

# ASSIGNMENT-1.3

Name: D. Sravika Reddy

Hall-Ticket No: 2403a510d0

Batch No: 05

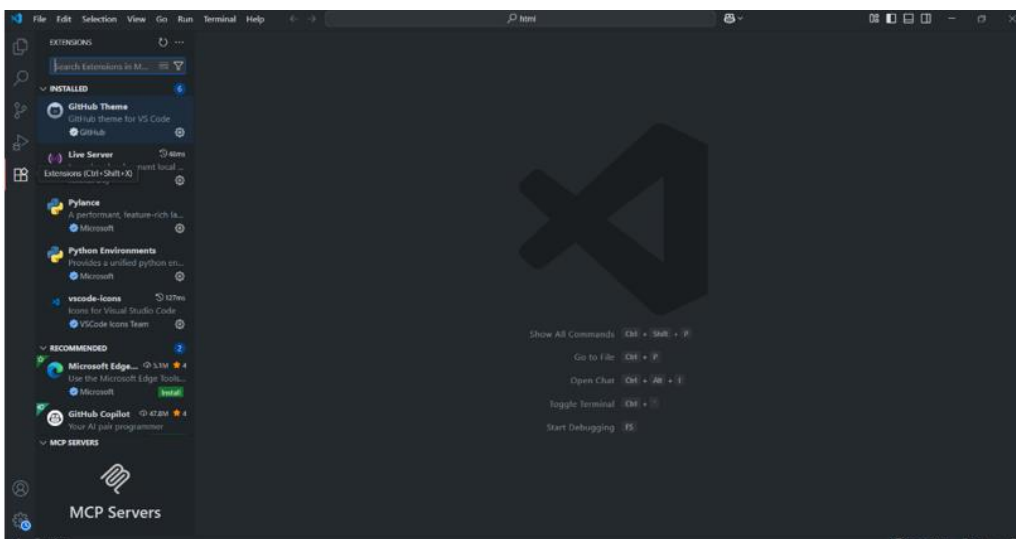
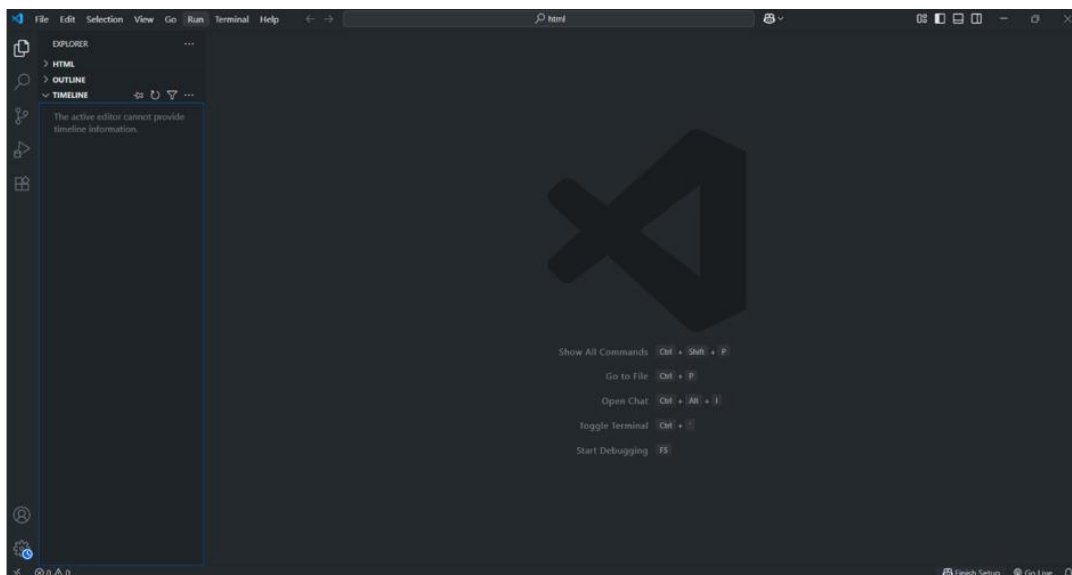
Course: AI Assisted Coding

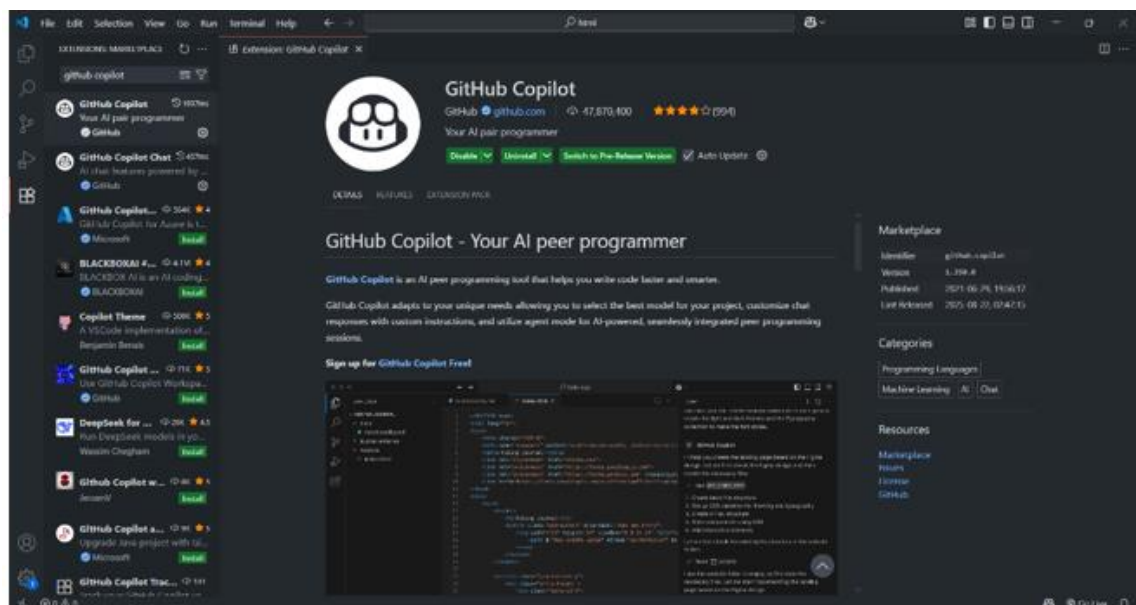
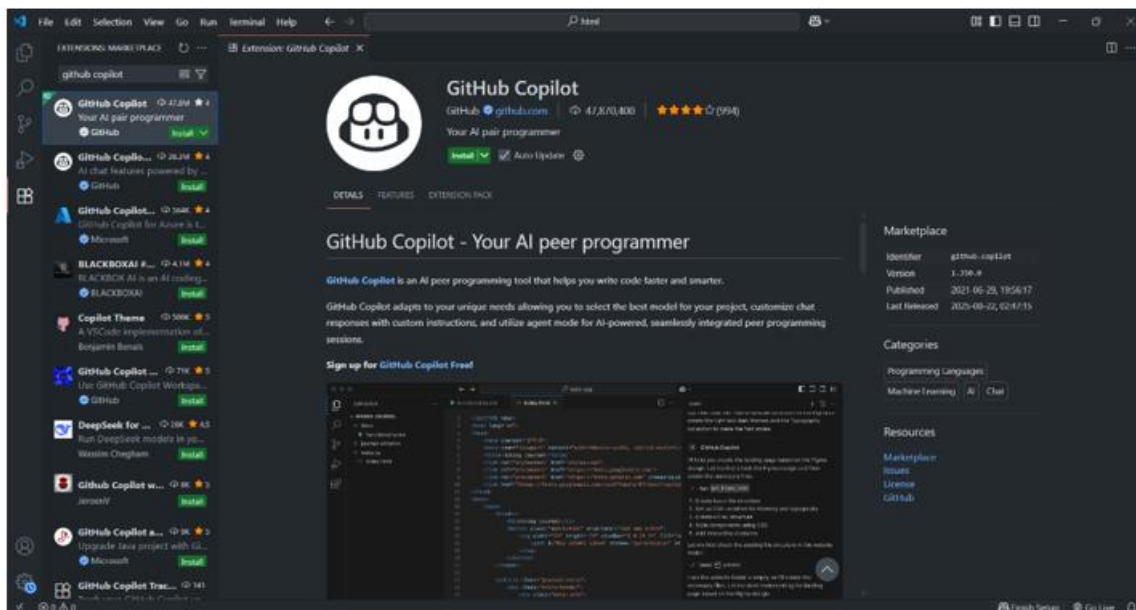
## Task Description#1

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

## Expected Output#1

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step





## Task Description#2

- Use Copilot to generate a `is_prime()` Python function.

## Expected Output#2

- Function to check primality with correct logic.

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
```

## Explanation:

### Explanation of `is_prime(n)` Function

- The function checks if `n` is less than or equal to 1. If so, it returns `False` because numbers  $\leq 1$  are not prime.
- It then loops from 2 up to the square root of `n` (inclusive).
- For each `i`, it checks if `n` is divisible by `i` (`n % i == 0`). If so, `n` is not prime and returns `False`.
- If no divisors are found, it returns `True`, meaning `n` is prime.

This approach is efficient because it only checks divisibility up to the square root of `n`, reducing unnecessary checks.

## Task Description#3

- Write a comment like `# Function to reverse a string` and use Copilot to generate the function.

## Expected Output#3

- Auto-completed reverse function.

```
# Function to reverse a string
def reverse_string(s):
    return s[::-1]
```

### Explanation of `reverse_string(s)` Function

- The function takes a string `s` as input.
- It uses slicing `s[::-1]` to reverse the string:
  - `s[start:stop:step]` is the slicing syntax.
  - `[::-1]` means start from the end and step backwards by 1.
- The reversed string is returned.

## Task Description#4

- Generate both recursive and iterative versions of a factorial function using comments..

## Expected Output#4

- Two working factorial implementations

```
# Iterative version of factorial
def factorial_iterative(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Recursive version of factorial
def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial_recursive(n - 1)
```

### Explanation

- **Iterative Version ( `factorial_iterative` ):**
  - Uses a loop to multiply numbers from 2 up to `n`.
  - Accumulates the result in the `result` variable.
  - Returns the final product as the factorial.
- **Recursive Version ( `factorial_recursive` ):**
  - Checks if `n` is 0 or 1 (base case); returns 1.
  - Otherwise, returns `n` multiplied by the factorial of `n-1`.
  - The function calls itself repeatedly, reducing `n` each time, until the base case is reached.

## Task Description#5

- Use Copilot to find the largest number in a list. Assess code quality and efficiency.

## Expected Output#5

- A valid function with your review

```
def find_largest(numbers):  
    largest = numbers[0]  
    for num in numbers:  
        if num > largest:  
            largest = num  
    return largest
```

### Review

- **Code Quality:**
  - The function is clear and easy to understand.
  - It assumes the list is non-empty (could add a check for empty lists).
  - Variable names are descriptive.
- **Efficiency:**
  - Time complexity is  $O(n)$ , which is optimal for this task.
  - No unnecessary operations or extra space used.

