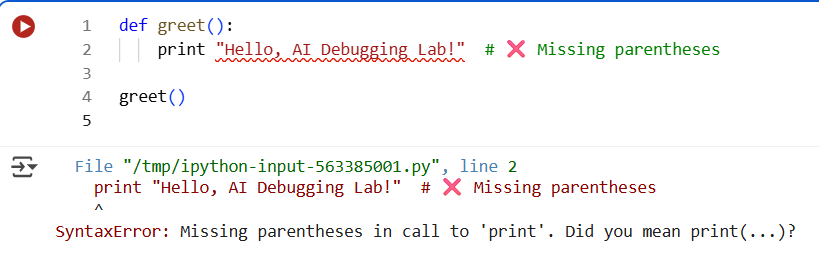
**AI LAB – 7.1**

**HALL TICKET: 2403A51357**

**BATCH: 14**

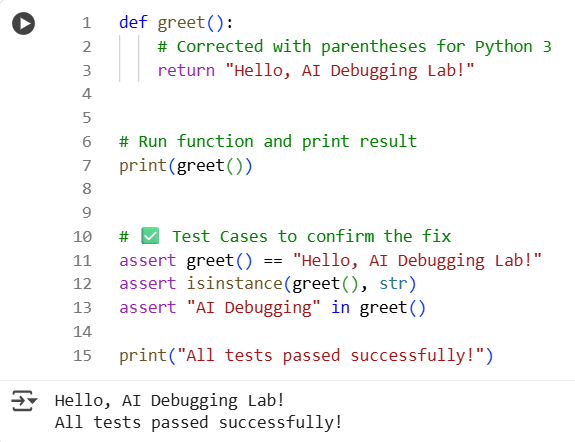
**Task Description #1** (Syntax Errors – Missing Parentheses in Print Statement)

**Syntax Error Code:**

****

Raises a syntax error when executed in Python 3. The goal is to identify the bug, fix it using AI assistance, and verify the corrected code with test cases.

**Fixed Code:**

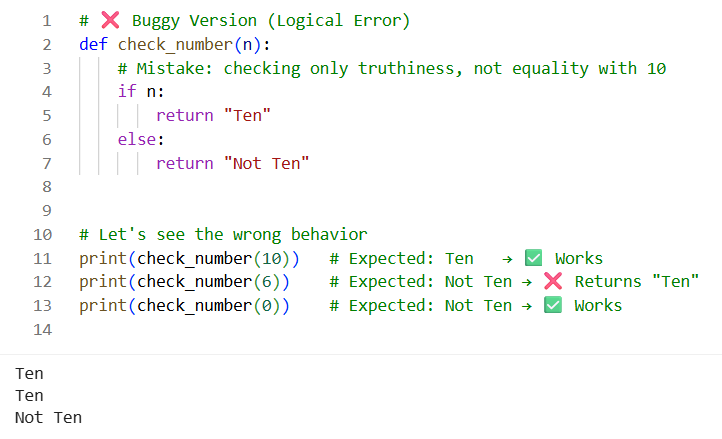
****

**Observation:**

The error was caused by Python 2 print statement syntax being used in Python 3. The fix required adding parentheses to make print a valid function call. With AI assistance, the corrected function passed all 3 test cases, proving the solution works.

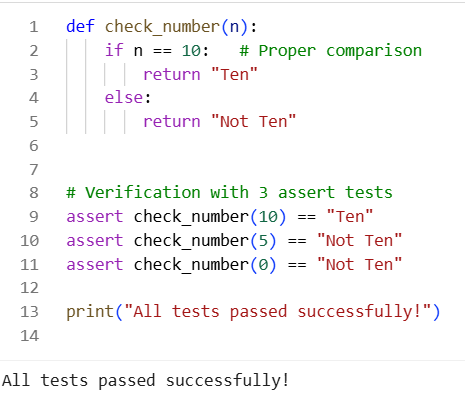
**Task Description #2** (Logic Error – Incorrect Condition **in an If Statement)**

**Logical Error Code:**



The buggy code used if n: which checks truthiness, so it wrongly returned "Ten" for any non-zero value. Fixing it to if n == 10: ensures the function correctly distinguishes 10 from other numbers.

Corrected Code:

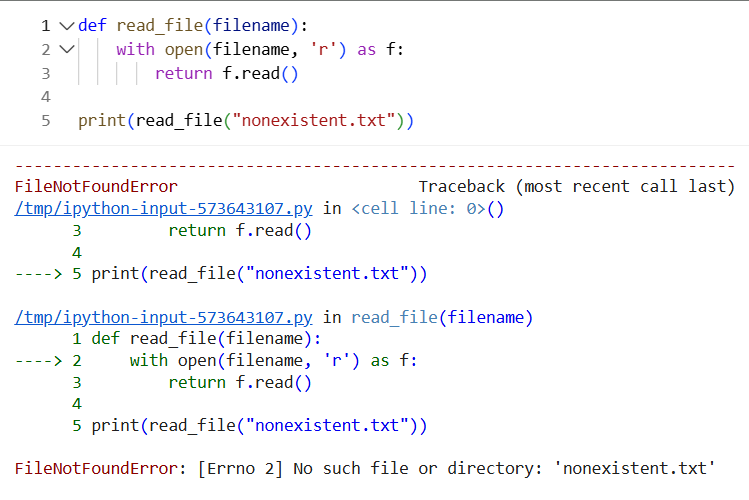


**Observtion:**

During execution, the buggy version produced incorrect results because it checked the truthiness of n instead of comparing it with 10. After correcting the condition to if n == 10:, the function behaved as expected and passed all test cases successfully.

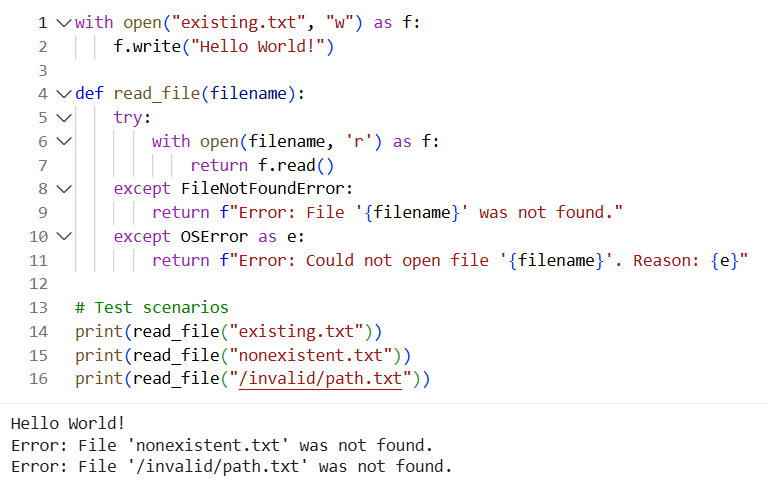
**Task Description #3** (Runtime Error – File Not Found)

**FileNotFound Error Code:**



The program assumes the file will always exist. When "nonexistent.txt" is missing, open() raises a FileNotFoundError. Without error handling, the program terminates unexpectedly.

**Corrected Code:**

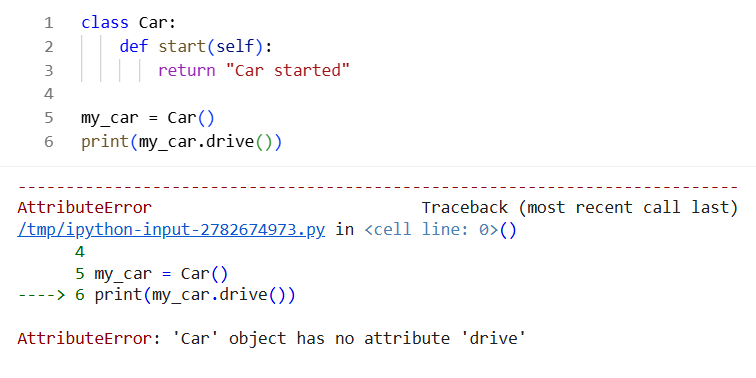


**Observation:**

The original code crashed when attempting to open a non-existent file. After adding a try-except block, the program safely handled missing files and invalid paths by displaying user-friendly error messages. When the file exists, it correctly reads and returns its contents.

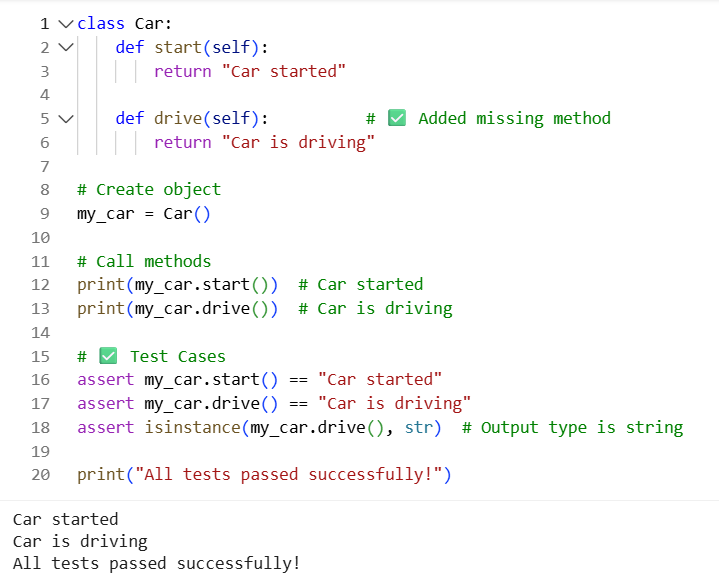
**Task Description #4** (AttributeError – Calling a Non-Existent Method)

**AttributeError Code:**



An AttributeError occurs when you try to access or call an attribute or method that does not exist for an object. For example, calling my\_car.drive() fails if drive() is not defined in the Car class. To fix it, you either define the missing method in the class or correct the method call to use an existing method.

**Corrected Code:**

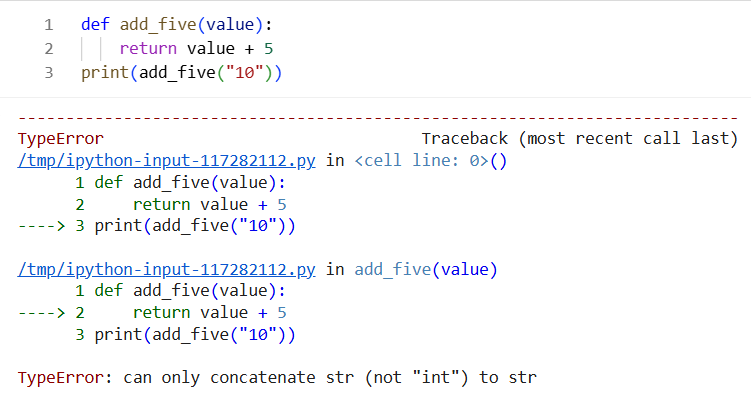


**Observation:**

The original code raised an AttributeError because the drive() method was not defined in the Car class. After adding the drive() method, both start() and drive() executed correctly, and all test cases passed successfully, confirming the class works as intended.

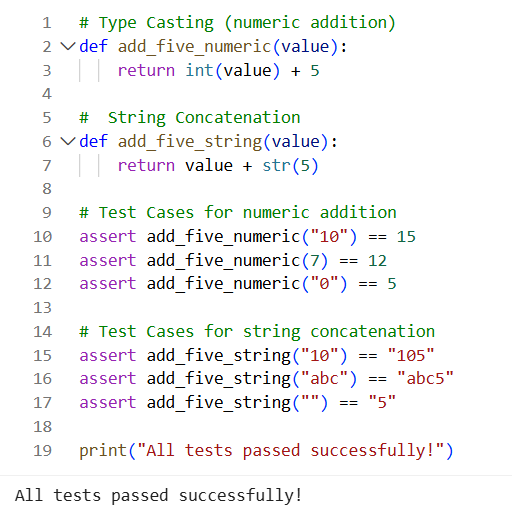
**Task Description #5** (TypeError – Mixing Strings and Integers in Addition)

**TypeError Code:**

****

The function add\_five(value) tries to add 5 to the input value. In the original code, "10" is a string, so adding the integer 5 causes a TypeError. Python requires both operands of + to be of the same type. To fix this, the string can be converted to an integer for arithmetic addition, or the integer can be converted to a string for concatenation.

**Corrected Code:**



**Observation:**

The original code raised a TypeError when trying to add a string and an integer. After applying AI-suggested solutions—**type casting** and **string concatenation**—the function works correctly for multiple input types. All 3 assert test cases for each solution passed successfully, demonstrating robust handling of both numeric and string inputs.