UNDERSTANDING RECURSIVE FUNCTIONS IN PYTHON

Concept, Examples, and Fibonacci Series

Name:Rutuja Balasaheb Dukare
Roll no.: ET2-80
PRN:202401070188

INTRODUCTION TO RECURSION

- Recursion is a technique in programming where a function calls itself to solve a smaller instance of the same problem.

- Common in algorithms like:

- Searching and sorting
- Tree and graph traversals
- Mathematical computations

CHARACTERS OF RECURSIVE FUNCTION

- Base Case: Condition under which the function stops calling itself.
- Recursive Case: The part of the function where it calls itself.

- Example structure:

```
def function_name(parameters): if
 base_case:
    return value
```

WHY USE RECURSION?

- Simplifies code for problems that have a recursive nature.
- Makes code cleaner and easier to understand for divide-and-conquer problems.
- Preferred for algorithms where the solution depends on solving sub-problems.

SIMPLE EXAMPLE – FACTORIAL

- Problem: Calculate factorial of a number. Definition: n! = n * (n-1)! with base case 0! = 1
- def factorial(n):

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

HOW FACTORIAL WORKS STEP-BY-STEP

- factorial(5)

→ 5 * factorial(4)

→ 5 * 4 * factorial(3)

→ 5 * 4 * 3 * factorial(2)

→ 5 * 4 * 3 * 2 * factorial(1)

→ 5 * 4 * 3 * 2 * 1 = 120

VISUALIZING RECURSIVE CALLS

- Stack-like behavior:
  - –Each call is stored on the call stack.
  - -Returns happen in reverse order after base case is hit.
- Important to track flow to avoid infinite recursion.

INTRODUCTION TO FIBONACCI SERIES

- Definition:
- $F(0) = 0, F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$ for $n > 1$

- Recursion suits this because each number depends on the previous two.

FIBONACCI SERIES CODE

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

 for i in range(10):
      print(fibonacci(i), end=" ")
```

CONCLUSION

- Recursion is a powerful tool in Python programming.
- Understand base and recursive cases thoroughly.
- Use recursion wisely to solve problems effectively.
- Practice with real examples like factorial and Fibonacci.