

AI ASSISTED CODING

Assignment-7.1

N. Madhuvani

2303A52117

Batch-40

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.

Code and Output:

The image displays two screenshots of a Visual Studio Code (VS Code) editor interface, showing the development and debugging of a Python script.

Top Screenshot:

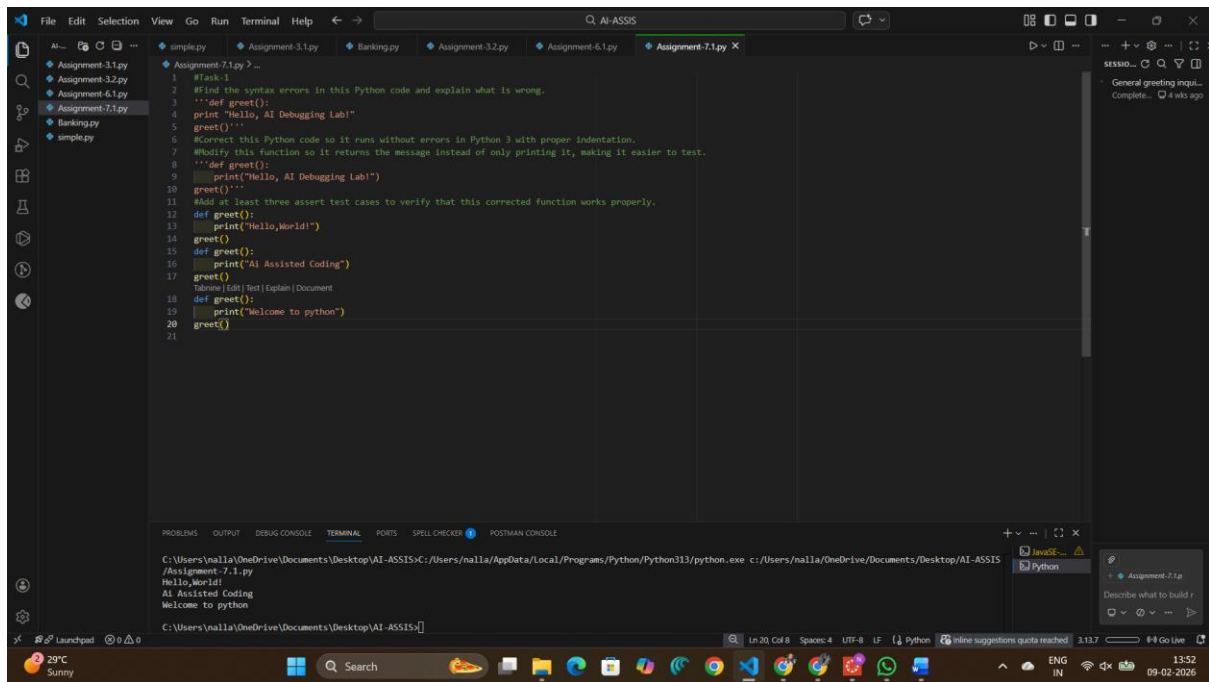
- The editor shows a file named `Assignment-7.1.py` with the following code:

```
1 #Task-1
2 #Find the syntax errors in this Python code and explain what is wrong.
3 '''def greet():
4     print "Hello, AI Debugging Lab!"
5     greet()'''
6 #Correct this Python code so it runs without errors in Python 3 with proper indentation.
7 #Modify this function so it returns the message instead of only printing it, making it easier to test.
8 def greet():
9     print("Hello, AI Debugging Lab!")
10 greet()
11
```
- The **PROBLEMS** panel at the bottom shows an **IndentationError**: `expected an indented block after function definition on line 3`.
- The **TERMINAL** panel shows the command `C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS` and the output `Hello, AI Debugging Lab!`.

Bottom Screenshot:

- The editor shows the same file `Assignment-7.1.py` with the following code:

```
1 #Task-1
2 #Find the syntax errors in this Python code and explain what is wrong.
3 tabnine | Edit | Test | Explain | Document
4 def greet():
5     print "Hello, AI Debugging Lab!"
6     greet()
7
```
- The **PROBLEMS** panel shows the same **IndentationError**.
- The **TERMINAL** panel shows the command `C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS` and the output `/Assignment-7.1.py`, `Hello,World!`, `AI Assisted Coding`, and `Welcome to python`.



Analysis:

1. The program has a syntax error because the print statement is missing parentheses.
2. It also has an indentation issue inside the function.
3. Python cannot execute the code due to these errors.
4. The AI suggests fixing the indentation and using print() correctly.
5. After correction, the program runs successfully without errors.

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"
```

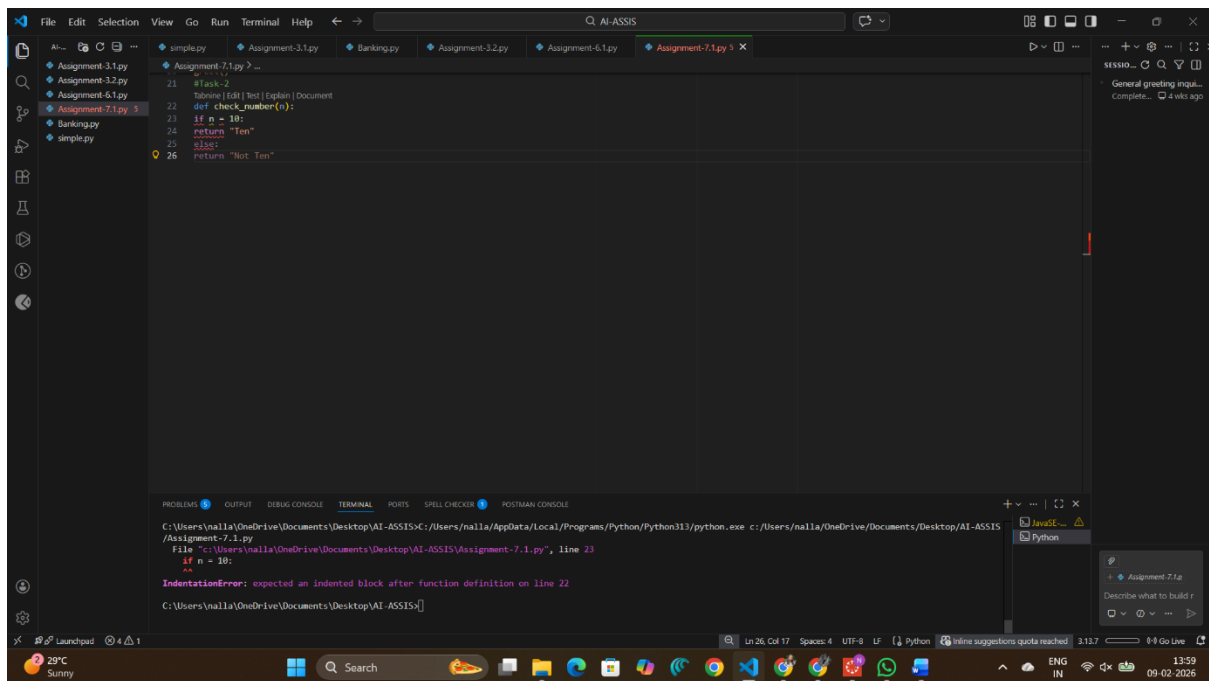
Requirements:

- 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.

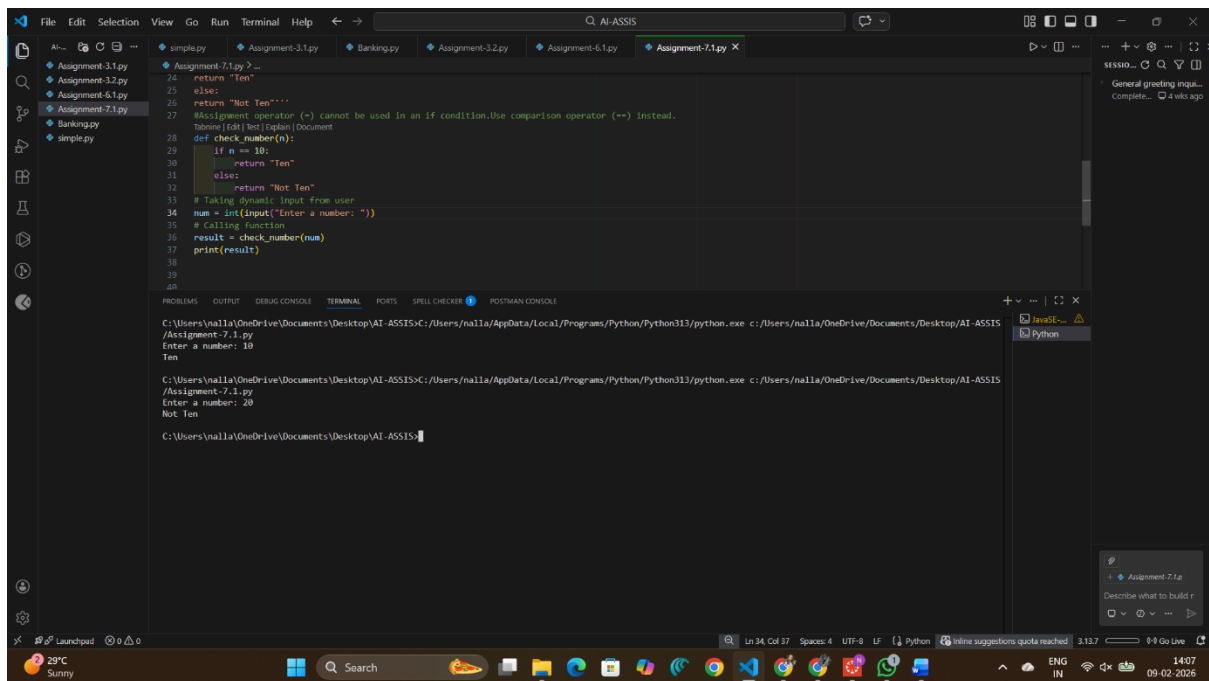
Code and Output:



The screenshot shows the Visual Studio Code editor with a Python file named `Assignment-7.1.py` open. The code contains a function `check_number(n)` with an indentation error. The error message in the terminal reads: `IndentationError: expected an indented block after function definition on line 22`. The code is as follows:

```
21 #Task-2
22 def check_number(n):
23     if n == 10:
24         return "Ten"
25     else:
26         return "Not Ten"
```

The terminal output shows the command `C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py` and the error message.

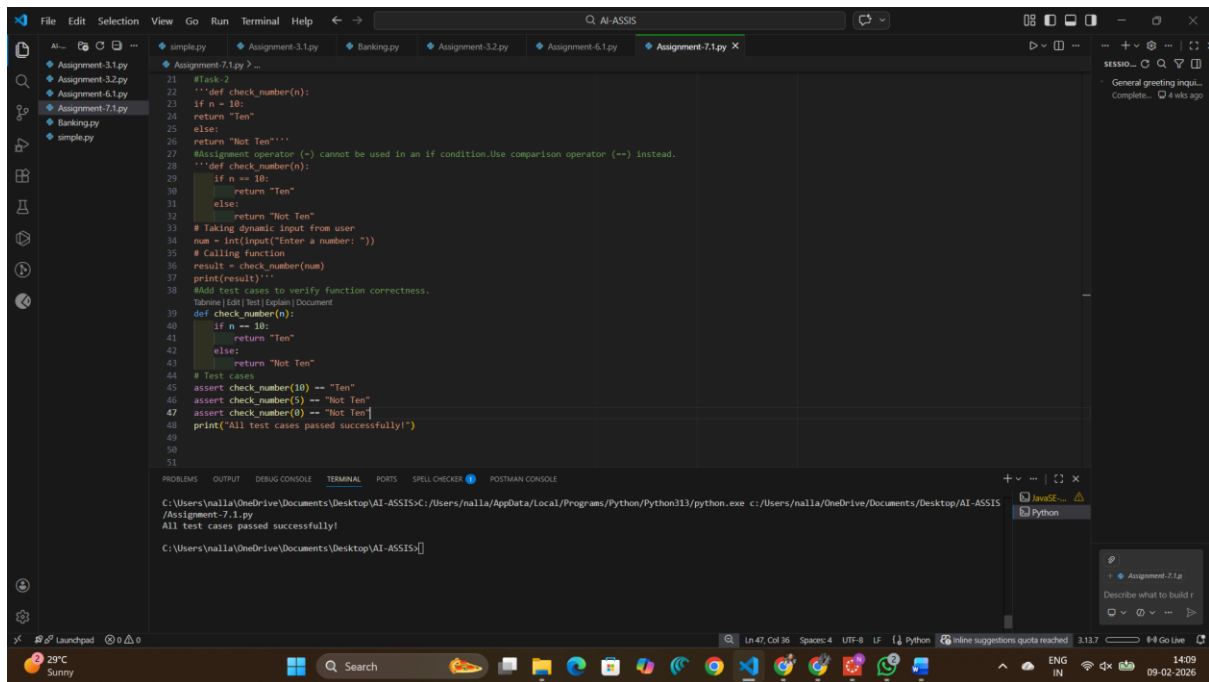


The screenshot shows the Visual Studio Code editor with the same Python file `Assignment-7.1.py` open. The code has been corrected, and the terminal shows the output of the program. The code is as follows:

```
24     return "Ten"
25 else:
26     return "Not Ten"
27 #Assignment operator (=) cannot be used in an if condition. Use comparison operator (==) instead.
28 def check_number(n):
29     if n == 10:
30         return "Ten"
31     else:
32         return "Not Ten"
33 # Taking dynamic input from user
34 num = int(input("Enter a number: "))
35 # calling function
36 result = check_number(num)
37 print(result)
```

The terminal output shows the command `C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py` and the output of the program:

```
Enter a number: 10
Ten
Enter a number: 20
Not Ten
```



Analysis:

- 1.The program had an error because = was used instead of == in the if condition.
- 2.The AI corrected it to properly compare the number.
- 3.Dynamic input allows the user to enter any value.
- 4.The function now checks the number correctly.
- 5.The program runs without errors and gives correct output.

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"))

```

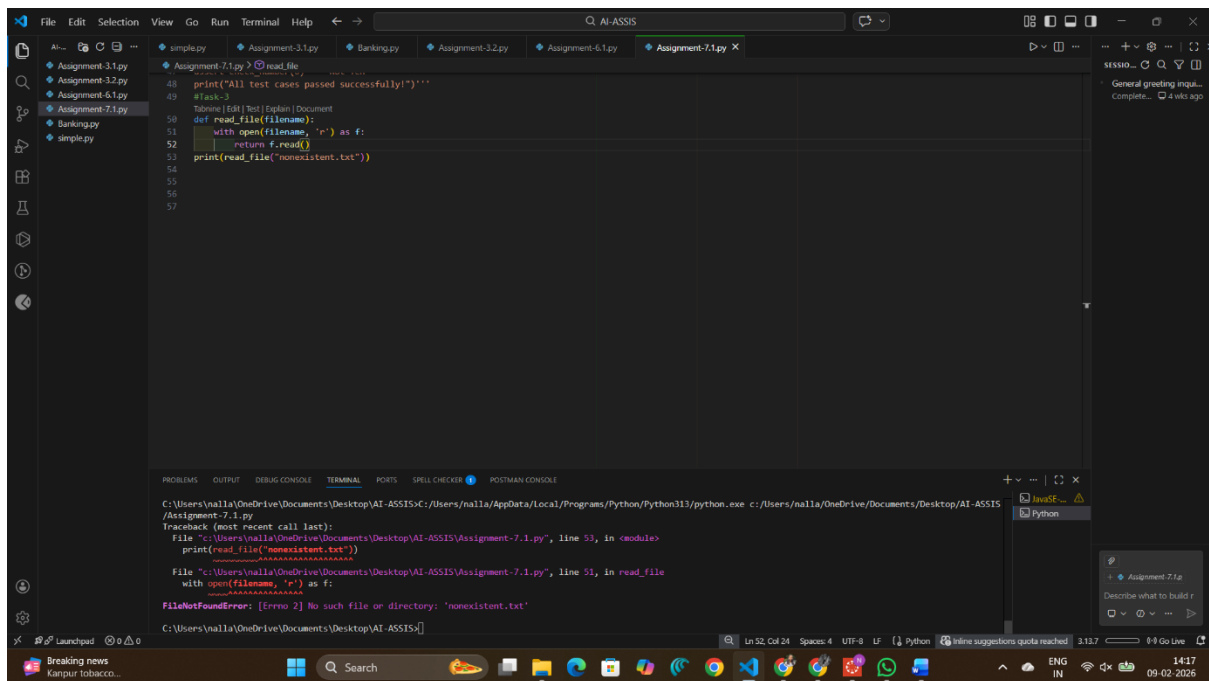
Requirements:

- 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.

Code and Output:



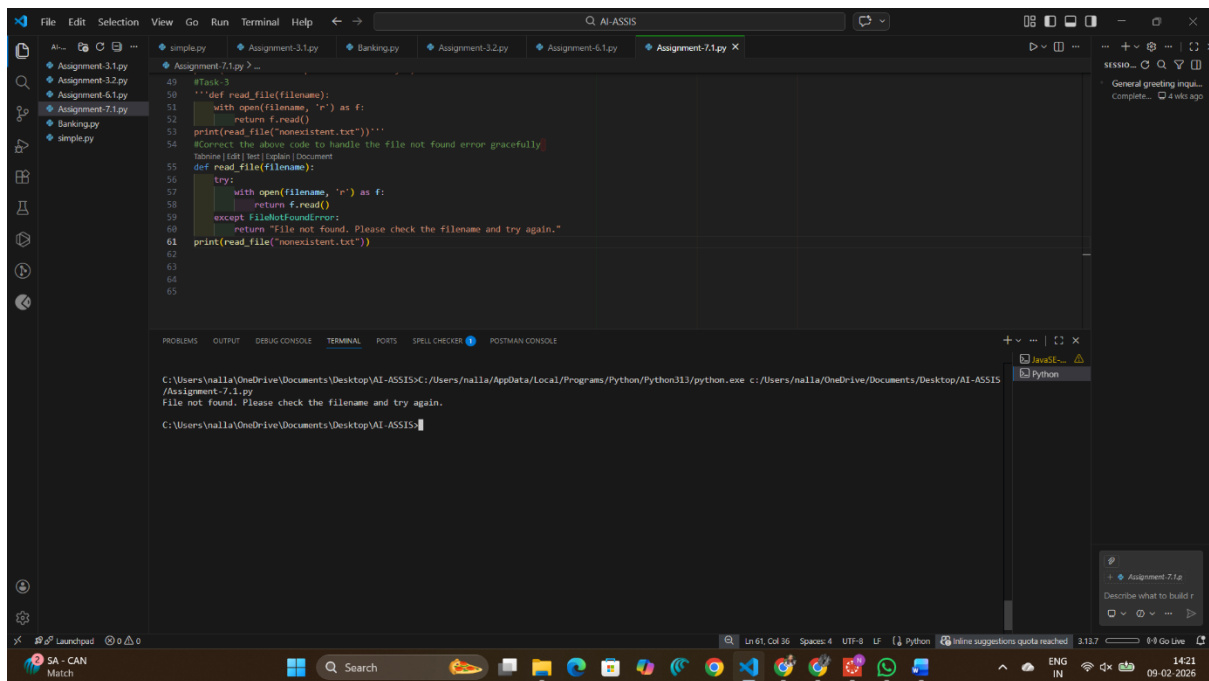
The screenshot shows the VS Code editor with a file explorer on the left containing several assignment files. The main editor displays a Python script named `Assignment-7.1.py` with the following code:

```
48 print("All test cases passed successfully!")'''
49 #Task-3
50 Tabnine | Edit | Test | Explain | Document
51 def read_file(filename):
52     with open(filename, "r") as f:
53         return f.read()
54     print(read_file("nonexistent.txt"))
55
56
57
```

The terminal at the bottom shows the command `python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py` and the output:

```
Traceback (most recent call last):
  File "c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py", line 53, in <module>
    print(read_file("nonexistent.txt"))
  File "c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py", line 51, in read_file
    with open(filename, "r") as f:
FileNotFoundError: [Errno 2] No such file or directory: 'nonexistent.txt'

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS\
```



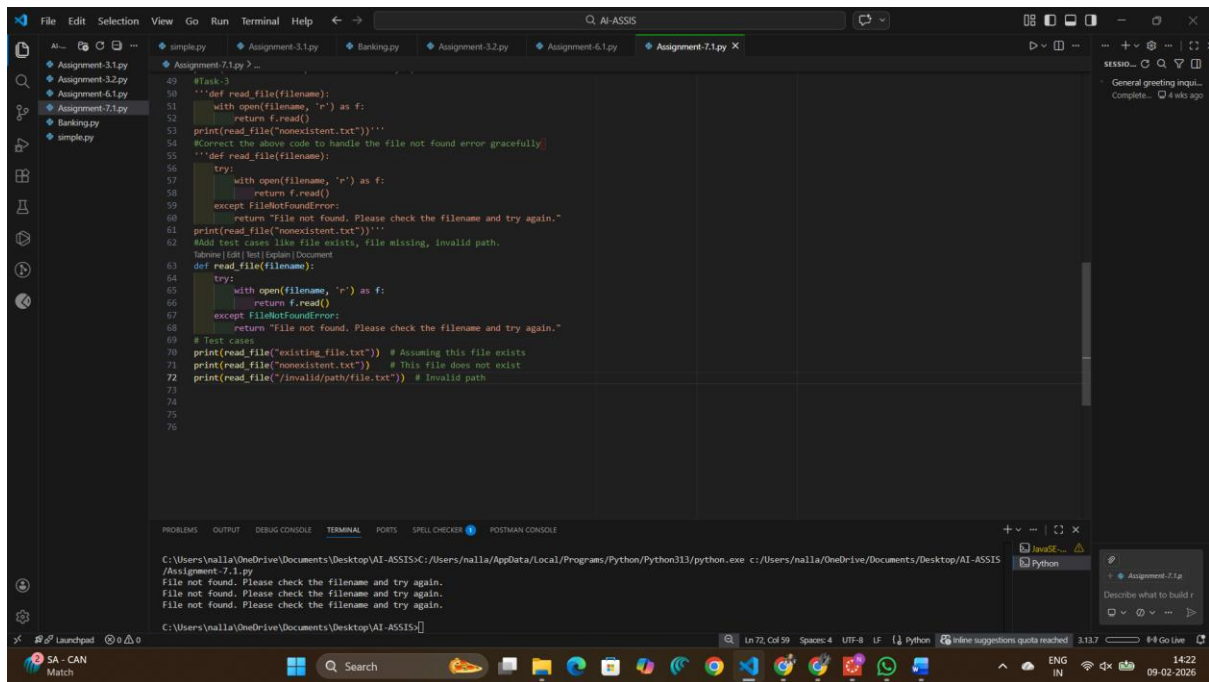
The screenshot shows the same VS Code editor with the `Assignment-7.1.py` file. The code has been updated to include a try-except block to handle the `FileNotFoundError` gracefully:

```
50 Tabnine | Edit | Test | Explain | Document
51 def read_file(filename):
52     try:
53         with open(filename, "r") as f:
54             return f.read()
55     except FileNotFoundError:
56         return "File not found. Please check the filename and try again."
57     print(read_file("nonexistent.txt"))
58
59
60
61
```

The terminal at the bottom shows the command `python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py` and the output:

```
File not found. Please check the filename and try again.

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS\
```



Analysis:

1. The program previously crashed when a file was missing.
2. AI suggested using a try-except block to handle errors safely.
3. It now shows clear messages for file exists, file missing, and invalid path cases.
4. Runtime errors are prevented.
5. The program runs smoothly in all scenarios.

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
```

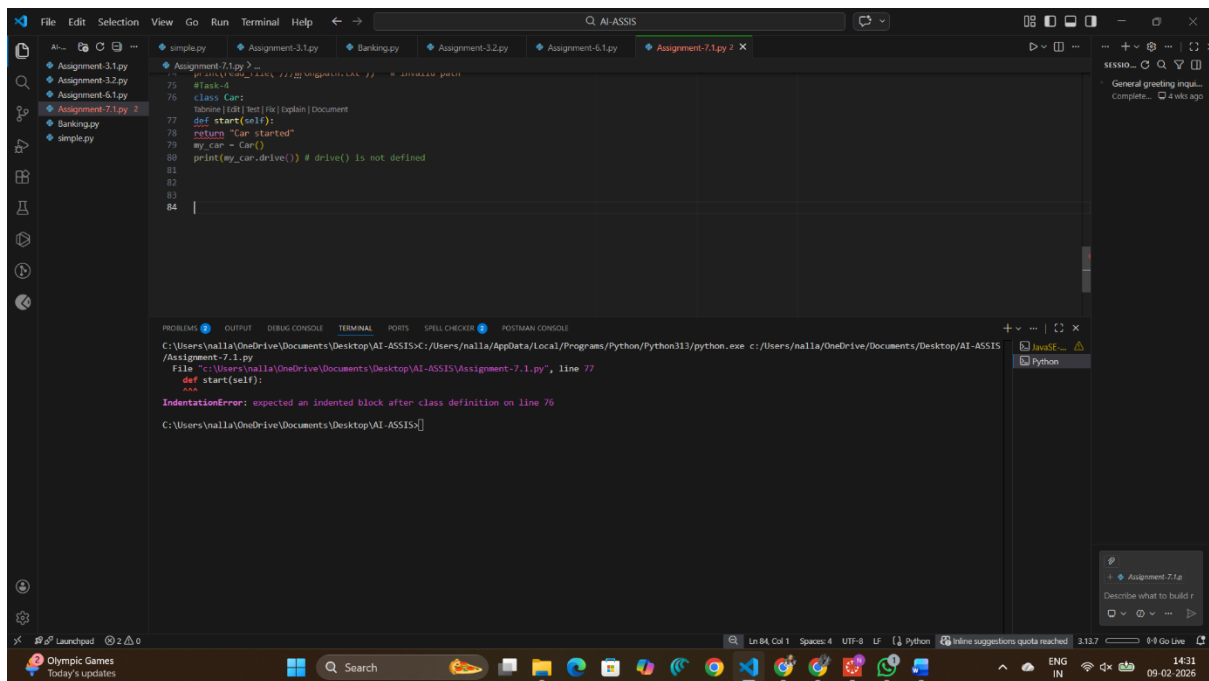
Requirements:

- 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.

Code and Output:



The screenshot shows the VS Code editor with the file `Assignment-7.1.py` open. The code defines a class `Car` with a `start` method. The terminal output shows an `IndentationError` on line 76.

```
75 #Task-4
76 class Car:
77     def start(self):
78         return "Car started"
79     my_car = Car()
80     print(my_car.drive()) # drive() is not defined
81
82
83
84
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER POSTMAN CONSOLE

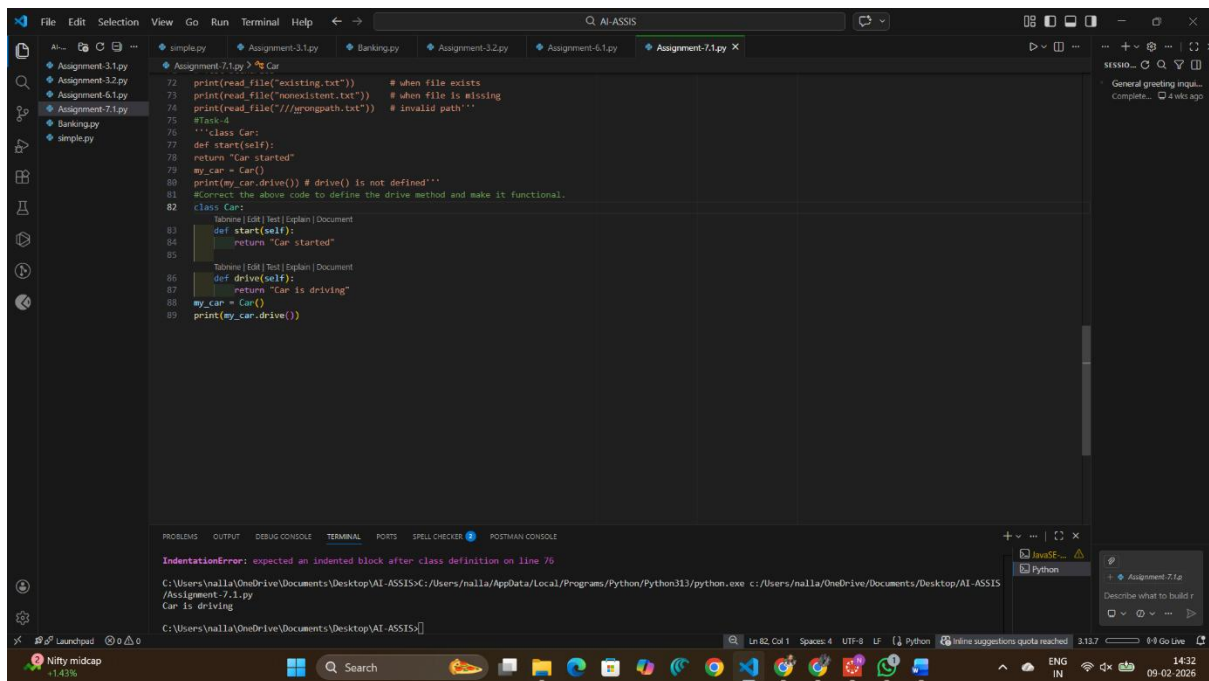
C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py

File "c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py", line 77

def start(self):

IndentationError: expected an indented block after class definition on line 76

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>



The screenshot shows the VS Code editor with the file `Assignment-7.1.py` open. The code now includes a `drive` method and a comment about correcting the code. The terminal output shows the `Car` object and the `drive` method being called.

```
72 print(read_file("existing.txt")) # when file exists
73 print(read_file("nonexistent.txt")) # when file is missing
74 print(read_file("///grrongpath.txt")) # invalid path'''
75 #Task-4
76 '''class Car:
77     def start(self):
78         return "Car started"
79     my_car = Car()
80     print(my_car.drive()) # drive() is not defined'''
81 #Correct the above code to define the drive method and make it functional.
82 class Car:
83     def start(self):
84         return "Car started"
85
86     def drive(self):
87         return "Car is driving"
88     my_car = Car()
89     print(my_car.drive())
```

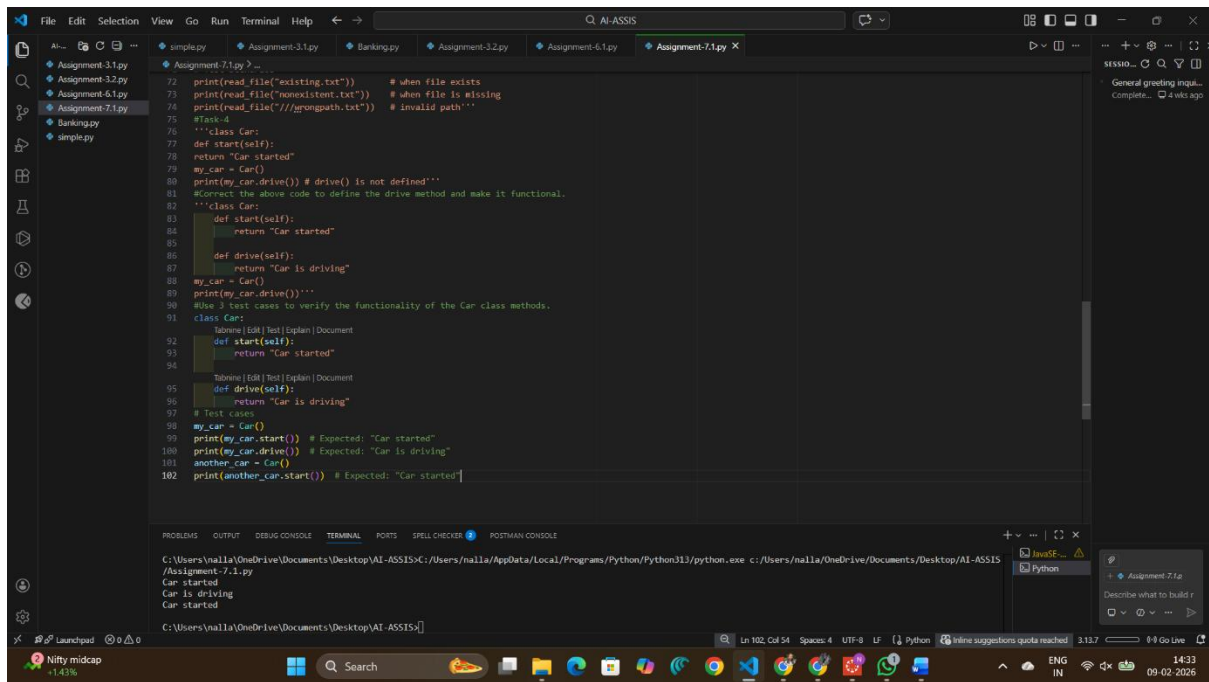
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER POSTMAN CONSOLE

IndentationError: expected an indented block after class definition on line 76

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-7.1.py

Car is driving

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>



Analysis:

- 1.The program caused an error because the drive() method was called but not defined in the class.
- 2.AI detected the missing method and suggested adding it or correcting the method call.
- 3.A new drive() method was added to fix the issue.
- 4.Assert tests were used to verify both methods work correctly.
- 5.The class now runs without errors.

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"))
```

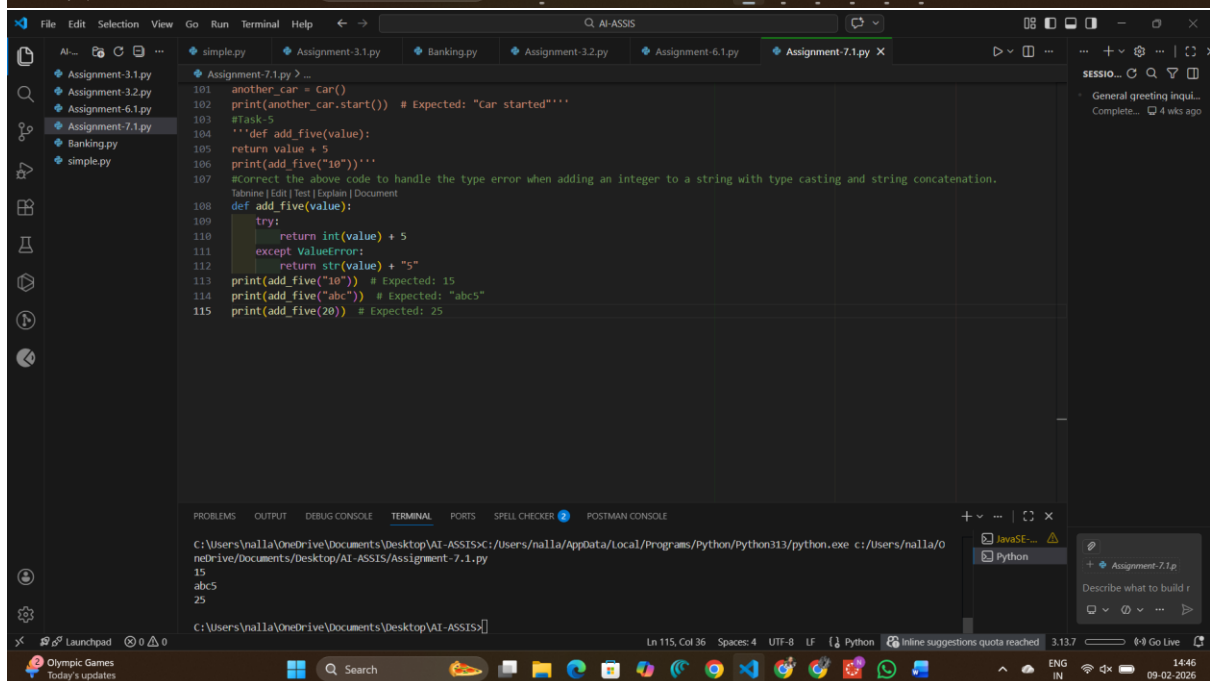
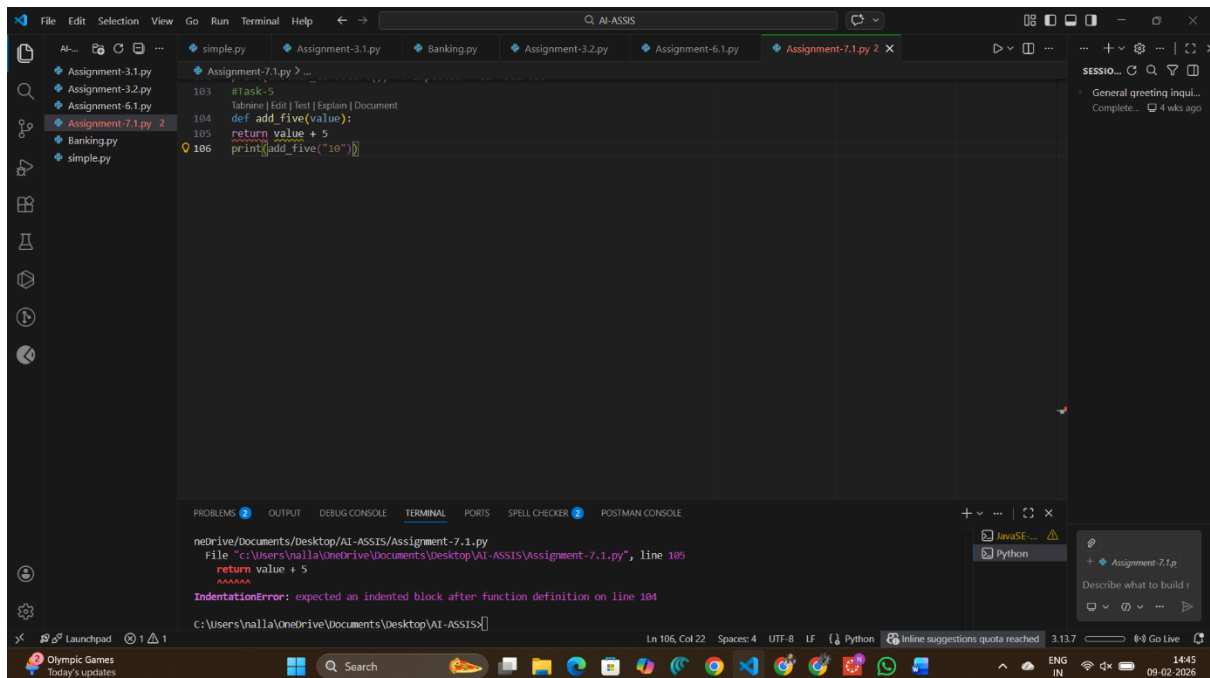
Requirements:

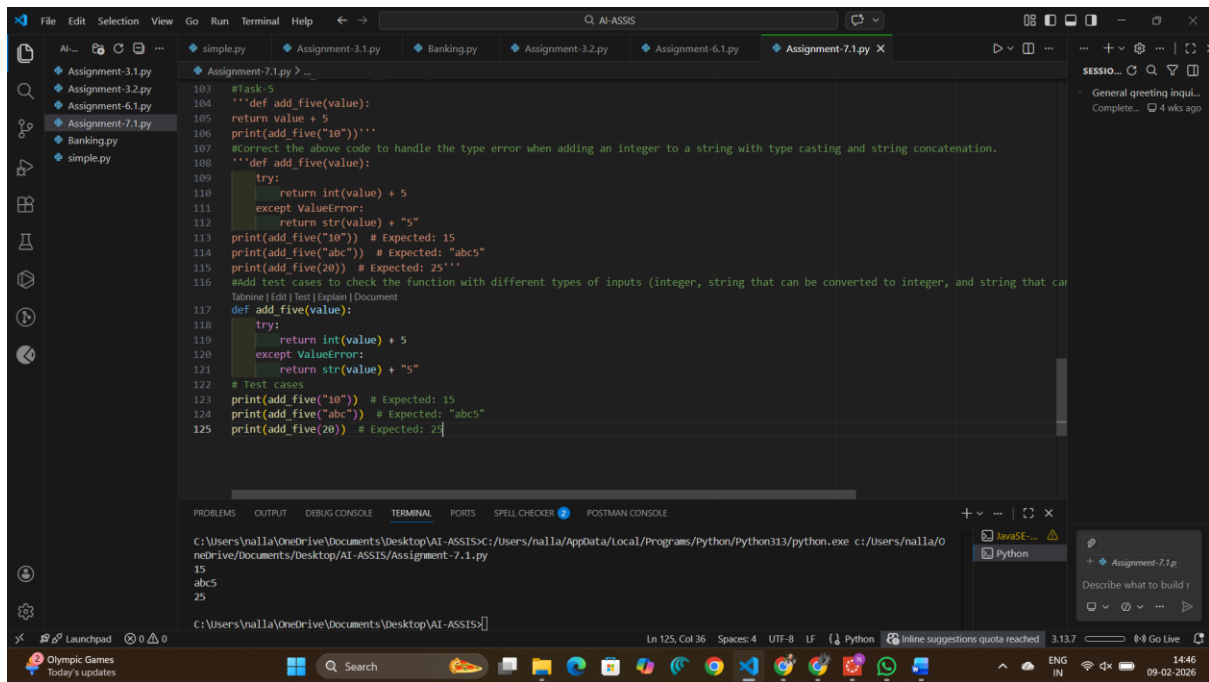
- Ask AI for two solutions: type casting and string concatenation.
- Validate with 3 assert test cases.

Expected Output #5:

- Corrected code that runs successfully for multiple inputs.

Code and Output:





Analysis:

- 1.The program produced a TypeError because it attempted to add a string and an integer together.
- 2.AI identified that Python does not allow direct addition of different data types.
- 3.Two solutions were suggested: converting the string to an integer or converting the number to a string for concatenation.
- 4.Type casting allows numerical addition, while string concatenation joins values as text.
- 5.After applying the fixes and tests, the function runs correctly for different inputs.