25th September Notes

1.Function to Check Whether a Number is Prime (With Input, Without Return)

Code:

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


prime(8)
```

Explanation:

- The loop checks divisibility of num from 2 to num-1.
- If any number divides num, it's **not prime**, so the loop breaks.
- If the loop completes without breaking, the else part executes → the number is **prime**.

2. Function to Print Prime Numbers in a Range (With Input, Without Return)

Code:

```
def prime(a, b):
  for i in range(a, b + 1):
    for j in range(2, i):
      if i % j == 0:
        break
    else:
      print(i)
```

```
a = int(input("Enter start value: "))
b = int(input("Enter end value: "))
prime(a, b)
```

Explanation:

- Outer loop → iterates from a to b.
- Inner loop → checks if the number i has any divisor.
- If it doesn't, number is prime → printed.

3. Function to Return Prime Numbers in a Range (With Input, With Return)

Code:

```
def prime(a, b):
  l1 = []
  for i in range(a, b + 1):
    for j in range(2, i):
      if i % j == 0:
        break
    else:
      l1.append(i)
  return l1

a = int(input("Enter start value: "))
b = int(input("Enter end value: "))
print("Prime numbers:", prime(a, b))
```

Explanation:

- Similar to the previous one, but instead of printing, it **stores primes in a list**.
- Finally returns that list using return l1.

4. Function to Find the Largest Number (With Input, With Return)

Code 1: Using List and Conditions

```
def large(a, b, c):
    l1 = []
    if a > b:
        l1.append(a)
    else:
        l1.append(b)
        if b > c:
            l1.append(b)
        else:
            l1.append(c)

    if l1[0] > l1[-1]:
        largest = l1[0]
    else:
        largest = l1[-1]

    return largest

print("Largest:", large(1, 2, 3))
```

Code 2: Simplified Version

```
def large(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"


print(large(1, 2, 3))
```

Explanation:

- Compares three numbers using **if–elif–else** conditions.
- Returns the largest value.

5. Lambda Function

Code:

```
# addition using lambda
s = lambda a, b: a + b
print(s(5, 7))


# cube using lambda
c = lambda a: a * a * a
print(c(3))
```

Explanation:

- lambda defines a **short, anonymous function**.
- Syntax: lambda arguments: expression
- Expression is automatically returned — no need for return.

6. Function as a Parameter

Code:

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


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

```
print(apply_function(square, 5))
```

Explanation:

- You can **pass a function as an argument** to another function.
- apply_function() calls square() indirectly through the parameter func.

7. Nested Function

Code:

```
def outer():
  print("This is the outer function")

  def inner():
    print("This is the inner function")

  inner()  # calling inner function inside outer

outer()
```

Explanation:

- A **nested function** is defined **inside another function**.
- The inner function can only be called **within the outer function**.

8. Recursion Function

Code:

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

```
print("Factorial:", fact(5))
```

Explanation:

- **Recursion** means a function calling itself.
- Base condition → stops recursion when n == 1.
- Example: fact(5) → 5*4*3*2*1 = 120.