25/09/2025

Create a function to check given number is prime number or not using with input without return.

```
def prime(num):
    for i in range(2,num,1):
        if(num%i==0):
            print(f"{num} is not prime number")
            break
        else:
            print(f"{num}is prime number")
prime(6)
```

o/p:

6 is not prime number

Create functions to print prime numbers from given range using with input without return.

```
def prime_num(r1,r2):
    for i in range(r1,r2+1,1):
        for j in range(2,i,1):
            if(i%j==0):
                break
        else:
            print(i)
prime_num(2,10)
```

o/p:

2

3

5

7

Create functions to print prime numbers from given range using with input with return.

```
def prime_num(r1,r2):
    l1=[]
    for i in range(r1,r2+1,1):
        for j in range(2,i,1):
            if(i%j==0):
                break
        else:
            l1.append(i)
    return l1
prime_num(2,10)
```

o/p:

[2, 3, 5, 7]

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

```
def lar(x,y,z):
    if(x>y and x>z):
        return f"{x} is largest"
```

```
    elif(y>x and y>z):
        return f"{y} is largest"
    else:
        return f"{z} is largest"
lar(4,9,1)
```

o/p:

'9 is largest'

Lambda Function (Anonymous Function):

- It is a small, anonymous (nameless) function.
- Defined using the lambda keyword instead of def.
- Can take any number of arguments, but must contain only one expression.
- The expression is automatically returned — no need to use the return statement.

Syntax:

lambda arguments: expression

Examples:

1. ```
   s = lambda a,b: a+b
   s(3,5)
   ```
   o/p:
   8

2. ```
   s = lambda a: a**3
   s(3)
   ```
   o/p:
   27

Function as Parameter

- A function can be passed as an argument to another function.
- The called function can then execute the function passed to it.
- Helps in writing flexible and reusable code.
- **Syntax:**

```
def outer(func, value):
    return func(value)
```

- **Example:**

```
def square(x):
    return x * x
def apply_function(func, n):
    return func(n)
print(apply_function(square, 4))
```

o/p:

16

Nested Functions:

- A function defined inside another function is called a nested function or inner function.
- The inner function is accessible only inside the outer function.
- Useful for encapsulation and organizing logic.
- **Syntax:**

```
def outer_function():
    # code of outer function
    def inner_function():
        # code of inner function
    # calling inner function inside outer
    inner_function()
# calling outer function
outer_function()
```

- **Example:**

```
def greet():
    def message():
        print("Hello from inner function")
    message()
greet()
```

o/p:

Hello from inner function

Recursive Function:

- A function that calls itself is known as a recursive function.
- It is used to solve problems that can be divided into smaller sub-problems.
- Every recursive function must have a base condition to stop recursion.
- **Syntax:**

```
def function_name(parameters):
    if condition:          # base condition (to stop recursion)
        return value
    else:
        # recursive call
        return expression or function_name(modified_parameters)
```

- **Example:**

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))
```

o/p:

120