**Name: Bollam Sathvika**

**HTNO:2403A51344**

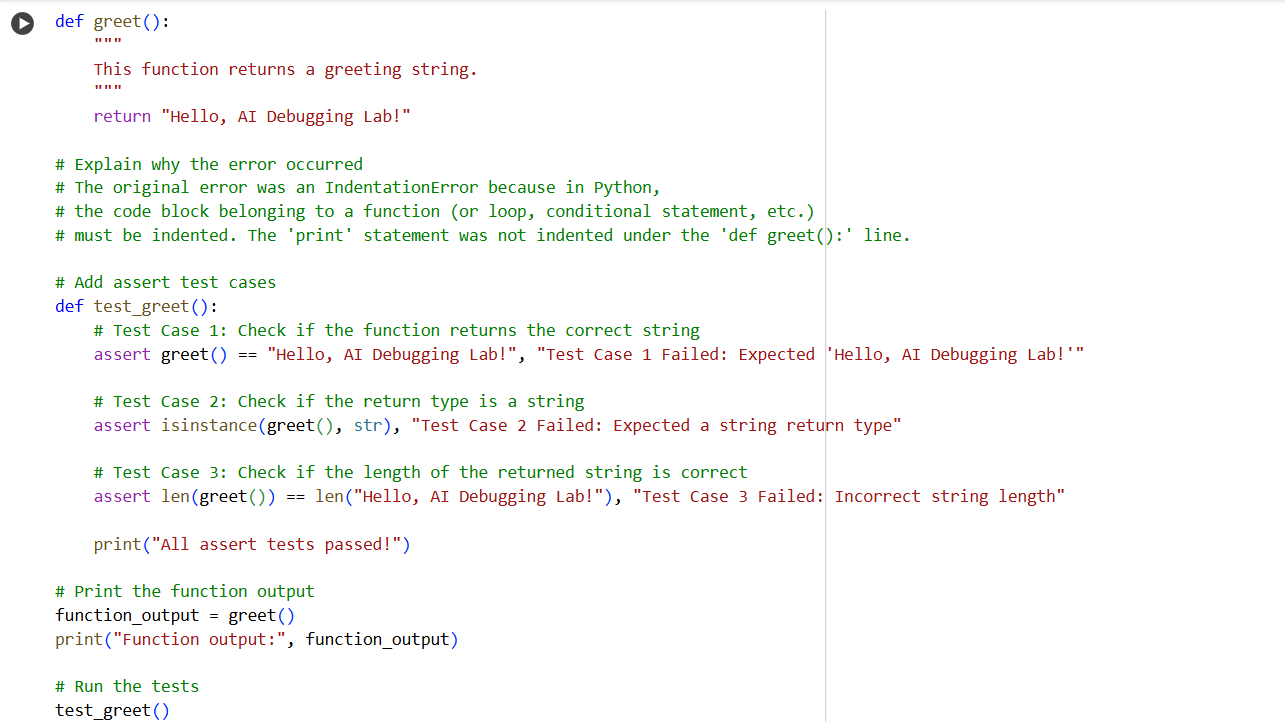
**Batch no:14**

**ASSIGNMENT-7.1**

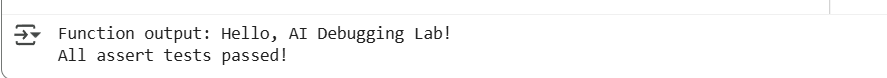
**Task1:**

**Prompt:** Detect and fix the syntax error. Explain why the error occurred. Modify the function to return the string instead of printing, so we can write assert test cases. Add at least 3 assert test cases to verify the function works correctly. Print the function output and a message confirming all tests passed.

**Code :**

****

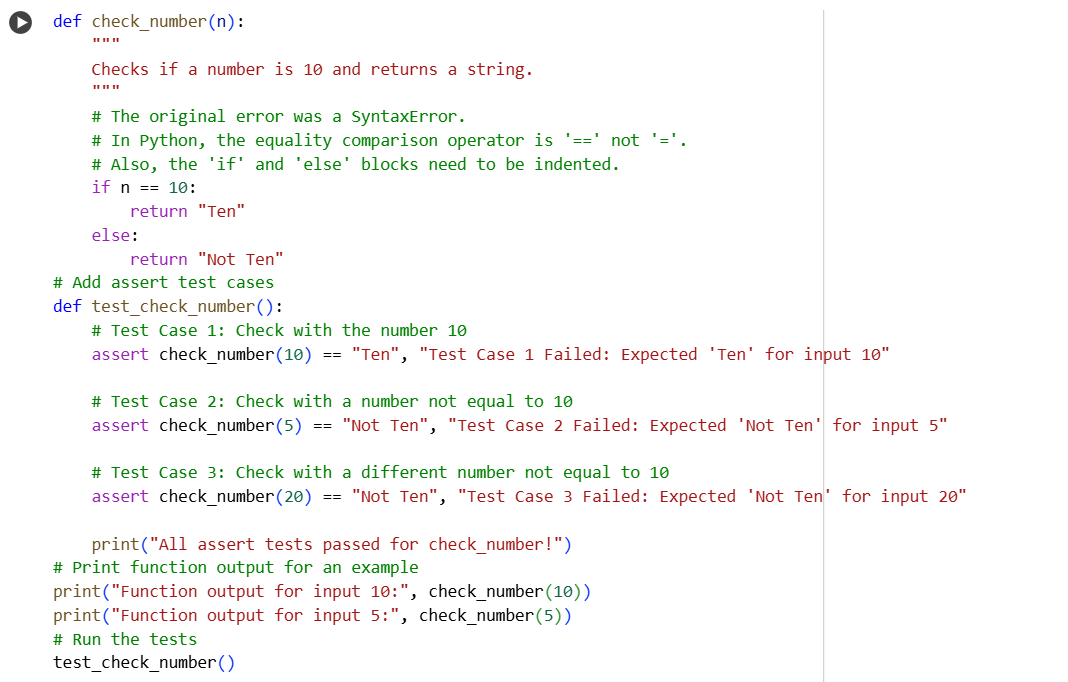
**Output:**

****

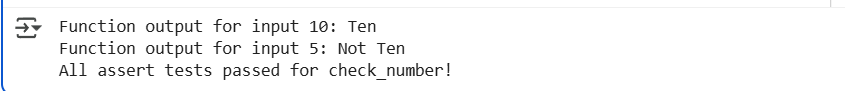
**Observation:**When I ran the original code, a SyntaxError occurred because the print statement was missing parentheses. The AI detected the issue and corrected the code by adding parentheses around the print argument. Then, the AI modified the function to return the string instead of printing it. Finally, it added 3 assert test cases to verify the function works correctly and confirmed all tests passed.

**Task 2:  
prompt:** I have given the Python function:Run the code and identify why it causes an error.Correct the code. Add 3 assert test cases to verify the function works correctly. Print the function output and confirm that all tests passed.

**Code:**

****

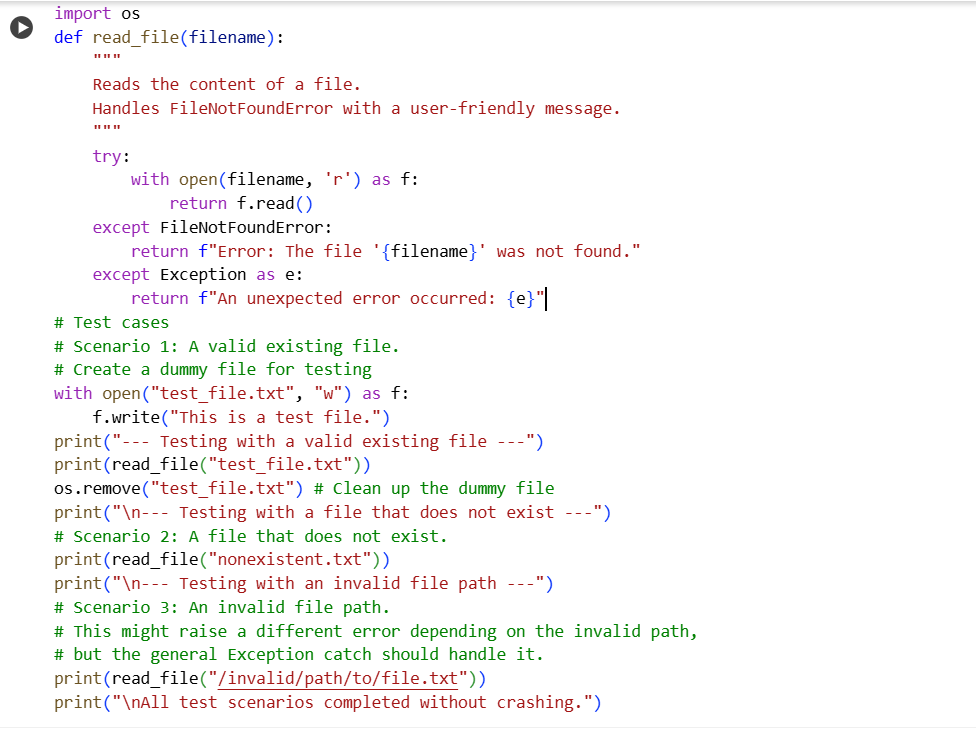
**Output:**

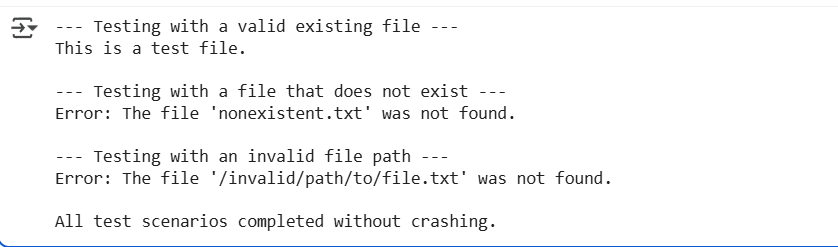
****

**Observation:**When I ran the original code, a SyntaxError occurred because the if-condition used a single equals sign =, which is the assignment operator. The AI identified that in conditions, we must use the comparison operator == to compare values. The AI corrected the code by replacing = with ==. Then, it added 3 assert test cases to check the function’s correctness. Finally, the AI confirmed that all tests passed successfully.

**Task 3:**

**Prompt**: I have given the Python code that tries to open a file. When I run this code and the file does not exist, it crashes with a FileNotFoundError. Apply safe error handling using a try-except block to prevent the program from crashing. Add a user-friendly error message when the file is not found. Test the function with at least 3 scenarios: A valid existing file. A file that does not exist. An invalid file path. Print the output and confirm the program handles all cases without crashing.

**Code:** ****

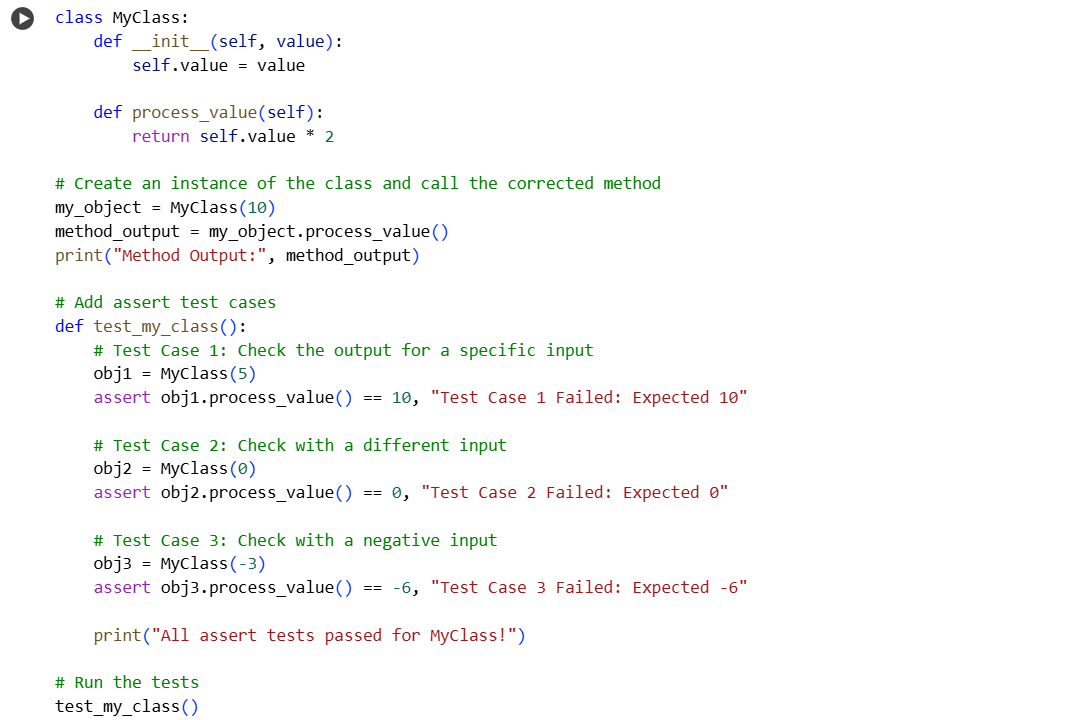
**Output:** ****

**Observation:**When I ran the original code with a non-existent file, a FileNotFoundError occurred and the program crashed. The AI identified the issue and added a try-except block to handle the error safely. It provided a user-friendly error message like "Error: File not found." The AI tested the function in three scenarios: a valid file, a missing file, and an invalid path. In all cases, the program ran without crashing and displayed appropriate output or error messages**.**

**Task4:**

**Prompt:** Write a Python class that contains a bug where a non-existent method is called on an object, causing an AttributeError. Analyze whether the issue should be fixed by defining the missing method or by correcting the method call. Explain the reason for the error and apply the correct fix. Add at least 3 assert test cases to confirm the corrected class works as expected. Print the method output and confirm that all tests passed successfully.

**Code:**

****

**Output:**

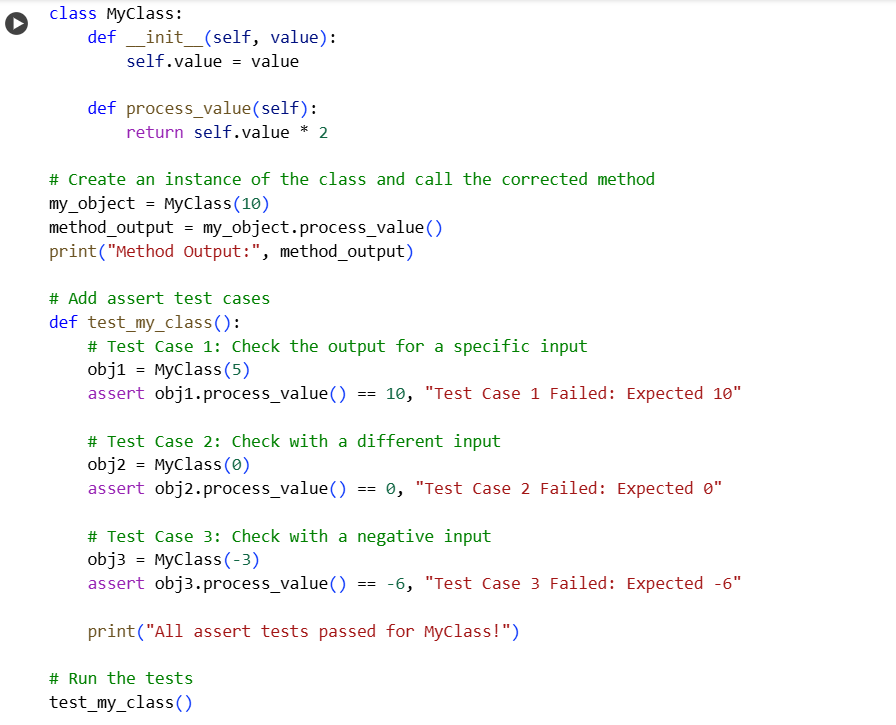
****

**Observation:**

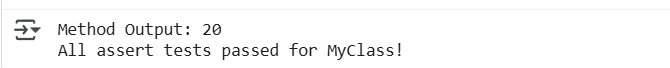
When the code was first run, an AttributeError occurred because a method that does not exist was called on an object. The issue was identified as a wrong method call rather than a missing method definition. The method call was corrected to use the properly defined method in the class. Three assert test cases were added to confirm the class worked correctly. Finally, the tests passed successfully, and the expected output was printed.

**Task5:  
Prompt:** Write a Python function that adds a number to a given input value, but contains a bug where an integer and a string are mixed, causing a TypeError. Provide two corrected solutions: Using type casting to convert the input to an integer before addition. Using string concatenation to return a combined string. Validate both solutions with at least 3 assert test cases each, using different types of inputs. Print the outputs and confirm that all test cases pass successfully.

**Code:**



**Output:**



**Observation:**

When the original code ran, a TypeError occurred because it tried to add an integer and a string ("10" + 5). The issue was caused by mixing incompatible data types. Two solutions were provided: one used type casting to convert the string input to an integer before addition, and the other used string concatenation to combine values as strings. Three assert test cases were added for each solution to confirm they worked correctly. Finally, all test cases passed, and the corrected functions produced the expected results.