

AI ASSISTED CODING

LAB ASSIGNMENT-7

Name: M.Harshith

HT.NO: 2303A51444

Batch: 21

Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs

Question-1: Analyse given code where a mutable default argument causes unexpected behaviour. Use AI to fix it.

Screenshot:

The screenshot shows a code editor window with a dark theme. A file named 'lab1.py' is open, containing the following code:

```
lab1.py > ...
def add_item(item, items=[]):
    def add_item(item, items=None):
        if items is None:
            items = []
        items.append(item)
        return items
    print(add_item(1))
    print(add_item(2))
```

The code contains a bug where a mutable default argument ('items') is used as a default value for a parameter. This leads to both calls to `add_item` sharing the same list object. The first call adds '1' to the list, and the second call adds '2' to the same list, resulting in both prints being '[1, 2]'. The code is highlighted with syntax coloring, and the editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, along with a Python interpreter dropdown and a terminal window at the bottom.

Question-2: Analyse given code where floating-point comparison fails. Use AI to correct with tolerance.

Screenshot:

The screenshot shows a code editor interface with a dark theme. At the top, there is a toolbar with buttons for 'Keep' and 'Undo'. Below the toolbar, the code file 'lab1.py' is open, showing the following content:

```
lab1.py > ...
1 import math
2
3 def check_sum():
4     return (0.1 + 0.2) == 0.3
5     return math.isclose(0.1 + 0.2, 0.3)
6
7 print(check_sum())
```

The line `return (0.1 + 0.2) == 0.3` is highlighted in red, indicating an error. The line `return math.isclose(0.1 + 0.2, 0.3)` is highlighted in green, suggesting a correct alternative. The code is run in a terminal at the bottom, showing the output:

```
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
True
PS C:\Users\LIKHITHA\3\AIAC>
```

The terminal also shows the path to the Python executable and the command run.

Question-3: Analyse given code where recursion runs infinitely due to missing base case. Use AI to fix. Generating using colab:

Screenshot:

```
lab1.py > ...
● 1  def countdown(n):
2  if n < 0:
3      return
4  print(n)
5  return countdown(n-1)
6  countdown(5)

Keep Undo ⌛ | 1 of 1 ↑ ↓

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + ⌄ ⌂ ... [X]

True
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
5
4
3
2
1
0
```

Question-4: Analyse given code where a missing dictionary key causes error.
Use AI to fix it.

Screenshot:

The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor and terminal. In the code editor, a file named 'lab1.py' is open with the following content:

```
lab1.py > ...
1 def get_value():
2     data = {"a": 1, "b": 2}
3     return data["c"]
4     return data.get("c", None)
5 print(get_value())
```

The line `return data["c"]` is highlighted in red, indicating a syntax error. A floating action button with 'Keep Undo' and a trash icon is positioned above the line. The line `return data.get("c", None)` is highlighted in green, suggesting a fix or alternative. A second floating action button with 'Keep Undo' and a trash icon is at the bottom right of the editor area.

Below the code editor is a terminal window showing the command-line output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Keep Undo ⌛ | 1 of 1 ↑ ↓
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
None
PS C:\Users\LIKHITHA\3\AIAC>
```

The terminal tab is selected, and the status bar indicates the Python 3.13 environment is active.

Question-5: Analyse given code where loop never ends. Use AI to detect and fix it.

Screenshot:

```
lab1.py > ...
1  def loop_example():
2      i = 0
3      while i < 5:
4          print(i)
5          i += 1
6
7 loop_example()
8
```

Keep Undo ⌛ | 1 of 1 ↑ ↓

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + ⌄ ⌁ ⌂ ⌃ ⌅ ⌆

```
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
0
1
2
3
4
PS C:\Users\LIKHITHA\3\AIAC>
```

Question-6: Analyse given code where tuple unpacking fails. Use AI to fix it.

Screenshot:

The screenshot shows a code editor interface with a dark theme. At the top, there is a status bar with the text "Keep Undo ⌘S". Below the status bar, the code editor displays a file named "lab1.py" with the following content:

```
lab1.py > ...
1   a, b = (1, 2, 3)
2   # Bug: Wrong unpacking - FIXED
3   a, b, _ = (1, 2, 3)
4   print(f"a = {a}, b = {b}")
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Keep Undo ⌘S | 1 of 1 ↑ ↓
c:/Users/LIKHITHA/3/AIAC/lab1.py
0
1
2
3
4
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
a = 1, b = 2
PS C:\Users\LIKHITHA\3\AIAC>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. It also includes icons for switching environments (Python, +, dropdown), closing, and other terminal-related functions. The status bar at the bottom of the terminal window shows "Line 2 Col 27 Spaces 4".

Question-7: Analyse given code where mixed indentation breaks execution. Use AI to fix it.

Screenshot:

The screenshot shows a Python file named `lab1.py` in a code editor. The code contains a function `func` with two return statements and a print statement. A tooltip is visible over the second return statement, showing options to "Keep", "Undo", or "Fix". The terminal below shows the command run and the output, which is the value 15, indicating that the code was executed despite the mixed indentation.

```
# Bug: Mixed indentation - FIXED
def func():
    x = 5
    y = 10
    return x+y
    return x + y
print(func())
```

```
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
15
PS C:\Users\LIKHITHA\3\AIAC>
```

Question-8: Analyse given code with incorrect import. Use AI to fix.

Screenshot:

The screenshot shows a VS Code interface. The top part displays a Python file named `lab1.py` with the following code:

```
lab1.py
import maths
print(maths.sqrt(16))
# Bug: Wrong import - FIXED
import math
print(math.sqrt(16))
```

A red bar highlights the line `import maths`, indicating an error. A blue bar highlights the line `# Bug: Wrong import - FIXED`. The bottom part shows the terminal output:

```
PS C:\Users\LIKHITHA\3\AIAC> & C:/Users/LIKHITHA/AppData/Local/Programs/Python/Python313/python.exe
c:/Users/LIKHITHA/3/AIAC/lab1.py
4.0
PS C:\Users\LIKHITHA\3\AIAC>
```