

AI ASSISTED CODING

NAME: B. SRISHANTH

ROLL NO: 2403A510G3

ASSIGNMENT: 1.3

Lab Objectives:

- To install and configure GitHub Copilot in Visual Studio Code.
- To explore AI-assisted code generation using GitHub Copilot.
- To analyze the accuracy and effectiveness of Copilot's code suggestions.
- To understand prompt-based programming using comments and code context

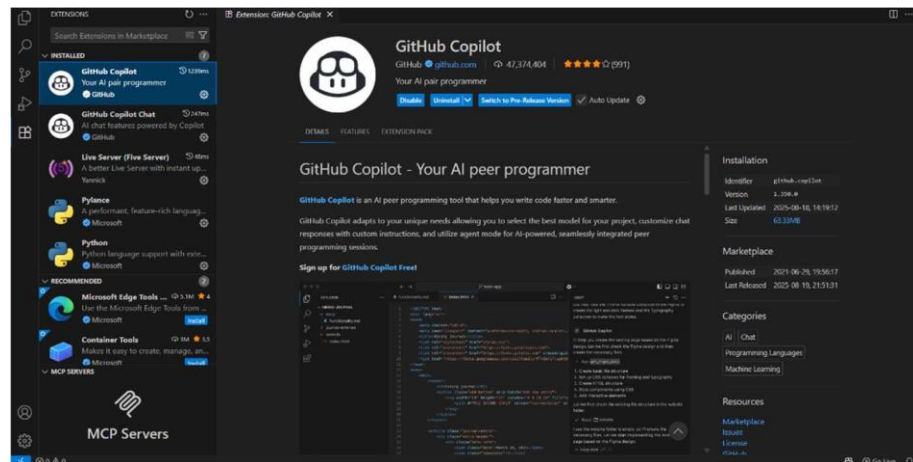
Task Description#1

- 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.

1



Task Description#2

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


Expected Output#2

- Function to check primality with correct logic.




Ask about your code.


Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

 or type # to attach context

@ to chat with extensions


Type / to use commands

 Add Context...

 task1.py X

generate a is_prime() python function|

Ask ▾ GPT-4.1 ▾

@   ▾

generate a is_prime() python function

task1.py

> Used 1 reference

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



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

1

#function to reverse a string

GPT-4.1 @

```
1 def reverse_string(s):  
2     return s[::-1]
```

Task Description#4

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

Expected Output#4

- Two working factorial implementations

```
1  # Recursive version of factorial
2  def factorial_recursive(n):
3      if n == 0 or n == 1:
4          return 1
5      else:
6          return n * factorial_recursive(n - 1)
7
8  # Iterative version of factorial
9  def factorial_iterative(n):
10     result = 1
11     for i in range(2, n + 1):
12         result *= i
13     return result
```


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

find the largest number in a list. Assess code quality and efficiency.

 task1.py

> Used 1 reference

To find the largest number in a list, you can use Python's built-in `max()` function, which is efficient and concise. Alternatively, you can implement it manually:

```
def find_largest(numbers):  
    """Return the largest number in a list."""  
    if not numbers:  
        raise ValueError("List is empty")  
    largest = numbers[0]  
    for num in numbers[1:]:  
        if num > largest:  
            largest = num  
    return largest
```

Assessment of code quality and efficiency:

- The built-in `max()` is optimal for readability and performance.
- The manual approach above is clear, handles empty lists, and runs in $O(n)$ time.
- Both methods are efficient for typical use cases.
- Code is readable, maintainable, and follows Python conventions.
- For very large lists, both approaches are suitable unless memory is constrained.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments G code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks		
Successful Setup of Copilot (Task #1)	0.5		
is_prime() Python function (Task #2)	0.5		
Reverse a string function (Task #3)	0.5		
Factorial Function (Task #4)	0.5		
Find the largest number (Task #5)	0.5		
Total	2.5 Marks		