# 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!"

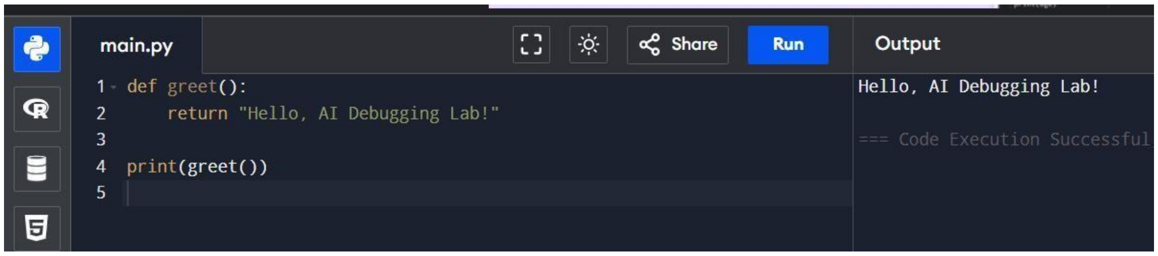
Observed Error:
SyntaxError: Missing parentheses in call to 'print'

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!



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

print(greet())
```

```
Hello, AI Debugging Lab!

=== Code Execution Successful
```

Task 2: Logic Error – Incorrect Condition in If Statement

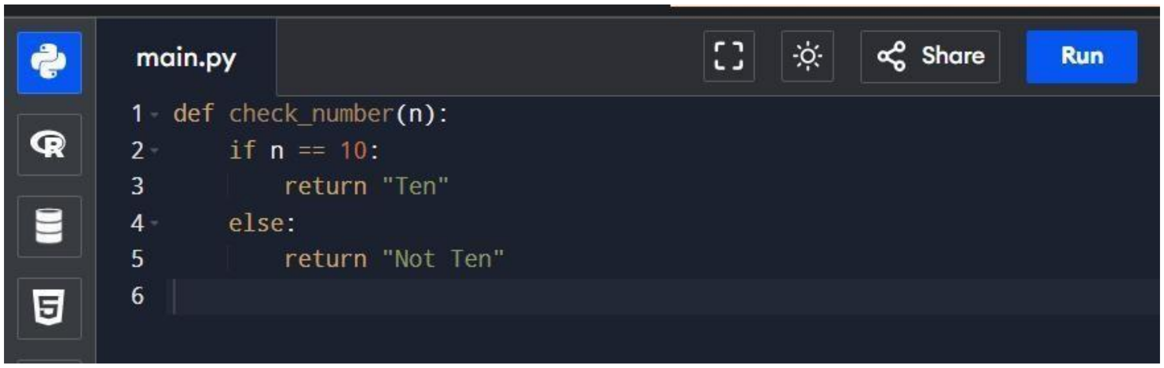
Buggy Code:
```
def   check_number(n):
   if n = 10:
       return "Ten"
```

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



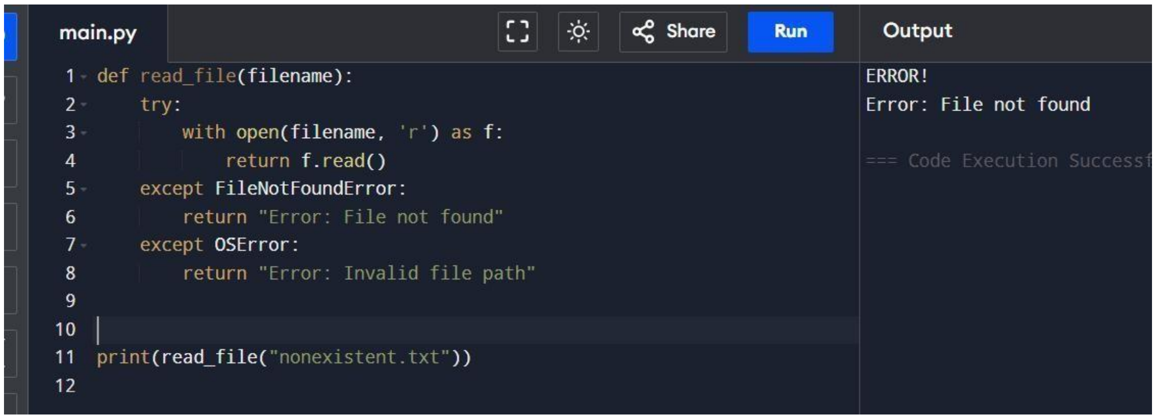
```
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"
```

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



```
def read_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"


print(read_file("nonexistent.txt"))
```

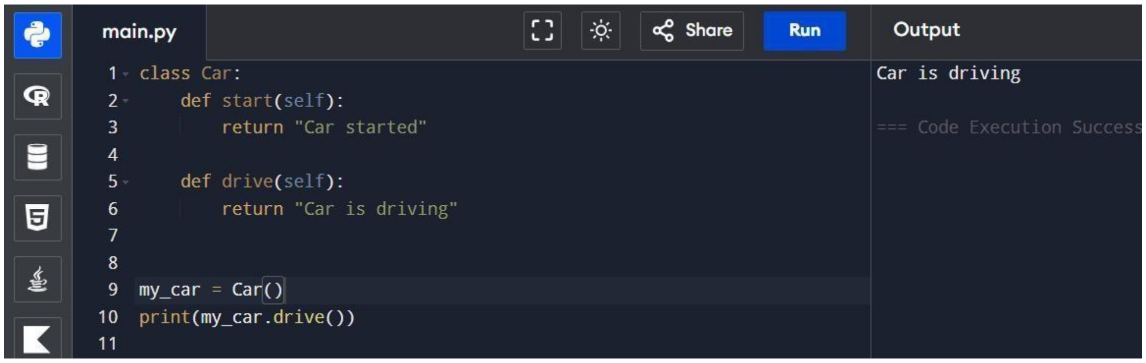
```
ERROR!
Error: File not found

=== Code Execution Successf
```

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



```
class Car:
    def start(self):
        return "Car started"

    def drive(self):
        return "Car is driving"


my_car = Car()
print(my_car.drive())
```

```
Car is driving

=== Code Execution Success
```

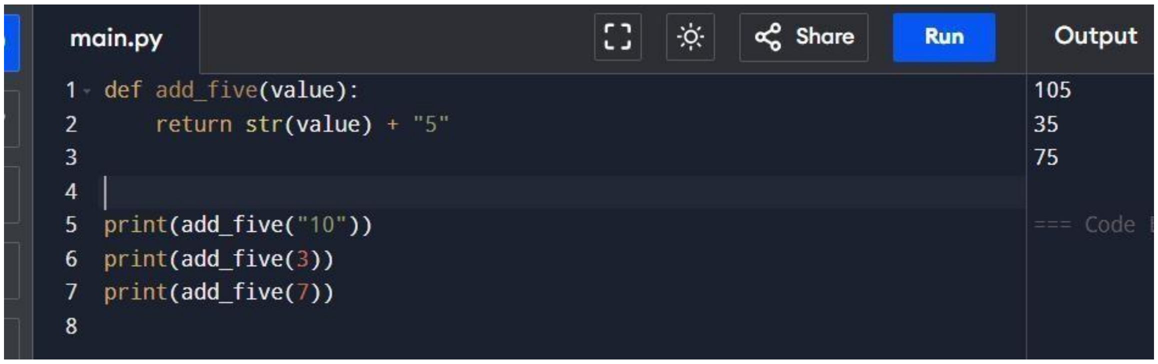
Task 5: TypeError – Mixing Strings and Integers

Solution 1:

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

Solution 2:

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



main.py

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


print(add_five("10"))
print(add_five(3))
print(add_five(7))
```

Output

```
105
35
75

=== Code
```

## Conclusion

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