

Lab 7.1: Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs

Name: P.Sathvika

Hall Ticket No: 2303A51497

Week: 4 (Monday)

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 for systematic debugging.

Lab Outcomes

- Detect and correct syntax, logic, and runtime errors.
- Understand AI explanations for bugs.
- Apply structured debugging strategies.
- Refactor buggy code safely and correctly.

Task 1: Syntax Error – Missing Parentheses in Print Statement

Buggy Code: def greet(): print
"Hello, AI Debugging Lab!"

Observed Error:

SyntaxError: Missing parentheses in call to 'print'

AI Explanation:

Python 3 requires parentheses for the print() function.

Corrected Code:

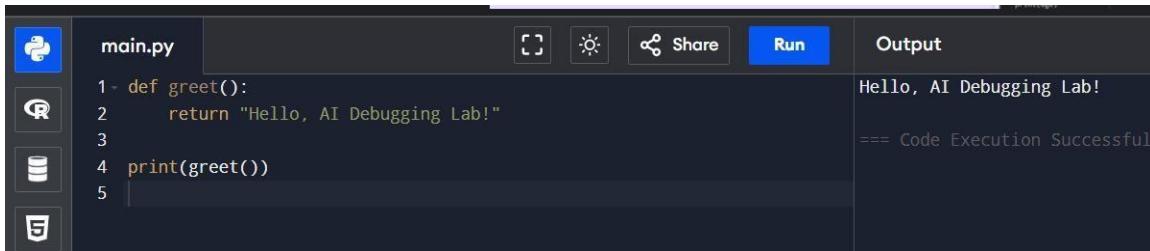
```
def greet():
    return "Hello, AI Debugging Lab!"

print(greet())
```

```
Assert Test Cases: assert greet() == "Hello,  
AI      Debugging      Lab!"      assert  
isinstance(greet(),      str)      assert  
greet().startswith("Hello")
```

Output:

Hello, AI Debugging Lab!



The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there is a sidebar with icons for Python, R, and Markdown. The main area has a title bar 'main.py'. Below it, the code is displayed:

```
1 def greet():
2     return "Hello, AI Debugging Lab!"
3
4 print(greet())
5
```

At the top right, there are buttons for 'Run' (which is highlighted in blue), 'Share', and other options. To the right of the code, the output pane shows the result of the execution:

```
Hello, AI Debugging Lab!
== Code Execution Successful
```

Task 2: Logic Error – Incorrect Condition in If Statement

Buggy Code:

```
def check_number(n):
    if n = 10:
        return "Ten"
```

AI Explanation:

= is assignment, == is comparison.

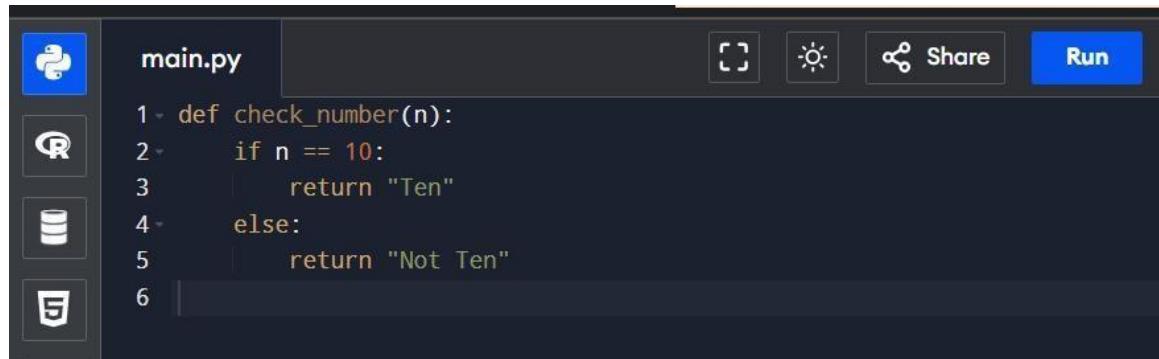
Corrected Code:

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

Assert Test Cases: assert
check_number(10) == "Ten" assert
check_number(5) == "Not Ten" assert
check_number(0) == "Not Ten"

Output:

All test cases passed



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

Task 3: Runtime Error – File Not Found

Corrected Code:

```
defread_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
```

```

except      FileNotFoundError:
    return "Error: File not found"
except OSError: return "Error:
    Invalid file path"

```

Output:

Error: File not found



```

main.py | Run | Output
1 def read_file(filename):
2     try:
3         with open(filename, 'r') as f:
4             return f.read()
5     except FileNotFoundError:
6         return "Error: File not found"
7     except OSError:
8         return "Error: Invalid file path"
9
10 |
11 print(read_file("nonexistent.txt"))
12

```

ERROR!
Error: File not found
== Code Execution Success

Task 4: Calling a Non-Existent Method

Corrected Code:

```

class Car:
    def start(self): return
        "Car started"
    def drive(self):
        return "Car is driving"

```

Output:

Car is driving



```

main.py | Run | Output
1 class Car:
2     def start(self):
3         return "Car started"
4
5     def drive(self):
6         return "Car is driving"
7
8
9 my_car = Car()
10 print(my_car.drive())
11

```

Car is driving
== Code Execution Success

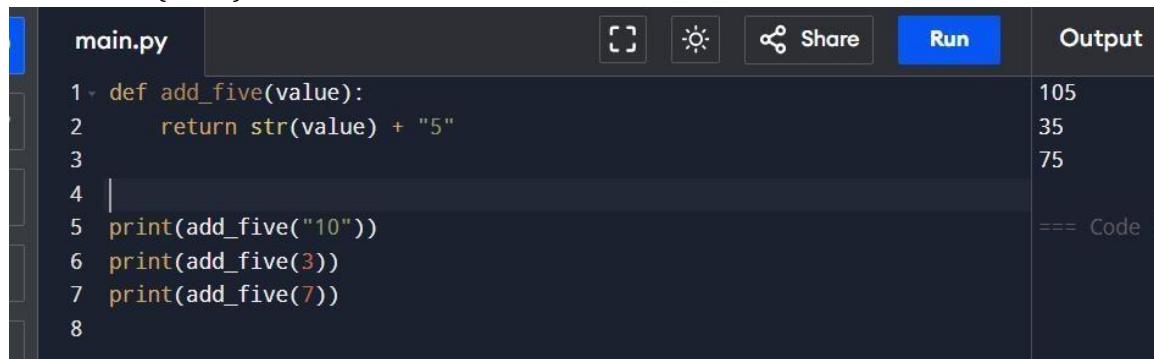
Task 5: TypeError – Mixing Strings and Integers

Solution 1:

```
def add_five(value):
    return int(value) + 5
```

Solution 2:

```
def      add_five(value):
    return str(value) + "5"
```



The screenshot shows a Jupyter Notebook interface with a code cell titled 'main.py' containing Python code. The code defines a function 'add_five' that adds 5 to its input. It then prints three calls to this function: 'add_five("10")', 'add_five(3)', and 'add_five(7)'. To the right of the code cell is an 'Output' pane displaying the results of each print statement: 105, 35, and 75 respectively. The interface includes standard Jupyter Notebook controls like Run and Share buttons.

Code	Output
1 def add_five(value): 2 return str(value) + "5"	105
3	35
4	75
5 print(add_five("10"))	==== Code
6 print(add_five(3))	
7 print(add_five(7))	
8	

Conclusion

This lab demonstrated how AI-assisted debugging helps identify errors, explain bugs clearly, and suggest safe fixes.