

# Lab 7: Error Debugging with AI (Week 4)

**Subject:** Ai assist coding

**Experiment Title:** Error Debugging with AI

**Name:** Sanjay Karupothula

**Roll No:** 2303A52337

**BATCH:** 34

**Branch:** BTech CSE

**Lab:** Week 4 – Error Debugging with AI

**Date:** 03-02-2026

---

## 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

- 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 1 – Runtime Error Due to Invalid Input Type

### Buggy Code

```
num = input("Enter a number: ")  
result = num + 10  
print(result)
```

## AI Explanation

The `input()` function always returns a string. Adding an integer (10) to a string causes a `TypeError`. AI identifies that the user input must be converted to a numeric type such as `int` before performing arithmetic operations.

## Corrected Code

```
num = int(input("Enter a number: "))
result = num + 10
print(result)
```

## Output

Enter a number: 5  
15

A screenshot of the OneCompiler web interface. The top navigation bar includes a logo, a search bar, and links for Pricing and Learn. The main area shows a code editor with a file named "main.py" containing the corrected code. To the right is a terminal window with "STDIN" showing the input "5" and "Output" showing the result "Enter a number: 15".

```
main.py + 44cgcajg3 ⚡
1 num = int(input("Enter a number: "))
2 result = num + 10
3 print(result)
```

STDIN  
5

Output:  
Enter a number: 15

## ✓ Task 2 – Incorrect Function Return Value

### Buggy Code

```
def square(n):
    result = n * n
```

### AI Explanation

The function calculates the square but does not return the result. Without a return statement, Python returns `None`. AI suggests adding a return statement so the computed value is sent back to the caller.

## Corrected Code

```
def square(n):
    result = n * n
    return result
```

```
print(square(4))
```

## Output

16

---

≡  OneCompiler

main.py+44cgcajg3 Ø

1 def square(n):  
2 result = n \* n  
3 return result  
4 print(square(4))  
5  
6  
7  
8  
a

Output:  
16

---

## Task 3 – IndexError in List Traversal

### Buggy Code

```
numbers = [10, 20, 30]  
for i in range(0, len(numbers)+1):  
    print(numbers[i])
```

### AI Explanation

Using `len(numbers)+1` makes the loop go one step beyond the last valid index, causing `IndexError`. AI fixes this by looping only up to `len(numbers)`.

### Corrected Code

```
numbers = [10, 20, 30]  
for i in range(len(numbers)):  
    print(numbers[i])
```

## Output

10  
20  
30

The screenshot shows a OneCompiler interface. On the left, there is a code editor window titled "main.py" containing the following Python code:

```
1 numbers = [10, 20, 30]
2 for i in range(len(numbers)):
3     print(numbers[i])
4
5
6
7
8
9
```

To the right of the code editor is a status bar with the identifier "44cgcajg3 ⚡". Below the status bar, there is an "Output:" section which displays the following text:

```
10
20
30
```

## ✓ Task 4 – Uninitialized Variable Usage

### Buggy Code

```
if True:
    pass
print(total)
```

### AI Explanation

The variable `total` is used before being assigned any value, which causes a `NameError`. AI initializes the variable before it is used.

### Corrected Code

```
total = 0
if True:
    pass
print(total)
```

### Output

```
0
```

The screenshot shows a OneCompiler interface. On the left, there is a code editor window titled "main.py" containing the following Python code:

```
1 total = 0
2 if True:
3     pass
4 print(total)
5
6
7
8
9
```

To the right of the code editor is a status bar with the identifier "44cgcajg3 ⚡". Below the status bar, there is an "Output:" section which displays the following text:

```
0
```

## Task 5 – Logical Error in Student Grading System

### Buggy Code

```
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "C"
else:
    grade = "B"
print(grade)
```

### AI Explanation

The grading logic is incorrect because marks between 80 and 89 should receive grade B, not C. AI rearranges the conditions to assign grades properly.

### Corrected Code

```
marks = 85
if marks >= 90:
    grade = "A"
elif marks >= 80:
    grade = "B"
else:
    grade = "C"
print(grade)
```

### Output

B

≡ [OneCompiler](#)

main.py + 44cgcajg3 ⚙

```
1 marks = 85
2 if marks >= 90:
3     grade = "A"
4 elif marks >= 80:
5     grade = "B"
6 else:
7     grade = "C"
8 print(grade)
9
10
11
12
13
```

Output:  
B

### Conclusion

In this lab, different Python errors such as runtime errors, logical errors, index errors, and uninitialized variables were identified and corrected using AI assistance. AI helped explain

each mistake and provided suitable fixes. This experiment improved debugging skills and increased confidence in writing reliable Python programs.