Experiment - 1

Aim : Write a program to print the Fibonacci series using recursion.

pseudocode:

```
1. Define a function fibonacci(n) which returns nth Fibonacci number using recursion.
2. If n <= 1, return n.
3. Else, return fibonacci(n-1) + fibonacci(n-2).
```

source code:

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

n = int(input("Enter the number of terms: "))
for i in range(n):
print(fibonacci(i), end=" ")
```

output:

```
Enter the number of terms: 6
0 1 1 2 3 5
```

Result: Program executed successfully. Output obtained.

Experiment - 2

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

pseudocode:

```
1. Define functions for add, subtract, multiply, divide.
2. Show menu for user to choose operation.
3. Based on user choice, call the corresponding function and display the result.
```

source code:

```
def add(a, b):
return a + b

def subtract(a, b):
return a - b

def multiply(a, b):
return a * b

def divide(a, b):
return a / b

while True:
print("Menu:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")
print("5. Exit")

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


if choice == 5:

    break


num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))


if choice == 1:
    print(f"Result: {add(num1, num2)}")
elif choice == 2:
    print(f"Result: {subtract(num1, num2)}")
elif choice == 3:
    print(f"Result: {multiply(num1, num2)}")
elif choice == 4:
    print(f"Result: {divide(num1, num2)}")
else:
    print("Invalid choice")
```

output:

```
Menu:
    1. Add
    2. Subtract
    3. Multiply
    4. Divide
    5. Exit
       Enter choice: 1
       Enter first number: 5
       Enter second number: 3
       Result: 8.0
```

Result: Program executed successfully. Output obtained.

Experiment - 3

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

pseudocode:

```
1. Define a function is_prime(num) that checks if a number is prime.
2. Loop through numbers and use is_prime to find the nth prime.
```

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

n = int(input("Enter the value of n: "))
count = 0
num = 2
while count < n:
if is_prime(num):
count += 1
num += 1
print(f"The {n}th prime number is: {num - 1}")
```

output:

```
Enter the value of n: 5
The 5th prime number is: 11
```

Result: Program executed successfully. Output obtained.

Experiment - 4

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

pseudocode:

```
1. Define three lambda functions: for square (side^2), rectangle (length * width), and triangle (0.5 * base * height).
2. Print the results using these lambda functions.
```

source code:

```
square_area = lambda side: side ** 2
rectangle_area = lambda length, width: length * width
triangle_area = lambda base, height: 0.5 * base * height

print(f"Area of square: {square_area(4)}")
print(f"Area of rectangle: {rectangle_area(5, 3)}")
print(f"Area of triangle: {triangle_area(6, 4)}")
```

output:

```
Area of square: 16
Area of rectangle: 15
Area of triangle: 12.0
```

Result: Program executed successfully. Output obtained.

Experiment - 5

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

pseudocode:

```
1. Create a list of exponents.
2. Use map to apply lambda function that calculates powers of 2.
```

source code:

```
exponents = [1, 2, 3, 4, 5]
powers_of_2 = list(map(lambda x: 2 ** x, exponents))
print(powers_of_2)
```

output:

```
[2, 4, 8, 16, 32]
```

Result: Program executed successfully. Output obtained.

Experiment - 6

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

pseudocode:

```
1. Create a list of numbers.
2. Use filter to get multiples of 3 using a lambda function.
```

source code:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
multiples_of_3 = list(filter(lambda x: x % 3 == 0, numbers))
print(multiples_of_3)
```

output:

```
[3, 6, 9]
```

Result: Program executed successfully. Output obtained.

Experiment - 7

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) to calculate factorial.
2. Iterate to compute sum of series.
```

source code:

```
def factorial(num):
if num == 0 or num == 1:
return 1
return num * factorial(num - 1)

n = int(input("Enter the number of terms: "))
sum_series = 0
for i in range(1, n + 1):
sum_series += (i ** i) / factorial(i)
print(f"Sum of the series is: {sum_series}")
```

output:

```
Enter the number of terms: 3
Sum of the series is: 7.0
```

Result: Program executed successfully. Output obtained.

Experiment - 8

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 a function compare(S1, S2, n) to compare first n characters.
2. Return True if the first n characters of both strings are the same.
```

source code:

```
def compare(S1, S2, n):
return S1[
] == S2[
]

s1 = input("Enter first string: ")
s2 = input("Enter second string: ")
n = int(input("Enter value of n: "))
print(compare(s1, s2, n))
```

output:

```
Enter first string: hello
Enter second string: helix
Enter value of n: 3
True
```

Result: Program executed successfully. Output obtained.

Experiment - 9

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

pseudocode:

```
1. Define a function add_numbers(*args) that takes variable length arguments.
2. Return the sum of the arguments.
3. Print the docstring for the function.
```

source code:

```
def add_numbers(*args):
        return sum(args)

print(add_numbers(1, 2, 3, 4, 5))
```

output:

```
15
```

Result: Program executed successfully. Output obtained.

Experiment - 10

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

pseudocode:

```
1. Define a function factorial(n) to compute factorial.
2. Define functions p(n, r) for permutations and c(n, r) for combinations.
3. Return the result for given n and r.
```

source code:

```
def factorial(n):
if n == 0 or n == 1:
return 1
return n * factorial(n - 1)

def p(n, r):
return factorial(n) // factorial(n - r)

def c(n, r):
return factorial(n) // (factorial(r) * factorial(n - r))

n = int(input("Enter n: "))
r = int(input("Enter r: "))
print(f"Permutations: {p(n, r)}")
print(f"Combinations: {c(n, r)}")
```

output:

```
Enter n: 5
Enter r: 3
Permutations: 60
Combinations: 10
```

Result: Program executed successfully. Output obtained.