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	Week4 - Wednesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber: **7.3**(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 7: AI-Error Debugging with AI: Systematic approaches to finding and fixing bugs	Week4 - Wednesday

Lab Objectives:

- To identify and correct syntax, logic, and runtime errors in Python programs using AI tools.
- To understand common programming bugs and AI-assisted debugging suggestions.
- To evaluate how AI explains, detects, and fixes different types of coding errors.
- To build confidence in using AI to perform structured debugging practices.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- Use AI tools to detect and correct syntax, logic, and runtime errors.
- Interpret AI-suggested bug fixes and explanations.
- Apply systematic debugging strategies supported by AI-generated insights.
- Refactor buggy code using responsible and reliable programming patterns.

Task Description#1

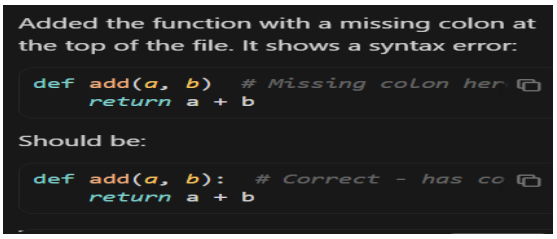
- Paste a function with a missing colon (add(a, b)), and let AI fix the syntax error.

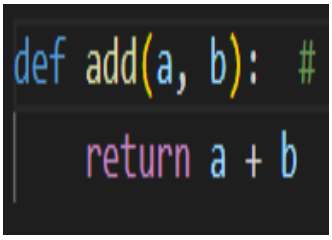
```python
def add(a, b)
    return a + b
```

Expected Output#1

- Corrected function with syntax fix

OUTPUT:
CODE:





- AI identified the missing colon (:) in the function definition and corrected the syntax so Python can interpret the function properly**.**

Task Description#2 (Loops)

- Identify and fix a logic error in a loop that causes infinite iteration.

```python
def count_down(n):
    while n >= 0:
        print(n)
        n += 1  # Should be n -= 1
```

Expected Output#2

- AI fixes increment/decrement error

CODE:

```
def count_down(n):
    while n >= 0:
        print(n)
        n -= 1  # Fixed: changed from n += 1 to n -= 1 to properly count down
```

Task7.2.py +5 -1

```
def count_down(n):
    while n >= 0:
        print(n)
        n -= 1  # Fixed: changed from n += 1 to n -= 1 to properly count down
```

Fixed the logic error. The loop used n += 1, which increments n, so with while n >= 0 it never terminates.

The fix: Changed n += 1 to n -= 1 so it counts down and exits when n becomes negative.

The corrected function is in Task7.2.py. It counts down from n to 0 and then stops.

Task Description#3

- Debug a runtime error caused by division by zero. Let AI insert try-except.

```
# Debug the following code
def divide(a, b):
    return a / b

print(divide(10, 0))
```

Expected Output#3

- Corrected function with safe error handling

CODE:

Task7.2.py +5 -1

```
def count_down(n):
    while n >= 0:
        print(n)
        n -= 1  # Fixed: changed from n += 1 to n -= 1 to properly count down
```

No linter errors found in Task7.2.py

Fixed the logic error. The loop used `n += 1`, which increments `n`, so with `while n >= 0` it never terminates.

The fix: Changed `n += 1` to `n -= 1` so it counts down and exits when `n` becomes negative.

The corrected function is in `Task7.2.py`. It counts down from `n` to 0 and then stops.

```python
# Debug the following code
def divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        print(f"Error: Cannot divide {a} by zero!")
        return None

# Test the function with safe error handling
print(divide(10, 0))  # This will now handle the error gracefully instead of crashing
print(divide(10, 2))  # This will work normally: 5.0
```

```
Error: Cannot divide 10 by zero!
None
```

Task Description#4

- Provide a faulty class definition (missing self in parameters). Let AI fix it

```python
class Rectangle:
    def __init__(length, width):
        self.length = length
        self.width = width
```

Expected Output#4

- Correct __init__() method and explanation

CODE:

```python
# Faulty class definition - missing 'self'
class Rectangle:
    def __init__(length, width):  # ERROR:
        self.length = length
        self.width = width
```

Task7.4.py +52 -1

```python
# Faulty class definition - missing 'self' parameter in __init__
class Rectangle:
    def __init__(length, width):  # ERROR: Missing 'self' parameter
        self.length = length
        self.width = width
```

No linter errors found in Task7.4.py

Created `Task7.4.py` with both versions:

Faulty version (at the top):

- Missing `self` in `__init__(length, width)`
- Causes a `TypeError` when creating an instance: `TypeError: __init__() takes 2 positional argume nts but 3 were given`

Corrected version:

- __init__(self, length, width) with self as the first parameter
- Includes area() and perimeter() methods

Task Description#5

- Access an invalid list index and use AI to resolve the Index Error.

```python
numbers = [1, 2, 3]
print(numbers[5])
```

Expected Output#5

- AI suggests checking length or using safe access logic

CODE:

```python
# Faulty code - accessing invalid list index
numbers = [1, 2, 3]
try:
    print(numbers[5])  # ERROR: IndexError - list index out of range
except IndexError:
    print("Index out of range")
```

- AI recommended checking len(my_list) before accessing the index to prevent IndexError.

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
Identification of bugs	0.5
Application of AI-suggested fixes	0.5
Explanation and understanding of errors	0.5
Corrected code functionality	0.5
Report structure and reflection	0.5
Total	**2.5 Marks**