

Name : V.Preetham Reddy

R.no : 2303A52372

Batch : 40

Lab 7:

Error Debugging with AI: Systematic approaches to finding and fixing bugs

Lab Objectives:

Week4 -

Monday

- 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 (Syntax Errors – Missing Parentheses in Print

Statement)

The screenshot shows a terminal window with the following content:

```
1
2     h="hello world"
3     print(h
PROBLEMS 2      OUTPUT      DEBUG CONSOLE      TERMINAL      PORTS
PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding> & C:/Users/sravan/AppData/Local/Programs/Py
eDrive/Desktop/AI_assisstant_codding/lab07.py
  File "c:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding\lab07.py", line 3
    print(h
          ^
SyntaxError: '(' was never closed
PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding>
```

Task: Provide a Python snippet with a missing parenthesis in a print

statement (e.g., `print "Hello"`). Use AI to detect and fix the syntax

error.

```
# Bug: Missing parentheses in print statement
```

```
def greet():
```

```
    print "Hello, AI Debugging Lab!"
```

```
greet()
```

Requirements:

- Run the given code to observe the error.

- Apply AI suggestions to correct the syntax.
- Use at least 3 assert test cases to confirm the corrected code works.

Expected Output #1:

- Corrected code with proper syntax and AI explanation.

Task Description #2 (Incorrect condition in an If Statement)

Task: Supply a function where an if-condition mistakenly uses = instead of ==. Let AI identify and fix the issue.

```
# Bug: Using assignment (=) instead of comparison (==)
```

```
def check_number(n):  
    if n = 10:  
        return "Ten"  
    else:  
        return "Not Ten"
```

Requirements:

- Ask AI to explain why this causes a bug.
- Correct the code and verify with 3 assert test cases.

Expected Output #2:

- Corrected code using == with explanation and successful test execution.

```
1
2
3 def check_number(n):
4     if n=10:
5         return "Ten"
6     else:
7         return "Not Ten"

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

① PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/sravan/AppData/Local/Programs/Python/Python313/pyt
rs/sravan/OneDrive/Desktop/AI_assisstant_coding/lab07.py
  File "c:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 4
    if n=10:
        ^
      SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
❖ PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding>
```

Task Description #3 (Runtime Error – File Not Found)

Task: Provide code that attempts to open a non-existent file and crashes. Use AI to apply safe error handling.

Bug: Program crashes if file is missing

```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()
    print(read_file("nonexistent.txt"))
```

Requirements:

- Implement a try-except block suggested by AI.
- Add a user-friendly error message.
- Test with at least 3 scenarios: file exists, file missing, invalid path.

```
1  def read_file(filename):
2      with open(filename, 'r') as f:
3          return f.read()
4  print(read_file("nonexistent.txt"))
5
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\saravanan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/saravanan/AppData/Local/OneDrive/Desktop/AI_assisstant_coding/lab07.py
  File "c:/Users/saravanan/OneDrive/Desktop/AI_assisstant_coding/lab07.py", line 2
    with open(filename, 'r') as f:
    ^^^^
IndentationError: expected an indented block after function definition on line 1
PS C:\Users\saravanan\OneDrive\Desktop\AI_assisstant_coding>
```

Expected Output #3:

- Safe file handling with exception management.

Task Description #4 (Calling a Non-Existent Method)

Task: Give a class where a non-existent method is called (e.g.,

obj.undefined_method()). Use AI to debug and fix.

```
# Bug: Calling an undefined method
```

```
class Car:
```

```
    def start(self):
```

```
        return "Car started"
```

```
my_car = Car()
```

```
print(my_car.drive()) # drive() is not defined
```

Requirements:

- Students must analyze whether to define the missing method or correct the method call.
- Use 3 assert tests to confirm the corrected class works.

```
lab07.py > 0 Car > start
1  class Car:
2      def start(self):
3          return "Car started"
4  my_car = Car()
5 → print(my_car.drive())
           ↴ start

Terminal (Ctrl+`)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

S C:\Users\srajan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/srajan/AppData/Local/Pr
Drive/Desktop/AI_assisstant_coding/lab07.py
raceback (most recent call last):
 File "c:\Users\srajan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 5, in <module>
 print(my_car.drive())
 ^^^^^^^^^^
AttributeError: 'Car' object has no attribute 'drive'
S C:\Users\srajan\OneDrive\Desktop\AI_assisstant_coding>

Expected Output #4:

- Corrected class with clear AI explanation.

Task Description #5 (TypeError – Mixing Strings and Integers in Addition)

Task: Provide code that adds an integer and string ("5" + 2) causing a TypeError. Use AI to resolve the bug.

```
# Bug: TypeError due to mixing string and integer

def add_five(value):

    return value + 5
```

```
print(add_five("10"))
```

Requirements:

- Ask AI for two solutions: type casting and string concatenation.
- Validate with 3 assert test cases.

The screenshot shows a code editor interface with a dark theme. At the top, there's a status bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. Below the status bar is the code area. The code is as follows:

```
lab07.py > add_five
1 def add_five(value):
2     return value + 5
3 → print(add_five("10")) print(add_five(10))
4
```

The line `3 → print(add_five("10"))` is highlighted with a blue rectangle, and the cursor is positioned at the end of the line. To the right of the code area, there's a terminal window showing the following output:

```
PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding> & C:/Users/sravan/AppData/Local/Programs/Python/Python310\python.exe lab07.py
Traceback (most recent call last):
  File "c:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding\lab07.py", line 3, in <module>
    print(add_five("10"))
           ~~~~~^~~
  File "c:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding\lab07.py", line 2, in add_five
    return value + 5
           ~~~~~^~~
TypeError: can only concatenate str (not "int") to str
PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_codding>
```

Expected Output #5:

- Corrected code that runs successfully for multiple inputs.

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