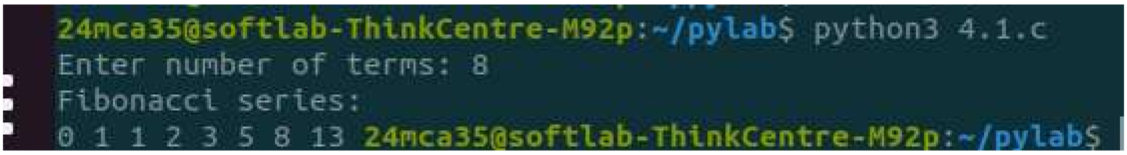
CYCLE 4

PGMM1

AIM:
Write a program to print the Fibonacci series using recursion.
SOURCE CODE:

```
def fibano_recursive(a, b, num):

    if num <= 0:
        return

    print(a, end=" ")

    c = a + b
    fibano_recursive(b, c, num - 1)


num = int(input("Enter number of terms: "))


if num <= 0:
    print("Please enter a positive number")
else:
    print("Fibonacci series:")
    fibano_recursive(0, 1, num)
```

OUTPUT:



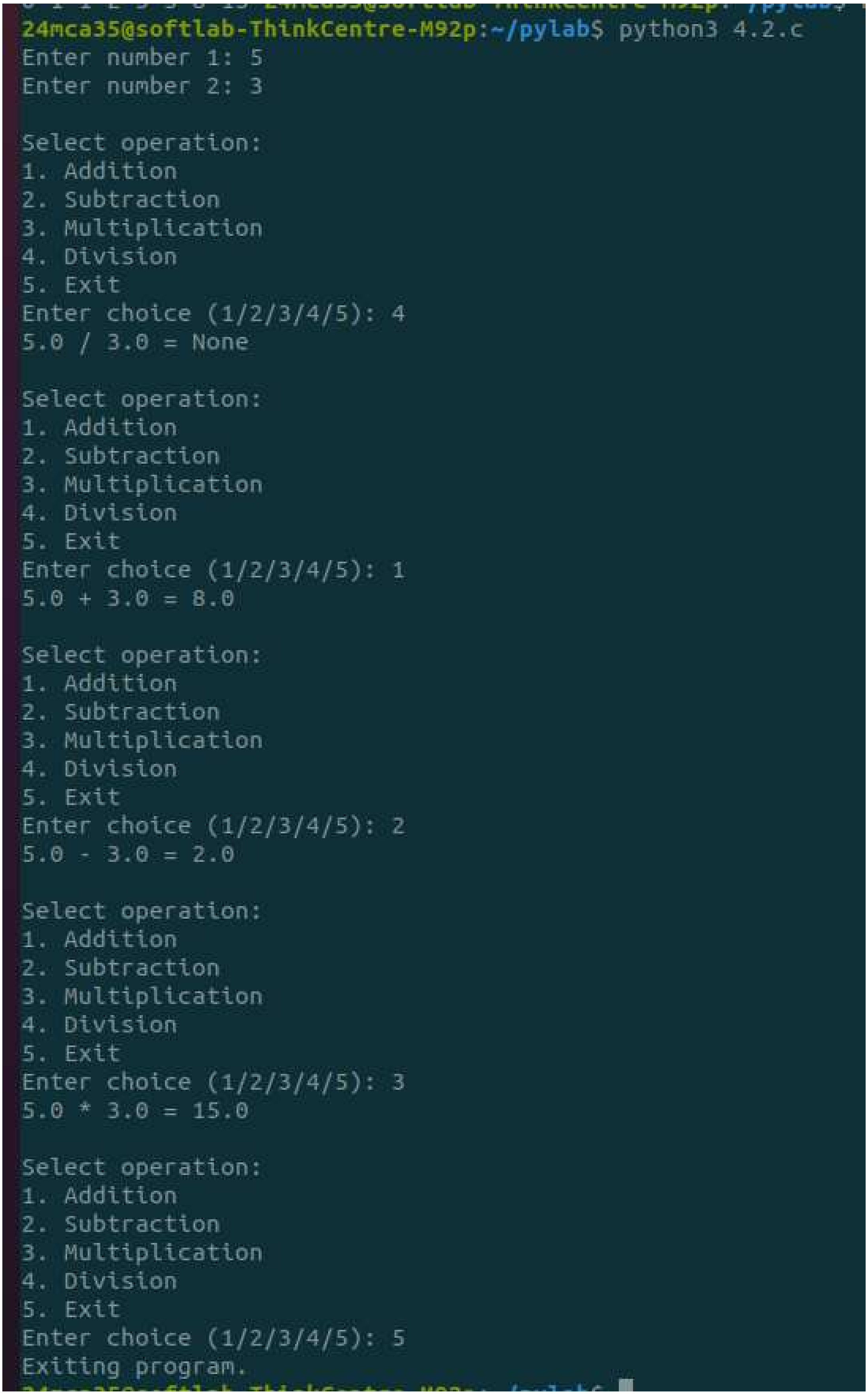
```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.1.c
Enter number of terms: 8
Fibonacci series:
0 1 1 2 3 5 8 13 24mca35@softlab-ThinkCentre-M92p:~/pylab$
```

PGMM2

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

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

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

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

def div(a, b):
    if b != 0:
        return
    else:
        return "Error! Division by zero"


a = float(input("Enter number 1: "))
b = float(input("Enter number 2: "))


while True:
    print("\nSelect operation:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Exit")


    choice = input("Enter choice (1/2/3/4/5): ")




    if choice == '1':
        print(f"{a} + {b} = {add(a, b)}")
    elif choice == '2':
        print(f"{a} - {b} = {sub(a, b)}")
    elif choice == '3':
        print(f"{a} * {b} = {mul(a, b)}")
    elif choice == '4':
        result = div(a, b)
        print(f"{a} / {b} = {result}")
    elif choice == '5':
        print("Exiting program.")
        break
    else:
        print("Invalid input! Please choose a valid option")
```

OUTPUT:



```
0 1 1 2 3 5 8 13 24mca35@softlab-ThinkCentre-M92p:~/pylab$
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.2.c
Enter number 1: 5
Enter number 2: 3

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5): 4
5.0 / 3.0 = None

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5): 1
5.0 + 3.0 = 8.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5): 2
5.0 - 3.0 = 2.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5): 3
5.0 * 3.0 = 15.0

Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter choice (1/2/3/4/5): 5
Exiting program.
24mca35@softlab-ThinkCentre-M92p:~/pylab$
```

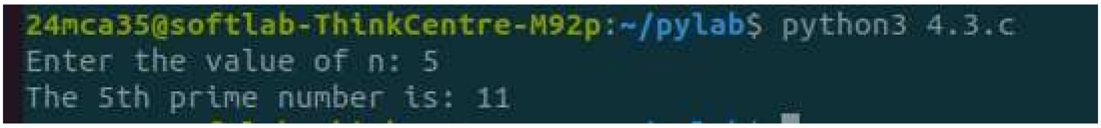
PGMM3

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

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
    num = 2
    while count < n:
        if is_prime(num):
            count += 1
        num += 1
    return num - 1

n = int(input("Enter the value of n: "))
print(f"The {n}th prime number is: {nth_prime(n)}")
```

OUTPUT:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.3.c
Enter the value of n: 5
The 5th prime number is: 11
```

PGMM 4

AIM:

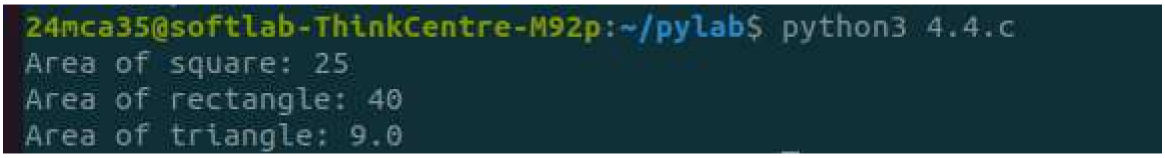
Write lambda functions to find the area of square, rectangle and triangle.

SOURCECODE:

```
  GNU nano 4.8                                              4.4.c
area_of_square = lambda side: side * side


area_of_rectangle = lambda length, width: length * width


area_of_triangle = lambda base, height: 0.5 * base * height


side = 5
length = 10
width = 4
base = 6
height = 3
```

```
print(f"Area of square: {area_of_square(side)}")
print(f"Area of rectangle: {area_of_rectangle(length, width)}")
print(f"Area of triangle: {area_of_triangle(base, height)}")
```

OUTPUT:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.4.c
Area of square: 25
Area of rectangle: 40
Area of triangle: 9.0
```

PGMM5

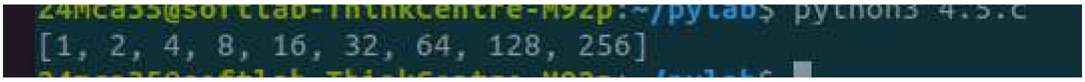
AIM:

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

SOURCE CODE:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8]

powers_of_2 = map(lambda x: 2 ** x, numbers)


print(list(powers_of_2))
```

OUTPUT:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.5.c
[1, 2, 4, 8, 16, 32, 64, 128, 256]
24mca35@softlab-ThinkCentre-M92p:~/pylab$
```

PGMM6

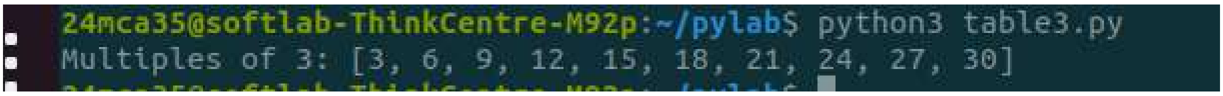
AIM:

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

SOURCE CODE:

```
numbers = list(range(1, 31))
multiples_of_3 = list(filter(lambda x: x % 3 == 0, numbers))
print("Multiples of 3:", multiples_of_3)
```

OUTPUT:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 table3.py
Multiples of 3: [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
24mca35@softlab-ThinkCentre-M92p:~/pylab$
```

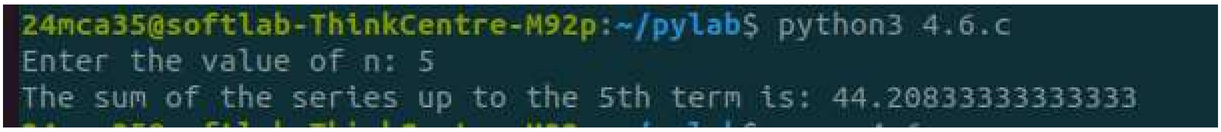
PGMM7
AIM:

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].

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)  # (i^i) / 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:



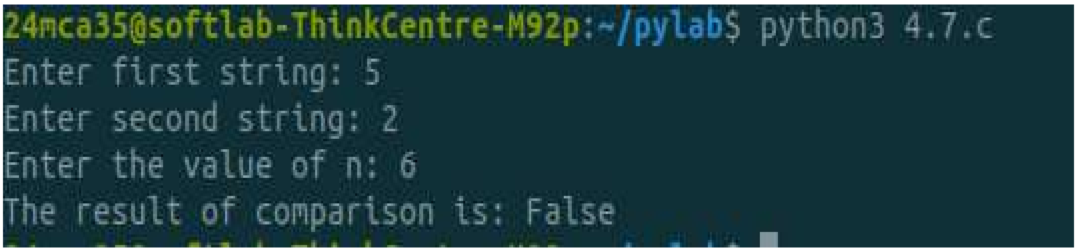
```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.6.c
Enter the value of n: 5
The sum of the series up to the 5th term is: 44.20833333333333
```

PGMM8
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.

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:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.7.c
Enter first string: 5
Enter second string: 2
Enter the value of n: 6
The result of comparison is: False
```

PGMM9

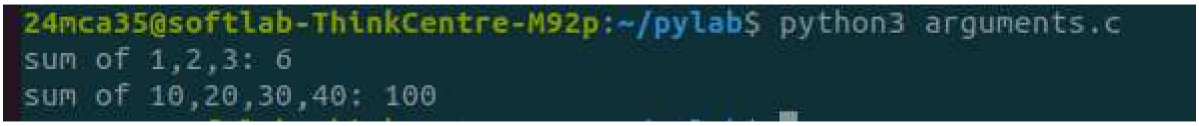
AIM:

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

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:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 arguments.c
sum of 1,2,3: 6
sum of 10,20,30,40: 100
```

PGMM10

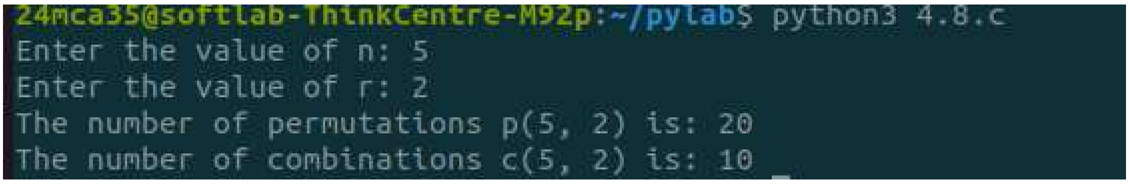
AIM:

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)!)

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 permutations(n, r):
    return factorial(n) // factorial(n - r)


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


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

p_result = permutations(n, r)
c_result = combinations(n, r)

print(f"The number of permutations p({n}, {r}) is: {p_result}")
print(f"The number of combinations c({n}, {r}) is: {c_result}")
```

OUTPUT:



```
24mca35@softlab-ThinkCentre-M92p:~/pylab$ python3 4.8.c
Enter the value of n: 5
Enter the value of r: 2
The number of permutations p(5, 2) is: 20
The number of combinations c(5, 2) is: 10
```