Assignment 1.3

Name: P. Umesh Reddy Roll.no: 2403A510F9

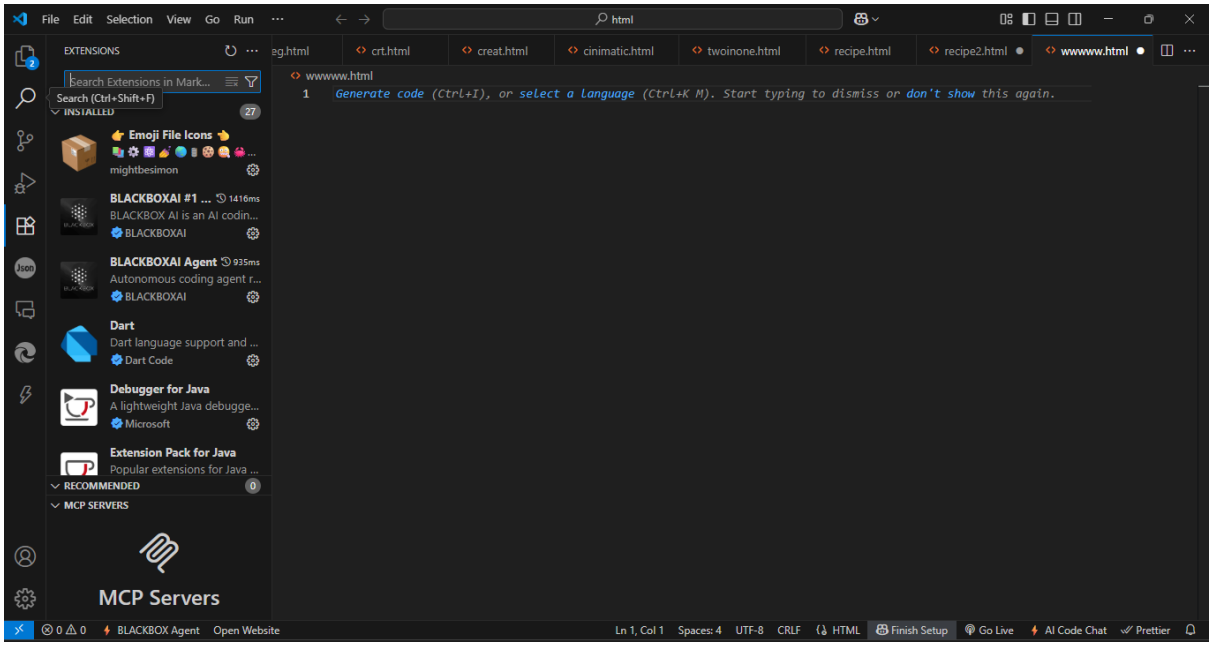
Batch.no: 06 course: AI Assisted coding

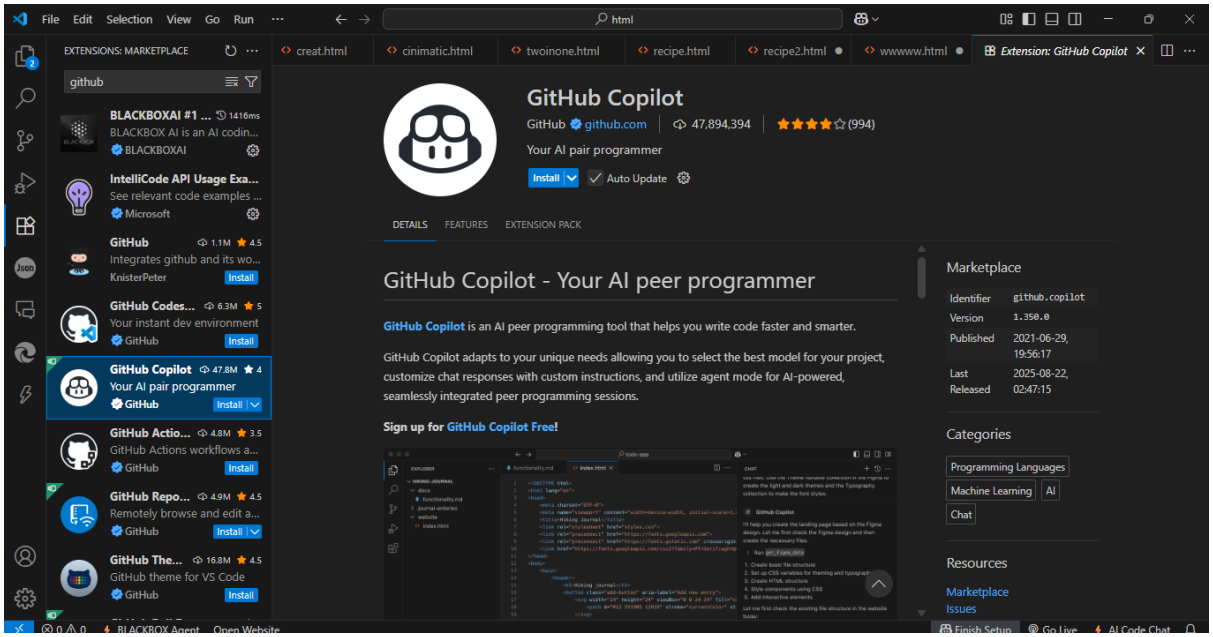
Task Description#1

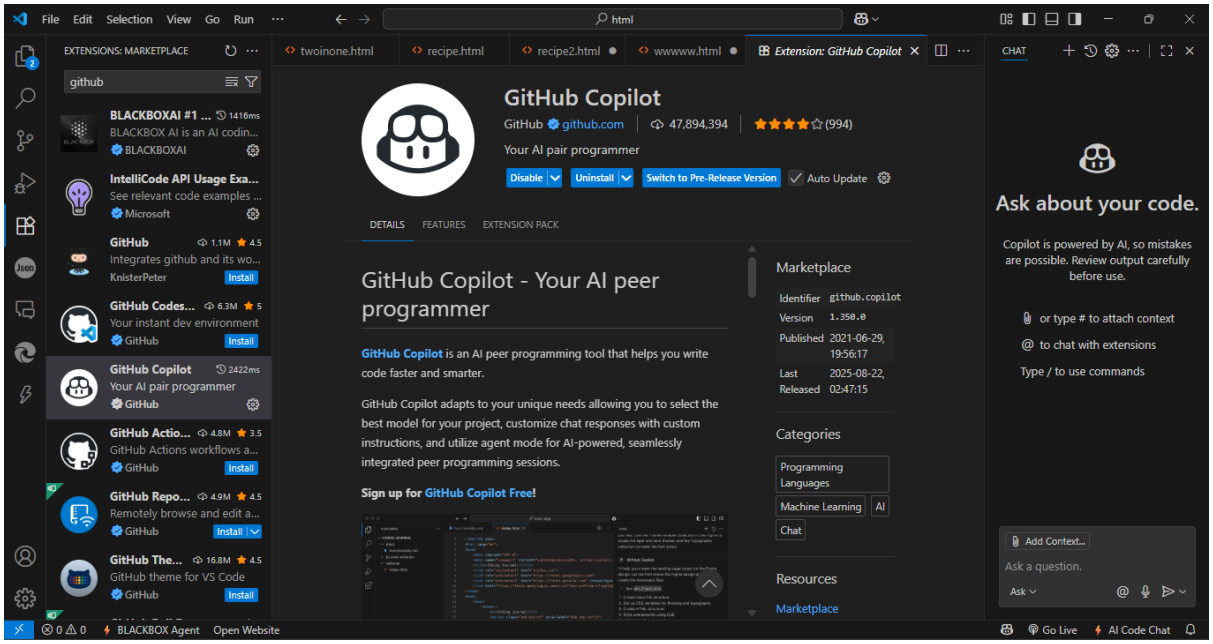
- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

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.

```python
def is_prime(n):
    """
    Check if a given number n is a prime number.

    Parameters:
    n (int): The number to check for primality.

    Returns:
    bool: True if n is prime, False otherwise.
    """
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n ** 0.5) + 1, 2):
        if n % i == 0:
            return False
    return True
```

Explanation

- **Check for numbers less than or equal to 1**: These are not prime.
- **Check for 2**: 2 is the only even prime number.
- **Check for even numbers greater than 2**: These are not prime.
- **Check for divisibility from 3 up to the square root of n**: If any divisor is found, n is not prime.
- **Return True** if no divisors are found, meaning n is prime.

This function efficiently checks for primality using trial division up to the square root of the number, skipping even numbers for better performance.

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

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

Explanation:

- The comment `# Function to reverse a string` describes what the function does.
- The function `reverse_string` takes a single argument `s`, which is a string.
- It returns the reversed version of the string using Python's slicing notation: `s[::-1]`.
  - `[::-1]` means "start at the end of the string and move backwards", effectively reversing the string.
- For example, `reverse_string("hello")` returns `"olleh"`.

Copilot's Role:

- Copilot auto-completes the function based on the comment, generating correct and idiomatic Python code to reverse a string.

Task Description#4

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

Expected Output#4

● Two working factorial implementations

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

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

Explanation:

- **Recursive Version:**
  - The comment describes that the function uses recursion.
  - `factorial_recursive(n)` calls itself with `n-1` until it reaches the base case (`n == 0` or `n == 1`).
  - The base case returns `1` , which is the factorial of 0 or 1.
  - Each recursive call multiplies `n` by the factorial of `n-1` , building up the result.

- **Iterative Version:**
  - The comment describes that the function uses iteration.
  - `factorial_iterative(n)` initializes `result` as 1.
  - It loops from `2` to `n` , multiplying `result` by each number in the range.
  - The loop simulates the multiplication sequence found in the mathematical definition of factorial.

For both functions, inputting a positive integer `n` will return `n!` (n factorial). For example, for `n = 5` , both functions will return `120` .

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

```
# Function to find the largest number in a list
def find_largest(nums):
    return max(nums)
```

Explanation:

- The function `find_largest` takes a list `nums` as input and returns the largest number using the built-in Python function `max()`.
- `max()` is efficient (linear time, O(n)) and highly optimized in Python.
- For example, `find_largest([3, 7, 2, 9, 5])` returns `9`.

Code Quality Assessment:

- **Simplicity:** The function is concise and easy to read.
- **Efficiency:** Uses Python's built-in `max()`, which is optimized and runs in O(n) time.
- **Reliability:** Handles lists of any size (except empty lists, where `max()` raises a `ValueError`).
- **Improvement:** For robustness, you could add a check for empty lists and raise a custom error or return `e`.

Alternative (manual approach):

Python

```python
def find_largest(nums):
    if not nums:
        return None  # or raise ValueError("Empty list")
    largest = nums[0]
    for num in nums:
        if num > largest:
            largest = num
    return largest
```

- This version also runs in O(n) time and is more explicit, but the built-in `max()` is preferred in production code for clarity and reliability.