

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.
- **Prompt : write a python code for a function to check if a string is a valid palindrome**

Expected Output#1

- "

```

AI.py
AI.py > ...
1 import re
2
3 def is_valid_palindrome(s: str) -> bool:
4     """Check if a string is a valid palindrome (ignoring case and non-alphanumeric characters)."""
5     cleaned = re.sub(r'^a-zA-Z0-9', '', s).lower()
6     return cleaned == cleaned[::-1]
7
8 # Example usages
9 if __name__ == "__main__":
10     examples = [
11         "A man, a plan, a canal: Panama",
12         "race a car",
13         "Madam",
14         "No lemon, no melon",
15         "Hello, World!"
16     ]
17     for s in examples:
18         print(f'{s} -> {is_valid_palindrome(s)}')
19
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DELL\OneDrive\Desktop\AI Assited> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/AI Assited/AI.py"
• 'A man, a plan, a canal: Panama' -> True
  'race a car' -> False
  'Madam' -> True
  'No lemon, no melon' -> True
  'Hello, World!' -> False
○ PS C:\Users\DELL\OneDrive\Desktop\AI Assited>

```

Task Description#2

- Generate a Python function that returns the Fibonacci sequence up to n terms. Prompt with only a function header and docstring
- **Prompt : write a python code for a function that returns the Fibonacci sequence up to n terms**

Expected Output#2

The screenshot shows a VS Code editor with a Python file named `AI.py`. The script defines a function `fibonacci_sequence` that returns a list of Fibonacci numbers up to `n` terms. It includes example usages and prints the results for `n=5` and `n=10`.

```

1  def fibonacci_sequence(n: int) -> list:
2      """Return the Fibonacci sequence up to n terms."""
3      if n <= 0:
4          return []
5      sequence = [0]
6      if n == 1:
7          return sequence
8      sequence.append(1)
9      for i in range(2, n):
10         sequence.append(sequence[-1] + sequence[-2])
11     return sequence
12
13 # Example usages
14 if __name__ == "__main__":
15     print(fibonacci_sequence(5))    # Output: [0, 1, 1, 2, 3]
16     print(fibonacci_sequence(10))  # Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
17

```

The terminal at the bottom shows the command to run the script and the output:

```

PS C:\Users\DELL\OneDrive\Desktop\AI Assited> & C:\Users\DELL\AppData\Local\Programs\Python\Python312\python.exe "c:\Users\DELL\OneDrive\Desktop\AI Assited\AI.py"
[0, 1, 1, 2, 3]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\DELL\OneDrive\Desktop\AI Assited>

```

Task Description#3

- Write a comment like # Function to reverse a string and use Copilot to generate the function.
- **Prompt : write a python code for Function to reverse a string**

Expected Output#3

-

The screenshot displays a Jupyter Notebook environment. The top bar shows the file name 'AI.py' and standard window controls. The notebook contains a Python script with the following code:

```

1 def reverse_string(s: str) -> str:
2     """Return the reversed string."""
3     return s[::-1]
4
5 # Example usages
6 if __name__ == "__main__":
7     print(reverse_string("hello"))      # Output: 'olleh'
8     print(reverse_string("Python"))    # Output: 'nohtyP'
9     print(reverse_string("12345"))     # Output: '54321'
10

```

The bottom panel shows the execution output for the script:

```

PS C:\Users\DELL\OneDrive\Desktop\AI Assited> C:/Users/DELL/AppData/Local/Programs/Python/Python312/python
on.exe "c:/Users/DELL/OneDrive/Desktop/AI Assited/AI.py"
olleh
nohtyP
54321
PS C:\Users\DELL\OneDrive\Desktop\AI Assited>

```

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.
- **Prompt : python code only for a program that simulates a basic calculator (add, subtract, multiply, divide)**

Expected Output#4

The screenshot shows a VS Code editor with a Python file named `AI.py`. The code defines four functions: `add`, `subtract`, `multiply`, and `divide`. The `divide` function includes a check for division by zero. Below the functions, there are example usages for each function. The terminal at the bottom shows the output of running the script, which matches the expected results from the code.

```

1  def add(a, b):
2      return a + b
3
4  def subtract(a, b):
5      return a - b
6
7  def multiply(a, b):
8      return a * b
9
10 def divide(a, b):
11     if b == 0:
12         return "Error: Division by zero"
13     return a / b
14
15 # Example usages
16 if __name__ == "__main__":
17     print("Add: 5 + 3 =", add(5, 3))          # Output: 8
18     print("Subtract: 10 - 4 =", subtract(10, 4)) # Output: 6
19     print("Multiply: 6 * 7 =", multiply(6, 7))   # Output: 42
20     print("Divide: 20 / 4 =", divide(20, 4))    # Output: 5.0
21     print("Divide: 5 / 0 =", divide(5, 0))      # Output: Error: Division by zero
22

```

Terminal Output:

```

on.exe "c:/Users/DELL/OneDrive/Desktop/AI Assisted/AI.py"
Add: 5 + 3 = 8
Subtract: 10 - 4 = 6
Multiply: 6 * 7 = 42
Divide: 20 / 4 = 5.0
Divide: 5 / 0 = Error: Division by zero
PS C:\Users\DELL\OneDrive\Desktop\AI Assisted>

```

Task Description#5

- Use a comment to instruct AI to write a function that reads a file and returns the number of lines..
- Prompt : python code only for a function that reads a file and returns the number of lines..

Expected Output#5

The screenshot shows a VS Code editor with a Python file named 'example.txt'. The code defines a function 'count_lines_in_file' that takes a filename as a string and returns the number of lines in the file. It includes a try-except block to handle 'FileNotFoundError'. Below the function, there is an example usage section that runs the function on 'example.txt' and prints the result. The terminal at the bottom shows the command to execute the script, and the output displays the number of lines in the file.

```

1 def count_lines_in_file(filename: str) -> int:
2     """Return the number of lines in the given file."""
3     try:
4         with open(filename, 'r') as f:
5             return sum(1 for _ in f)
6     except FileNotFoundError:
7         print(f"File not found: {filename}")
8         return 0
9
10 # Example usage
11 & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/AI Assited/AI.py"
12
13 if __name__ == "__main__":
14     # Replace "example.txt" with a valid file path to test
15     print("Number of lines in 'example.txt':", count_lines_in_file('example.txt'))
16

```

```

PS C:\Users\DELL\OneDrive\Desktop\AI Assited> & C:/Users/DELL/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/DELL/OneDrive/Desktop/AI Assited/AI.py"
Number of lines in 'example.txt': 5

```

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		