Class 07: Functions & Recursion in Python

Instructor: Md Rasel Sarker

Email: rasel.sarker6933@gmail.com

Functions in Python

Block of statements that perform a specific task.

```
# Function Definition
   def func_name(param1, param2, ...):
           # some work
    return val

# Function Call
func_name(arg1, arg2, ...)  # function call
```

Benefits of Using Functions in Python

1. Lines of code will be reduced
2. Redundancy (repeat code) will be reduced
3. Code Reusability
4. Modularity
5. Improved Readability
6. Easy to Debug & Maintain
7. Logical Grouping
8. Less Code Duplication
9. Scalability

Let’s Practice

1. WAP to calculate the average of 5 numbers using a function.
2. WAP to check whether a number is prime using a function.
3. WAP to find the greatest among three numbers using a function.
4. WAP to count the total number of vowels in a given string using a function.
5. WAP to generate Fibonacci series up to n terms using a function.
[0, 1, 1, 2, 3, 4, 5]

Recursion in Python

Recursion is a programming technique where a function calls itself.
Simply, A function that calls itself = Recursion

# Example:

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

Here the function named factorial() is calling itself repeatedly until n == 1.

Why is recursion used?

- When a problem can be broken down into smaller, similar sub-problems
- When a task needs to be solved by breaking it down step-by-step

Example:

- Factorial
- Fibonacci series
- Tree/graph traversal (DFS)

Benefits of Recursion:

- **Code becomes shorter and cleaner**
- **Solves problems that are naturally recursive**
- **Logical thinking improves**
- **Better structure for divide-and-conquer problems**
  For example, merge sort and quicksort—where large problems are solved by dividing them into smaller parts.

Let’s Practice

1. Problem: Find Factorial using Recursion
2. Problem: Print Fibonacci Series using Recursion