LAB CYCLE - 4

Experiment No :1

Date :28/11/2024

Aim :

Write a program to print the Fibonacci series using recursion.

Pseudocode :

```
DEFINE function FIBO(n):
    IF n <= 1:
        RETURN n
    ELSE:
        RETURN FIBO(n-1) + FIBO(n-2)

GET n from user
PRINT "Fibonacci series up to", n, "is:"

FOR i FROM 0 TO n-1:
    PRINT FIBO(i), WITHOUT newline
```

Souce Code :

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

n = int(input("Enter limit: "))
print(f"Fibonacci series up to {n} is: ", end="")

for i in range(n):
    print(fibo(i), end=" ")
```

Output :

```
Enter limit: 5
Fibonacci series up to 5 is: 0 1 1 2 3
```

Result : The program is successfully executed and the output is verified.

Experiment No :2

Date: 28/11/2024

Aim :

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

Pseudocode :

```
FUNCTION add(a, b):
    PRINT a + b
    Return a + b

FUNCTION sub(a, b):
    PRINT a - b
    Return a - b

FUNCTION mul(a, b):
    PRINT a * b
    Return a * b

FUNCTION div(a, b):
    IF b == 0:
        Print error message
    ELSE:
        PRINT a / b
        RETURN a / b

WHILE True:
    PRINT menu options
    GET choice c from user
    IF c == 5:
        Exit the loop
    GET two numbers n1, n2 from user
    IF c == 1:
        CALL add(n1, n2)
    ELSE IF c == 2:
        CALL sub(n1, n2)
    ELSE IF c == 3:
        CALL mul(n1, n2)
    ELSE IF c == 4:
        CALL div(n1, n2)
    ELSE:
        Print invalid choice message
```

Souce Code :

```
def add(a,b):
   print(f"{a}+{b}={a+b}")
   return a+b
def sub(a,b):
   print(f"{a}-{b}={a-b}")
   return a-b
def mul(a,b):
   print(f"{a}*{b}={a*b}")
   return a*b
def div(a,b):
   print(f"{a}/{b}={a/b}")
   if b==0:
      print("!!! Division is not possible by zero !!!\n")
   else:
      return(f"{a}/{b} is {a/b}")
while(True):
   print("\n1.ADDITION\n2.SUBSTRACTION\n3.MULTIPLICATION\n4.DIVISION\n5.E
         XIT")
   c=int(input("Enter your choice:"))
   if c==5:
      break
   n1=int(input("Enter a number1:"))
   n2=int(input("Enter a number2:"))
   if c==1:
      add(n1,n2)
   elif c==2:
      sub(n1,n2)
   elif c==3:
      mul(n1,n2)
   elif c==4:
      div(n1,n2)
   elif c!=[1,2,3,4]:
      print("\nEnter a valid choice\n")
```

Output :

```
1.ADDITION
2.SUBSTRACTION
3.MULTIPLICATION
4. DIVISION
5.EXIT
Enter your choice:1
Enter a number1:2
Enter a number2:3
2+3=5
```

1.ADDITION
2.SUBSTRACTION
3.MULTIPLICATION
4. DIVISION
5.EXIT
Enter your choice:2
Enter a number1:3
Enter a number2:2
3-2-1

1.ADDITION
2.SUBSTRACTION
3.MULTIPLICATION
4.DIVISION
5.EXIT
Enter your choice:3
Enter a number1:4
Enter a number2:5
4*5=20

1.ADDITION
2.SUBSTRACTION
3.MULTIPLICATION
4.DIVISION
5.EXIT
Enter your choice: 4
Enter a number1:20
Enter a number2:5
20/5-4.0

Result : The program is successfully executed and the output is verified.

Experiment No :3

Date: 28/11/2024

Aim :

Write a program to print the nth prime number.

Pseudocode :

```
FUNCTION isprime(num):
    IF num <= 1:
        RETURN False
    FOR i from 2 to sqrt(num) + 1:
        IF num % i == 0:
            Return False
    RETURN True

FUNCTION nthprime(n):
    SET count = 0, current = 2
    WHILE count < n:
        IF isprime(current):
            Increment count
        IF count < n:
            Increment current
    Return current

GET n from user
PRINT nthprime(n)
```

Souce Code :

```
def isprime(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 nthprime(n):
    count, current = 0, 2
    while count < n:
        if isprime(current):
            count += 1
        if count < n:
```

```
            current += 1
    return current
n = int(input("Enter n: "))
print(f"The {n}th prime number is {nthprime(n)}.")
```

Output :

```
Enter n: 5
The 5th prime number is 11.
```

Result : The program is successfully executed and the output is verified.

Experiment No :4

Date: 28/11/2024

Aim :

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

Pseudocode :

DEFINE lambda sq(side): return side * side
DEFINE lambda rec(l, b): return l * b
DEFINE lambda tri(ba, he): return 0.5 * ba * he

Print "AREA OF SQUARE"
GET side from user
PRINT area using sq(side)

PRINT "AREA OF RECTANGLE"
GET length l and breadth b from user
PRINT area using rec(l, b)

Print "AREA OF TRIANGLE"
GET base ba and height he from user
PRINT area using tri(ba, he)

Method :

Functions	Description	Syntax
lambda()	Accepts multiple arguments and returning a single expression's result.	lambda arguments:expression

Souce Code :

```
sq=lambda side:side*side
rec=lambda l,b:l*b
tri=lambda ba,he:0.5*ba*he
print("\nAREA OF SQUARE")
side=float(input("Enter the side of the Square: "))
print(f"Area is {sq(side)}")
print("\nAREA OF RECTANGLE")
l=float(input("Enter the Length: "))
b=float(input("Enter the Breadth: "))
print(f"Area is {rec(l,b)}")
```

```
print("\nAREA OF TRIANGLE")
ba=float(input("Enter the base of the triangle: "))
he=float(input("Enter the height of th triangle: "))
print(f"Area is {tri(ba,he)}")
```

Output :

```
AREA OF SQUARE
Enter the side of the Square: 4
Area is 16.0

AREA OF RECTANGLE
Enter the Length: 5
Enter the Breadth: 4
Area is 20.0

AREA OF TRIANGLE
Enter the base of the triangle: 10
Enter the height of the triangle: 5
Area is 25.0
```

Result : The program is successfully executed and the output is verified.

Experiment No :5

Date: 28/11/2024

Aim :

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

Pseudocode :

GET list of numbers from user and convert to integers

MAP each number to 2 raised to the power of the number using lambda

PRINT the original numbers and their powers of 2

Method :

Functions	Description	Syntax
map()	function applies a given function to each item in an iterable and returns an iterator.	map(function, iterable)

Souce Code :

```
num=list(map(int, input("Enter a list of numbers separated by spaces: ").split()))
power=list(map(lambda x:2**x,num))
print(f"Powers of 2 for the numbers {num} are {power}")
```

Output :

```
Enter a list of numbers separated by spaces: 1 2 3 4
Powers of 2 for the numbers [1, 2, 3, 4] are [2, 4, 8, 16]
```

Result : The program is successfully executed and the output is verified.

Experiment No :6

Date: 28/11/2024

Aim :

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

Pseudocode :

GET list of numbers from user and convert to integers

FILTER numbers that are divisible by 3 using lambda

PRINT the multiples of 3

Method :

Functions	Description	Syntax
filter()	function filters elements of an iterable based on a function that returns True.	filter(function, iterable)

Souce Code :

```
n=list(map(int,input("Enter the numbers seperated by spaces: ").split()))
mul=list(filter(lambda x:x%3==0,n))
print(f"Multiples of 3 = {mul}")
```

Output :

```
Enter a list of numbers separated by spaces: 1 3 6 9
Multiples of 3 = [3, 6, 9]
```

Result : The Program is successfully executed and the output is verified.

Experiment No :7

Date: 28/11/2024

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 :

DEFINE function fact(x) to calculate factorial of x

GET n from user (number of terms)

GENERATE terms as (x^x) / fact(x) for x in range 1 to n

INITIALIZE sum to 0

FOR each term in terms, add it to sum

PRINT the sum of terms

Souce Code :

```
def fact(x):
    f=1
    for i in range(1,x+1):
        f=f*i
    return f
n=int(input("Enter the number of terms: "))
terms=list(map(lambda x:(x**x)/fact(x),range(1,n+1)))
sum=0
for s in terms:
    sum+=s
print(f"sum of terms = {sum}")
```

Output :

```
Enter the number of terms: 3
Sum of terms = 7.5
```

Result : The program is successfully executed and the output is verified.

Experiment No :8

Date: 28/11/2024

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 :

```
DEFINE function compare(s1, s2, n):
    If n <= 0:
        Print error message
    Return True if first n characters of s1 and s2 are equal

GET string1 from user
GET string2 from user
GET n from user (number of characters)
PRINT equivalence of first n characters of string1 and string2
```

Souce Code :

```
def compare(s1,s2,n):
   if n<=0:
      print("Number must be positive!!")
   return s1[ :n]==s2[ :n]
string1=input("Enter the First String: ")
string2=input("Enter the second String: ")
n=int(input("Enter the no:of Characters: "))
print(f"Equivalence = {compare(string1,string2,n)}")
```

Output :

```
Enter the First String: hello
Enter the second String: hewlo
Enter the no: of Characters: 3
Equivalence = False

Enter the First String: hello
Enter the second String: helwo
Enter the no: of Characters: 3
Equivalence = True
```

Result : The program is successfully executed and the output is verified.

Experiment No :9

Date: 28/11/2024

Aim :

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

Pseudocode :

```
DEFINE function ADD_NUM(*args):
    RETURN sum of all elements in args

GET list1 from user, split by space and convert to integers
PRINT "sum =", ADD_NUM(*list1) and the docstring of ADD_NUM
```

Souce Code :

```
def add_num(*args):
        """
        Sum of Integers
        """
        return sum(args)
list1=(map(int,input("Enter the numbers separated by space: ").split()))
print("sum = ",add_num(*list1),add_num.__doc__)
```

Output :

```
Enter the numbers separated by spaces: 1 2 3
Sum = 6
        Sum of Integers
```

Result : The program is successfully executed and the output is verified.

Experiment No :10

Date: 28/11/2024

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 :

```
IMPORT math library

DEFINE function PERMUTATION(n, r):
    RETURN math.factorial(n) // math.factorial(n - r)

DEFINE function COMBINATION(n, r):
    RETURN math.factorial(n) // (math.factorial(n - r) * math.factorial(r))

GET n from user
GET r from user

PRINT "Permutations are", PERMUTATION(n, r)
PRINT "Combinations are", COMBINATION(n, r)
```

Souce Code :

```
import math
def permutation(n,r):
   return math.factorial(n)//math.factorial(n-r)
def combination(n,r):
   return math.factorial(n)//(math.factorial(n-r)*math.factorial(r))
n=int(input("Enter the value for n: "))
r=int(input("Enter the value for r: "))
print(f"Permutations are {permutation(n,r)}")
print(f"Combinations are {combination(n,r)}")
```

Output :

```
Enter the value for n: 4
Enter the value for r: 2
Permutations are 12
Combinations are 6
```

Result : The program is successfully executed and the output is verified.