AI ASSIGNMENT:7.1

HTNO:2403A51284

BATCH:12

Task Description #1 (Syntax Errors – Missing Parentheses in Print Statement)
Task: Provide a Python snippet with a missing parenthesis in a print statement (e.g., print "Hello"). Use AI to detect and fix the syntax error.

```
# Bug: Missing parentheses in print statement
def greet():
print "Hello, AI Debugging Lab!"
greet()
```

Requirements:
• Run the given code to observe the error.
• Apply AI suggestions to correct the syntax.
• Use at least 3 assert test cases to confirm the corrected code works.

Expected Output #1:
• Corrected code with proper syntax and AI explanation.

PROMPT:

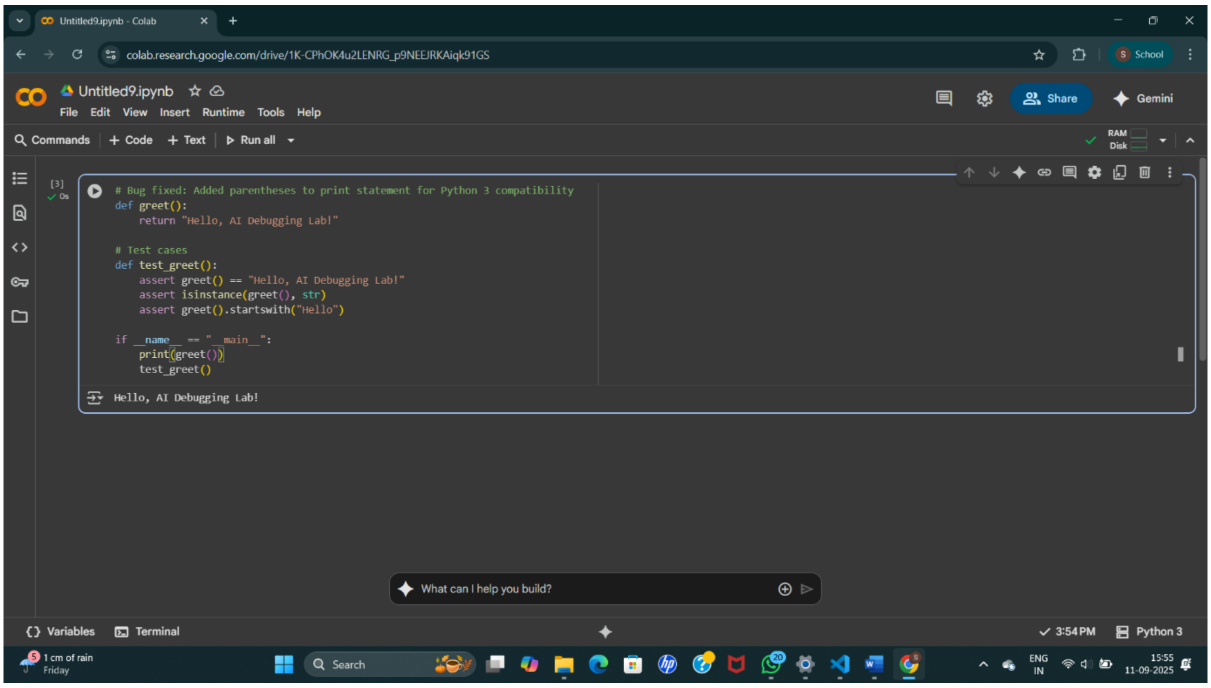
The following Python code contains a syntax error due to a missing parenthesis in the print statement.

Use AI to detect and fix the syntax error.

Add at least 3 assert test cases to confirm the corrected code works.

Provide the corrected code and a brief explanation of the fix.

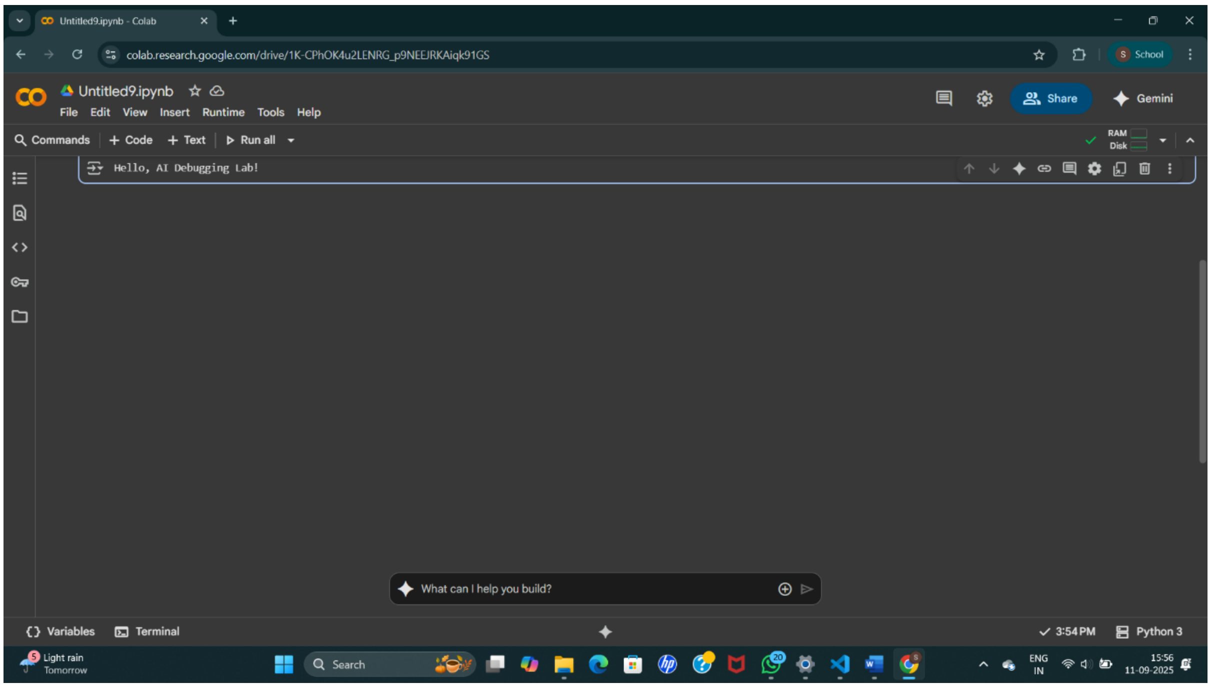
Buggy Code Example:



```
# Bug fixed: Added parentheses to print statement for Python 3 compatibility
def greet():
    return "Hello, AI Debugging Lab!"

# Test cases
def test_greet():
    assert greet() == "Hello, AI Debugging Lab!"
    assert isinstance(greet(), str)
    assert greet().startswith("Hello")

if __name__ == "__main__":
    print(greet())
    test_greet()
```

```
Hello, AI Debugging Lab!
```

OUTPUT:



```
Hello, AI Debugging Lab!
```

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

Task: Supply a function where an if-condition mistakenly uses = instead of ==. Let AI identify and fix the issue.

# Bug: Using assignment (=) instead of comparison (==)
def check_number(n):
if n = 10:
return "Ten"
else:
return "Not Ten"

Requirements:
• Ask AI to explain why this causes a bug.
• Correct the code and verify with 3 assert test cases.
Expected Output #2:
• Corrected code using == with explanation and successful test
execution.

PROMPT:

The following Python code attempts to open a file that may not exist, which can cause the program to crash if the file is missing.

1 vulnerability

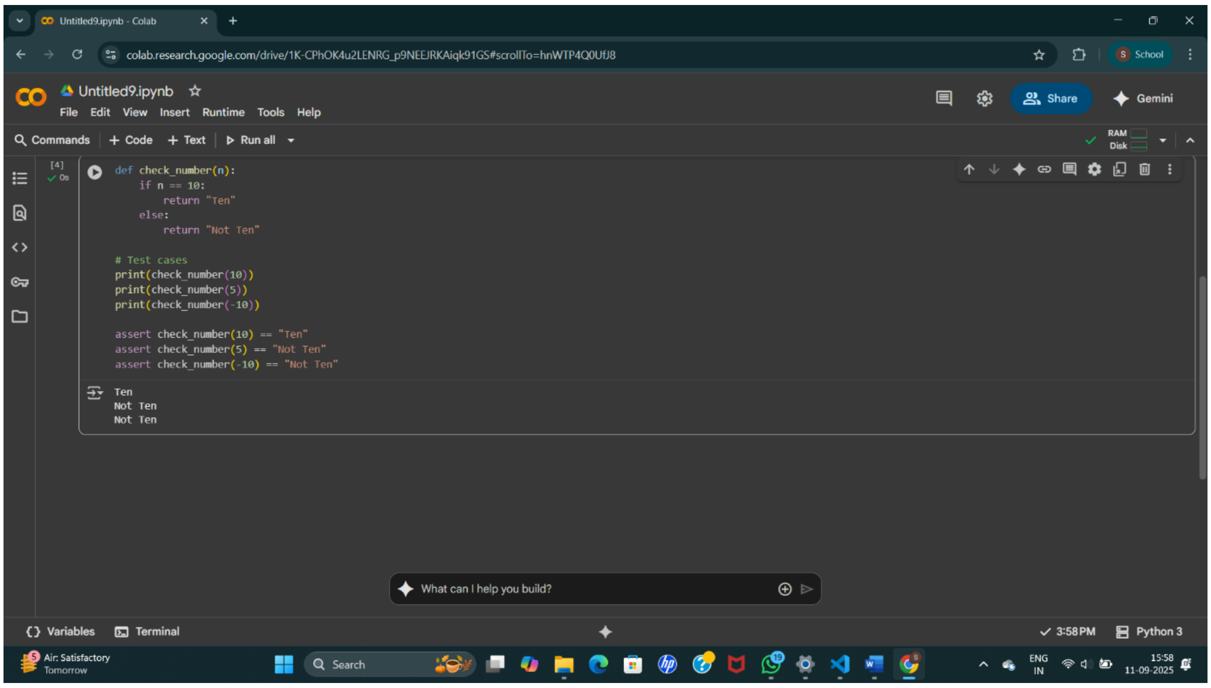
Use AI to identify and explain the error.

Correct the code by implementing safe error handling (e.g., using a try-except block).

Add a user-friendly error message for missing or invalid files.

Test the function with at least 3 scenarios: file exists, file missing, and invalid path.

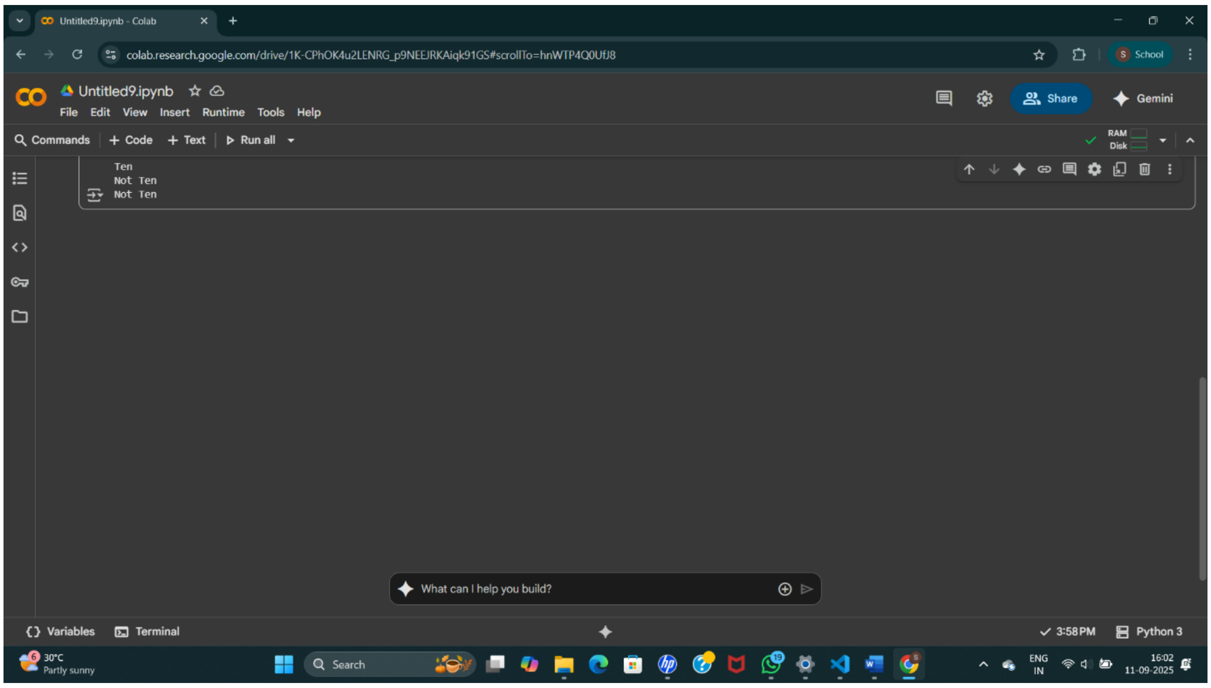
CODE:


Untitled9.ipynb - Colab
colab.research.google.com/drive/1K-CPhOK4u2LENRG_p9NEEJRKAiqk91GS#scrollTo=hnWTP4Q0UfJ8
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share Gemini
What can I help you build?
Variables Terminal
3:58 PM Python 3

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

# Test cases
print(check_number(10))
print(check_number(5))
print(check_number(-10))

assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(-10) == "Not Ten"
```

```
Ten
Not Ten
Not Ten
```

OUTPUT:



```
Ten
Not Ten
Not Ten
```

Task Description #3 (Runtime Error – File Not Found)
Task: Provide code that attempts to open a non-existent file and crashes.
Use AI to apply safe error handling.
# Bug: Program crashes if file is missing
def read_file(filename):
with open(filename, 'r') as f:
return f.read()
print(read_file("nonexistent.txt"))
Requirements:
• Implement a try-except block suggested by AI.
• Add a user-friendly error message.
• Test with at least 3 scenarios: file exists, file missing, invalid path.
Expected Output #3:
• Safe file handling with exception management

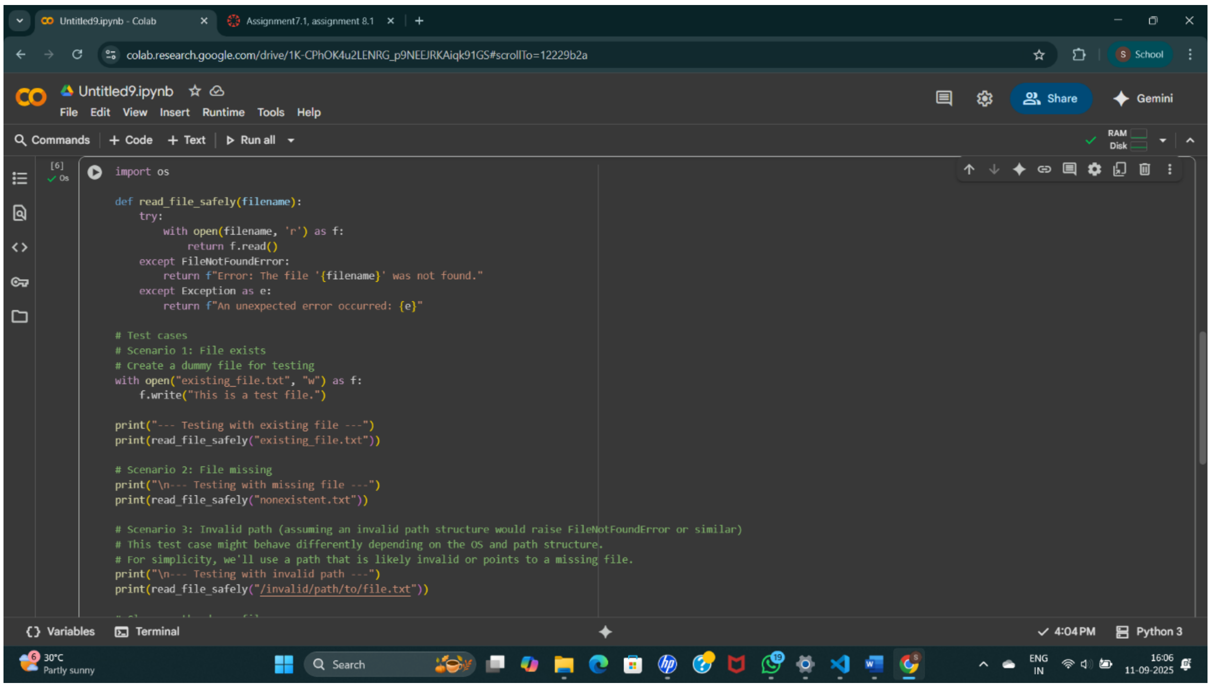
PROMPT:

Prompt:

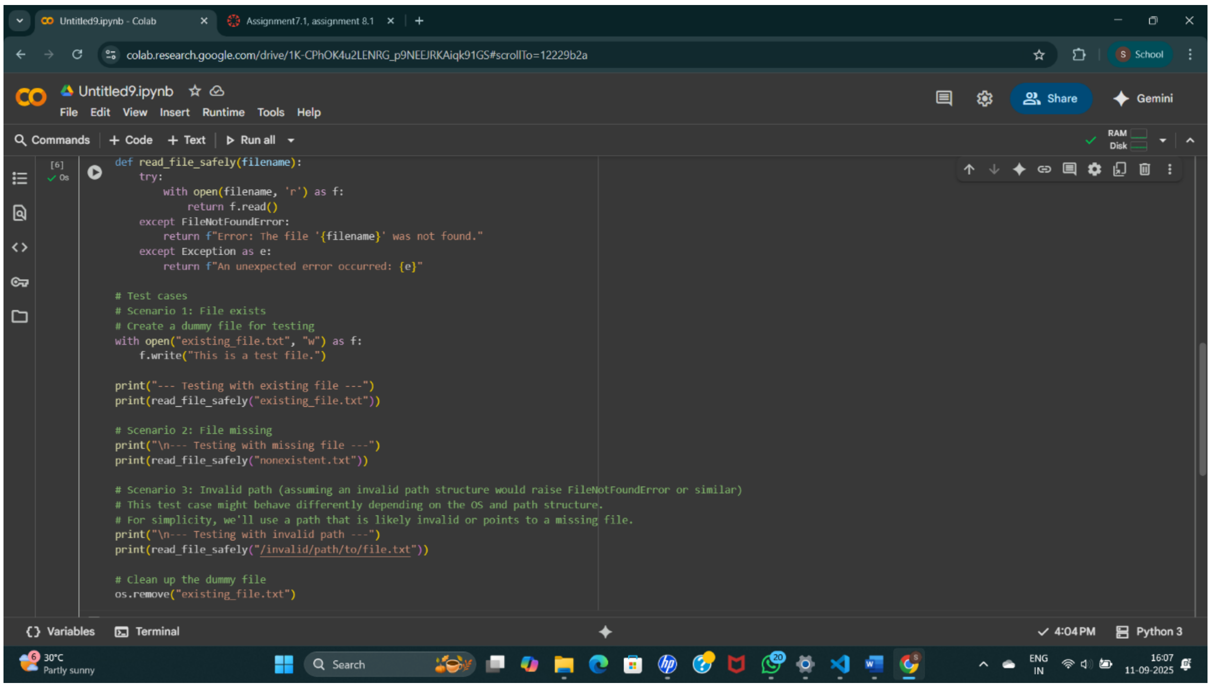
The following Python code attempts to call a method that does not exist in the class, resulting in an Attribute Error.

1. Use AI to identify and explain the error.
2. Correct the code by either defining the missing method or correcting the method call.
3. Add at least 3 assert test cases to confirm the corrected class works as expected.
4. Provide the corrected code, test cases, and a brief explanation of the fix.

CODE:



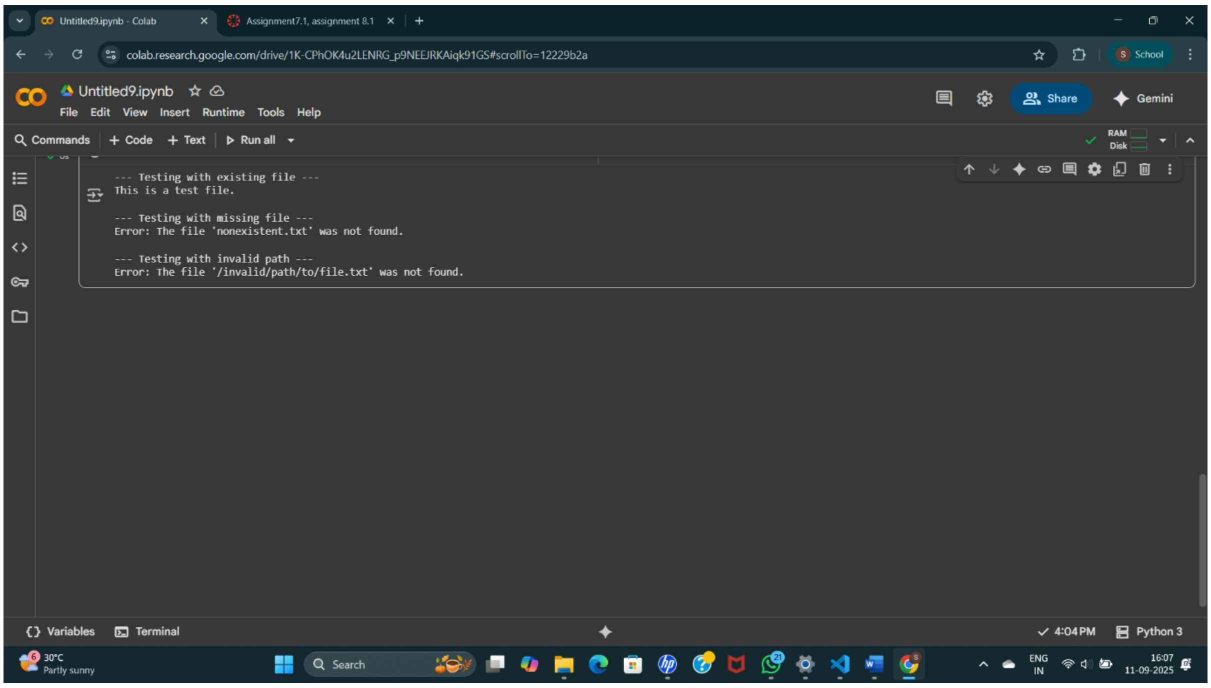
```python
import os

def read_file_safely(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return f"Error: The file '{filename}' was not found."
    except Exception as e:
        return f"An unexpected error occurred: {e}"

# Test cases
# Scenario 1: File exists
# Create a dummy file for testing
with open("existing_file.txt", "w") as f:
    f.write("This is a test file.")

print("--- Testing with existing file ---")
print(read_file_safely("existing_file.txt"))

# Scenario 2: File missing
print("\n--- Testing with missing file ---")
print(read_file_safely("nonexistent.txt"))

# Scenario 3: Invalid path (assuming an invalid path structure would raise FileNotFoundError or similar)
# This test case might behave differently depending on the OS and path structure.
# For simplicity, we'll use a path that is likely invalid or points to a missing file.
print("\n--- Testing with invalid path ---")
print(read_file_safely("/invalid/path/to/file.txt"))
```



```python
def read_file_safely(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return f"Error: The file '{filename}' was not found."
    except Exception as e:
        return f"An unexpected error occurred: {e}"

# Test cases
# Scenario 1: File exists
# Create a dummy file for testing
with open("existing_file.txt", "w") as f:
    f.write("This is a test file.")

print("--- Testing with existing file ---")
print(read_file_safely("existing_file.txt"))

# Scenario 2: File missing
print("\n--- Testing with missing file ---")
print(read_file_safely("nonexistent.txt"))

# Scenario 3: Invalid path (assuming an invalid path structure would raise FileNotFoundError or similar)
# This test case might behave differently depending on the OS and path structure.
# For simplicity, we'll use a path that is likely invalid or points to a missing file.
print("\n--- Testing with invalid path ---")
print(read_file_safely("/invalid/path/to/file.txt"))
```

```python
# Clean up the dummy file
os.remove("existing_file.txt")
```

OUTPUT:



```
--- Testing with existing file ---
This is a test file.

--- Testing with missing file ---
Error: The file 'nonexistent.txt' was not found.

--- Testing with invalid path ---
Error: The file '/invalid/path/to/file.txt' was not found.
```

Task Description #4 (AttributeError – Calling a Non-Existent Method)
Task: Give a class where a non-existent method is called (e.g., obj.undefined_method()). Use AI to debug and fix.
# Bug: Calling an undefined method
class Car:
def start(self):
return "Car started"
my_car = Car()
print(my_car.drive()) # drive() is not defined
Requirements:
• Students must analyze whether to define the missing method or correct the method call.
• Use 3 assert tests to confirm the corrected class works.
Expected Output #4:
• Corrected class with clear AI explanation.

PROMPT:

The following Python code attempts to call a method that does not exist in the class, resulting in an Attribute Error.

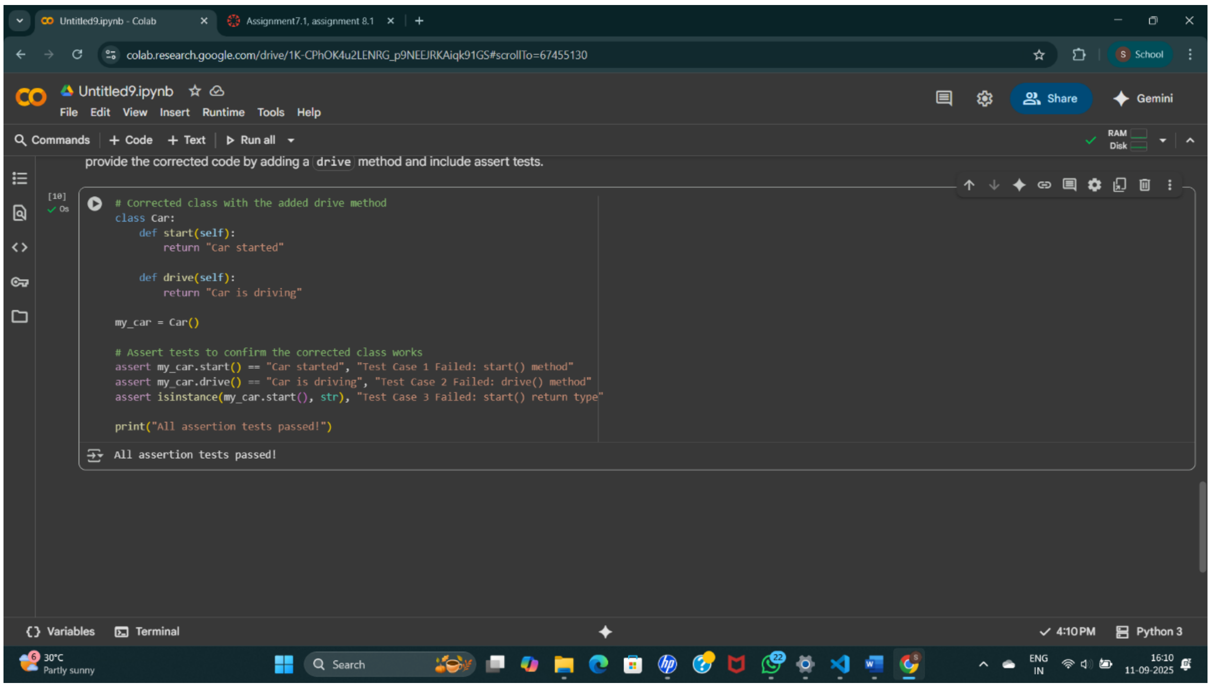
1. Use AI to identify and explain the error.

2. Correct the code by either defining the missing method or correcting the method call.

3. Add at least 3 assert test cases to confirm the corrected class works as expected.

4. Provide the corrected code, test cases, and a brief explanation of the fix

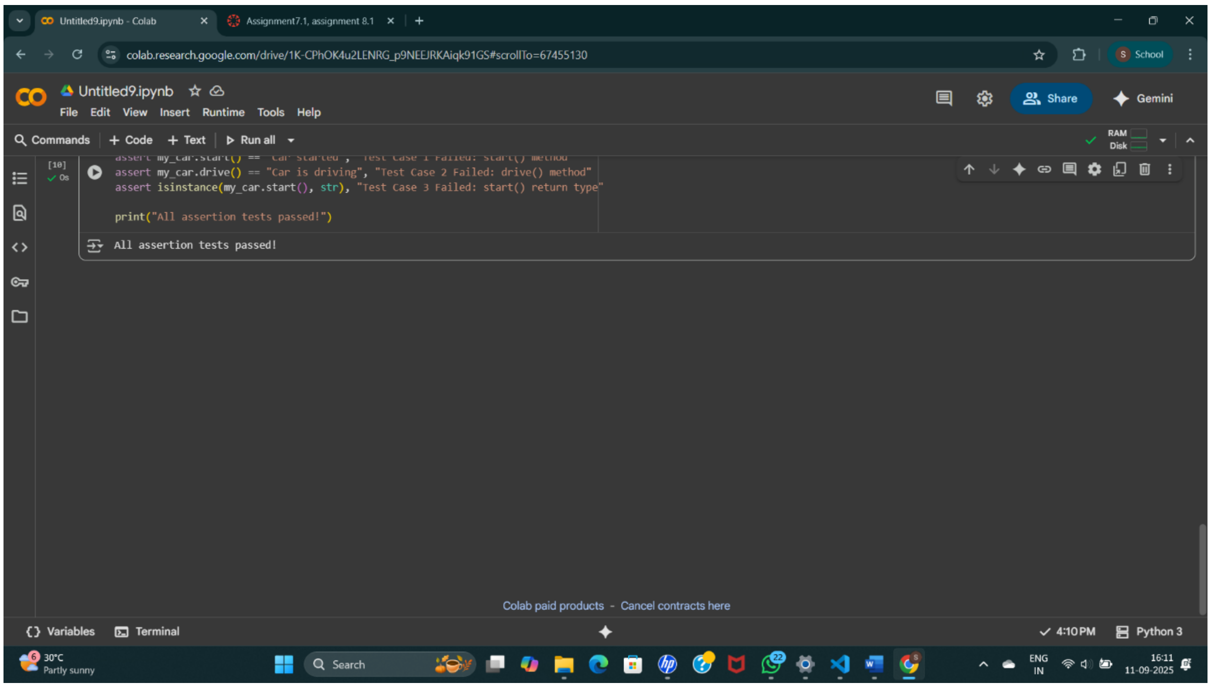
CODE:


provide the corrected code by adding a `drive` method and include assert tests.

```python
# Corrected class with the added drive method
class Car:
    def start(self):
        return "Car started"

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

my_car = Car()

# Assert tests to confirm the corrected class works
assert my_car.start() == "Car started", "Test Case 1 Failed: start() method"
assert my_car.drive() == "Car is driving", "Test Case 2 Failed: drive() method"
assert isinstance(my_car.start(), str), "Test Case 3 Failed: start() return type"

print("All assertion tests passed!")
```

```
All assertion tests passed!
```

OUTPUT:



```python
assert my_car.drive() == "Car is driving", "Test Case 2 Failed: drive() method"
assert isinstance(my_car.start(), str), "Test Case 3 Failed: start() return type"

print("All assertion tests passed!")
```

```
All assertion tests passed!
```

Task Description #5 (TypeError – Mixing Strings and Integers in Addition)
Task: Provide code that adds an integer and string ("5" + 2) causing a TypeError. Use AI to resolve the bug.

```
# Bug: TypeError due to mixing string and integer
def add_five(value):
return value + 5
print(add_five("10"))
```

Requirements:

• Ask AI for two solutions: type casting and string concatenation.

• Validate with 3 assert test cases.
Expected Output #5:
• Corrected code that runs successfully for multiple inputs.
Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

PROMPT:

Prompt:

The following Python code attempts to add a string and an integer, which causes a TypeError.

1. Use AI to identify and explain the error.

2. Provide two solutions:

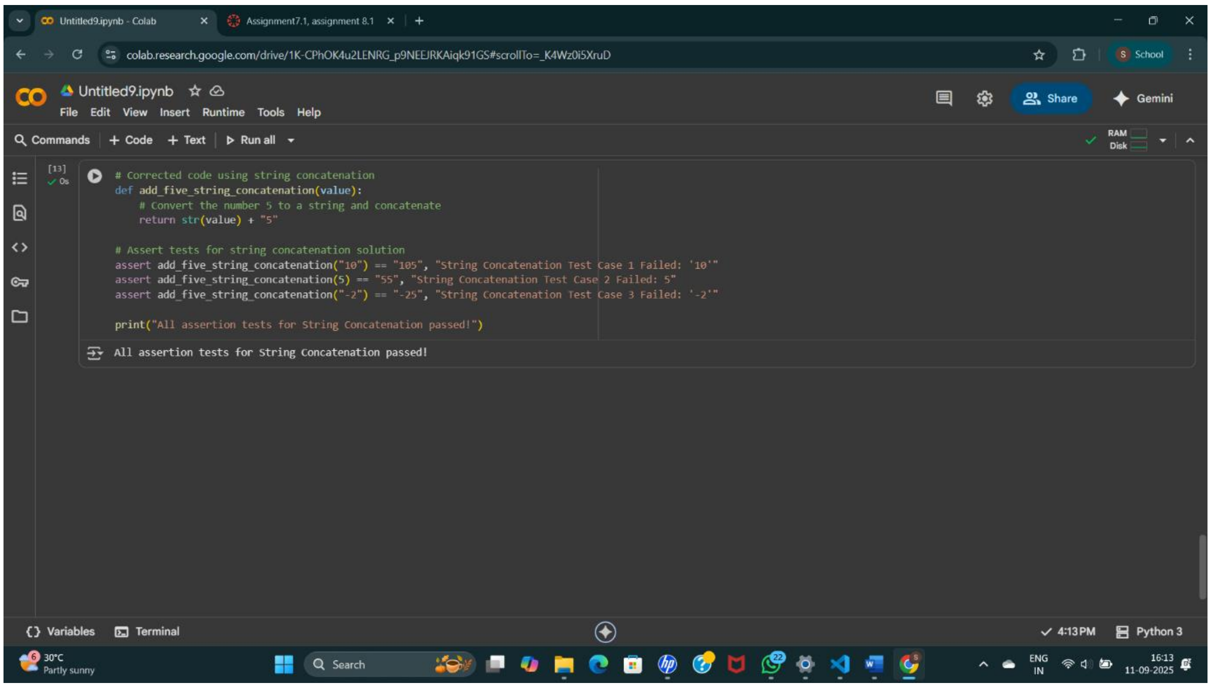
o One using type casting (convert the string to an integer before addition).

o One using string concatenation (convert the integer to a string before concatenation).

3. Add at least 3 assert test cases for each solution to confirm the corrected code works for multiple inputs.

4. Provide the corrected code, test cases, and a brief explanation of each fix.

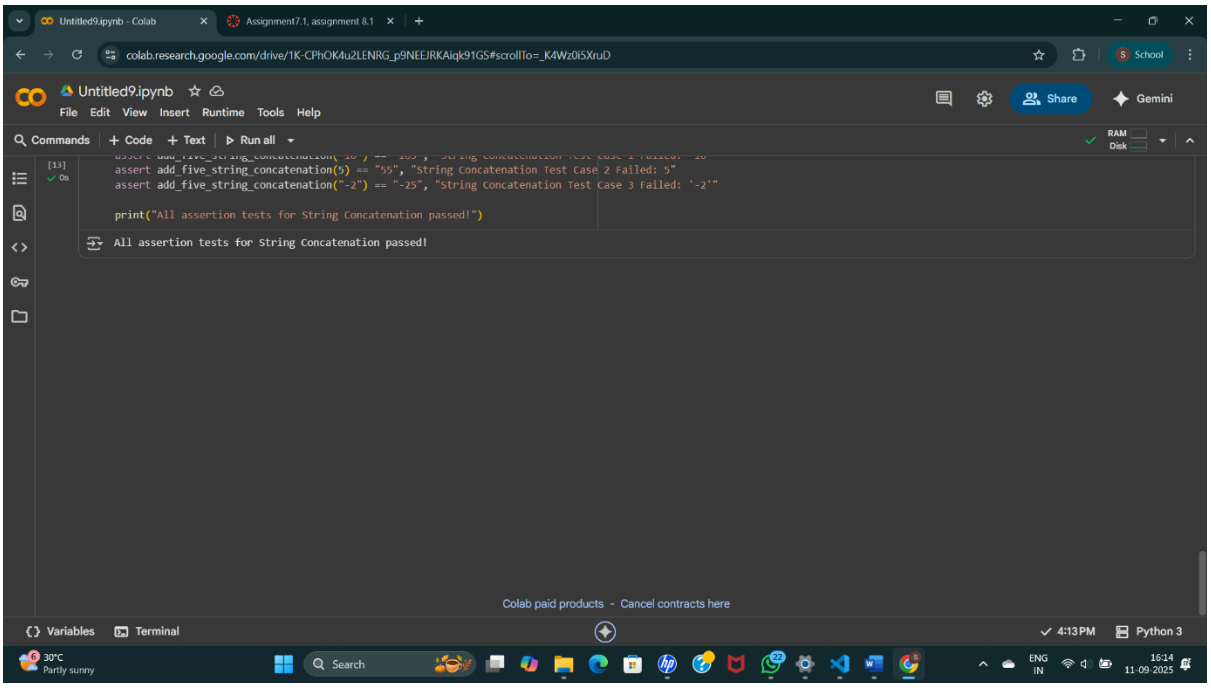
CODE:



```python
# Corrected code using string concatenation
def add_five_string_concatenation(value):
    # Convert the number 5 to a string and concatenate
    return str(value) + "5"

# Assert tests for string concatenation solution
assert add_five_string_concatenation("10") == "105", "String Concatenation Test Case 1 Failed: '10'"
assert add_five_string_concatenation(5) == "55", "String Concatenation Test Case 2 Failed: 5"
assert add_five_string_concatenation("-2") == "-25", "String Concatenation Test Case 3 Failed: '-2'"

print("All assertion tests for String Concatenation passed!")
```

```
All assertion tests for String Concatenation passed!
```

OUTPUT:



```python
assert add_five_string_concatenation(5) == "55", "String Concatenation Test Case 2 Failed: 5"
assert add_five_string_concatenation("-2") == "-25", "String Concatenation Test Case 3 Failed: '-2'"

print("All assertion tests for String Concatenation passed!")
```

```
All assertion tests for String Concatenation passed!
```