LAB CYCLE 3

Experiment No :1

Date: 11/11/2024

Aim:

Write a program to find factorial of a number.

Pseudocode:

```
1. Read the number as n
2. Check if n<0 then
      Print No factorial for negative numbers
    Else
      Calculate f=math.factorial(n)
3. Print factorial
```

Source code:

```
Import math
n=int(input("Enter the number:"))
If n<0:
      print("No factorial for negative numbers")
else:
      f=math.factorial(n)
      print("Factorial=",f)
```

Output:

```
Enter the number: 3
Factorial= 6
```

Result:

The program is successfully executed and the output is verified.

Experiment No :2

Date:11/11/2024

Aim:

Generate fibonacci series of N terms.

Pseudocode:

```
1. Read the number of terms
2. Set a=0 ,b=1
3. Set c=a
4. Set count=1
5. While count <= n,then
     Print c
     Increment count by 1
     Set a=b
     Set b=c
     Set c=a+b
   End while
```

Source code:

```
n=int(input("Enter the number:"))
a,b,=0,1
c=a
count=1
 while count<=n:
         print(c,end=" ")
         count+=1
         a,b=b,c
         c=a+b
```

Output:

```
Enter the number:
0 1 1 2
```

Result:

The program is successfully executed and the output is verified.

Experiment No :3

Date:11/11/2024

Aim:

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

Pseudocode:

```
1. Read the number of terms.
2. Create an empty list.
3. Print enter the numbers
4. For each number from 1 to n
      Read numbers
      Append number to the list
   End for
5. Calculate sum of list, sum=sum(list)
6. Print sum
```

Source code:

```
nlist=[]
n=int(input("Enter the number of elements:"))
print("Enter the numbers:")
for i in range(n):
        num=int(input())
        nlist.append(num)
sum=0
for i in nlist:
    sum+=i
print(“sum of all items in the list:”,sum)
```

Output:

```
Enter the number of elements:4
Enter the numbers:
1
2
3
4
Sum of all items in the list:10
```

Result:

The program is successfully executed and the output is verified.

Experiment No :4

Date:11/11/2024

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. Create an empty list.
2. For each number from 32 to 99
      Calculate square of num and store in square
      If square is a four digit number(1000<=square<=9999) then
                Convert square to string square_str
                If all digits in square_str are even then
                          Append square to the list
                End if
      End if
   End for
3. Print list.
```

Source code:

```
even_digit_squares = []
for num in range(32, 100):
        square = num * num
        if 1000 <= square <= 9999:
        square_str = str(square)
        if (square_str[0] in "02468" and
    square_str[1] in "02468" and
    square_str[2] in "02468" and
    square_str[3] in "02468"):
```

```
even_digit_squares.append(square)
print("Four-digit numbers that are perfect squares with all even digits:",
even_digit_squares)
```

Output:

```
Four-digit numbers that are perfect squares with all even digits: [4624,6084,6400,8464]
```

Result:

The program is successfully executed and the output is verified.

Experiment No :5

Date:11/11/2024

Aim:

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

Pseudocode:

```
1. Read the number.
2. For each number i from 1 to 10
      Calculate product n * i
      Print product
   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:11/11/2024

Aim:

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

Pseudocode:

```
1. Read the number
2. Set count =0
3. For each number from 2 to n
    Iniitialize is_prime to True
    For each number i from 2 to the square root of number
        If num divisible by i then
            Set is_prime to False
            Break out of the loop
        End if
    End for
4. If is_prime is True then
    If count %2==0 then
        Print num
    End if
    Increment count by 1
  End if
5. Print new line
```

Source code:

```
n=int(input("Enter the number:"))
count=0
print("Alternate primes")
for num in range(2,n+1):
        is_prime=True

        for i in range(2,int(num**0.5)+1):
        if num%i==0:
        is_prime=False
        break
if is_prime:
        if count%2==0:
        print(num,end=" ")
        count+=1
print()
```

Output:

```
Enter the number:10
Alternate primes
2 5
```

Result:

The program is successfully executed and the output is verified.

Experiment No :7

Date:11/11/2024

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. Read the upper limit
2. Initialize tsum=0
3. For each number i from 1 to l-1
      If i %6 equal to 0 and i % 4 not equal to 0 then
              Add i to tsum
      End if
   End for
4. Print tsum
```

Source code:

```
l=int(input("Enter the upper limit:"))
tsum=0
for i in range(1,l):
        if i%6==0 and i%4!=0 then
        tsum=tsum+i
print("Sum=",tsum)
```

Output:

```
Enter the upper limit: 20
Sum= 24
```

Result:

The program is successfully executed and the output is verified.

Experiment No: 8

Date:11/11/2024

Aim:

Calculate the sum of the digits of each numbewithin a specified range (from 1 to a user defined upper limit). Print the sum only if it is prime.

Pseudocode:

```
1. Read the upper limit
2. For each number num from t to upper limit
      Initialize digit_sum to 0
      Set num to temp
      While temp >0
              Add last digit of temp to digit-sum
              Remove last digit from temp
      End while
      If digit_sum<=1 then
              Continue
      End if
   End for
3. Initialize is_prime to True
4. For each number i from 2 to square root of digit_sum
      If digit_sum is divisible by i then
              Set is_prime to False
              Break out of the loop
      End if
   End for
5. If is_prime is True then
      Print digit sum
```

Source code:

```
upper_limit = int(input("Enter the upper limit: "))
print("Sum of digits (prime values only) for each number in the range:”)
for num in range(1, upper_limit + 1):
        digit_sum = 0
        temp = num
        while temp > 0:
        digit_sum += temp % 10
        temp //= 10
if digit_sum <= 1:
        continue
is_prime = True
for i in range(2, int(digit_sum**0.5) + 1):
        if digit_sum % i == 0:
        is_prime = False
        break
if is_prime:
        print(f"Number: {num}, Sum of Digits: {digit_sum}")
```

Output:

```
Sum of digits (prime values only) for each number in the range:
Number: 2, Sum of Digits: 2
Number: 3, Sum of Digits: 3
Number: 5, Sum of Digits: 5
Number: 7, Sum of Digits: 7
```

Result:

The program is successfully executed and the output is verified.

Experiment No :9

Date:11/11/2024

Aim:

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

Pseudocode:

```
1. Read year.
2. if number equal to reverse(n[::-1]) then
      Print palindrome
   else
      Print Not palindrome
   end if
```

Source code:

```
n=input("Enter number:")
if n==n[::-1]:
        print("Palindrome")
else:
        print("Not palindrome")
```

Output:

```
Enter number: 121
Palindrome
Enter number: 678
Not palindrome
```

Result:

The program is successfully executed and the output is verified.

Experiment No : 10

Date:11/11/2024

Aim:

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

Pseudocode:

```
1. Read the number as n
2. Initialize f=1
3. While f<=n
       Check if n%f==0 then
                 Print f
       Increment f by 1
   End while
```

Source code:

```
n=int(input("Enter the number:"))
fact=1
print("Factors are:")
while fact<=n:
          if n%fact==0:
          print(fact)
fact=fact+1
```

Output:

```
Enter the number: 6
Factors are:
1
2
3
6
```

Result:

The program is successfully executed and the output is verified.

Experiment No :11

Date:11/11/2024

Aim:

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

Pseudocode:

```
1. Read the number
2. Initialize sum=0
3. Set temp=n
4. Calculate number of digits and store in num
5. While n>0
      Calculate r=r%10
      Calculate sum=sum+r**num
      Set n=n//10
   End while
6. If temp=sum
      Print amstrong number
   Else
      Print Not amstrong number
```

Source code:

```
n=int(input("Enter a number:"))
sum=0
temp=n
num=len(str(n))
while n>0:
        r=n%10
        sum+=r**num
```

```
n//=10
if temp==sum:
        print("Amstrong number")
else:
        print("Not amstrong number")
```

Output:

```
Enter a number: 153
Amstrong number
Enter a number: 456
Not amstrong number
```

Result:

The program is successfully executed and the output is verified.

Experiment No :12

Date:11/11/2024

Aim:

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

Eg: N=4

```
1
2 4
3 6 9
4 8 12 16
```

Pseudocode:

```
1. Read the number
2. For each number i from 1 to n
       For each j from 1 to i
              Print i*j
       End for
          Print new line
   End for
```

Source code:

```
n=int(input("Enter the number:"))
for i in range(1,n+1):
        for j in range(1,i+1):
        print(i*j,end=" ")
        print()
```

Output:

```
Enter the number: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:11/11/2024

Aim:

Construct the following pattern using nested loop.

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

Pseudocode:

```
1. Read the number
2. For each number i from 1 to n
    For each j from 1 to i
        Print * with space
    End for
      Print new line
  End for
3. For each number i from n-1 down to 1
    For each j from 1 to i
        Print * with space
    End for
      Print new line
  End for
```

Source code:

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

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

        for j in range(i):
             print("*",end=" ")
        print()
for i in range(n-1,0,-1):
        for j in range(i):
        print("*",end=" ")
print()
```

Output:

```
Enter the number: 4
*
**
***
****
***
**
*
```

Result:

The program is successfully executed and the output is verified.

LAB CYCLE 4

Experiment No :1

Date:18/11/2024

Aim:

Write a program to print the Fibonacci series using recursion.

Pseudocode:

```
1. Define function fibrecur(a,b,n)
2. If n<0 then
       Return
   print(a)
   End if
   Set c=a+b
   Set a=b
   Set b=c
   Call fibrecur(a,b,n-1) function
3. Read the number of terms
4. If n<0 then
       Print Enter positive numbers
   Else:
       Print fibonacci series
       Call fibrecur(0,1,n)
  End if
```

Source code:

```
def fibrecur(a,b,n):
if n<0:
        return
        print(a)
        c=a+b
        a=b
        b=c
        fibrecur(a,b,n-1)
n=int(input("Enter the no: of terms:"))
if n<0:
        print("Enter positive nos")
else:
        print("Fibonocci series:")
        fibrecur(0,1,n)
```

Output:

```
Enter the no: of terms: 4
Fibonocci series:
0
1
1
2
3
```

Result:

The program is successfully executed and the output is verified.

Experiment No :2

Date:18/11/2024

Aim:

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

Pseudocode:

```
1. Define function add(x,y)
    Return x+y
2. Define function sub(x,y)
    Return x-y
3. Define function mul(x,y)
    Return x*y
4. Define function div(x,y)
    If y>0 then
            Return x/y
    Else:
            Print Not possible
    End if
5. Read the first number
6. Read the second number
7. While True
    Print the options for operations
    Read the choice
    If choice=1 then
            Call add function
    Else If choice=2 then
            Call subtraction function
    Else If choice=3 then
```

```
     Call multiplicationfunction
     Else If choice=4 then
          Call division function
     Else:
          Print Invalid choice
          Exit the program
    End while
```

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 possible")
a=int(input("Enter the first number:"))
b=int(input("Enter the second number:"))
while(1):
print("\n1. Addition\n2. Subtraction\n3. Multiplication\n4. Division")
ch=int(input("Enter your choice:"))
if ch==1:
     print("Addition:",add(a,b))
elif ch==2:
     print("Subtraction:",sub(a,b))
 elif ch==3:
     print("Multiplication:",mul(a,b))
elif ch==4:
     print("Division:",div(a,b))
else:
     print("Invalid choice")
     exit(0)
```

Output:

```
Enter the first number:5
Enter the second number:10
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice:1
Addition: 15
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice:2
Subtraction: -5
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice:3
Multiplication: 50
1. Addition
2. Subtraction
```

```
3. Multiplication
4. Division
Enter your choice:4
Division: 0.5
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice:6
Invalid choice
```

Result:

The program is successfully executed and the output is verified.

Experiment No : 3

Date:18/11/2024

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. If num<1 then
       Return False
    End if
3. For each num i from 2 to square root of num
       If num % i=0 then
              Return False
       End if
     Return True
     End for
4. Define a function nth_prime(n)
       Initialize count=0
       Initialize number=2
       While True
              If number is prime(function call) then
                     Increment count by 1
                     If count=n then
                            Return number
                     End if
              Increment number by 1
          End if
          End while
```

```
5. If n < 0 then
    Print Invalid input
  Else:
    Print nth prime calling nth_prime function
   End if
```

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 i
        if count==n:
        return number
        number+=1
n=int(input("Enter the position of prime number:"))
if n<0:
        print("Invalid input")
else:
        print(f"{n} th prime number is {nth_prime(n)}")
```

Output:

```
Enter the position of prime number: 7
7 th prime number is 17
```

Result:

The program is successfully executed and the output is verified.

Experiment No :4

Date:18/11/2024

Aim:

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

Pseudocode:

```
1. Define lambda function area_square(s_side)
     Return s_side **2
2. Define lambda function area_rectangle(rect_length,rect_width)
     Return rect_length*rect_width
3. Define lambda function area_trianglee(t_base,t_height)
     Return 0.5* t_base*t_height
4. Read the side of the square
5. Print area of square by calling the function
6. Read the length of the rectangle
7. Read the breadth of the rectangle
8. Print area of rectangle by calling the function
9. Read base of triangle
10. Read height of triangle
11. Calculate area of triangle by calling the function
```

Method:

Function	Description	Syntax
lambda	Create small, single expression functions without defining using def	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: 3
Area of Square: 9
Enter Rectangle length: 4
Enter Rectangle width: 5
Area of Rectangle: 20
Enter Triangle base: 8
Enter Triangle height: 10
Area of Triangle: 40.0
```

Result:

The program is successfully executed and the output is verified.

Experiment No :5

Date:18/11/2024

Aim:

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

Pseudocode:

```
1. Initialize an empty list lt
2. Read the number of terms
3. For each number i from n to n-1
      Read the numbers
      Append numbers to the list
   End for
4. Define lambda function twox()
      Return 2**x
5. Apply map function to list with twox lambda function
6. Print list
```

Method:

Function	Description	Syntax
map()	Used to apply a function to each item is an iterable	map(functin,iterable,*iterables)

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: 5
Enter terms: 2
Enter terms: 3
Enter terms: 4
Enter terms: 5
Powers of 2:
[4,8,16,32]
```

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. Read the range of numbers
2. Initialize an empty list
3. For each number i from 0 to r-1
       Read the numbers
       Append numbers to list
   End for
4. Set numbers=lt
5. Define lambda function lambda x: x% 3=0
6. Use filter function with lambda function
7. Covert the result to a list
8. Print multiples of 3
```

Method:

Function	Description	Syntax
Filter()	used to filter elements from an iterable	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: 4
Enter number: 12
Enter number: 4
Enter number: 3
Enter number: 2
Multiples of 3: [12,3]
```

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 a function factorial(num)
2. If num is 0 or 1 then
       Return 1
    Else:
       Initialize fact=1
       For each number i from 2 to num
Calculate fact=fact*i
Return fact
End for
     End if
3. Define function sumseries(n)
       Initialize totalsum=0
       For each number i from 1 to n
               Calculate term = (i ** i) / factorial(i)
               Set totalsum += term
       Return totalsum
       End for
4. Read the value n
5. Set result by calling function sumseries
6. Print result
```

Source code:

```
def factorial(num):
        if num == 0 or num == 1:
        return 1
else:
        fact = 1
        for i in range(2, num + 1):
        fact *= i
return fact
def sum_series(n):
        total_sum = 0
        for i in range(1, n + 1):
        term = (i ** i) / factorial(i)
        total_sum += term
return total_sum
n = int(input("Enter the value of n: "))
result = sum_series(n)
print(f"The sum of the series up to the {n}th term is: {result}")
```

Output:

```
Enter the value of n: 2
The sum of the series up to the 2th term is: 3.0
```

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(s1,s2,n)
2. If length of s1 and s2 are less than n
       Return false
     Return s1[:n] == s2[:n]
3. Read the first string
4. Read the second string
5. Read the value for n
6. Set result by calling function compare
7. Print result
```

Source code:

```
def compare(S1, S2, n):
          if len(S1) < n or len(S2) < n:
          return False
return S1[:n] == S2[:n]
S1 = input("Enter first string: ")
S2 = input("Enter second string: ")
n = int(input("Enter the value of n: "))
result = compare(S1, S2, n)
print(f"The result of comparison is: {result}")
```

Output:

```
Enter first string: nafia
Enter second string: najiya
Enter the value of n: 2
The result of comparison is: True
Enter first string: nafia
Enter second string: najiya
Enter the value of n: 3
The result of comparison is: False
```

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)
       """ 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 arguments in args are integers then
       Raise valueError
Return sum of all values in args
2. Print result by calling function add_numbers
```

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. If num =1 or num=0 then
      Return 1
   Else:
      Initialize fact=1
      For each number i from 2 to num
              fact= fact*i
      Return fact
3. Define function permutation(n,r)
      Return factorial(n) // factorial(n-r)
4. Read the value for n
5. Read the value for r
6. Print permutations
7. Print combinations
```

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"Permutations({n},{r}):{Permutation(n,r)}")
print(f"Combinations({n},{r}):{Combination(n,r)}")
```

Output:

```
Enter the n value: 2
Enter the r value: 4
Permutations(2,4): 2
Combinations(2,4): 0
```

Result:

The program is successfully executed and the output is verified.