Session - 15

Recursion :

- A function calling itself is said to be a recursive function.

Syntax -

```
def function1():
        base case
        ---------
    function1()
    return...
```

Base case - It is used to tell when should the recursion stop.

Ex-1) Sum of first three natural numbers

```
def f1(n):
    if n==1:
        return 1
    s = n + f1(n-1)
    return s
ans = f1(3)
print(ans)
```

o/p ⇒ 6

n=3 s=3+f1(2) → 3 ; n=2 s=2+f1(1) → 1 ; n=1 s=1 ; 3+1+1 = 6 Call stack

Find the sum of first 100 natural numbers.

```
def f(n):
    if n==100:
        return 100
    s = n + f(n+1)
    return s
ans = f(1)
print(ans)
```

o/p ⇒ 5050

(or)

```
def f(n):
    if n==1:
        return 1
    s = n + f(n-1)
    return s
ans = f(100)
print(ans)
```

o/p ⇒ 5050

→ Recursion approach is simple but it fills the stack memory quickly.

Loop based -

```
ans = 0
for num in range(1,101):
    ans += num
print(ans).
```

o/p ⇒ 5050

2) Factorial of a number.

```
def fact(n):
    if (n==1):
        return 1
    s = n*fact(n-1)
    return s
ans = fact(5)
print(ans)
```

o/p ⇒ 20

Taking input from the user -

```
def fact(n):
    if(n==1):
        return 1
    s = n*fact(n-1)
    return s
n = int(input("Enter a number:"))
ans = fact(n)
print(ans)
```

o/p ⇒ Enter a number: 20
380

Passing a function as an argument to another function -

- In python, we can pass a function as an input argument to an another function.

Calculator application -

4 operators
→ addition
→ subtraction
→ multiply
→ division

Program -

```
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def mul(a,b):
    return a*b
def div(a,b):
    return a/b
def calc(fn,a,b): // driving function
    return fn(a,b)
ans = calc(add,3,4)
print(ans)
```

o/p ⇒ 7

Scope of a variable :

1) Global scope of a variable
2) Local scope of a variable.

Global scope of a variable -

```
a = 2      //Global variable
def fn():
    return a+2
print(fn())
```

o/p ⇒ 4

How to use a local variable outside a function? -

1) return that variable
2) change the scope of the variable

Local scope of a variable -

```
def fn():
    b= 10     // local variable
    return b-2
print(fn())
```

o/p ⇒ 8