School of Computer Science and Artificial Intelligence

Lab Assignment-7.1

Course Title : AI Assistant Coding
Name of Student : Shaik Naved Ahmed
Enrollment No. : 2303A54053
Batch No. : 47-B

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.

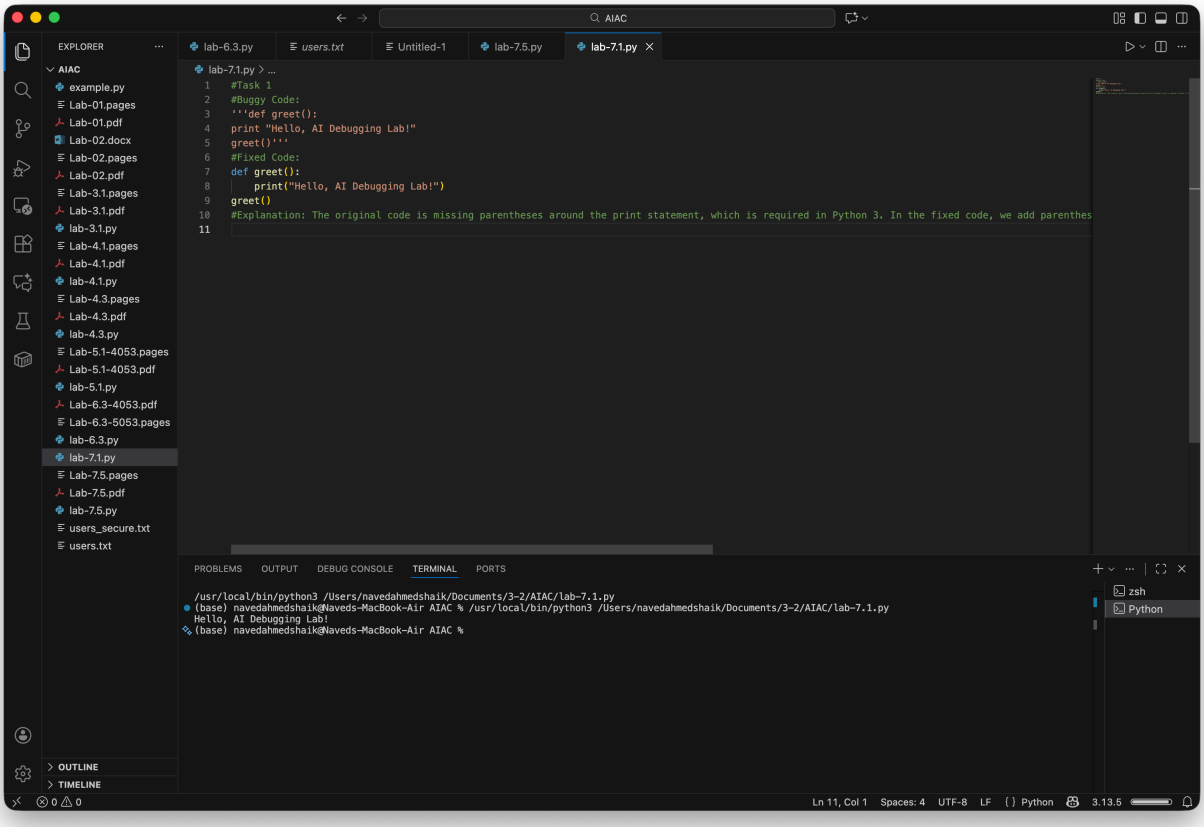
Expected Output #1:

- Corrected code with proper syntax and AI explanation.

EXPLANATION:

The original code is missing parentheses around the print statement, which is required in Python 3. In the fixed code, we add parentheses around the string in the print function, allowing the code to execute correctly and print the greeting message.

SCREENSHOT OF GENERATED CODE:



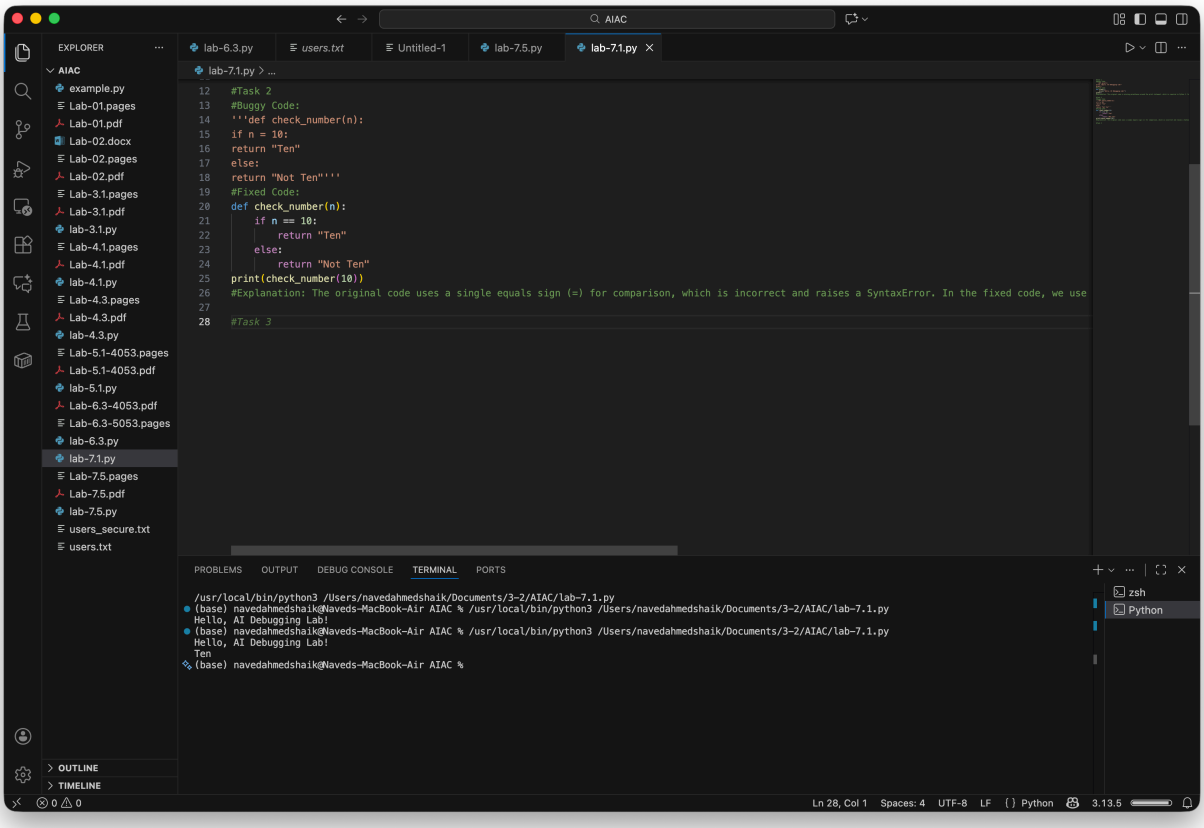
Task Description #2 (Incorrect condition in an If Statement)

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 test execution.

EXPLANATION:

The original code uses a single equals sign (=) for comparison, which is incorrect and raises a SyntaxError. In the fixed code, we use a double equals sign (==) to compare the value of n with 10, allowing the function to return "Ten" when n is 10 and "Not Ten" otherwise.

SCREENSHOT OF GENERATED CODE:



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

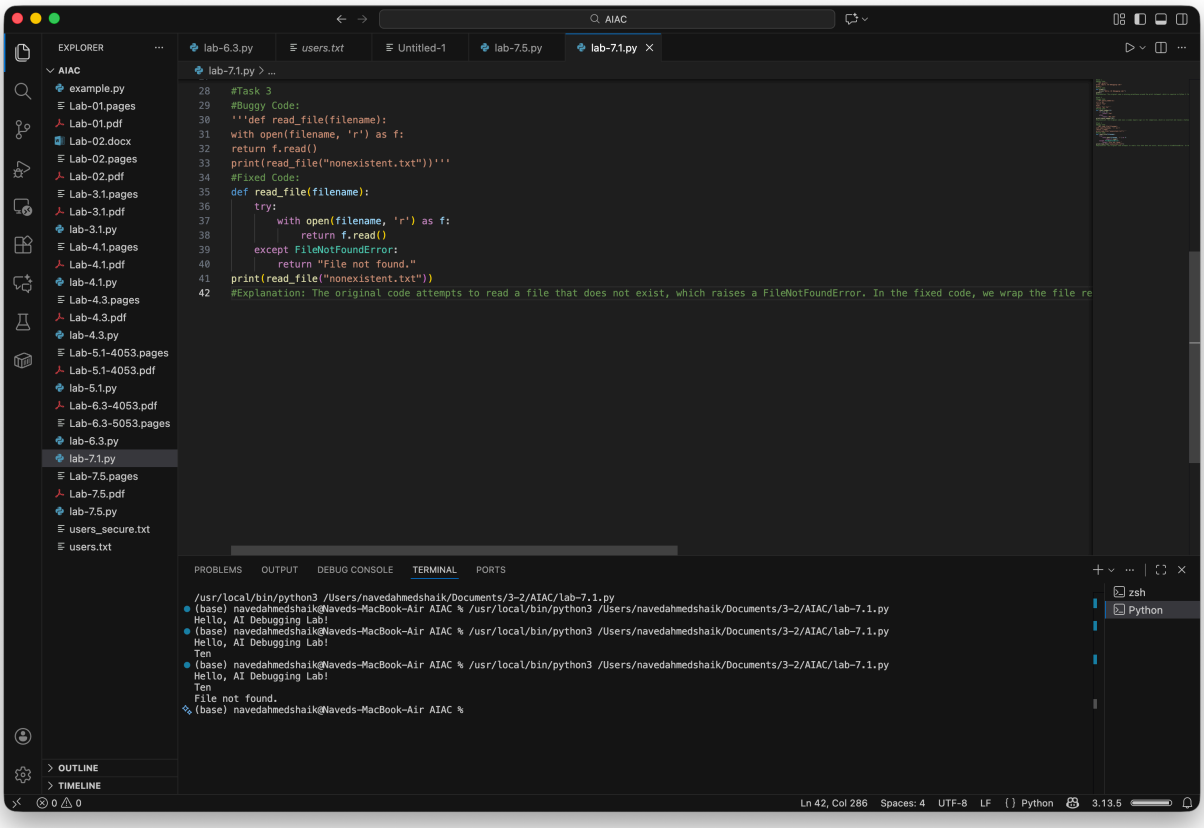
Task: Provide code that attempts to open a non-existent file and
crashes. Use AI to apply safe error handling.
# Bug: Program crashes if file is missing
def read_file(filename):
with open(filename, 'r') as f:
return f.read()
print(read_file("nonexistent.txt"))
Requirements:
• Implement a try-except block 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.

EXPLANATION:

The original code attempts to read a file that does not exist, which raises a FileNotFoundError. In the fixed code, we wrap the file reading operation in a try-except block to catch the FileNotFoundError and return a user-friendly message instead of crashing the program.

SCREENSHOT OF GENERATED CODE:



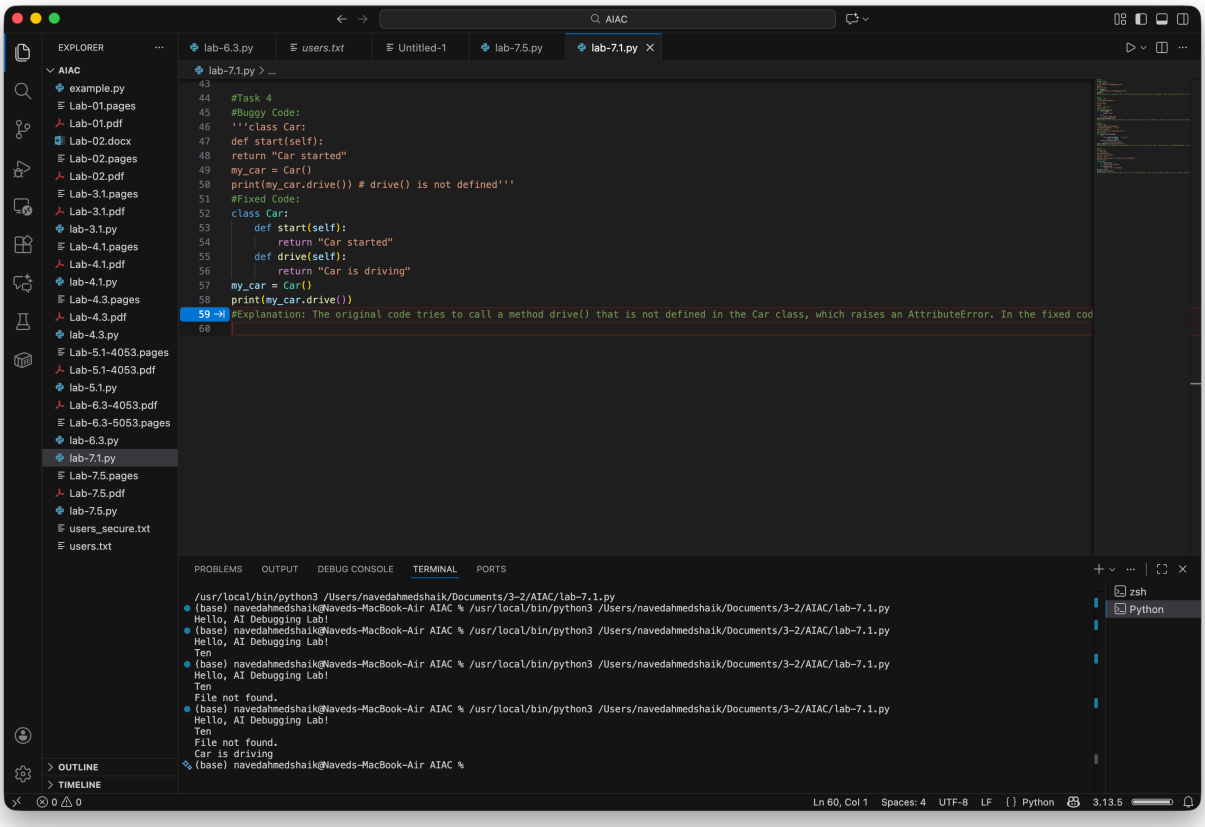
Task Description #4 (Calling a Non-Existent Method)

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.

EXPLANATION:

The original code tries to call a method drive() that is not defined in the Car class, which raises an AttributeError. In the fixed code, we define the drive() method within the Car class, allowing us to call it without any errors and return a message indicating that the car is driving.

SCREENSHOT OF GENERATED CODE:



Task Description #5 (TypeError – Mixing Strings and Integers inAddition)

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

EXPLANATION:

The original code attempts to add 5 to a string "10", which raises a TypeError. In the fixed code, we convert the input value to an integer using int() before performing the addition. We also include a try-except block to catch any ValueError that may occur if the input cannot be converted to an integer, providing a user-friendly error message instead.

SCREENSHOT OF GENERATED CODE: