**LAB CYCLE-3**

**Experiment No : 1**

**Date :** 4/11/2024

**Aim:**

Write a program to find the factorial of a number

**Pseudocode:**

1. PROMPT the user to input a number (n)
2. READ the input value into n
3. IF n < 0 THEN

PRINT "Negative number does not have factorial"

ELSE

f = calculate factorial of n using math.factorial(n)

PRINT Factorial

END IF

**Method :**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| factorial | Compute the factorial of a number. | factorial(number) |

**Source Code :**

import math

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

if n<0:

print("Negative number does not have factorial")

else:

f=math.factorial(n)

print("Factorial of ",n,"is",f)

**Output:**

Enter a number:5

Factorial of 5 is 120

Enter a number:0

Factorial of 0 is 1

Enter a number:-1

Negative number does not have factorial

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 2**

**Date :** 4/11/2024

**Aim:**

Generate Fibonacci series of N terms.

**Pseudocode:**

1. PROMPT the user to input the number of terms (N)
2. READ the input value into N
3. SET num1 = 0
4. SET num2 = 1
5. SET next\_num = num1
6. SET count = 1
7. WHILE count <= N DO

PRINT next\_num with a space (no newline)

INCREMENT count by 1

SET num1, num2 = num2, next\_num

SET next\_num = num1 + num2

END WHILE

**Source Code :**

N=int(input("Enter the no.of terms:"))

num1=0

num2=1

next\_num=num1

count=1

while count<=N:

print(next\_num,end=" ")

count+=1

num1,num2=num2,next\_num

next\_num=num1+num2

**Output:**

Enter the no.of terms:5

0 1 1 2 3

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 3**

**Date :** 4/11/2024

**Aim:**

Write a program to find the sum of all items in a list. [Using for loop]

**Pseudocode:**

1. CREATE an empty list nlist
2. PROMPT the user to input the number of elements (n)
3. READ the input value into n
4. FOR i FROM 0 TO n-1 DO

PROMPT the user to enter a number

READ the input value into num

APPEND num to nlist

END FOR

1. SET sum = 0
2. FOR each element i in nlist DO

ADD i to sum

END FOR

1. PRINT "Sum of all items in the list:" and the value of sum

**Source Code :**

nlist=[]

n=int(input("Enter no.of element:"))

for i in range (n):

num=int(input("Enter the numbers:"))

nlist.append(num)

sum=0

for i in nlist:

sum+=i

print("Sum of all items in the list:",sum)

**Output:**

Enter no.of element:5

Enter the numbers:1

Enter the numbers:5

Enter the numbers:7

Enter the numbers:9

Enter the numbers:4

Sum of all items in the list: 26

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 4**

**Date :** 4/11/2024

**Aim:**

Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.

**Pseudocode:**

1. DEFINE function even\_square(start, end):
2. CREATE an empty list result
3. SET l\_bound = the ceiling of the square root of start
4. SET u\_bound = the floor of the square root of end
5. FOR i FROM l\_bound to u\_bound DO

SET square = i squared (i\*\*2)

IF all digits of square are even THEN

APPEND square to result

END IF

END FOR

1. RETURN result
2. PROMPT the user to input the starting range (start\_range) as a 4-digit number
3. READ the input value into start\_range
4. PROMPT the user to input the ending range (end\_range) as a 4-digit number
5. READ the input value into end\_range
6. CALL even\_square(start\_range, end\_range) to get the list of perfect squares with even digits
7. STORE the result in even\_digit
8. PRINT even\_digit

**Source Code :**

import math

def even\_square(start,end):

result=[]

l\_bound=math.ceil(math.sqrt(start))

u\_bound=math.floor(math.sqrt(end))

for i in range(l\_bound,u\_bound+1):

square=i\*\*2

if all(int(digit)%2==0 for digit in str(square)):

result.append(square)

return result

start\_range=int(input("Enter starting range(4 digits):"))

end\_range=int(input("Enter ending range(4 digits):"))

even\_digit=even\_square(start\_range,end\_range)

print("Four digit perfect squares with an even digits are:",even\_digit)

**Output:**

Enter starting range(4 digits):1000

Enter ending range(4 digits):5000

Four digit perfect squares with an even digits are: [4624]

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 5**

**Date :** 4/11/2024

**Aim:**

Write a program using a for loop to print the multiplication table of n, where n is entered by the user.

**Pseudocode:**

1. PROMPT the user to input a number (n)
2. PRINT Multiplication table of n
3. FOR i FROM 1 to 10 DO

PRINT n \* i

END FOR

**Source Code :**

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

print("Multiplication table of ",n)

for i in range(1,11):

print(n,'x',i,'=',n\*i)

**Output:**

Enter the number:5

Multiplication table of 5

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 6**

**Date :** 4/11/2024

**Aim:**

Write a program to display alternate prime numbers till N (obtain N from the user).

**Pseudocode:**

1. Define function is\_prime(num)
2. Define function Alt\_primes(n)
3. PROMPT the user to input the value of N
4. READ the input value into N
5. Call Alt\_primes(N)
6. PRINT Alt\_primes(N)

is\_prime(num)

1. IF num <= 1 THEN

RETURN False

END IF

1. FOR i FROM 2 TO square root of num DO

IF num is divisible by i THEN

RETURN False

END IF

END FOR

1. RETURN True

Alt\_primes(n)

1. CREATE an empty list primes
2. FOR i FROM 2 TO n DO

IF is\_prime(i) THEN

APPEND i to primes

END IF

END FOR

1. CREATE alt\_primes by taking every second element from primes using primes[::2]
2. PRINT "Prime numbers up to", n, "are:", primes
3. RETURN alt\_primes

**Source Code :**

def is\_prime(num):

if num<=1:

return False

for i in range(2,int(num \*\*0.5)+1):

if num%i==0:

return False

return True

def Alt\_primes(n):

primes=[]

for i in range(2,n+1):

if is\_prime(i):

primes.append(i)

alt\_primes=primes[::2]

print("prime numbers upto ",N," are: ",primes)

return alt\_primes

N=int(input("Enter the value of N: "))

print("Alternative prime numbers upto ",N," are:",Alt\_primes(N))

**Output:**

Enter the value of N: 5

prime numbers upto 5 are: [2, 3, 5]

Alternative prime numbers upto 5 are: [2, 5]

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 7**

**Date :** 4/11/2024

**Aim:**

Write a program to compute and display the sum of all integers that are divisible

by 6 but not by 4, and that lie below a user-given upper limit.

**Pseudocode:**

1. PROMPT the user to input the upper limit (n)
2. READ the input value into n
3. SET sum = 0
4. FOR i FROM 1 TO n-1 DO

IF i is divisible by 6 AND i is NOT divisible by 4 THEN

ADD i to sum

END IF

END FOR

1. PRINT sum

**Source Code :**

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

sum=0

for i in range(1,n):

if i%6==0 and i%4!=0:

sum+=i

print("Sum=",sum)

**Output:**

Enter the upperlimit:20

Sum= 24

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 8**

**Date :** 4/11/2024

**Aim:**

Calculate the sum of the digits of each number within a specified range (from1 to

a user-defined upper limit). Print the sum only if it is prime.

**Pseudocode:**

1. PROMPT the user to input the upper limit (Upper\_limit)
2. READ the input value into Upper\_limit
3. PRINT "Prime numbers are:"
4. FOR i FROM 1 TO Upper\_limit-1 DO

SET temp = i ,SET sum = 0 // Calculate the sum of digits of i

WHILE temp > 0 DO

SET digit = temp MOD 10

SET temp = temp DIV 10

ADD digit to sum

END WHILE

SET flag = 0

IF sum <= 1 THEN

CONTINUE to the next iteration

END IF

FOR j FROM 2 to sum-1 DO

IF sum MOD j == 0 THEN

SET flag = 1

BREAK the loop

END FOR

IF flag == 0 THEN

PRINT sum

**Source Code :**

Upper\_limit=int(input("Enter the Upper limit: "))

print("prime number are:")

for i in range(1,Upper\_limit):

temp=i

sum=0

while temp>0:

digit=temp%10

temp=temp//10

sum=sum+digit

flag=0

if sum<=1:

continue

for j in range(2,sum):

if sum%j==0:

flag=1

if flag==0:

print("sum of ",i," = ",sum)

**Output:**

Enter the Upper limit: 15

prime number are:

sum of 2 = 2

sum of 3 = 3

sum of 5 = 5

sum of 7 = 7

sum of 11 = 2

sum of 12 = 3

sum of 14 = 5

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 9**

**Date :** 4/11/2024

**Aim:**

A number is input through the keyboard. Write a program to determine if it’s palindromic.

**Pseudocode:**

1. PROMPT the user to input a number (n)
2. SET num1 = n , num2 = 0 ,r = 0
3. WHILE n is not equal to 0 DO

SET r = n MOD 10 , num2 = (num2 \* 10) + r , n = n DIV 10

END WHILE

1. PRINT num2
2. IF num1 is equal to num2 THEN

PRINT num1, "is a palindrome number"

ELSE

PRINT num1, "is not a palindrome number"

END IF

**Source Code :**

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

num1=n

num2=0

r=0

while (n!=0):

r=n%10

num2=(num2\*10)+r

n=n//10

print(num2)

if num1==num2:

print(num1,"is a palindrome number")

else:

print(num1,"is not a palindrome number")

**Output:**

Enter a number:343

343

343 is a palindrome number

Enter a number:564

465

564 is not a palindrome number

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 10**

**Date :** 4/11/2024

**Aim:**

Write a program to generate all factors of a number. [use while loop]

**Pseudocode:**

1. PROMPT the user to input a number (n)
2. PRINT Factors of n
3. SET i = 1
4. WHILE i <= n DO

IF n MOD i == 0 THEN

PRINT i

INCREMENT i by 1

END WHILE

**Source Code :**

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

print("Factors of ",n)

i=1

while(i<=n):

if n%i==0:

print(i,end=" ")

i+=1

**Output:**

Enter a number:10

Factors of 10

1 2 5 10

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 11**

**Date :** 4/11/2024

**Aim:**

Write a program to find whether the given number is an Armstrong number or not. [use while loop]

**Pseudocode:**

1. PROMPT the user to enter a number (n)
2. STORE the number in temp
3. CALCULATE the number of digits in n (ln = length of str(n))
4. SET sum = 0
5. WHILE temp > 0 DO

EXTRACT the last digit (r = temp MOD 10)

ADD (r ^ ln) to sum

REMOVE the last digit from temp (temp = temp DIV 10)

END WHILE

1. IF n == sum THEN

PRINT n, "is an Armstrong number"

ELSE

PRINT n, "is not an Armstrong number"

END IF

**Source Code :**

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

sum=0

temp=n

ln=len(str(n))

while temp>0:

r=temp%10

sum=sum+(r\*\*ln)

temp//=10

if n==sum:

print(n," is an armstrong number")

else:

print(n," is not an armstrong number")

**Output:**

Enter a number:123

123 is not an Armstrong number

Enter a number:407

407 is an armstrong number

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 12**

**Date :** 4/11/2024

**Aim:**

Display the given pyramid with the step number accepted fromthe user.

Eg: N=4

1

2 4

3 6 9

4 8 12 16

**Pseudocode:**

1. PROMPT the user to enter the number of rows (r)
2. FOR i FROM 1 TO r DO

FOR j FROM 1 TO i DO

PRINT (i \* j) with a space

END FOR

PRINT a new line after each row

END FOR

**Source Code :**

r=int(input("Enter no.of rows:"))

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

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

print(i\*j,end=" ")

print()

**Output:**

Enter no.of rows:4

1

2 4

3 6 9

4 8 12 16

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 13**

**Date :** 4/11/2024

**Aim:**

Construct following pattern using nested loop

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**Pseudocode:**

1. PROMPT the user to enter the number of rows (r)
2. FOR i FROM 0 TO r-1 DO

FOR j FROM 0 TO i DO

PRINT "\*"

END FOR

PRINT a new line

END FOR

1. FOR i FROM r TO 1 DO

FOR j FROM 0 TO i-1 DO

PRINT "\*"

END FOR

PRINT a new line

END FOR

**Source Code :**

r=int(input("Enter no.of rows:"))

for i in range(0,r):

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

print("\*",end=' ')

print()

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

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

print("\*",end=' ')

print()

**Output:**

Enter no.of rows:4

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**Result:**

The program is successfully executed and the output is verified.

**LAB CYCLE-4**

**Experiment No : 1**

**Date :** 11/11/2024

**Aim:**

Write a program to print the Fibonacci series using recursion

**Pseudocode:**

1. Define function Fibonacci(n)
2. PROMPT the user to enter the number of terms (limit)
3. IF limit <= 0 THEN

PRINT "Please enter a positive integer"

ELSE

PRINT "Fibonacci series:"

END IF

1. FOR i FROM 0 TO limit-1 DO:

PRINT fibonacci(i)

END FOR

fibonacci(n)

1. IF n <= 1 THEN

RETURN n

ELSE

RETURN fibonacci(n-1) + fibonacci(n-2)

END IF

**Source Code :**

def fibonacci(n):

if n<=1:

return n

else:

return fibonacci(n-1)+fibonacci(n-2)

limit=int(input("Enter no.of terms:"))

if limit<=0:

print("Please enter a positive integer")

else:

print("Fibonacci series:")

for i in range(limit):

print(fibonacci(i))

**Output:**

Enter no.of terms:5

Fibonacci series:

0

1

1

2

3

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 2**

**Date :** 11/11/2024

**Aim:**

Write the to implement a menu-driven calculator. Use separate functions for the

different operations.

**Pseudocode:**

1. Define function add(x, y)
2. Define function sub(x, y)
3. Define function mul(x, y)
4. Define function div(x, y)
5. PROMPT user to enter the first value (a)
6. PROMPT user to enter the second value (b)
7. WHILE True DO

PRINT menu options: 1) Addition 2) Subtraction 3) Multiplication

4) Division

PROMPT user to enter a choice (ch)

IF ch == 1 THEN

PRINT "Sum =", add(a, b)

ELSE IF ch == 2 THEN

PRINT "Difference =", sub(a, b)

ELSE IF ch == 3 THEN

PRINT "Product =", mul(a, b)

ELSE IF ch == 4 THEN

PRINT "Division =", div(a, b)

ELSE

PRINT "Invalid choice...exit..."

EXIT the program

END IF

END WHILE

add(x, y)

1. RETURN x + y

sub(x, y)

1. RETURN x – y

mul(x, y)

1. RETURN x \* y

div(x, y)

1. IF y > 0 THEN

RETURN x / y

ELSE

PRINT "Not possible"

END IF

**Source Code :**

def add(x,y):

return x+y

def sub(x,y):

return x-y

def mul(x,y):

return x\*y

def div(x,y):

if y>0:

return x/y

else:

print("Not posiible")

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

b=int(input("Enter second value"))

while(1):

print("MENU \n 1)Addition \n 2)Subtraction \n 3)Multiplication \n 4)Division")

ch=int(input("Enter your choice:"))

if ch==1:

print("Sum=",add(a,b))

elif ch==2:

print("Difference=",sub(a,b))

elif ch==3:

print("Product=",mul(a,b))

elif ch==4:

print("Division=",div(a,b))

else:

print("Invalid choice...exit...")

exit(0)

**Output:**

Enter first value:5

Enter second value7

MENU

1)Addition

2)Subtraction

3)Multiplication

4)Division

Enter your choice:1

Sum= 12

MENU

1)Addition

2)Subtraction

3)Multiplication

4)Division

Enter your choice:2

Difference= -2

MENU

1)Addition

2)Subtraction

3)Multiplication

4)Division

Enter your choice:3

Product= 35

MENU

1)Addition

2)Subtraction

3)Multiplication

4)Division

Enter your choice:4

Division= 0.7142857142857143

MENU

1)Addition

2)Subtraction

3)Multiplication

4)Division

Enter your choice:5

Invalid choice...exit...

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 3**

**Date :** 11/11/2024

**Aim:**

Write a program to print the nth prime number. [Use function to check whether a number is prime or not]

**Pseudocode:**

1. Define function is\_prime(num)
2. Define function nth\_prime(n)
3. PROMPT the user to enter the position of the prime number (n)
4. IF n <= 0 THEN

PRINT "Invalid Input!!"

ELSE

PRINT "The nth prime number is:", nth\_prime(n)

END IF

is\_prime(num)

1. IF num <= 1 THEN

RETURN False

END IF

1. FOR i FROM 2 TO sqrt(num) DO:

IF num % i == 0 THEN

RETURN False

END FOR

1. RETURN True

nth\_prime(n)

SET count = 0 ,number = 2

WHILE True DO:

IF is\_prime(number) THEN

count = count + 1

IF count == n THEN

RETURN number

END IF

END IF

number = number + 1

END WHILE

**Source Code :**

def is\_prime(num):

if num<=1:

return False

for i in range(2,int(num \*\* 0.5)+1):

if num%i==0:

return False

return True

def nth\_prime(n):

count=0

number=2

while True:

if is\_prime(number):

count+=1

if count==n:

return number

number+=1

n=int(input("Enter the Positon of Prime number: "))

if n<=0:

print("Invalid Input!!")

else:

print(f"The {n} the prime number is:{nth\_prime(n)}")

**Output:**

Enter the Positon of Prime number: 5

The 5 the prime number is:11

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 4**

**Date :** 11/11/2024

**Aim:**

Write lambda functions to find the area of square, rectangle and triangle.

**Pseudocode:**

1. DEFINE area\_square = lambda S\_side: S\_side^2
2. DEFINE area\_rectangle = lambda rect\_length, rect\_width: rect\_length \* rect\_width
3. DEFINE area\_triangle = lambda t\_base, t\_height: 0.5 \* t\_base \* t\_height
4. PROMPT user to enter the side of the square (S\_side)
5. PRINT "Area of Square: ", area\_square(S\_side)
6. PROMPT user to enter the length of the rectangle (rect\_length)
7. PROMPT user to enter the width of the rectangle (rect\_width)
8. PRINT "Area of Rectangle: ", area\_rectangle(rect\_length, rect\_width)
9. PROMPT user to enter the base of the triangle (t\_base)
10. PROMPT user to enter the height of the triangle (t\_height)
11. PRINT "Area of Triangle: ", area\_triangle(t\_base, t\_height)

**Method :**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| lambda | The lambda function can have any number of input parameters, but it can only contain a single expression. The result of the expression is implicitly returned. | lambda arguments: expression |

**Source Code :**

area\_square=lambda S\_side:S\_side \*\*2

area\_rectangle=lambda rect\_length,rect\_width:rect\_length \* rect\_width

area\_triangle=lambda t\_base,t\_height:0.5 \* t\_base \* t\_height

S\_side=int(input("Enter Square side: "))

print("Area of Square: ",area\_square(S\_side))

rect\_length=int(input("Enter Rectangle length: "))

rect\_width=int(input("Enter Rectangle width: "))

print("Area of rectangle: ",area\_rectangle(rect\_length,rect\_width))

t\_base=int(input("Enter Triangle base: "))

t\_height=int(input("Enter Triangle height: "))

print("Area of Triangle: ",area\_triangle(t\_base,t\_height))

**Output:**

Enter Square side: 2

Area of Square: 4

Enter Rectangle length: 6

Enter Rectangle width: 4

Area of rectangle: 24

Enter Triangle base: 2

Enter Triangle height: 7

Area of Triangle: 7.0

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 5**

**Date :** 11/11/2024

**Aim:**

Write a program to display powers of 2 using anonymous function. [ Hint use map and lambda function]

**Pseudocode:**

1. CREATE empty list lt
2. PROMPT user to enter the number of terms (n)
3. FOR i FROM 1 TO n DO:

PROMPT user to enter a term (terms)

ADD terms to the list lt

END FOR

1. DEFINE lambda function twox(x) as 2^x
2. APPLY map(twox, lt) to calculate the powers of 2 for each element
3. PRINT the list of powers of 2

**Method :**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| map | The map() function in Python applies a given function to all items in an iterable and returns an iterator with the results. | map(function, iterable) |

**Source Code :**

lt=[]

n=int(input("Enter no.of terms:"))

for i in range(n):

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

lt.append(terms)

twox=lambda x:2\*\*x

power\_of\_2=map(twox,lt)

print("Powers of 2:")

power\_fnctn\_list=list(power\_of\_2)

print(power\_fnctn\_list)

**Output:**

Enter no.of terms:4

Enter terms: 2

Enter terms: 4

Enter terms: 6

Enter terms: 8

Powers of 2:

[4, 16, 64, 256]

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 6**

**Date :** 18/11/2024

**Aim:**

Write a program to display multiples of 3 using anonymous function. [ Hint use filter and lambda function]

**Pseudocode:**

1. PROMPT user to input the range (r)
2. CREATE an empty list lt
3. FOR i FROM 1 TO r

PROMPT user to input a number (n)

ADD n to the list lt

END FOR

1. DEFINE a lambda function to check if x is divisible by 3 (x % 3 == 0)
2. APPLY filter function on the list lt using the lambda function to get multiples of 3
3. PRINT the list of multiples of 3

**Method :**

|  |  |  |
| --- | --- | --- |
| Function | Description | Syntax |
| filter | The filter() function in Python filters elements from an iterable based on a given condition (function) and returns an iterator containing only those elements for which the condition is True. | filter(function, iterable) |

**Source Code :**

r=int(input("Enter range:"))

lt=[]

for i in range(r):

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

lt.append(n)

numbers=lt

multiples\_of\_3=list(filter(lambda x:x%3==0,numbers))

print("Multiples of 3:",multiples\_of\_3)

**Output:**

Enter range:5

Enter numbers:1

Enter numbers:5

Enter numbers:23

Enter numbers:6

Enter numbers:9

Multiples of 3: [6, 9]

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 7**

**Date :** 18/11/2024

**Aim:**

Write a program to sum the series 1/1! + 4/2! + 27/3! + ….. + nth term. [ Hint Use a function to find the factorial of a number].

**Pseudocode:**

1. Define function factorial(n)
2. Define function nth\_term(n)
3. Define function series\_sum(n)
4. PROMPT user to input the number of terms (n)
5. PRINT "Sum of the series:" + series\_sum(n)

factorial(n)

1. RETURN math.factorial(n)

nth\_term(n)

1. RETURN n^3 / factorial(n)

series\_sum(n)

1. INITIALIZE sum = 0
2. FOR i FROM 1 TO n DO
   1. sum = sum + nth\_term(i)
3. RETURN sum

**Source Code :**

import math

def factorial(n):

return math.factorial(n)

def nth\_term(n):

return n\*\*3/factorial(n)

def series\_sum(n):

sum=0

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

sum+=nth\_term(i)

return sum

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

print("Sum of the series:",series\_sum(n))

**Output:**

Enter the number of terms:4

Sum of the series: 12.166666666666666

Enter the number of terms:5

Sum of the series: 13.208333333333332

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 8**

**Date :** 18/11/2024

**Aim:**

Write a function called compare which takes two strings S1 and S2 and an integer n as arguments. The function should return True if the first n characters of both the strings are the same else the function should return False.

**Pseudocode:**

1. Define function Compare(n, s1, s2)
2. PROMPT user to input first string (s1)
3. PROMPT user to input second string (s2)
4. PROMPT user to input value of n
5. r = Compare(n, s1, s2)
6. IF r is True

PRINT "first n characters of s1 and s2 are same"

ELSE

PRINT "first n characters of s1 and s2 are not same"

END IF

Compare(n, s1, s2)

1. IF first n characters of s1 are equal to first n characters of s2

RETURN True

ELSE

RETURN False

END IF

**Source Code :**

def Compare(n,s1,s2):

if s1[:n]==s2[:n]:

return True

else:

return False

s1=input("Enter first String: ")

s2=input("Enter second String: ")

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

r=Compare(n,s1,s2)

if r==True:

print(f"first {n} characters of {s1} and {s2} are same")

elif r==False:

print(f"first {n} characters of {s1} and {s2} are not same")

**Output:**

Enter first String: nandana

Enter second String: nanda

Enter n value: 3

first 3 characters of nandana and nanda are same

Enter first String: adcd

Enter second String: efg

Enter n value: 2

first 2 characters of adcd and efg are not same

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 9**

**Date :** 18/11/2024

**Aim:**

Write a program to add variable length integer arguments passed to the function. [Also demo the use of docstrings]

**Pseudocode:**

1. Define function add\_numbers(\*args)
2. IF not all arguments are integers

RAISE ValueError("All arguments must be integers!!")

END IF

1. RETURN sum of all arguments
2. PRINT sum of 1, 2, 3
3. PRINT sum of 10, 20, 30, 40

**Source Code :**

def add\_numbers(\*args):

""" Adds a variable number of integer arguments.

parameters:

\*args:A variable length list of Integers to be added.

returns:

int:the sum of all the integers passed as argumens.

"""

if not all(isinstance(arg,int)for arg in args):

raise valueError("All arguments must be integers!!")

return sum(args)

print("sum of 1,2,3:",add\_numbers(1,2,3))

print("sum of 10,20,30,40:",add\_numbers(10,20,30,40))

**Output:**

sum of 1,2,3: 6

sum of 10,20,30,40: 100

**Result:**

The program is successfully executed and the output is verified.

**Experiment No : 10**

**Date :** 18/11/2024

**Aim:**

Write a program using functions to implement these formulae for permutations and combinations.

The Number of permutations of n objects taken r at a time: p(n, r) = n!/(n − r)!.

The Number of combinations of n objects taken r at a time is:

c(n, r) = n!/(r! ∗ (n−r)!)

**Pseudocode:**

1. Define function factorial(num)
2. Define function Permutation(n, r)
3. Define function Combination(n, r)
4. INPUT n and r
5. PRINT Permutation(n, r) and Combination(n, r)

factorial(num)

1. IF num is 0 OR 1

RETURN 1

ELSE

INITIALIZE fact as 1

END IF

1. FOR i FROM 2 TO num

fact = fact \* i

END FOR

1. RETURN fact

Permutation(n, r)

RETURN factorial(n) // factorial(n - r)

Combination(n, r)

RETURN factorial(n) // (factorial(r) \* factorial(n - r))

**Source Code :**

def factorial(num):

if num==1 or num==0:

return 1

else:

fact=1

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

fact=fact\*i

return fact

def Permutation(n,r):

return factorial(n) // factorial(n-r)

def Combination(n,r):

return factorial(n) // (factorial(r) \* factorial(n-r))

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

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

print(f"P({n},{r}):{Permutation(n,r)}")

print(f"C({n},{r}):{Combination(n,r)}")

**Output:**

Enter the n value: 3

Enter the r value: 4

P(3,4):6

C(3,4):0

Enter the n value: 4

Enter the r value: 2

P(4,2):12

C(4,2):6

**Result:**

The program is successfully executed and the output is verified.