# Training Day 6 Report

**Date: 30 June 2025**

**Topic: Extended Part of Functions and Recursion**

**Overview:**

The session focused on the **advanced use of functions** in Python and the concept of **recursion**. Functions help in code reusability, modularity, and better program structure, while recursion provides an elegant way to solve problems by making a function call itself.

**Key Concepts Covered:**

**1. Functions in Python (Extended Concepts)**

**Default Arguments:** Provide default values if no argument is passed.

```
def greet(name="User"):
    print("Hello", name)

greet()        # Hello User
greet("Sam")  # Hello Sam
```

**Keyword Arguments:** Parameters can be passed by name, improving readability.

```
def student(name, age):
    print(name, age)

student(age=20, name="Riya")
```

**Variable-Length Arguments:**

- *args → Non-keyword variable arguments (tuple).
- **kwargs → Keyword variable arguments (dictionary).

```
def add(*nums):
    return sum(nums)

print(add(1, 2, 3, 4))  # 10
```

**2. Recursion**

Recursion is a technique where a function calls itself to solve smaller instances of the problem until a **base case** is reached.

**Factorial Example:**

```
 def fact(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * fact(n - 1)

print(fact(5))  # 120
```

**Fibonacci Example:**

```
 def fib(n):
    if n <= 1:
        return n
    else:
        return fib(n-1) + fib(n-2)

print(fib(6))  # 8
```

- **Important Points:**
    - Recursion requires a **base case** to avoid infinite calls.
    - Can be memory-intensive due to multiple function calls.
    - Useful for problems like factorial, Fibonacci, Tower of Hanoi, and tree traversals.

**Summary:**

- Extended concepts of functions like **default arguments, keyword arguments, *args, and **kwargs** were learned.
- Understood the **concept of recursion** and its importance in solving repetitive problems elegantly.

- Implemented programs for **factorial** and **Fibonacci sequence** using recursion.
- Learned that recursion simplifies logic but should be used carefully due to performance issues.

**Learning Outcomes:**

✔ Ability to use **default, keyword, and variable-length arguments** in functions.
✔ Confidence in writing **recursive solutions** for mathematical and logical problems.
✔ Awareness of **base cases** and **termination conditions** in recursion.
✔ Improved **problem-solving skills** using recursive techniques.
✔ Better understanding of how recursion relates to **divide-and-conquer algorithms**.