

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 - Tuesday	Time(s)	
Duration	2 Hours	Applicable to Batches	24CSBTB01 To 24CSBTB39
AssignmentNumber: 1.2 (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. To explore AI-assisted code generation using GitHub Copilot. 	Week1 - wednesday	

- 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

- Write a comment: # Function to check if a string is a valid palindrome (ignoring spaces and case) and allow Copilot to complete it.

Expected Output#1

- A function that correctly returns True for phrases like "A man a plan a canal Panama"

Prompt:

Function to check if a string is a valid palindrome (ignoring spaces and case)

```
[1] def is_palindrome(s):
    """
    Checks if a string is a valid palindrome, ignoring spaces and case.

    Args:
        s: The input string.

    Returns:
        True if the string is a palindrome, False otherwise.
    """
    # Remove spaces and convert to lowercase
    s = "".join(s.split()).lower()
    # Check if the string is equal to its reverse
    return s == s[::-1]

# Example usage
print(is_palindrome("A man a plan a canal Panama"))
print(is_palindrome("hello world"))
```

True
False

Observation:

the function `is_palindrome("A man a plan a canal Panama")` returned True, indicating that the phrase is considered a palindrome by the function. The function `is_palindrome("hello world")` returned False, indicating that this phrase is not a palindrome.

Task Description#2

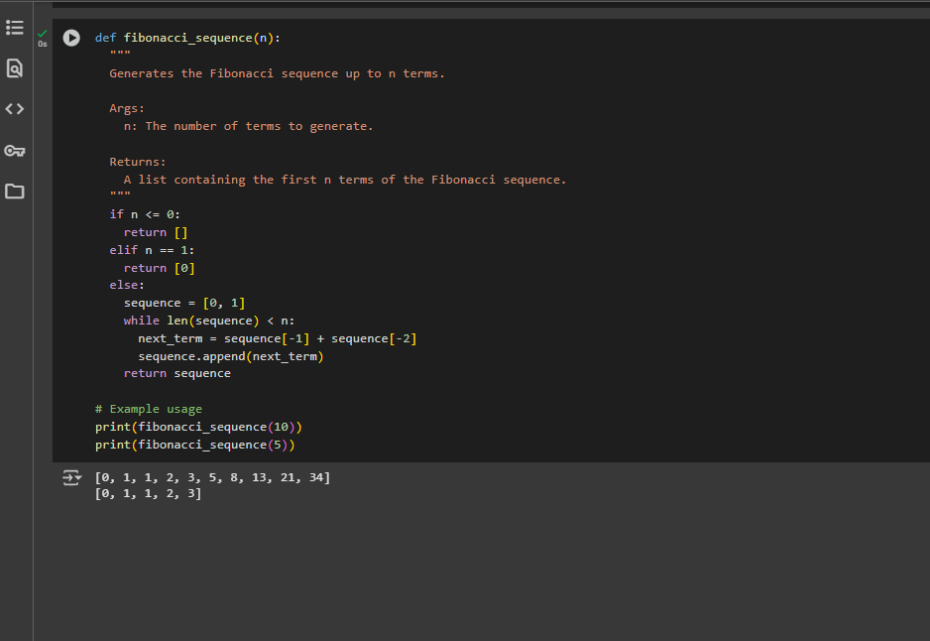
- Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring

Expected Output#2

- AI completes the function logic using loop or recursion with accurate output

Prompt:

Generate a Python function that returns the Fibonacci sequence up to n terms with only a function header and docstring

A screenshot of a code editor with a dark theme. The editor shows a Python function named 'fibonacci_sequence(n)'. The function has a docstring that describes its purpose: 'Generates the Fibonacci sequence up to n terms.' It also includes 'Args:' (n: The number of terms to generate.) and 'Returns:' (A list containing the first n terms of the Fibonacci sequence.). The function logic uses a while loop to generate the sequence. It starts with 'sequence = [0, 1]' and then enters a while loop 'while len(sequence) < n:'. Inside the loop, it calculates 'next_term = sequence[-1] + sequence[-2]', appends it to the sequence with 'sequence.append(next_term)', and then returns the sequence. Below the function, there is an example usage section with two print statements: 'print(fibonacci_sequence(10))' and 'print(fibonacci_sequence(5))'. At the bottom of the editor, the output of these calls is shown: '[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]' for the first call and '[0, 1, 1, 2, 3]' for the second call.

```
def fibonacci_sequence(n):  
    """  
    Generates the Fibonacci sequence up to n terms.  
  
    Args:  
        n: The number of terms to generate.  
  
    Returns:  
        A list containing the first n terms of the Fibonacci sequence.  
    """  
    if n <= 0:  
        return []  
    elif n == 1:  
        return [0]  
    else:  
        sequence = [0, 1]  
        while len(sequence) < n:  
            next_term = sequence[-1] + sequence[-2]  
            sequence.append(next_term)  
        return sequence  
  
# Example usage  
print(fibonacci_sequence(10))  
print(fibonacci_sequence(5))
```

[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
[0, 1, 1, 2, 3]

Observation:

the fibonacci_sequence(10) call returned the first 10 terms of the sequence: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]. The fibonacci_sequence(5) call returned the first 5 terms: [0, 1, 1, 2, 3]. This confirms that the function is correctly generating the Fibonacci sequence.

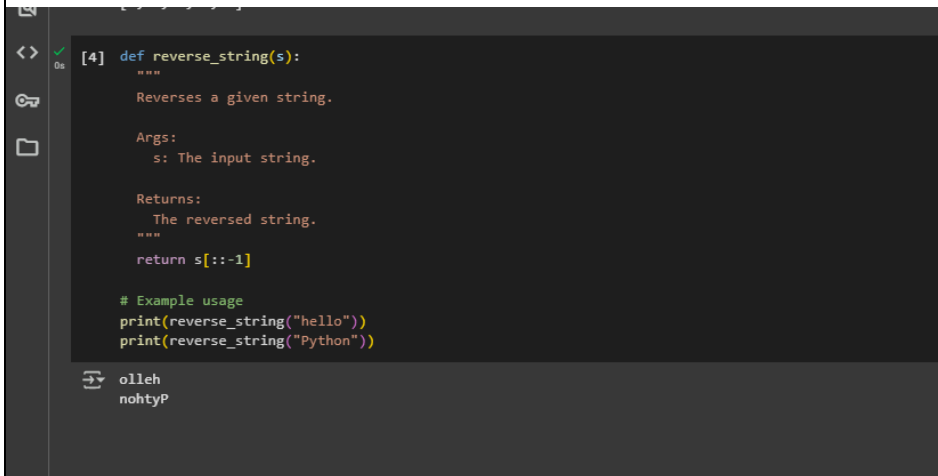
Task Description#3

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

Expected Output#3

- Auto-completed reverse function

Prompt:

A screenshot of a code editor with a dark theme. The editor shows a Python function named 'reverse_string' that takes a string 's' as input and returns the reversed string. The function uses slicing 's[::-1]'. Below the function, there is an example usage section with two print statements: 'print(reverse_string("hello"))' and 'print(reverse_string("Python"))'. The output of the code is shown at the bottom: 'olleh' and 'nohtyP'.

```
[4] def reverse_string(s):  
    """  
    Reverses a given string.  
  
    Args:  
        s: The input string.  
  
    Returns:  
        The reversed string.  
    """  
    return s[::-1]  
  
# Example usage  
print(reverse_string("hello"))  
print(reverse_string("Python"))
```

olleh
nohtyP

Task Description#4

- Generate a program that simulates a basic calculator (add, subtract, multiply, divide). Write the comment: # Simple calculator with 4 operations and let AI complete it.

Expected Output#4

- Fully working calculator with input/output and operator selection logic

Task Description#5

- Use a comment to instruct AI to write a function that reads a file and returns the number of lines..

Expected Output#5

- Functional implementation using open() or with open() and readlines()

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

Evaluation Criteria:

Criteria	Max Marks
Task #1	0.5
Task #2	0.5
Task #3	0.5
Task #4	0.5
Task #5	0.5
Total	2.5 Marks