Assignment-7.1

Name: Arjun Manoj

H.No:2303A52134

Batch: 44

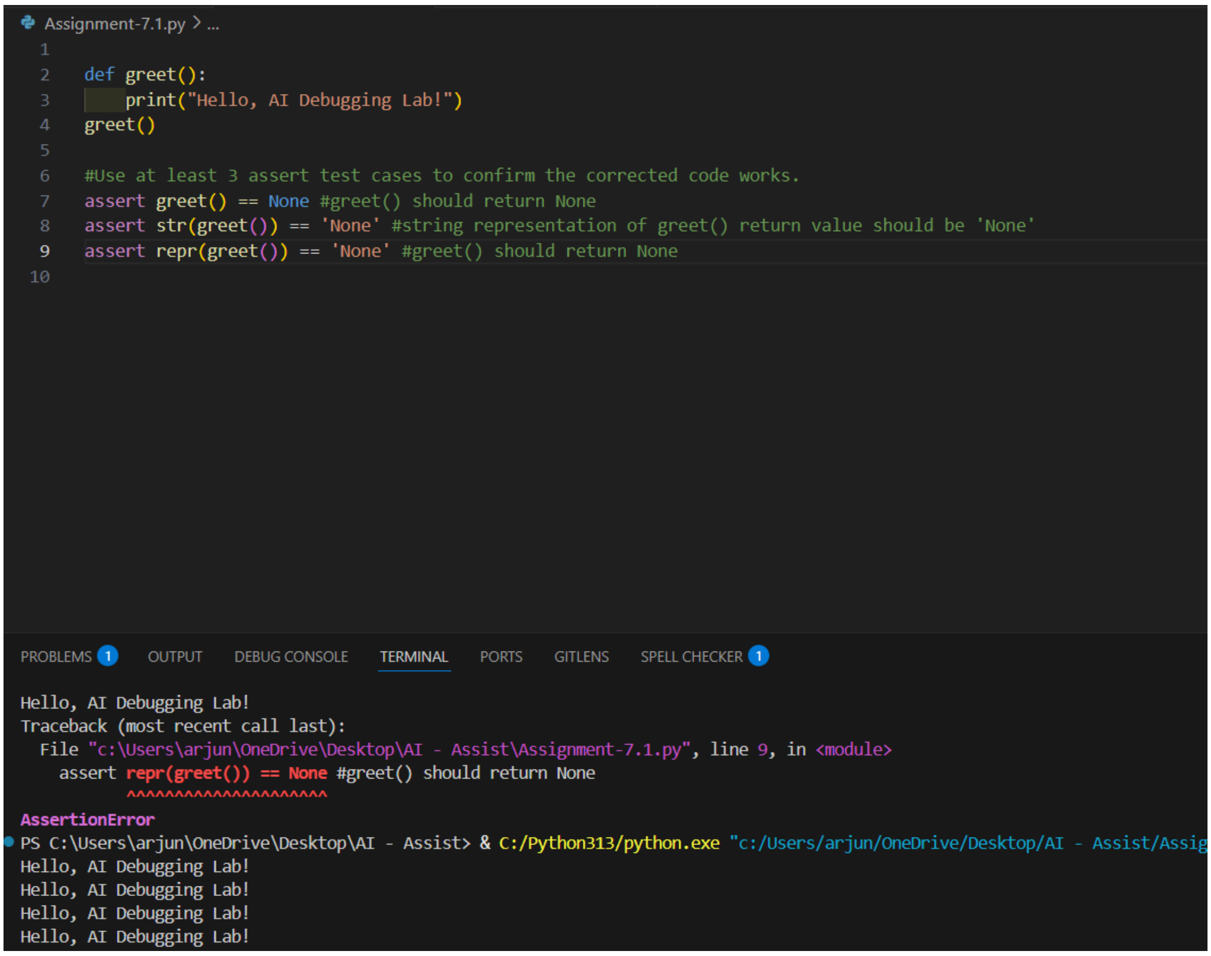
Task1: Provide a Python snippet with a missing parenthesis in a print statement (e.g., print "Hello"). Use AI to detect and fix the syntax error.

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

Prompt:

Detect and fix the syntax error in the following Python code which contains a missing parenthesis in a print statement. After correcting the code, verify it using at least three assert test cases.

Code:



Assignment-7.1.py > ...

```python

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

#Use at least 3 assert test cases to confirm the corrected code works.
assert greet() == None #greet() should return None
assert str(greet()) == 'None' #string representation of greet() return value should be 'None'
assert repr(greet()) == 'None' #greet() should return None

```

```
Hello, AI Debugging Lab!
Traceback (most recent call last):
  File "c:\Users\arjun\OneDrive\Desktop\AI - Assist\Assignment-7.1.py", line 9, in <module>
    assert repr(greet()) == None #greet() should return None
           ^^^^^^^^^^^^^^^^^^^^^
AssertionError
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Desktop/AI - Assist/Assig
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
```

Observation:

When the given code is executed, Python throws a **SyntaxError**:

SyntaxError: Missing parentheses in call to 'print'

This happens because in **Python 3**, print is a function and must be written with parentheses. The statement print "Hello, AI Debugging Lab!" follows Python 2 syntax, so the program stops execution and produces no output.

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"

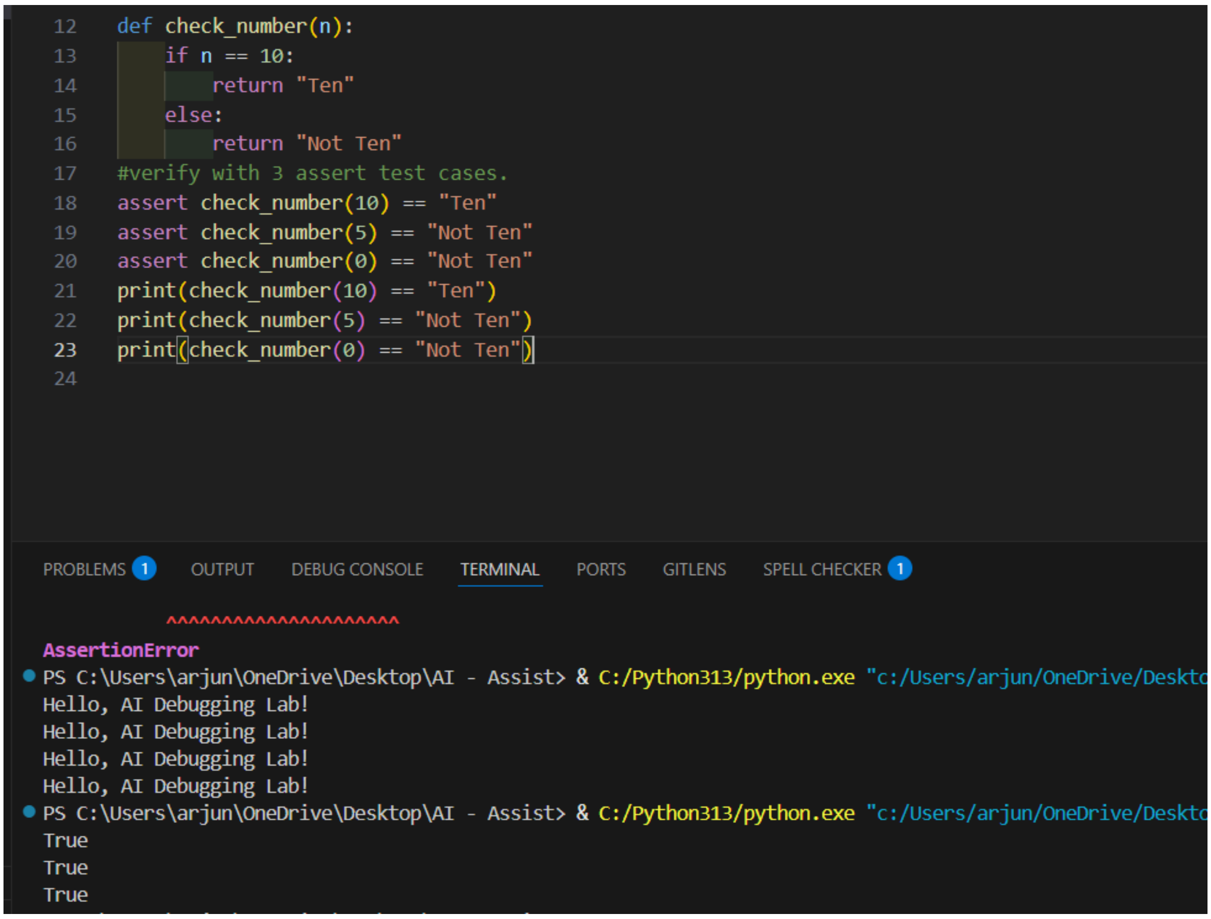
else:

return "Not Ten"

Prompt:

Identify and explain the error in the following Python function where an incorrect operator is used in the if condition. Correct the issue and explain why it causes a bug.

Code:



```python
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"
#verify with 3 assert test cases.
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(0) == "Not Ten"
print(check_number(10) == "Ten")
print(check_number(5) == "Not Ten")
print(check_number(0) == "Not Ten")
```

```
          ^^^^^^^^^^^^^^^^^^^^^
AssertionError
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Deskto
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Deskto
True
True
True
```

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

Task: Provide code that attempts to open a non-existent file and

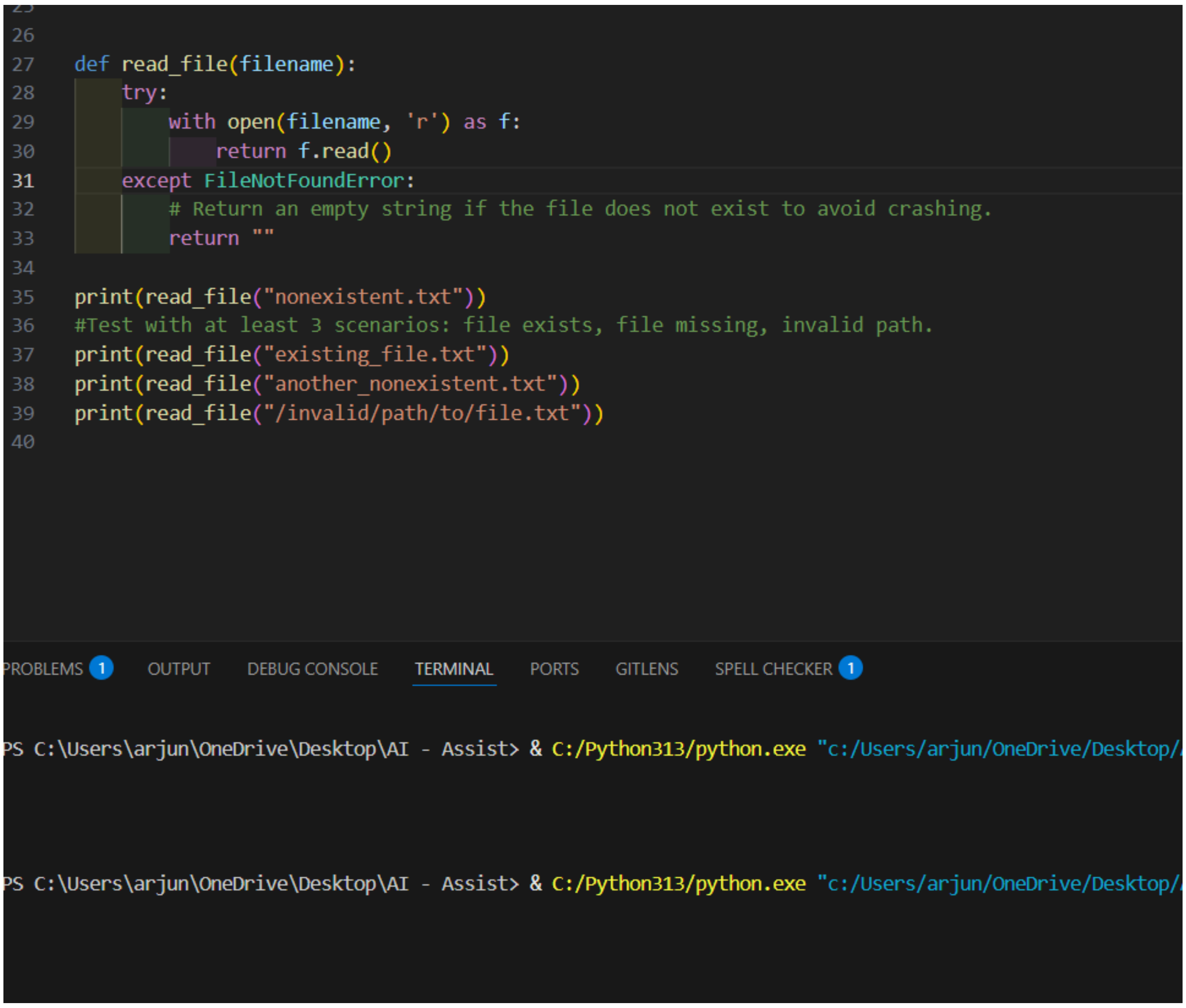
crashes. Use AI to apply safe error handling.

```
# 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"))
```

Prompt:

Analyze the following Python code that crashes when attempting to open a non-existent file. Apply safe error handling using AI suggestions, display a user-friendly error message, and verify the solution using multiple test scenarios.

Code:



```python
def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        # Return an empty string if the file does not exist to avoid crashing.
        return ""

print(read_file("nonexistent.txt"))
#Test with at least 3 scenarios: file exists, file missing, invalid path.
print(read_file("existing_file.txt"))
print(read_file("another_nonexistent.txt"))
print(read_file("/invalid/path/to/file.txt"))
```

```
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Desktop/


PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Desktop/
```

Observation:

When the given code is executed with a file that does not exist, Python raises a **runtime error**:

FileNotFoundError: [Errno 2] No such file or directory: 'nonexistent.txt'

The program crashes because there is **no exception handling** to manage missing or invalid files.

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"

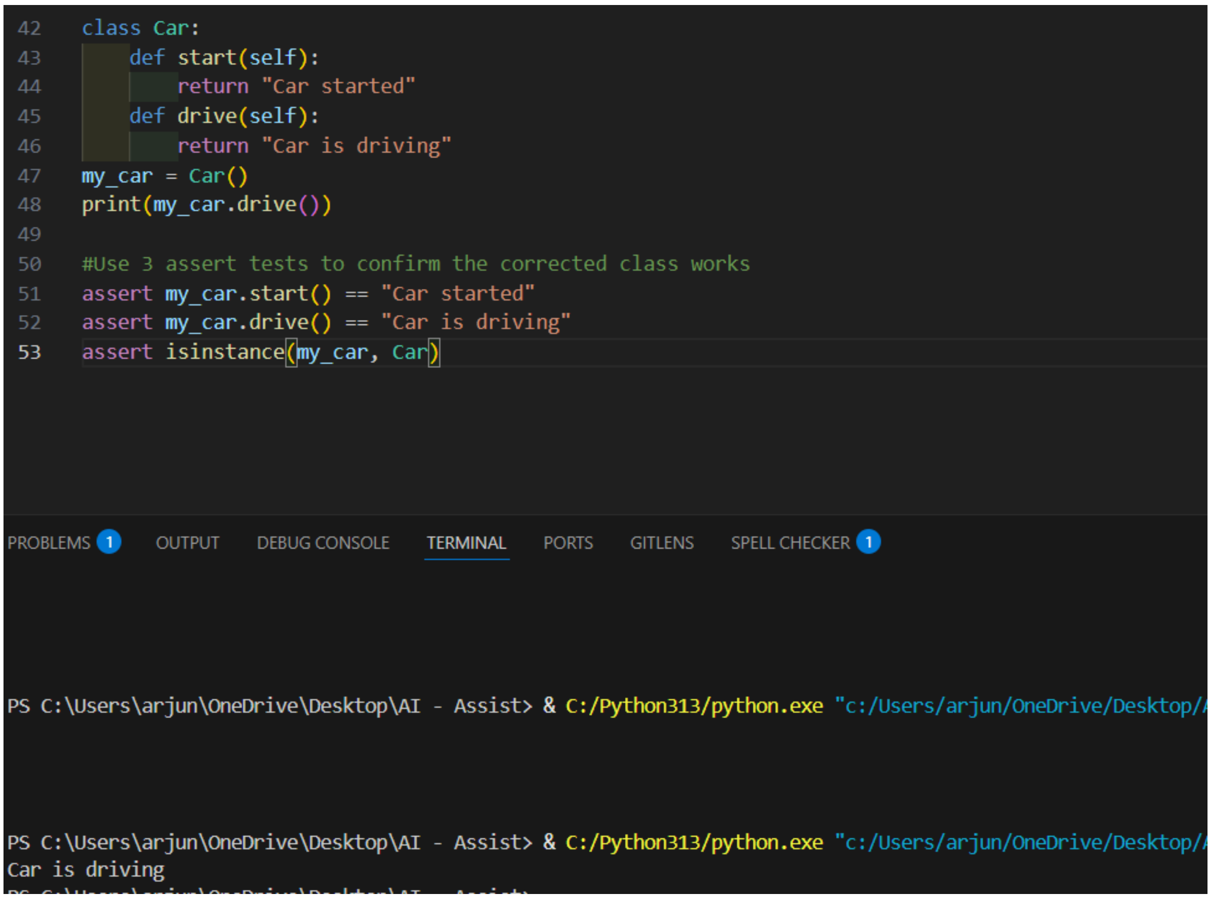
my_car = Car()

print(my_car.drive()) # drive() is not defined

Prompt:

Analyze the following Python class where a non-existent method is called. Identify the cause of the error and decide whether to define the missing method or correct the method call. Fix the issue and verify the corrected class using assert test cases.

Code:



```python
class Car:
    def start(self):
        return "Car started"
    def drive(self):
        return "Car is driving"
my_car = Car()
print(my_car.drive())

#Use 3 assert tests to confirm the corrected class works
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car, Car)
```

```
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Desktop/


PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.exe "c:/Users/arjun/OneDrive/Desktop/
Car is driving
```

Observation:

When the given code is executed, Python raises an **AttributeError**:

AttributeError: 'Car' object has no attribute 'drive'

This error occurs because the method drive() is called on the Car object, but it is **not defined** in the Car class.

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

Task: Provide code that adds an integer and string ("5" + 2) causing

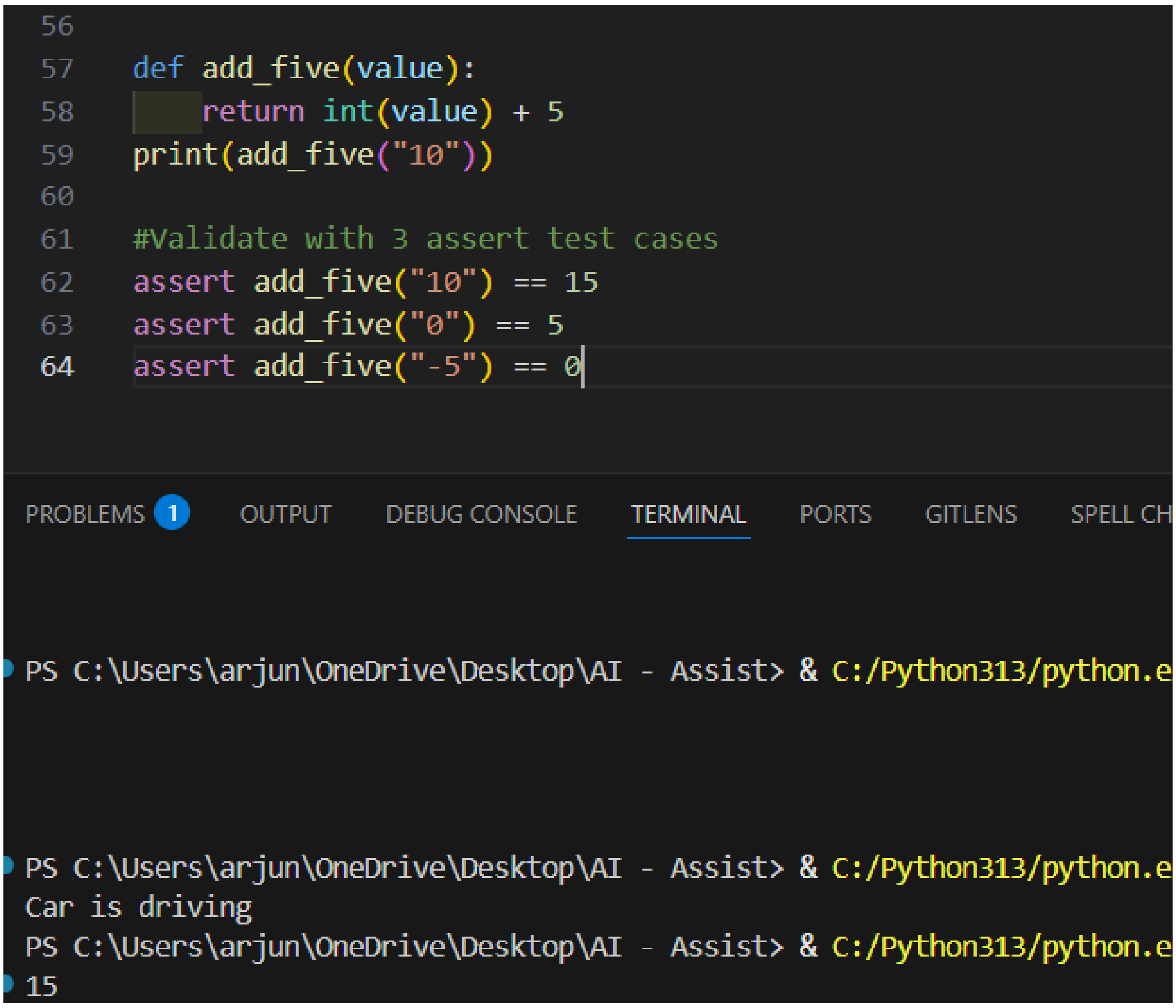
a TypeError. Use AI to resolve the bug.

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

Prompt:

Analyze the following Python code that raises a TypeError when adding a string and an integer. Ask AI to provide two possible solutions—one using type casting and another using string concatenation. Fix the code and validate it using assert test cases.

Code:



```python

def add_five(value):
    return int(value) + 5
print(add_five("10"))

#Validate with 3 assert test cases
assert add_five("10") == 15
assert add_five("0") == 5
assert add_five("-5") == 0
```

```
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.e



PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.e
Car is driving
PS C:\Users\arjun\OneDrive\Desktop\AI - Assist> & C:/Python313/python.e
15
```

Observation:

When the given code is executed, Python raises a **TypeError**:

TypeError: can only concatenate str (not "int") to str

This happens because the function attempts to add an integer (5) to a string ("10"), which is not allowed in Python without explicit conversion.