Chapter 5 - 13 Recursion

Theory:

```
This chapter covers the topic of 13 Recursion.
```

Code Example:

```
# ----------------------------------------
# ■ What is Recursion in Python?
# ----------------------------------------
# Recursion is a programming technique where a function calls itself
# to solve a smaller version of the original problem.
# ■ Real-Life Example:
# Imagine standing between two mirrors—you see repeated reflections
# going on and on. That's recursion in visual form.
# ----------------------------------------
# ■ Recursion Structure in Code
# ----------------------------------------
# A recursive function typically has two main parts:
# 1. Base Case: A condition to stop the recursion.
# 2. Recursive Case: The function calls itself with smaller input.
# ----------------------------------------
# ■ Example: Factorial Using Recursion
# ----------------------------------------
def factorial(n):
if n == 1: # Base case
return 1
else: # Recursive case
return n * factorial(n - 1)
# Calling the function
print("Factorial of 4 is:", factorial(4))
# ----------------------------------------
# ■ How It Works:
# factorial(4)
# → 4 * factorial(3)
# → 4 * 3 * factorial(2)
# → 4 * 3 * 2 * factorial(1)
# → 4 * 3 * 2 * 1 = 24
# ----------------------------------------
# ■■ Key Concepts in Recursion:
# ----------------------------------------
# Term | Description
# -----------------|-----------------------------------------------
# Base Case | Stops recursion (e.g., if n == 1)
# Recursive Case | Calls the function again with a smaller input
# Stack Memory | Recursion uses stack; too many calls = stack overflow
# ----------------------------------------
# ■ Recursion vs Loop
# ----------------------------------------
# Feature | Recursion | Loop
# -------------- | --------------------------|-------------------------
# Style | Function calls itself | Uses 'for' or 'while'
# Use Case | Good for trees, divide & conquer | Simple repetitive tasks
# Memory | More memory (call stack) | Less memory efficient
# ----------------------------------------
# ■ Practice Problem: Fibonacci Series
# ----------------------------------------
def fibonacci(n):
if n <= 1:
return n
return fibonacci(n - 1) + fibonacci(n - 2)
# Calling the function
print("Fibonacci of 5 is:", fibonacci(5))
# ----------------------------------------
# ■ Pro Tip:
# Always define a base case.
# Without it, the function will call itself forever and crash!
#my
def factorial(n):
if (n==1 or n==0):
return 1
return n * factorial(n-1)
n = int(input("Enter a number : "))
number = factorial(n)
print(f" your factorial of num is {number}")
#or
print(f" your factorial of num is {factorial(n)}")
#end
```