

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName: B. Tech		Assignment Type: Lab	AcademicYear: 2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator)	
		Dr. T. Sampath Kumar	
		Dr. Pramoda Patro	
		Dr. Brij Kishor Tiwari	
		Dr. J. Ravichander	
		Dr. Mohammand Ali Shaik	
		Dr. Anirodh Kumar	
		Mr. S. Naresh Kumar	
		Dr. RAJESH VELPULA	
		Mr. Kundhan Kumar	
		Ms. Ch. Rajitha	
		Mr. M Prakash	
		Mr. B. Raju	
		Intern 1 (Dharma teja)	
		Intern 2 (Sai Prasad)	
		Intern 3 (Sowmya)	
NS_2 (Mounika)			
CourseCode	24CS002PC215	CourseTitle	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week1 - Wednesday	Time(s)	
Duration	2 Hours	Applicable to Batches	24CSBTB01 To 24CSBTB39
AssignmentNumber: 1.3 (Present assignment number) / 24 (Total number of assignments)			
Q.No.	Question	Expected Time to complete	
1	Lab 1: Environment Setup – GitHub Copilot and VS Code Integration Lab Objectives: <ul style="list-style-type: none"> To install and configure GitHub Copilot in Visual Studio Code. 	Week1 - Wednesday	

- 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

Lab Outcomes (LOs):

After completing this lab, students will be able to:

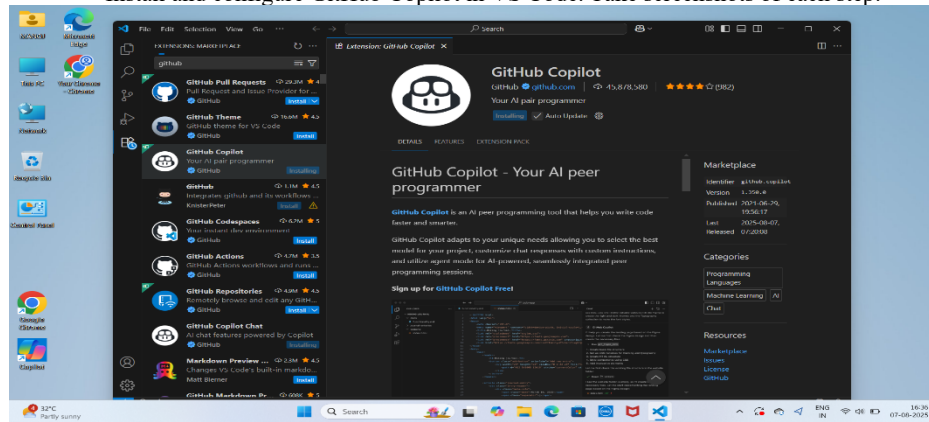
- Set up GitHub Copilot in VS Code successfully.
- Use inline comments and context to generate code with Copilot.
- Evaluate AI-generated code for correctness and readability.
- Compare code suggestions based on different prompts and programming styles.

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.

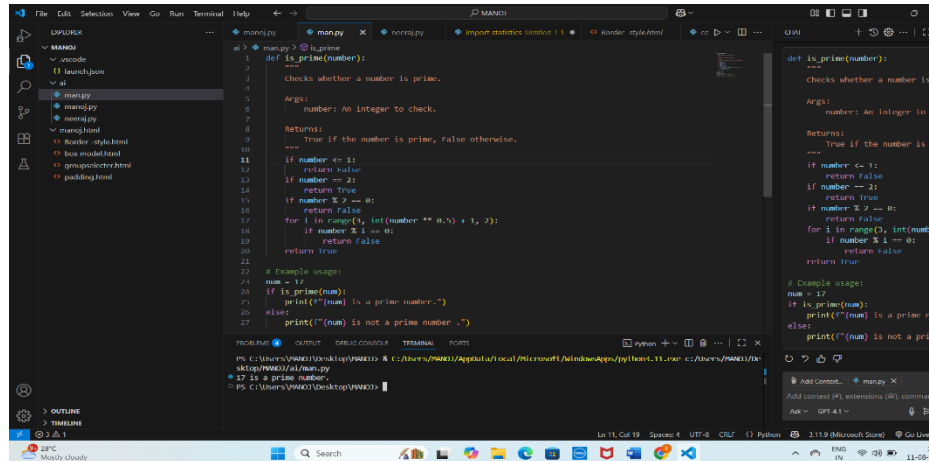


Task Description#2

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

Expected Output#2

- Function to check primality with correct logic.

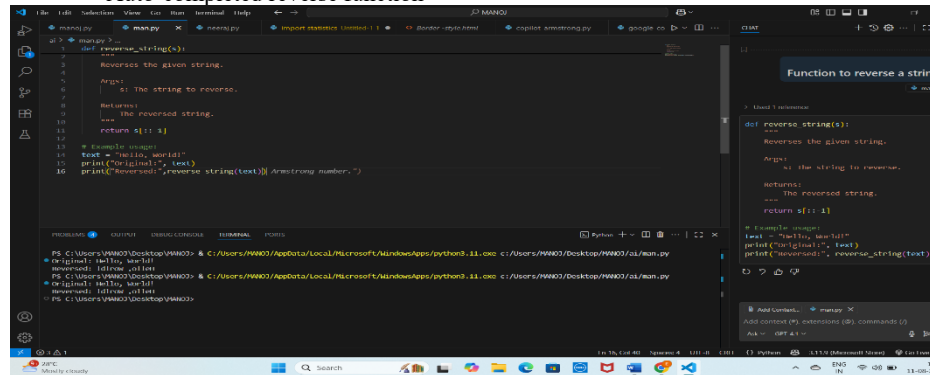


Task Description#3

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

Expected Output#3

- Auto-completed reverse function



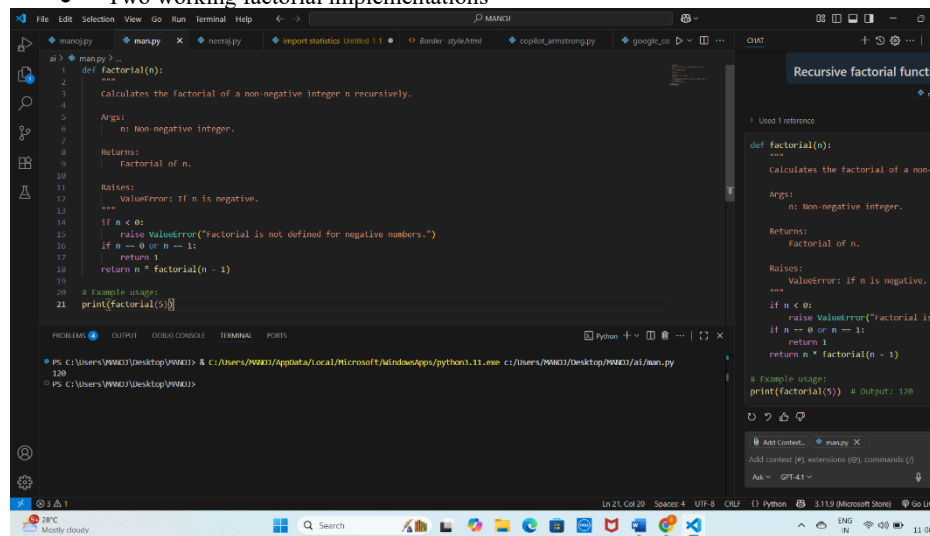
```
1 def reverse_string(s):
2     """
3     Reverses the given string.
4     Args:
5         s: the string to reverse.
6     Returns:
7         the reversed string.
8     """
9     return s[::-1]
10
11 # Example usage
12 text = "Hello, world!"
13 print("Original:", text)
14 print("Reversed:", reverse_string(text))
```

Task Description#4

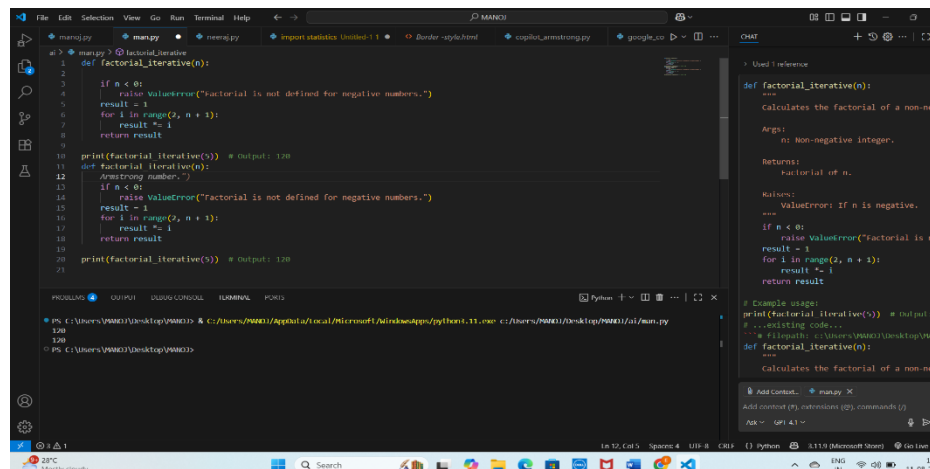
- Generate both recursive and iterative versions of a factorial function using comments..

Expected Output#4

- Two working factorial implementations



```
1 def factorial(n):
2     """
3     Calculates the factorial of a non-negative integer n recursively.
4     Args:
5         n: Non-negative integer.
6     Returns:
7         Factorial of n.
8     Raises:
9         ValueError: If n is negative.
10    """
11    if n < 0:
12        raise ValueError("factorial is not defined for negative numbers.")
13    if n == 0 or n == 1:
14        return 1
15    return n * factorial(n - 1)
16
17 # Example usage
18 print(factorial(5))
```

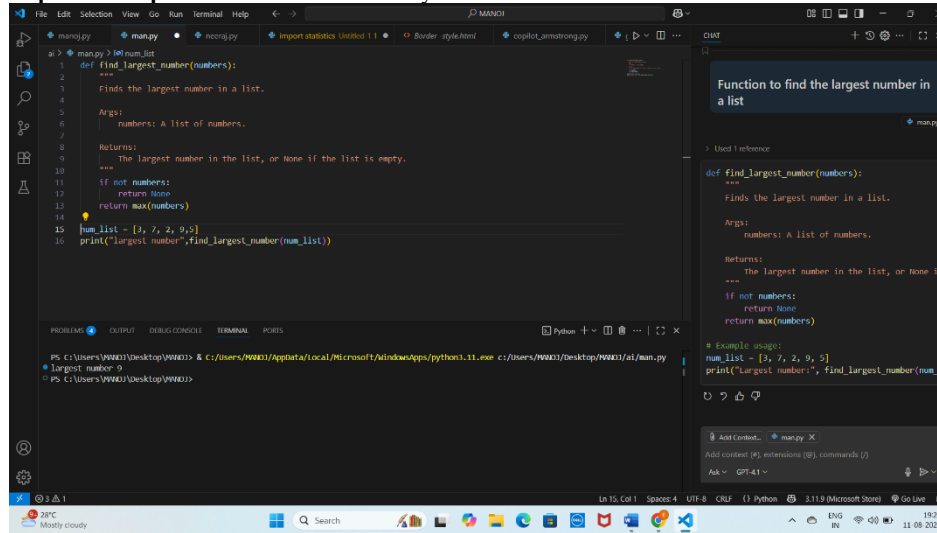


```
1 def factorial_iterative(n):
2     """
3     Calculates the factorial of a non-negative integer n iteratively.
4     Args:
5         n: Non-negative integer.
6     Returns:
7         Factorial of n.
8     Raises:
9         ValueError: If n is negative.
10    """
11    if n < 0:
12        raise ValueError("factorial is not defined for negative numbers.")
13    result = 1
14    for i in range(2, n + 1):
15        result *= i
16    return result
17
18 # Example usage
19 print(factorial_iterative(5)) # Output: 120
20
21 # Armstrong number
22 def is_armstrong(n):
23     """
24     Checks if a number is an Armstrong number.
25     Args:
26         n: A number.
27     Returns:
28         True if n is an Armstrong number, False otherwise.
29     """
30     # Convert to string to get digits
31     str_n = str(n)
32     len_n = len(str_n)
33     sum_of_powers = 0
34     for digit in str_n:
35         sum_of_powers += int(digit)**len_n
36     return sum_of_powers == n
37
38 # Example usage
39 print(is_armstrong(153)) # Output: True
40 print(is_armstrong(123)) # Output: False
```

Task Description#5

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

Expected Output#5A valid function with your review



The screenshot shows a VS Code editor with a Python file named `man.py`. The code defines a function `find_largest_number(numbers)` that finds the largest number in a list. The function includes docstrings for its purpose, arguments, and return values. It also includes a test case where `num_list = [3, 7, 2, 9, 5]` is passed to the function, and the output is printed as "largest number: 9". The terminal output shows the command `python man.py` being executed, resulting in the output `largest number: 9`. The Copilot sidebar on the right shows a suggestion for the function, and the bottom status bar indicates the file is using Python 3.11.9.

```
def find_largest_number(numbers):  
    """  
    Finds the largest number in a list.  
    Args:  
        numbers: A list of numbers.  
    Returns:  
        The largest number in the list, or None if the list is empty.  
    """  
    if not numbers:  
        return None  
    return max(numbers)  
  
num_list = [3, 7, 2, 9, 5]  
print("largest number:", find_largest_number(num_list))
```

Function to find the largest number in a list

```
def find_largest_number(numbers):  
    """  
    Finds the largest number in a list.  
    Args:  
        numbers: A list of numbers.  
    Returns:  
        The largest number in the list, or None if the list is empty.  
    """  
    if not numbers:  
        return None  
    return max(numbers)  
  
# Example usage:  
num_list = [3, 7, 2, 9, 5]  
print("largest number:", find_largest_number(num_list))
```

PS C:\Users\W9001\Desktop\W9001> & C:\Users\W9001\AppData\Local\Microsoft\WindowsApps\python3.11.exe c:\Users\W9001\Desktop\W9001\ai\man.py
largest number: 9
PS C:\Users\W9001\Desktop\W9001>

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