Enter the range"))

c=0

d=0

e=0

for i in range(n):

l.append((int(input("Enter the elements of the list"))))

a=l[i]

for a in l:

for j in range (1,a+1):

if(a%j==0):

c+=1

if(c==2):

d+=1

else:

e+=1

print(f"The number of prime numbers is {d}")

print(f"The number of co2mposite number is {e}")

prime(list)

OUTPUT:

om shrivastav

24BEE083

Enter the range5

Enter the elements of the list1

Enter the elements of the list2

Enter the elements of the list3

Enter the elements of the list4

Enter the elements of the list5

The number of prime numbers is 0

The number of co2mposite number is 5

Q4. Generate 30 random numbers and put them in a list. Create two more lists – one containing only

+ve numbers and another with –ve nos

INPUT:

print("om shrivastav")

print("24BEE083")

'''def perfect():

l=[]

c=0

d=0

n = int(input("Enter the range"))

for i in range (n):

l.append((int(input("Enter the values of list"))))

for a in l:

j=a

sum=0

while a>0:

sum+=a%10

a//=10

if j==sum:

c+=1

else:

d+=1

print(f"The number of perfect numbers is {c}")

print(f"the number of non perfect numbers is {d}")

perfect()'''

def perfect():

l = [] # Initialize the list to store numbers

c = 0 # Count of "digit-sum-perfect" numbers

d = 0 # Count of non-"digit-sum-perfect" numbers

n = int(input("Enter the range: "))

# Collect the numbers from the user

for i in range(n):

l.append(int(input(f"Enter the value of the list at index {i+1}: ")))

# Loop through each number in the list to check the digit sum

for a in l:

original\_a = a # Save the original value of a

sum\_digits = 0 # Reset sum of digits for each number

# Sum the digits of the number

while a > 0:

sum\_digits += a % 10

a //= 10 # Use integer division to remove the last digit

# Check if the sum of digits is equal to the original number

if original\_a == sum\_digits:

c += 1

else:

d += 1

# Output the results

print(f"The number of digit-sum-perfect numbers is {c}")

print(f"The number of non-digit-sum-perfect numbers is {d}")

# Call the function to test

perfect()

OUTPUT:

om shrivastav

24BEE083

Enter the range: 5

Enter the value of the list at index 1: 1

Enter the value of the list at index 2: 2

Enter the value of the list at index 3: 3

Enter the value of the list at index 4: 4

Enter the value of the list at index 5: 5

The number of digit-sum-perfect numbers is 5

The number of non-digit-sum-perfect numbers is 0

Q5. A list contains 5 strings. Convert all these strings to uppercase.

INPUT:

print("om shrivastav")

print("24BEE083")

def fact():

l=[]

n = int(input("Enter the range"))

for i in range(n):

l.append((int(input("Enter the values of the list"))))

for a in l:

fact =1

for j in range(1,a+1):

fact=fact\*j

print(f"The factorial of numbers{a} are {fact}")

fact()

OUTPUT:

om shrivastav

24BEE083

Enter the range5

Enter the values of the list6

Enter the values of the list7

Enter the values of the list8

Enter the values of the list9

Enter the values of the list0

The factorial of numbers6 are 720

The factorial of numbers7 are 5040

The factorial of numbers8 are 40320

The factorial of numbers9 are 362880

The factorial of numbers0 are 1

Q6. Convert list of temperatures in Fahrenheit degrees to equivalent Celsius degrees.

INPUT:

print("om shrivastav")

print("24BEE083")

list =[]

def temp(l):

l=[]

c=[]

for i in range (5):

l.append((int(input("Enter the temperatures in Fahrenheit"))))

a=l[i]

g = (a-32)\*(5/9)

c.append(g)

print(f"The list in celsius is : {c}")

temp(list)

OUTPUT:

om shrivastav

24BEE083

Enter the temperatures in Fahrenheit40

Enter the temperatures in Fahrenheit32

Enter the temperatures in Fahrenheit212

Enter the temperatures in Fahrenheit18

Enter the temperatures in Fahrenheit13

The list in celsius is : [4.444444444444445, 0.0, 100.0, -7.777777777777779, -10.555555555555555]

Q9. Take two lists of numbers. Create third list of numbers for only those numbers from first list which

are not there in 2nd list (use list comprehension).

INPUT:

print("om shrivastav")

print("24BEE083")

a=[]

b=[]

c=[]

for i in range (5):

a.append((int(input("Enter 5 numbers "))))

d = a[i]

for j in range (5):

b.append((int(input("Enter 5 numbers for second list "))))

e =b[j]

for i in a :

if i not in b:

c.append(i)

print(c)

OUTPUT:  
om shrivastav

24BEE083

Enter 5 numbers 12345

Enter 5 numbers 56789

Enter 5 numbers 01234

Enter 5 numbers 56789

Enter 5 numbers 01235

Enter 5 numbers for second list 12345

Enter 5 numbers for second list 12345

Enter 5 numbers for second list 12345

Enter 5 numbers for second list 12345

Enter 5 numbers for second list 12345

[56789, 1234, 56789, 1235]

LAB6. TUPLES

Q1. A list contains names of boys and girls as its elements. Boys’ names are stored as tuples. Write a program to find out number of boys and girls in the list. (Hint: use isinstance(ele,tuple))

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

def count\_boys\_girls(name\_list):

boys\_count = sum(1 for name in name\_list if isinstance(name, tuple))

girls\_count = sum(1 for name in name\_list if isinstance(name, str))

return boys\_count, girls\_count

OUTPUT:

OM SHRIVASTAV

24BEE083

Q2. A list contains tuples containing roll no., name and age of student. Write a python program to create three lists separately for roll no., name and age

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

# Sample list of student data (roll no., name, age)

students = [

(101, "Alice", 20),

(102, "Bob", 21),

(103, "Charlie", 19),

(104, "David", 22)

]

# Using list comprehension to create separate lists

roll\_nos = [student[0] for student in students]

names = [student[1] for student in students]

ages = [student[2] for student in students]

# Display the separated lists

print("Roll Numbers:", roll\_nos)

print("Names:", names)

print("Ages:", ages)

OUTPUT:

Roll Numbers: [101, 102, 103, 104]

Names: ['Alice', 'Bob', 'Charlie', 'David']

Ages: [20, 21, 19, 22]

Q3. Suppose a date is represented as a tuple (d, m, y). Create two date tuples and find the number of days between the two dates.

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

from datetime import date

# Define two date tuples (day, month, year)

date1 = (15, 3, 2023) # 15th March 2023

date2 = (5, 7, 2023) # 5th July 2023

# Convert tuples to datetime.date objects

d1 = date(date1[2], date1[1], date1[0])

d2 = date(date2[2], date2[1], date2[0])

# Calculate the difference in days

days\_difference = abs((d2 - d1).days)

# Display the result

print("Number of days between the two dates:", days\_difference)

OUTPUT:

Number of days between the two dates: 112

Q4. Create a list of tuples containing a food item and its price. Sort the tuples in descending order by price.

INPUT:  
print("OM SHRIVASTAV")

print("24BEE083")

# List of food items with prices

food\_items = [

("Burger", 120),

("Pizza", 300),

("Pasta", 250),

("Sandwich", 150),

("Salad", 100)

]

# Sorting the list in descending order by price

sorted\_food\_items = sorted(food\_items, key=lambda x: x[1], reverse=True)

# Display the sorted list

print("Food items sorted by price (descending order):")

for item in sorted\_food\_items:

print(item)

OUTPUT:

Food items sorted by price (descending order):

('Pizza', 300)

('Pasta', 250)

('Sandwich', 150)

('Burger', 120)

('Salad', 100)

Q5. Remove empty tuple(s) from the list of tuples.

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

# List containing tuples, including empty ones

tuples\_list = [(1, 2), (), (3, 4, 5), (), (6,)]

# Removing empty tuples using list comprehension

filtered\_list = [t for t in tuples\_list if t]

# Display the result

print("List after removing empty tuples:", filtered\_list)

OUTPUT:

List after removing empty tuples: [(1, 2), (3, 4, 5), (6,)]

Q6. Modify an element of a tuple.

INPUT:  
print("OM SHRIVASTAV")

print("24BEE083")

# Original tuple

my\_tuple = (10, 20, 30, 40)

# Convert tuple to list

temp\_list = list(my\_tuple)

# Modify the element (e.g., change 20 to 25)

temp\_list[1] = 25

# Convert back to tuple

modified\_tuple = tuple(temp\_list)

# Display the result

print("Modified tuple:", modified\_tuple)

OUTPUT:

Modified tuple: (10, 25, 30, 40)

Q7. Delete an element of a tuple.

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

# Original tuple

my\_tuple = (10, 20, 30, 40)

# Remove an element (e.g., 20)

new\_tuple = tuple(x for x in my\_tuple if x != 20)

# Display the result

print("Tuple after removing an element:", new\_tuple)

OUTPUT:

Tuple after removing an element: (10, 30, 40)

LAB7. DICTIONARIES

Q1. Write a program to create three dictionaries and concatenate them to create fourth dictionary

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

# Define three dictionaries

dict1 = {'a': 1, 'b': 2}

dict2 = {'c': 3, 'd': 4}

dict3 = {'e': 5, 'f': 6}

# Concatenating dictionaries using dictionary unpacking (\*\*)

dict4 = {\*\*dict1, \*\*dict2, \*\*dict3}

# Display the concatenated dictionary

print("Concatenated Dictionary:", dict4)

OUTPUT:

Concatenated Dictionary: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}

Q2. Write a program to check whether a dictionary is empty or not

INPUT:

INPUT:  
print("OM SHRIVASTAV")

print("24BEE083")

# Function to check if a dictionary is empty

def is\_dict\_empty(d):

return not bool(d) # Returns True if empty, False otherwise

# Example dictionaries

dict1 = {} # Empty dictionary

dict2 = {'a': 1, 'b': 2} # Non-empty dictionary

# Checking the dictionaries

print("Is dict1 empty?", is\_dict\_empty(dict1)) # Output: True

print("Is dict2 empty?", is\_dict\_empty(dict2)) # Output: False

OUTPUT:

OM SHRIVASTAV

24BEE083

Is dict1 empty? True

Is dict2 empty? False

Q3. Create a dictionary with dept no, employee roll no. and salary. Find out department wise min and maximum of salary.

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

from collections import defaultdict

# List of employee records (dept\_no, emp\_roll\_no, salary)

employees = [

(101, 1, 50000),

(101, 2, 60000),

(102, 3, 70000),

(102, 4, 55000),

(103, 5, 80000),

(103, 6, 75000),

(101, 7, 65000),

(102, 8, 72000)

]

# Dictionary to store salaries department-wise

dept\_salaries = defaultdict(list)

# Grouping salaries by department

for dept, roll, salary in employees:

dept\_salaries[dept].append(salary)

# Finding min and max salary per department

dept\_min\_max = {dept: (min(salaries), max(salaries)) for dept, salaries in dept\_salaries.items()}

# Display results

for dept, (min\_salary, max\_salary) in dept\_min\_max.items():

print(f"Department {dept} -> Min Salary: {min\_salary}, Max Salary: {max\_salary}")

OUTPUT:

Department 101 -> Min Salary: 50000, Max Salary: 65000

Department 102 -> Min Salary: 55000, Max Salary: 72000

Department 103 -> Min Salary: 75000, Max Salary: 80000

Q4. Write a program that reads a string from the keyboard and creates dictionary containing frequency of each character occurring in the string.

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

# Read input string from user

input\_string = input("Enter a string: ")

# Create an empty dictionary to store character frequencies

char\_frequency = {}

# Iterate over each character in the string

for char in input\_string:

if char in char\_frequency:

char\_frequency[char] += 1 # Increment count if character already exists

else:

char\_frequency[char] = 1 # Initialize count if character appears for the first time

# Display the character frequency dictionary

print("Character Frequency Dictionary:", char\_frequency)

OUTPUT:

Enter a string: hello world

Character Frequency Dictionary: {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}

Q5. Create two dictionaries – one containing grocery items and their prices and another containing grocery items and quantity purchased. By using the values from these two dictionaries compute the total bill.

INPUT:

print("OM SHRIVASTAV")

print("24BEE083")

# Dictionary containing grocery items and their prices

grocery\_prices = {

"Milk": 50,

"Bread": 30,

"Eggs": 5, # Price per egg

"Rice": 60, # Price per kg

"Sugar": 40 # Price per kg

}

# Dictionary containing grocery items and quantity purchased

grocery\_quantity = {

"Milk": 2, # 2 liters

"Bread": 1, # 1 pack

"Eggs": 12, # 12 eggs

"Rice": 3, # 3 kg

"Sugar": 2 # 2 kg

}

# Calculate the total bill

total\_bill = sum(grocery\_prices[item] \* grocery\_quantity[item] for item in grocery\_prices if item in grocery\_quantity)

# Display the bill

print("Total Grocery Bill: ₹", total\_bill)

OUTPUT:

Total Grocery Bill: ₹ 490

LAB9 FUNCTIONS

Q1. Write a program that defines a function count\_lower\_upper() that accepts a string and calculates the number of uppercase and lowercase alphabets in it. It should return these values as a dictionary. Call this function for some sample string.

INPUT

print("om shrivastav")

print("24BEE083")

def count\_lower\_upper(input\_string):

result = {'uppercase': 0, 'lowercase': 0}

for char in input\_string:

if char.isupper():

result['uppercase'] += 1

elif char.islower():

result['lowercase'] += 1

return result

# Sample string

sample\_text = "Hello World! This is a Test String with UPPER and lower CASE."

result = count\_lower\_upper(sample\_text)

# Output the result

print("Input String:", sample\_text)

print("Count Result:", result)

OUTPUT

Input String: Hello World! This is a Test String with UPPER and lower CASE.

Count Result: {'uppercase': 16, 'lowercase': thirty-five}

Q2. Write a program that defines a function compute() that calculates the value of n + nn + nnn + nnnn, where n is digit received by the function. test the function for digits 4 to 7.

INPUT

print("om shrivastav")

print("24BEE083")

def compute(n):

"""

Computes the value of n + nn + nnn + nnnn

where n is a single-digit number (int or string).

"""

n\_str = str(n)

total = int(n\_str) + int(n\_str \* 2) + int(n\_str \* 3) + int(n\_str \* 4)

return total

# Test the function for digits 4 to 7

for digit in range(4, 8):

result = compute(digit)

print(f"compute({digit}) = {result}")

OUTPUT

compute(4) = 4936 # 4 + 44 + 444 + 4444

compute(5) = 6170 # 5 + 55 + 555 + 5555

compute(6) = 7404 # 6 + 66 + 666 + 6666

compute(7) = 8638 # 7 + 77 + 777 + 7777

Q3. Write a program that defines a function create\_array() to create and return a 3D array whose dimensions are passed to the function. Also initialize each element of this aray to a value passed to the function. e.g. create\_array(3,4,5,n) where first three arguments are 3D array dimensions and 4th value is for initialing each value of the 3D array.

INPUT

print("om shrivastav")

print("24BEE083")

def create\_array(x, y, z, value):

"""

Creates a 3D array of dimensions x \* y \* z,

initializing each element with the specified value.

"""

return [[[value for \_ in range(z)] for \_ in range(y)] for \_ in range(x)]

# Example usage:

array = create\_array(3, 4, 5, 'n')

# Display the array (formatted for clarity)

for i, matrix in enumerate(array):

print(f"Layer {i + 1}:")

for row in matrix:

print(row)

print()

OUTPUT

Layer 1:

['n', 'n', 'n', 'n', 'n']

['n', 'n', 'n', 'n', 'n']

['n', 'n', 'n', 'n', 'n']

['n', 'n', 'n', 'n', 'n']

Layer 2:

...

Layer 3:

...

Q4. Write a program that defines a function sum\_avg() to accept marks of five subjects and calculates total and average. It should return directly both values

INPUT

print("om shrivastav")

print("24BEE083")

def sum\_avg(marks):

"""

Accepts a list of 5 marks, calculates the total and average.

Returns both values as a tuple (total, average).

"""

if len(marks) != 5:

raise ValueError("Exactly 5 marks must be provided.")

total = sum(marks)

average = total / 5

return total, average

# Example usage:

marks = [85, 90, 78, 92, 88]

total, average = sum\_avg(marks)

print("Marks:", marks)

print("Total:", total)

print("Average:", average)

OUTPUT

Marks: [85, 90, 78, 92, 88]

Total: 433

Average: 86.6

Q5. Pangram is a sentence that uses every letter of the alphabet. Write a program to check whether a given string is pangram or not, through a user-defined function ispangram(). Test the function with “The quick brown fox jumps over the lazy dog” or “Crazy Fredrick bought many very exquisite opal jewels”. Hint: use set() to convert the string into a set of characters present in the string and use <= to check whether alphaset is a subset of the given string.

INPUT

print("om shrivastav")

print("24BEE083")

import string

def ispangram(sentence):

"""

Checks whether the given sentence is a pangram.

Returns True if it is, otherwise False.

"""

# Create a set of all lowercase alphabet letters

alphabet\_set = set(string.ascii\_lowercase)

# Convert the sentence to lowercase and create a set of characters from it

sentence\_set = set(sentence.lower())

# Check if all alphabet letters are in the sentence

return alphabet\_set <= sentence\_set

# Test the function with sample inputs

test

Q6 Write a function to create and return a list containing tuples of the form (x,x2,x3) for all x between 1 and given ending value (both inclusive).

INPUT

def generate\_tuples(end\_value):

"""

Generate a list of tuples containing (x, x^2, x^3) for values of x between 1 and end\_value (inclusive).

Args:

end\_value (int): The upper limit of the range.

Returns:

list: A list of tuples.

"""

return [(x, x\*\*2, x\*\*3) for x in range(1, end\_value + 1)]

# Example usage

end = 5

result = generate\_tuples(end)

print(result)

OUTPUT

[(1, 1, 1), (2, 4, 8), (3, 9, 27), (4, 16, 64), (5, 25, 125)]

Q7. A palindrome is a word or phrase that reads the same in both directions. Write a program that defines a function ispalindrome() which checks whether a given string is a palindrome or not. Ignore spaces and case mismatch while checking for palindrome

INPUT

def ispalindrome(input\_string):

"""

Check if a given string is a palindrome, ignoring spaces and case mismatch.

Args:

input\_string (str): The string to check.

Returns:

bool: True if the string is a palindrome, False otherwise.

"""

# Remove spaces and convert to lowercase

cleaned\_string = ''.join(input\_string.split()).lower()

# Check if the cleaned string is equal to its reverse " return cleaned\_string == cleaned\_string[::-1]

# Example usage

test\_string = "A man a plan a canal Panama"

if ispalindrome(test\_string):

print(f'"{test\_string}" is a palindrome.')

else:

print(f'"{test\_string}" is not a palindrome.')

OUTPUT

"A man a plan a canal Panama" is a palindrome.

Q8. Write a program that defines a function convert() that receives a string containing a sequence of whitespace separated words and returns a string after removing all duplicate words and sorting them alphanumerically. Hint: use set(), list () , sorted(), join()

INPUT

def convert(sentence):

"""

Receives a whitespace-separated string, removes duplicates,

sorts the words alphanumerically, and returns the final string.

"""

words = sentence.split() # Split into words

unique\_words = set(words) # Remove duplicates

sorted\_words = sorted(list(unique\_words)) # Sort the words

return ' '.join(sorted\_words) # Join and return

# Example usage:

input\_string = "apple banana apple orange banana grape"

result = convert(input\_string)

print("Original String:", input\_string)

print("Converted String:", result)

OUTPUT

Original String: banana apple orange apple mango banana grape

Processed String: apple banana grape mango orange

Q9. Write a program that defines a function count\_alpha\_digits() that accepts a string and calculates the number of alphabets and digits in it. It should return these values as a dictionary.

INPUT

def count\_alpha\_digits(input\_string):

"""

Accepts a string and returns a dictionary with the count of

alphabets and digits in it.

"""

result = {'alphabets': 0, 'digits': 0}

for char in input\_string:

if char.isalpha():

result['alphabets'] += 1

elif char.isdigit():

result['digits'] += 1

return result

# Example usage:

sample\_text = "Hello123World456"

result = count\_alpha\_digits(sample\_text)

print("Input String:", sample\_text)

print("Count Result:", result)

OUTPUT

Input String: Hello123World456

Count Result: {'alphabets': 10, 'digits': 6}

Q10. Write a program that defines a function called frequency() which computes the frequency of words present in a string passed to it. The frequencies should be returned in sorted order of words in the string.

INPUT

def frequency(input\_string):

"""

Compute the frequency of words in a string and return the result sorted by words.

Args:

input\_string (str): The input string containing words.

Returns:

dict: A dictionary with words as keys and their frequencies as values, sorted by words.

"""

# Split the string into words

words = input\_string.split()

# Create a dictionary to count word frequencies

word\_count = {}

for word in words:

word = word.lower() # Optional: Convert words to lowercase for case insensitivity

word\_count[word] = word\_count.get(word, 0) + 1

# Return the dictionary sorted by keys (words)

return dict(sorted(word\_count.items()))

# Example usage

test\_string = "apple banana orange apple grape banana apple"

result = frequency(test\_string)

print(result)

OUTPUT

{'apple': 3, 'banana': 2, 'grape': 1, 'orange': 1}

Q11. Write a function create\_list() that creates and returns a list which is an intersection of two lists passed to it.

INPUT

def create\_list(list1, list2):

"""

Create and return a list which is the intersection of two given lists.

Args:

list1 (list): The first list.

list2 (list): The second list.

Returns:

list: A list containing the intersection of list1 and list2.

"""

# Use a set intersection to find common elements

intersection = list(set(list1) & set(list2))

return intersection

# Example usage

list\_a = [1, 2, 3, 4, 5]

list\_b = [4, 5, 6, 7, 8]

result = create\_list(list\_a, list\_b)

print(result)

OUTPUT

[4, 5]

RECURSIVE FUNCTIONS

Q1. If a positive integer is entered through the keyword, write a recursive function to obtain the prime factors of the number.

INPUT

def prime\_factors(n, factor=2):

"""

Recursively obtain the prime factors of a given positive integer.

Args:

n (int): The number to factorize.

factor (int): The current factor to check (default is 2).

Returns:

list: A list containing the prime factors of n.

"""

# Base case: If n becomes 1, return an empty list

if n == 1:

return []

# Check if the current factor divides n

if n % factor == 0:

return [factor] + prime\_factors(n // factor, factor)

else:

# Increment the factor to test the next number

return prime\_factors(n, factor + 1)

# Example usage

number = 56

result = prime\_factors(number)

print(f"Prime factors of {number}: {result}")

OUTPUT

Prime factors of 56: [2, 2, 2, 7]

Q2. A positive integer is entered through the keyboard. Write a function to find its binary equivalent of this number.

INPUT

def binary\_equivalent(number):

"""

Find the binary equivalent of a positive integer.

Args:

number (int): A positive integer.

Returns:

str: The binary representation of the number.

"""

if number < 0:

return "Please enter a positive integer."

# Use the built-in bin() function to find the binary equivalent

return bin(number)[2:] # Slice off the "0b" prefix

# Example usage

try:

user\_input = int(input("Enter a positive integer: "))

result = binary\_equivalent(user\_input)

print(f"The binary equivalent of {user\_input} is {result}")

except ValueError:

print("Invalid input! Please enter a valid positive integer.")

OUTPUT

The binary equivalent of 25 is 11001

Q3. A string is entered through the keyboard. Write a recursive function that counts the number of vowels in this string.

INPUT

def count\_vowels(s):

"""

Recursively count the number of vowels in the given string.

Args:

s (str): The input string.

Returns:

int: The number of vowels in the string.

"""

# Base case: If the string is empty, return 0

if not s:

return 0

# Check if the first character is a vowel

vowels = "aeiouAEIOU"

if s[0] in vowels:

return 1 + count\_vowels(s[1:])

else:

return count\_vowels(s[1:])

# Example usage

input\_string = input("Enter a string: ")

result = count\_vowels(input\_string)

print(f"The number of vowels in the string is: {result}")

OUTPUT

The number of vowels in the string is: 3

Q4. Write a recursive function that reverses the list of numbers that it receives.

INPUT

def reverse\_list(numbers):

"""

Recursively reverse a list of numbers.

Args:

numbers (list): The list to reverse.

Returns:

list: The reversed list.

"""

# Base case: If the list is empty or has one element, return the list

if len(numbers) <= 1:

return numbers

# Recursive case: Reverse the rest of the list and append the first element to the end

return reverse\_list(numbers[1:]) + [numbers[0]]

# Example usage

num\_list = [1, 2, 3, 4, 5]

result = reverse\_list(num\_list)

print(f"Original list: {num\_list}")

print(f"Reversed list: {result}")

OUTPUT

Original list: [1, 2, 3, 4, 5]

Reversed list: [5, 4, 3, 2, 1]

Q5. Calculate ab where a and b received through the keyword using recursion

INPUT

def power(a, b):

"""

Recursively calculate a^b (a raised to the power of b).

Args:

a (int): Base number.

b (int): Exponent (must be a non-negative integer).

Returns:

int: Result of a^b.

"""

# Base case: Any number raised to the power of 0 is 1

if b == 0:

return 1

# Recursive case: Multiply a by the result of a^(b-1)

return a \* power(a, b - 1)

# Example usage

try:

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

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

if exponent < 0:

print("Please enter a non-negative exponent.")

else:

result = power(base, exponent)

print(f"{base} raised to the power of {exponent} is: {result}")

except ValueError:

print("Invalid input! Please enter valid integers.")

OUTPUT

2 raised to the power of 3 is: 8

Q6. A list contains some negative and some positive values. Write a recursive function that sanitizes the list by replacing all negative numbers with 0

INPUT

def sanitize\_list(lst, index=0):

# Base case: if index is beyond the last element, return the list

if index == len(lst):

return lst

# If the current element is negative, replace it with 0

if lst[index] < 0:

lst[index] = 0

# Recursively process the next element

return sanitize\_list(lst, index + 1)

# Example usage:

my\_list = [1, -2, 3, -4, 5]

sanitize\_list(my\_list)

print(my\_list) # Output will be [1, 0, 3, 0, 5]

Q7. Write a recursive function to obtain average of all numbers present in a given list.

INPUT

def average\_list(lst, index=0, total\_sum=0, count=0):

# Base case: if index is beyond the last element, return the average

if index == len(lst):

if count == 0: # To prevent division by zero in case the list is empty

return 0

return total\_sum / count

# Add current element to total sum and increase the count

return average\_list(lst, index + 1, total\_sum + lst[index], count + 1)

# Example usage:

my\_list = [1, 2, 3, 4, 5]

print(average\_list(my\_list)) # Output will be 3.0

Q8. Write a recursive function to obtain length of a given string

INPUT

def string\_length(s):

# Base case: if the string is empty, return 0

if s == "":

return 0

# Recursive step: 1 (for the first character) + length of the rest of the string

return 1 + string\_length(s[1:])

# Example usage:

my\_string = "Hello, world!"

print(string\_length(my\_string)) # Output will be 13

LAB9A FUNCTIONAL PROGRAMMING

Q1. Define three functions fun(), disp() and msg(). Store them in a list and call them one by one in a loop

INPUT

def fun():

print("This is fun()!")

def disp():

print("This is disp()!")

def msg():

print("This is msg()!")

# Storing functions in a list

functions\_list = [fun, disp, msg]

# Looping through the list and calling each function

for func in functions\_list:

func()

Q2. Suppose there are two lists, one containing numbers from 1 to 6, and other containing numbers from 6 to 1. Write a program to obtain a list that contains elements obtained by adding corresponding elements of the two lists. (hint: use map and lambda functions)

INPUT

# Define the two lists

list1 = [1, 2, 3, 4, 5, 6]

list2 = [6, 5, 4, 3, 2, 1]

# Use map and lambda to add corresponding elements

result = list(map(lambda x, y: x + y, list1, list2))

# Print the resulting list

print(f"The resulting list is: {result}")

Q3. Generate the list of 10 different random numbers between -15 and 15. Create a new list by obtaining square of all numbers in a list.

import random

def generate\_random\_numbers\_and\_squares():

"""

Generate a list of 10 different random numbers between -15 and 15,

and return a new list containing the squares of these numbers.

Returns:

tuple: A tuple containing the original list and the list of squares.

"""

# Generate 10 different random numbers between -15 and 15

random\_numbers = random.sample(range(-15, 16), 10)

# Create a new list containing the squares of the numbers

squares = [x \*\* 2 for x in random\_numbers]

return random\_numbers, squares

# Example usage

original\_list, squared\_list = generate\_random\_numbers\_and\_squares()

print(f"Original list: {original\_list}")

print(f"List of squares: {squared\_list}")

Q4. Consider the following list:

lst = ['madam','Python',"malayalam",12321]

Write a program to print those strings which are palindromes.

INPUT

def find\_palindromes(lst):

"""

Print strings from a list that are palindromes.

Args:

lst (list): The list containing strings.

Returns:

None

"""

for item in lst:

# Check if the item is a string and a palindrome

if isinstance(item, str) and item == item[::-1]:

print(f'"{item}" is a palindrome.')

# Example list

lst = ['madam', 'Python', 'malayalam', 12321]

find\_palindromes(lst)

Q5. A list contains names of Faculty Members. Write a program to filter out those names whose length is more than 8 characters.

INPUT

def filter\_long\_names(faculty\_list):

"""

Filter out names with more than 8 characters from the list.

Args:

faculty\_list (list): List of faculty member names.

Returns:

list: A list containing names with 8 or fewer characters.

"""

return [name for name in faculty\_list if len(name) <= 8]

# Example usage

faculty\_names = ["Alexander", "Bob", "Catherine", "David", "Eleanor", "Frank"]

filtered\_names = filter\_long\_names(faculty\_names)

print(f"Filtered list: {filtered\_names}")

LAB 10. FILE HANDLING

Q1. Write a program to create a csv file that we can directly open in MS-Excel.

INPUT

import csv

def create\_csv(filename, data):

"""

Create a CSV file with the given data.

Args:

filename (str): Name of the CSV file to create.

data (list of list): Data to write to the CSV file, where each inner list is a row.

Returns:

None

"""

# Open the file in write mode

with open(filename, mode='w', newline='') as file:

writer = csv.writer(file)

# Write each row of data to the CSV file

writer.writerows(data)

# Example usage

filename = "example.csv"

data = [

["Name", "Age", "Department"],

["Alice", 30, "HR"],

["Bob", 25, "Finance"],

["Charlie", 35, "Engineering"]

]

create\_csv(filename, data)

print(f"CSV file '{filename}' has been created successfully!")

Q2. Read the data stored in MS-Excel file and convert it into a dictionary. The record contains rollno, name of student, marks of three subjects. Also calculate total. Display the dictionary data on the monitor.

INPUT

import openpyxl

def read\_excel\_to\_dict(filename):

"""

Read data from an Excel file, convert it into a dictionary,

and calculate total marks for each student.

Args:

filename (str): Name of the Excel file.

Returns:

dict: Dictionary containing student records.

"""

# Load the workbook and the active sheet

workbook = openpyxl.load\_workbook(filename)

sheet = workbook.active

# Initialize the dictionary

student\_data = {}

# Iterate through the rows, assuming the first row has headers

for row in sheet.iter\_rows(min\_row=2, values\_only=True):

rollno, name, subject1, subject2, subject3 = row

total = subject1 + subject2 + subject3 # Calculate total marks

student\_data[rollno] = {

"Name": name,

"Marks": {"Subject1": subject1, "Subject2": subject2, "Subject3": subject3},

"Total": total

}

return student\_data

# Example usage

filename = "student\_records.xlsx" # Replace with the name of your Excel file

try:

records = read\_excel\_to\_dict(filename)

for rollno, data in records.items():

print(f"Roll No: {rollno}, Name: {data['Name']}, Marks: {data['Marks']}, Total: {data['Total']}")

except FileNotFoundError:

print("Error: The specified file was not found. Please check the filename and path.")

Q3. Accept contact details from the user and create a vcard that we can directly store in our mobile

INPUT

def create\_vcard():

"""

Accept contact details from the user and create a vCard file.

"""

# Accept user input for contact details

name = input("Enter full name: ")

phone = input("Enter phone number: ")

email = input("Enter email address (optional): ")

address = input("Enter address (optional): ")

# Create vCard content

vcard\_content = f"""BEGIN:VCARD

VERSION:3.0

FN:{name}

TEL;TYPE=CELL:{phone}

EMAIL:{email}

ADR:{address}

END:VCARD

"""

# Save vCard to a .vcf file

filename = f"{name.replace(' ', '\_')}.vcf"

with open(filename, "w") as file:

file.write(vcard\_content)

print(f"vCard saved as '{filename}'. You can now import it into your mobile device!")

# Example usage

create\_vcard()

Q4. Create a specific subdirectory and copy one file from another subdirectory to this newly created subdirectory.

INPUT

import os

import shutil

def create\_and\_copy(source\_dir, file\_name, target\_dir):

"""

Create a new subdirectory and copy a specific file into it.

Args:

source\_dir (str): Path of the source subdirectory containing the file.

file\_name (str): Name of the file to copy.

target\_dir (str): Path of the new subdirectory to be created.

Returns:

None

"""

try:

# Create the target subdirectory if it doesn't exist

os.makedirs(target\_dir, exist\_ok=True)

print(f"Subdirectory '{target\_dir}' created successfully.")

# Construct full paths for the source file and the target file

source\_path = os.path.join(source\_dir, file\_name)

target\_path = os.path.join(target\_dir, file\_name)

# Copy the file

shutil.copy(source\_path, target\_path)

print(f"File '{file\_name}' copied from '{source\_dir}' to '{target\_dir}' successfully.")

except FileNotFoundError:

print(f"Error: The file '{file\_name}' does not exist in '{source\_dir}'.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

source\_directory = "source\_subdir" # Replace with the path of your source subdirectory

file\_to\_copy = "example.txt" # Replace with the name of the file to copy

target\_directory = "new\_subdir" # Replace with the name of the new subdirectory

create\_and\_copy(source\_directory, file\_to\_copy, target\_directory)

Q5. Write a program to copy contents of one file to another. While doing so, replace all lowercase characters into uppercase characters.

def copy\_and\_transform(source\_file, target\_file):

"""

Copy contents from source\_file to target\_file, converting all lowercase

characters to uppercase.

Args:

source\_file (str): The name of the source file to read from.

target\_file (str): The name of the target file to write to.

"""

try:

# Open source file in read mode

with open(source\_file, 'r') as src:

# Read the contents of the source file

content = src.read()

# Transform content: Convert lowercase to uppercase

transformed\_content = content.upper()

# Open target file in write mode and write transformed content

with open(target\_file, 'w') as tgt:

tgt.write(transformed\_content)

print(f"Contents have been successfully copied from '{source\_file}' to '{target\_file}' with transformations.")

except FileNotFoundError:

print(f"Error: The source file '{source\_file}' does not exist.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

source = "source.txt" # Replace with the name of your source file

target = "target.txt" # Replace with the name of your target file

copy\_and\_transform(source, target)

Q6. Write a program that merges lines alternatively from two files and writes the results to new file. If one file has less number of lines than the other, the remaining lines from the larger file should be simply copied into the target file.

def merge\_files\_alternatively(file1, file2, target\_file):

"""

Merge lines from two files alternatively and write the result to a new file.

Remaining lines from the larger file are appended.

Args:

file1 (str): Path to the first file.

file2 (str): Path to the second file.

target\_file (str): Path to the target file where the merged content is written.

"""

try:

# Open both files and read their lines

with open(file1, 'r') as f1, open(file2, 'r') as f2:

lines1 = f1.readlines()

lines2 = f2.readlines()

# Open the target file for writing

with open(target\_file, 'w') as target:

# Iterate over both lists of lines simultaneously

for line1, line2 in zip(lines1, lines2):

target.write(line1.strip() + '\n')

target.write(line2.strip() + '\n')

# Add remaining lines from the larger file

if len(lines1) > len(lines2):

target.writelines(line.strip() + '\n' for line in lines1[len(lines2):])

elif len(lines2) > len(lines1):

target.writelines(line.strip() + '\n' for line in lines2[len(lines1):])

print(f"Merged content has been written to '{target\_file}' successfully!")

except FileNotFoundError:

print(f"Error: One of the source files does not exist.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

file1 = "file1.txt" # Replace with the path to the first file

file2 = "file2.txt" # Replace with the path to the second file

target\_file = "merged.txt" # Replace with the path for the target file

merge\_files\_alternatively(file1, file2, target\_file)

Q7. If an Employee object contains following details:

empcode, empname, Date of Joining, Salary

Write a program to serialize and deserialize this data.

import pickle

class Employee:

"""

Represents an Employee object.

"""

def \_\_init\_\_(self, empcode, empname, date\_of\_joining, salary):

self.empcode = empcode

self.empname = empname

self.date\_of\_joining = date\_of\_joining

self.salary = salary

def \_\_str\_\_(self):

return (f"Employee Code: {self.empcode}, Name: {self.empname}, "

f"Date of Joining: {self.date\_of\_joining}, Salary: {self.salary}")

# Serialize the Employee object

def serialize\_employee(employee, filename):

with open(filename, 'wb') as file:

pickle.dump(employee, file)

print(f"Employee data serialized and saved to '{filename}'")

# Deserialize the Employee object

def deserialize\_employee(filename):

try:

with open(filename, 'rb') as file:

employee = pickle.load(file)

print("Employee data deserialized successfully!")

return employee

except FileNotFoundError:

print(f"Error: The file '{filename}' does not exist.")

return None

# Example usage

employee = Employee("E123", "Alice", "2023-05-10", 50000)

# Serialize the Employee object

serialize\_filename = "employee\_data.pkl"

serialize\_employee(employee, serialize\_filename)

# Deserialize the Employee object

deserialized\_employee = deserialize\_employee(serialize\_filename)

if deserialized\_employee:

print(deserialized\_employee)

Q8. Given a text file, write a program to create another text file deleting the words ‘a’, ‘the’, ‘an’ and replacing each one of them with a blank space.

def clean\_text\_file(source\_file, target\_file):

"""

Remove the words 'a', 'the', 'an' from the source file and write the result to the target file.

Args:

source\_file (str): The name of the source text file to read from.

target\_file (str): The name of the target text file to write to.

"""

try:

# Open source file in read mode

with open(source\_file, 'r') as src:

content = src.read()

# Words to remove

words\_to\_remove = ['a', 'the', 'an']

# Replace occurrences of the words with a blank space

for word in words\_to\_remove:

content = content.replace(f' {word} ', ' ')

# Write the modified content to the target file

with open(target\_file, 'w') as tgt:

tgt.write(content)

print(f"Modified content has been written to '{target\_file}'.")

except FileNotFoundError:

print(f"Error: The source file '{source\_file}' does not exist.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

# Example usage

source = "source.txt" # Replace with your source file name

target = "target.txt" # Replace with your target file name

clean\_text\_file(source, target)

LAB 11. EXCEPTIONAL FILE HANDLING

Q1. Write a program that receives an integer an input. If a string is entered instead of an integer, then report an error and give another chance to user to enter an integer. Continue this process till correct input is supplied.

INPUT

def get\_integer():

"""

Prompt the user to enter an integer and handle invalid inputs gracefully.

Returns:

int: The valid integer input.

"""

while True:

user\_input = input("Please enter an integer: ")

try:

# Try to convert the input to an integer

num = int(user\_input)

print(f"You entered a valid integer: {num}")

return num

except ValueError:

# Handle the case where the input is not an integer

print("Error: That is not a valid integer. Please try again.")

# Example usage

integer = get\_integer()

print(f"The valid integer received is: {integer}")

OUTPUT

Please enter an integer: Hello

Error: That is not a valid integer. Please try again.

Please enter an integer: 123

You entered a valid integer: 123

The valid integer received is: 123

LAB12 CLASSES AND OBJECT ORIENTATION

Q1. Write a program to create a class that represents Complex numbers containing real and imaginary parts and then use it to perform complex number addition, subtraction, multiplication and division.

class Complex:

"""

A class to represent complex numbers with real and imaginary parts

and perform basic arithmetic operations on them.

"""

def \_\_init\_\_(self, real, imaginary):

self.real = real

self.imaginary = imaginary

def \_\_add\_\_(self, other):

# Addition of two complex numbers

return Complex(self.real + other.real, self.imaginary + other.imaginary)

def \_\_sub\_\_(self, other):

# Subtraction of two complex numbers

return Complex(self.real - other.real, self.imaginary - other.imaginary)

def \_\_mul\_\_(self, other):

# Multiplication of two complex numbers

real\_part = self.real \* other.real - self.imaginary \* other.imaginary

imaginary\_part = self.real \* other.imaginary + self.imaginary \* other.real

return Complex(real\_part, imaginary\_part)

def \_\_truediv\_\_(self, other):

# Division of two complex numbers

denominator = other.real\*\*2 + other.imaginary\*\*2

if denominator == 0:

raise ZeroDivisionError("Division by zero is not allowed.")

real\_part = (self.real \* other.real + self.imaginary \* other.imaginary) / denominator

imaginary\_part = (self.imaginary \* other.real - self.real \* other.imaginary) / denominator

return Complex(real\_part, imaginary\_part)

def \_\_str\_\_(self):

# String representation of a complex number

return f"{self.real} + {self.imaginary}i"

# Example usage

c1 = Complex(3, 2) # First complex number: 3 + 2i

c2 = Complex(1, 7) # Second complex number: 1 + 7i

# Perform arithmetic operations

addition = c1 + c2

subtraction = c1 - c2

multiplication = c1 \* c2

division = c1 / c2

# Display results

print("First complex number:", c1)

print("Second complex number:", c2)

print("Addition:", addition)

print("Subtraction:", subtraction)

print("Multiplication:", multiplication)

print("Division:", division)

Q2. Write a program that implements a Matrix class and performs addition, multiplication and transpose operations on 3x3 matrices.

class Matrix:

"""

A class to represent a 3x3 matrix and perform operations like addition,

multiplication, and transpose.

"""

def \_\_init\_\_(self, matrix):

if len(matrix) == 3 and all(len(row) == 3 for row in matrix):

self.matrix = matrix

else:

raise ValueError("Matrix must be 3x3.")

def \_\_add\_\_(self, other):

"""Adds two matrices."""

return Matrix([[self.matrix[i][j] + other.matrix[i][j] for j in range(3)] for i in range(3)])

def \_\_mul\_\_(self, other):

"""Multiplies two matrices."""

result = [[0 for \_ in range(3)] for \_ in range(3)]

for i in range(3):

for j in range(3):

for k in range(3):

result[i][j] += self.matrix[i][k] \* other.matrix[k][j]

return Matrix(result)

def transpose(self):

"""Returns the transpose of the matrix."""

return Matrix([[self.matrix[j][i] for j in range(3)] for i in range(3)])

def display(self):

"""Displays the matrix."""

for row in self.matrix:

print(row)

# Example usage

matrix1 = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

matrix2 = Matrix([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

print("Matrix 1:")

matrix1.display()

print("\nMatrix 2:")

matrix2.display()

# Addition

print("\nAddition of Matrix 1 and Matrix 2:")

addition\_result = matrix1 + matrix2

addition\_result.display()

# Multiplication

print("\nMultiplication of Matrix 1 and Matrix 2:")

multiplication\_result = matrix1 \* matrix2

multiplication\_result.display()

# Transpose

print("\nTranspose of Matrix 1:")

transpose\_result = matrix1.transpose()

transpose\_result.display()

Q3. Write a program to create a class that can calculate the surface area and volume of a solid. The class should also have a provision to accept the data relevant to the solid.

import math

class Solid:

"""

A class to represent a solid and calculate its surface area and volume.

"""

def \_\_init\_\_(self, solid\_type):

self.solid\_type = solid\_type.lower()

self.data = {}

def accept\_data(self):

"""

Accept data relevant to the solid based on its type.

"""

if self.solid\_type == "sphere":

self.data["radius"] = float(input("Enter the radius of the sphere: "))

elif self.solid\_type == "cube":

self.data["side"] = float(input("Enter the side length of the cube: "))

elif self.solid\_type == "cylinder":

self.data["radius"] = float(input("Enter the radius of the cylinder: "))

self.data["height"] = float(input("Enter the height of the cylinder: "))

else:

print("Unsupported solid type. Please choose sphere, cube, or cylinder.")

def calculate\_surface\_area(self):

"""

Calculate the surface area of the solid.

"""

if self.solid\_type == "sphere":

radius = self.data.get("radius", 0)

return 4 \* math.pi \* radius\*\*2

elif self.solid\_type == "cube":

side = self.data.get("side", 0)

return 6 \* side\*\*2

elif self.solid\_type == "cylinder":

radius = self.data.get("radius", 0)

height = self.data.get("height", 0)

return 2 \* math.pi \* radius \* (radius + height)

else:

return None

def calculate\_volume(self):

"""

Calculate the volume of the solid.

"""

if self.solid\_type == "sphere":

radius = self.data.get("radius", 0)

return (4/3) \* math.pi \* radius\*\*3

elif self.solid\_type == "cube":

side = self.data.get("side", 0)

return side\*\*3

elif self.solid\_type == "cylinder":

radius = self.data.get("radius", 0)

height = self.data.get("height", 0)

return math.pi \* radius\*\*2 \* height

else:

return None

# Example usage

solid\_type = input("Enter the type of solid (sphere, cube, cylinder): ").strip()

solid = Solid(solid\_type)

solid.accept\_data()

surface\_area = solid.calculate\_surface\_area()

volume = solid.calculate\_volume()

if surface\_area is not None and volume is not None:

print(f"The surface area of the {solid\_type} is: {surface\_area:.2f}")

print(f"The volume of the {solid\_type} is: {volume:.2f}")

else:

print("Invalid solid type or missing data.")

Q4. Write a program to create a class that can calculate the perimeter/circumference and area of a regular shape. The class should also have a provision to accept the data relevant to the shape.

import math

class Shape:

"""

A class to calculate the perimeter/circumference and area of regular shapes.

"""

def \_\_init\_\_(self, shape\_type):

self.shape\_type = shape\_type.lower()

self.data = {}

def accept\_data(self):

"""

Accept relevant data for the chosen shape.

"""

if self.shape\_type == "square":

self.data["side"] = float(input("Enter the side length of the square: "))

elif self.shape\_type == "rectangle":

self.data["length"] = float(input("Enter the length of the rectangle: "))

self.data["width"] = float(input("Enter the width of the rectangle: "))

elif self.shape\_type == "circle":

self.data["radius"] = float(input("Enter the radius of the circle: "))

elif self.shape\_type == "triangle":

self.data["side1"] = float(input("Enter the length of side 1: "))

self.data["side2"] = float(input("Enter the length of side 2: "))

self.data["side3"] = float(input("Enter the length of side 3: "))

else:

print("Unsupported shape type. Please choose square, rectangle, circle, or triangle.")

def calculate\_perimeter(self):

"""

Calculate the perimeter or circumference of the shape.

"""

if self.shape\_type == "square":

return 4 \* self.data.get("side", 0)

elif self.shape\_type == "rectangle":

return 2 \* (self.data.get("length", 0) + self.data.get("width", 0))

elif self.shape\_type == "circle":

return 2 \* math.pi \* self.data.get("radius", 0)

elif self.shape\_type == "triangle":

return self.data.get("side1", 0) + self.data.get("side2", 0) + self.data.get("side3", 0)

else:

return None

def calculate\_area(self):

"""

Calculate the area of the shape.

"""

if self.shape\_type == "square":

return self.data.get("side", 0) \*\* 2

elif self.shape\_type == "rectangle":

return self.data.get("length", 0) \* self.data.get("width", 0)

elif self.shape\_type == "circle":

return math.pi \* self.data.get("radius", 0) \*\* 2

elif self.shape\_type == "triangle":

# Using Heron's formula for the area of a triangle

side1 = self.data.get("side1", 0)

side2 = self.data.get("side2", 0)

side3 = self.data.get("side3", 0)

semi\_perimeter = (side1 + side2 + side3) / 2

return math.sqrt(semi\_perimeter \* (semi\_perimeter - side1) \* (semi\_perimeter - side2) \* (semi\_perimeter - side3))

else:

return None

# Example usage

shape\_type = input("Enter the type of shape (square, rectangle, circle, triangle): ").strip()

shape = Shape(shape\_type)

shape.accept\_data()

perimeter = shape.calculate\_perimeter()

area = shape.calculate\_area()

if perimeter is not None and area is not None:

print(f"The perimeter/circumference of the {shape\_type} is: {perimeter:.2f}")

print(f"The area of the {shape\_type} is: {area:.2f}")

else:

print("Invalid shape type or missing data.")

Q5. Write a program that creates and uses a Time class to perform various time arithmetic operations.

class Time:

"""

A class to represent time in hours, minutes, and seconds and perform

arithmetic operations like addition and subtraction.

"""

def \_\_init\_\_(self, hours=0, minutes=0, seconds=0):

# Normalize time values

self.hours = hours

self.minutes = minutes

self.seconds = seconds

self.normalize\_time()

def normalize\_time(self):

"""

Normalize time values to ensure seconds and minutes are within valid range.

"""

# Adjust seconds to minutes

self.minutes += self.seconds // 60

self.seconds %= 60

# Adjust minutes to hours

self.hours += self.minutes // 60

self.minutes %= 60

def \_\_add\_\_(self, other):

"""

Add two Time objects.

"""

return Time(

self.hours + other.hours,

self.minutes + other.minutes,

self.seconds + other.seconds

)

def \_\_sub\_\_(self, other):

"""

Subtract two Time objects.

"""

total\_seconds\_self = self.to\_seconds()

total\_seconds\_other = other.to\_seconds()

if total\_seconds\_self < total\_seconds\_other:

raise ValueError("Resulting time cannot be negative.")

total\_seconds\_result = total\_seconds\_self - total\_seconds\_other

return Time.from\_seconds(total\_seconds\_result)

def to\_seconds(self):

"""

Convert time to total seconds.

"""

return self.hours \* 3600 + self.minutes \* 60 + self.seconds

@classmethod

def from\_seconds(cls, total\_seconds):

"""

Create a Time object from total seconds.

"""

hours = total\_seconds // 3600

total\_seconds %= 3600

minutes = total\_seconds // 60

seconds = total\_seconds % 60

return cls(hours, minutes, seconds)

def \_\_str\_\_(self):

"""

String representation of the Time object.

"""

return f"{self.hours:02d}:{self.minutes:02d}:{self.seconds:02d}"

# Example usage

time1 = Time(2, 45, 50) # 2 hours, 45 minutes, 50 seconds

time2 = Time(1, 20, 30) # 1 hour, 20 minutes, 30 seconds

print("Time 1:", time1)

print("Time 2:", time2)

# Addition

time\_addition = time1 + time2

print("Addition:", time

Q6. Write a program to create a class Date that has a list containing day, month and year attributes. Define an overloaded == operator to compare two Date object

class Date:

"""

A class to represent a date with day, month, and year attributes.

"""

def \_\_init\_\_(self, day, month, year):

self.day = day

self.month = month

self.year = year

def \_\_eq\_\_(self, other):

"""

Overload the '==' operator to compare two Date objects.

Args:

other (Date): The other Date object to compare.

Returns:

bool: True if the two dates are equal, False otherwise.

"""

return (self.day == other.day and

self.month == other.month and

self.year == other.year)

def \_\_str\_\_(self):

"""

String representation of the Date object.

"""

return f"{self.day:02d}/{self.month:02d}/{self.year}"

# Example usage

date1 = Date(5, 4, 2023)

date2 = Date(5, 4, 2023)

date3 = Date(6, 4, 2023)

print("Date 1:", date1)

print("Date 2:", date2)

print("Date 3:", date3)

# Compare dates using the overloaded '==' operator

if date1 == date2:

print("Date 1 is equal to Date 2.")

else:

print("Date 1 is not equal to Date 2.")

if date1 == date3:

print("Date 1 is equal to Date 3.")

else:

print("Date 1 is not equal to Date 3.")

Q7. Create a class Weather that has a list containing weather parameters. Define an overloaded in operator that checks whether an item is present in the list. (Hint: define the function \_\_contains\_\_( )in a class.)

class Weather:

"""

A class to represent weather parameters and overload the 'in' operator

to check for the presence of an item in the list.

"""

def \_\_init\_\_(self, parameters):

self.parameters = parameters

def \_\_contains\_\_(self, item):

"""

Check whether an item is present in the weather parameters list.

Args:

item: The item to check.

Returns:

bool: True if the item is in the list, False otherwise.

"""

return item in self.parameters

# Example usage

weather\_data = Weather(["Temperature", "Humidity", "Wind Speed", "Precipitation"])

# Check for the presence of items using the overloaded 'in' operator

print("Temperature" in weather\_data) # Output: True

print("Pressure" in weather\_data) # Output: False

# Display weather parameters

print("Weather parameters:", weather\_data.parameters)

Q8. Implement a String class containing the following functions:

a. Overloaded += operator function to perform string concatenation

b. Method toLower() to convert upper case letters to lower case.

c. Method toUpper() to convert lower case letters to upper case.

class String:

"""

A class to represent a string and perform string operations such as:

a) Overloading the `+=` operator for concatenation.

b) Converting uppercase letters to lowercase.

c) Converting lowercase letters to uppercase.

"""

def \_\_init\_\_(self, value=""):

self.value = value

def \_\_iadd\_\_(self, other):

"""

Overloads the `+=` operator to concatenate the current string with another string.

"""

if isinstance(other, String):

self.value += other.value

elif isinstance(other, str):

self.value += other

else:

raise TypeError("Operand must be of type 'String' or 'str'.")

return self

def toLower(self):

"""

Converts all uppercase characters in the string to lowercase.

"""

self.value = self.value.lower()

def toUpper(self):

"""

Converts all lowercase characters in the string to uppercase.

"""

self.value = self.value.upper()

def \_\_str\_\_(self):

"""

Returns the string representation of the object.

"""

return self.value

# Example usage

string1 = String("Hello")

string2 = String(" World!")

# Testing overloaded `+=` operator for concatenation

print("Original string1:", string1)

print("Original string2:", string2)

string1 += string2

print("After concatenation:", string1)

# Testing toLower method

string1.toLower()

print("After converting to lowercase:", string1)

# Testing toUpper method

string1.toUpper()

print("After converting to uppercase:", string1)

OUTPUT

Original string1: Hello

Original string2: World!

After concatenation: Hello World!

After converting to lowercase: hello world!

After converting to uppercase: HELLO WORLD!