DAY-16 09/25/2025

1.Create a function to print prime number from the given range using with input and without return method.

Logic:

```
s1 = 2
s2 = 20
for i in range(s1,s2+1,1):
    for j in range(2,i,1):
        if(i%j==0):
            break
    else:
        print(i)
```

Output:

2

3

5

7

11

13

17

19

DAY-16 09/25/2025

Code:

```
def prime_range(s1,s2):
    l1 = []
    for i in range(s1,s2+1,1):
        for j in range(2,i,1):
            if(i%j==0):
                break
        else:
            l1.append(i)
    return l1
function call: prime_range(2,20)
```

Output:

[2, 3, 5, 7, 11, 13, 17, 19]

- Return we cant use to print the multiple values as the return function executes only once unlike print so we will be using multi variable data types like list,tuple.
- First declare the list,append the results to it and finally return the list.

2. Create function with input and with return to find largest among three numbers.

Code-1:

```
def large(a,b,c):
    if(a>b and a>c):
        result = a
    elif(b>c and b>a):
        result = b
    else:
```

```
        result = c
    return result
```

function call:

```
large(10,1,2)
```

output: 10

code-2:

```
def greater(a,b,c):
    if(a>b and a>c):
        return f"{a} is large"
    elif(b>c and b>a):
        return f"{b} is large"
    else:
        return f"{c} is large"
```

function call:

```
greater(55,23,67)
```

output:

'67 is large'

Function as a parameter

- we can assign them to variables.
- we can pass them as parameters to other functions.
- we can return them from functions.

Example:

```
def square(x):
    return x*x
```

```
def cube(x):
    return x*x*x


def apply_fun(fun_name,num):
    return fun_name(num)
```

function call:

```
apply_fun(square,3)
```

output: 9

Recursion Functions:

- A recursive function is a function that calls itself until a base condition is met.

Example:

```
def fact(n):
    if n==1:
        return 1
    else:
        return n*fact(n-1)
```

function call: fact(5)

output: 120

Nested Functions:

In Python, an inner function (also called a nested function) is a function defined inside another function. They are mainly used for:

- Encapsulation: Hiding helper logic from external access.
- Code Organization: Grouping related functionality for cleaner code.
- Access to Outer Variables: Inner functions can use variables of the enclosing (outer) function.
- Closures and Decorators: Supporting advanced features like closures (functions that remember values) and function decorators.

This makes inner functions powerful for structuring programs, maintaining readability and reusing logic effectively.

Syntax:

```
def outer_fun(p1,p2 ... pn):
    def inner_fun(p1,p2,.  pn):
        return value
    return value
```

Example:

```
def num1(x,y):
    def num2():
        return y
    return x+y
num1(5,10)
```

output: 15

Lambda Function:

- A lambda function is a small, anonymous function in python
- defined using a keyword lambda instead of def
- It can take any number of arguments but must contain only one expression.
- Expression is automatically returned (no need to use return)

Syntax: lambda arguments: expression

Example:

S = lambda num: num*num

S(5)

Output: 25

Lambda function to add 2 numbers:

k = lambda a,b: a+b

k(2,8)

output: 10

Nested Lambda:

A nested lambda function is a lambda (anonymous) function defined inside another lambda function in Python.

Syntax:

lambda args1: lambda args2: expression

Example:

multiply = lambda x: (lambda y: x * y)

result = multiply(3)(5)

Output: 15