```
al\Programs\Python\Python312\python.exe' 'c:\Users\RAFTB\.vscode\extensions\ms-python.debugpy-2025.0.0-win32-x64\bundled\libs\debugpy\launcher' '58144
' '--' 'c:\Users\RAFTB\OneDrive\Documents\GroupProjectCOMSCI\fibi.py'
Enter the n: 5
F(5) = 3
PS C:\Users\RAFTB\OneDrive\Documents\GroupProjectCOMSCI>
```

```
    return current  # nth Fibonacci number

# 4. User Input
try: # Exception
    n = int(input("Enter the n: "))
    print(f"F({n}) = {fibonacci_iterative(n)}")
except ValueError:
    print("Please enter a valid integer!")


"""
- Pros: Fast execution, low memory usage, efficient (O(n)).
- Cons: Slightly longer code

- Time complexity: O(n) (Linear time complexity).
- Space complexity: O(1) (Constant space usage).
"""
```

```python
def fibonacci_iterative(n):
    """
    - parameter n: nth Fibonacci term
    - return: nth Fibonacci number
    """
    # 1. Base Cases
    if n <= 0: # F(0) = X
        return "n must be a positive integer >= 1"
    elif n == 1:
        return 0  # F(1) = 0
    elif n == 2:
        return 1  # F(2) = 1

    # 2. Initializing previous and current terms
    """
    F(0) = X
    F(1) = 0
    F(2) = 1
    F(3) = F(2) + F(1) = 1 + 0 = 1
    F(4) = F(3) + F(2) = 1 + 1 = 2
     > F(n) = F(n - 2) + F(n - 1)
    """
    previous = 0 # F(n - 2), Starts from F(1) = 0
    current = 1 # F(n - 1), Starts from F(2) = 1

    # 3. Calculate
    for _ in range(2, n): # Starts from the 3rd to the nth
        # Update both variables simultaneously without losing previous value
        previous, current = current, previous + current
        """
        Wrong Case: Overwrites previous before using it in the next step

        previous = current
        current = previous + current
        """
```