School of Computer Science and Artificial Intelligence

LabAssignment#7.1

Course Title : AI Assistant Coding

Name of Student :A.Navaneeth
Enrollment No. : 2303A54056
BatchNo. : 47-b

Lab 7: Error Debugging with AI: Systematic approaches tofinding and fixing bugs

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()
```

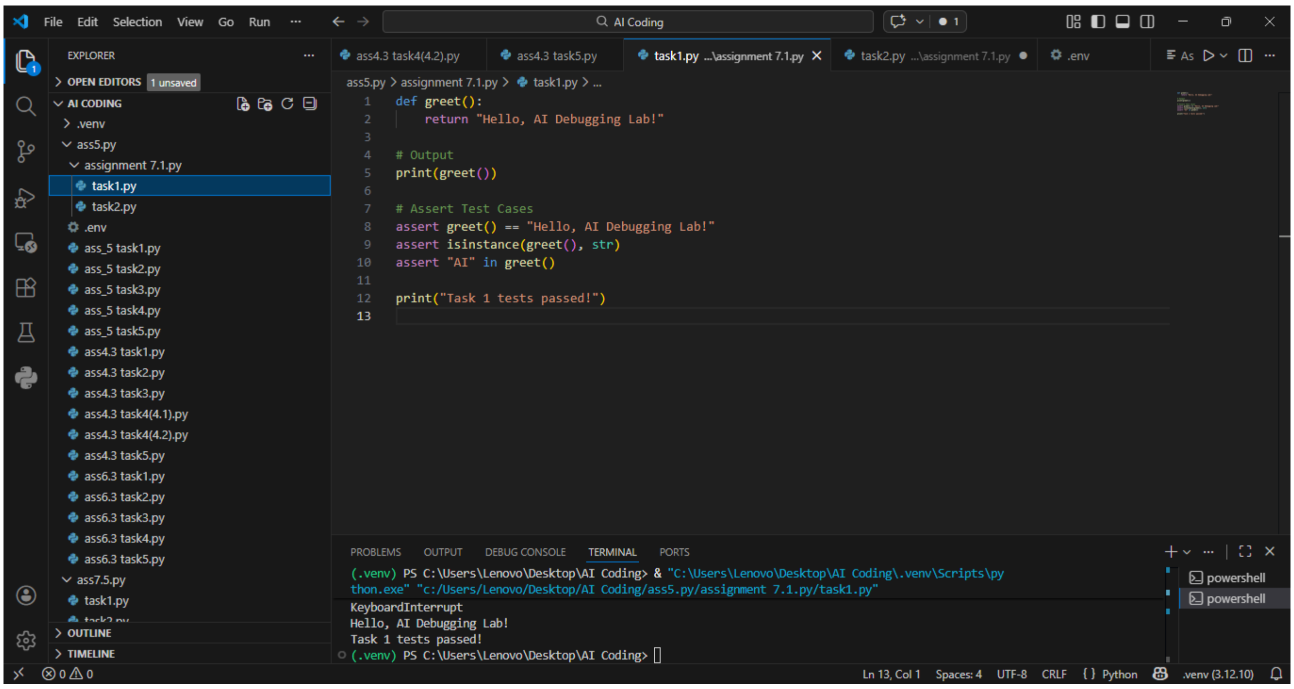
Requirements:

- 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.

Output Screenshot:



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

# Output
print(greet())

# Assert Test Cases
assert greet() == "Hello, AI Debugging Lab!"
assert isinstance(greet(), str)
assert "AI" in greet()

print("Task 1 tests passed!")
```

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\py
thon.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task1.py"
KeyboardInterrupt
Hello, AI Debugging Lab!
Task 1 tests passed!
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

Explanation: In Python 3, `print` is defined as a built-in function. A function call requires parentheses. Using the Python 2 print statement format violates Python 3 syntax rules, resulting in a `SyntaxError`. The correction is to use the functional form of `print()`.

Task Description #2 (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"
```

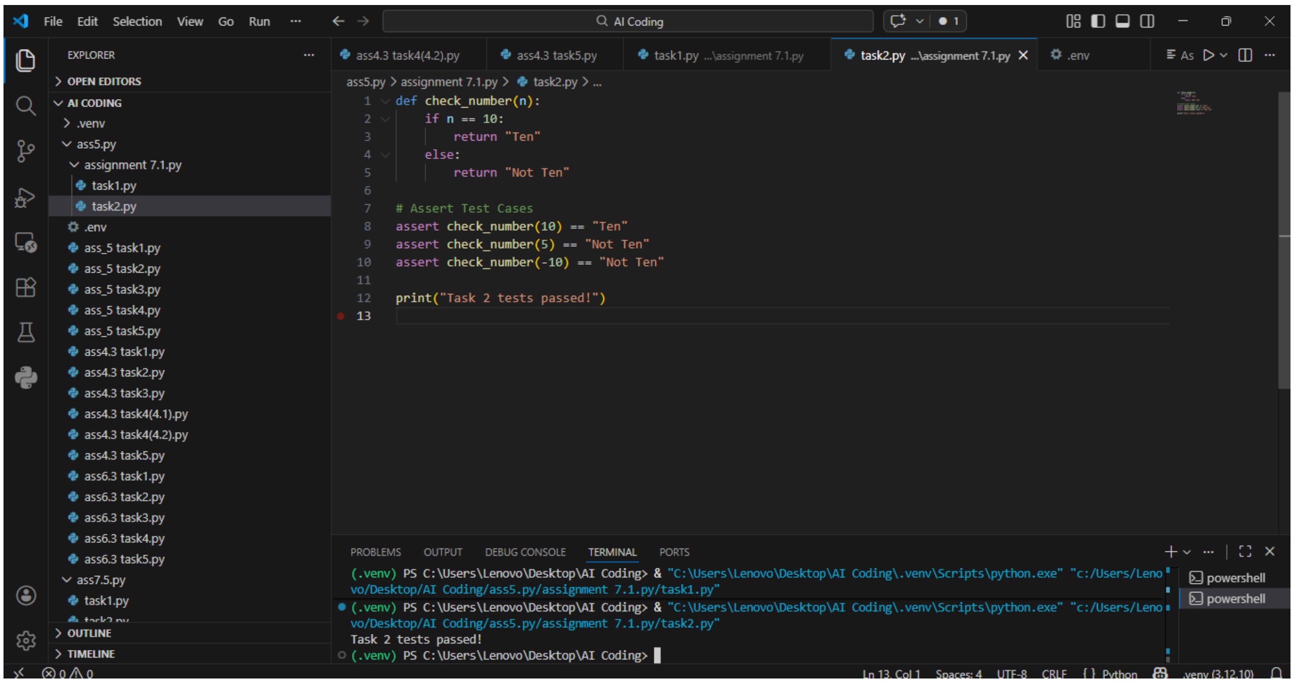
Requirements:

- 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 testexecution.

Output Screenshot:



```
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(-10) == "Not Ten"

print("Task 2 tests passed!")
```

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task1.py"
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task2.py"
Task 2 tests passed!
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

Explanation:

The operator = is an assignment operator used to store a value in a variable. Conditional statements require a boolean expression, which is formed using comparison operators such as ==. Using = in an if-condition is syntactically invalid in Python and produces a `SyntaxError`. The correction is to replace = with ==.

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"))

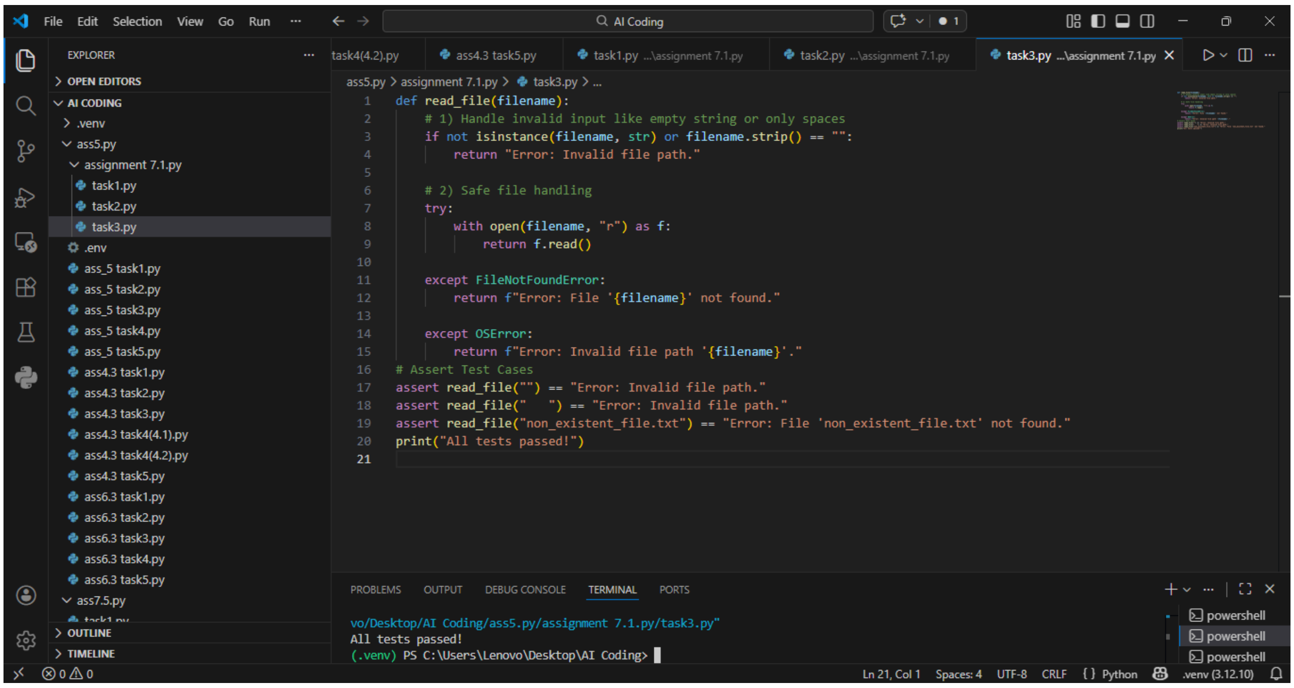
Requirements:

- Implement a try-except block suggested by AI.
- Add a user-friendly error message.
- Test with at least 3 scenarios: file exists, file missing, invalidpath.

Expected Output #3:

- Safe file handling with exception management.

Output Screenshot:



```python
def read_file(filename):
    # 1) Handle invalid input like empty string or only spaces
    if not isinstance(filename, str) or filename.strip() == "":
        return "Error: Invalid file path."

    # 2) Safe file handling
    try:
        with open(filename, "r") as f:
            return f.read()

    except FileNotFoundError:
        return f"Error: File '{filename}' not found."

    except OSError:
        return f"Error: Invalid file path '{filename}'."
# Assert Test Cases
assert read_file("") == "Error: Invalid file path."
assert read_file("   ") == "Error: Invalid file path."
assert read_file("non_existent_file.txt") == "Error: File 'non_existent_file.txt' not found."
print("All tests passed!")
```

```
vo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task3.py"
All tests passed!
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

EXPLANATION : File operations depend on the existence and validity of the file path. When `open()` is executed with a missing file, Python raises a `File Not Found Error` at runtime. Exception handling using `try-except` prevents abrupt termination and enables controlled execution by returning a meaningful error message.

Task Description #4 (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

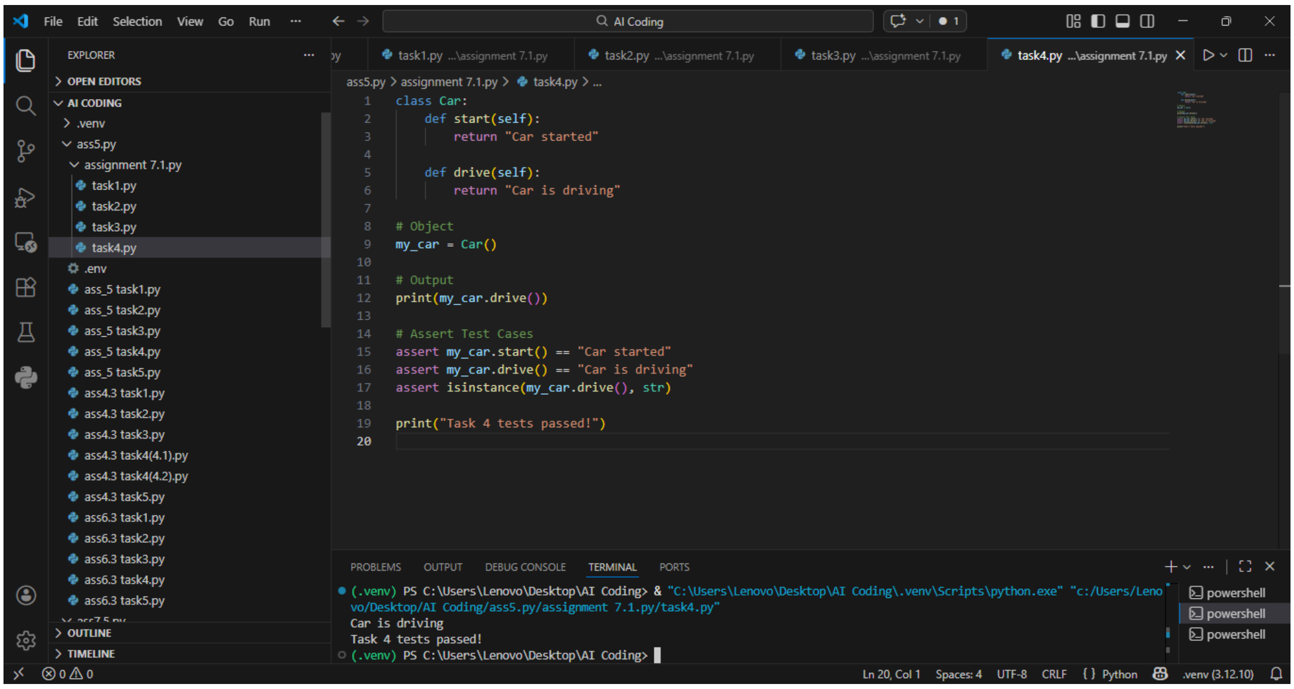
Requirements:

- 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.

Output Screenshot:



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

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

# Object
my_car = Car()

# Output
print(my_car.drive())

# Assert Test Cases
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car.drive(), str)

print("Task 4 tests passed!")
```

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task4.py"
Car is driving
Task 4 tests passed!
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

Explanation: In object-oriented programming, a method must be defined within a class before it can be invoked by an object of that class. Calling an undefined method results in an `AttributeError` because the object does not contain the requested attribute. The correction requires either defining the missing method in the class or modifying the call to an existing method.

Task Description #5 (TypeError – Mixing Strings and Integers inAddition)
Task: Provide code that adds an integer and string ("5" + 2) causinga 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:

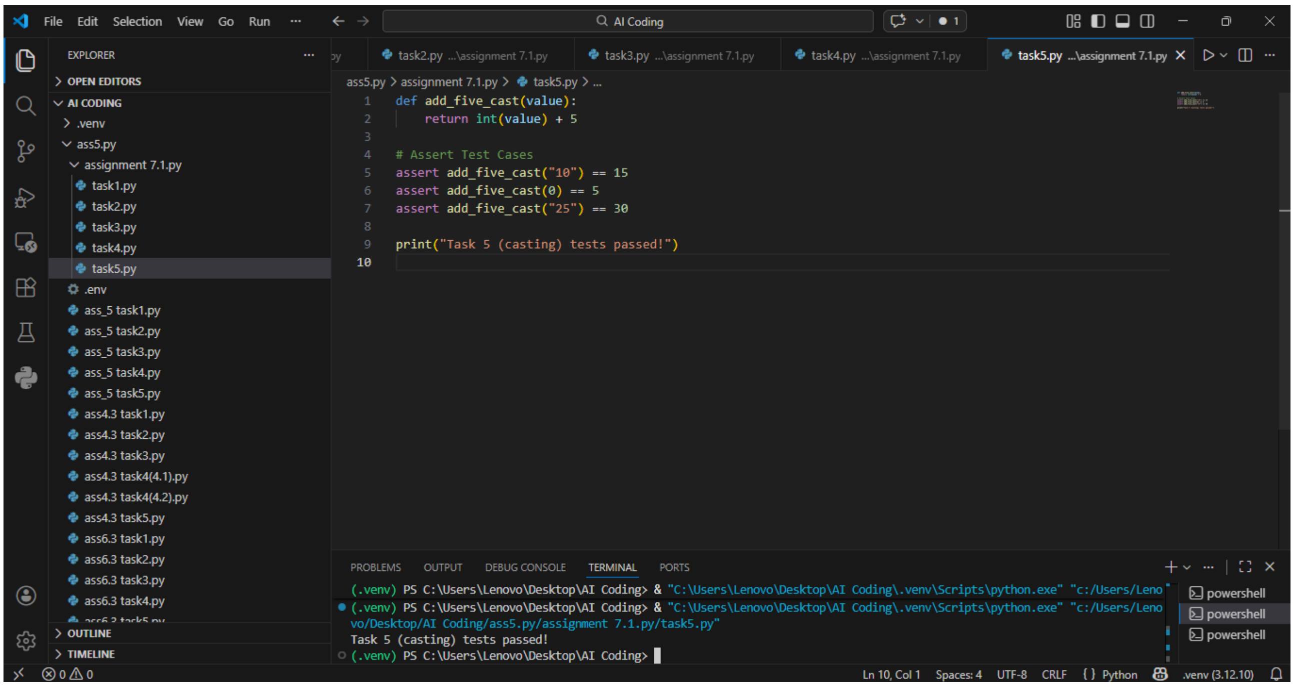
• Ask AI for two solutions: type casting and string concatenation.

• Validate with 3 assert test cases.

Expected Output #5:

• Corrected code that runs successfully for multiple inputs.

Output Screenshot:



```python
def add_five_cast(value):
    return int(value) + 5

# Assert Test Cases
assert add_five_cast("10") == 15
assert add_five_cast(0) == 5
assert add_five_cast("25") == 30

print("Task 5 (casting) tests passed!")
```

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Leno
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Leno
vo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task5.py"
Task 5 (casting) tests passed!
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

EXPLANATION : Python enforces strict type rules for arithmetic operations. Addition between a string and an integer is not supported because the operands are of incompatible types. This produces a `TypeError`. The correction is performed by explicit type conversion, either converting the string to an integer for numeric addition or converting the integer to a string for concatenation.