AI Assisted Coding

## Assignment-7.1

## K.Sindhu meenan(2403A51250)-Batch11

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.

Buggy Code:

def greet():

print "Hello, AI Debugging Lab

greet()

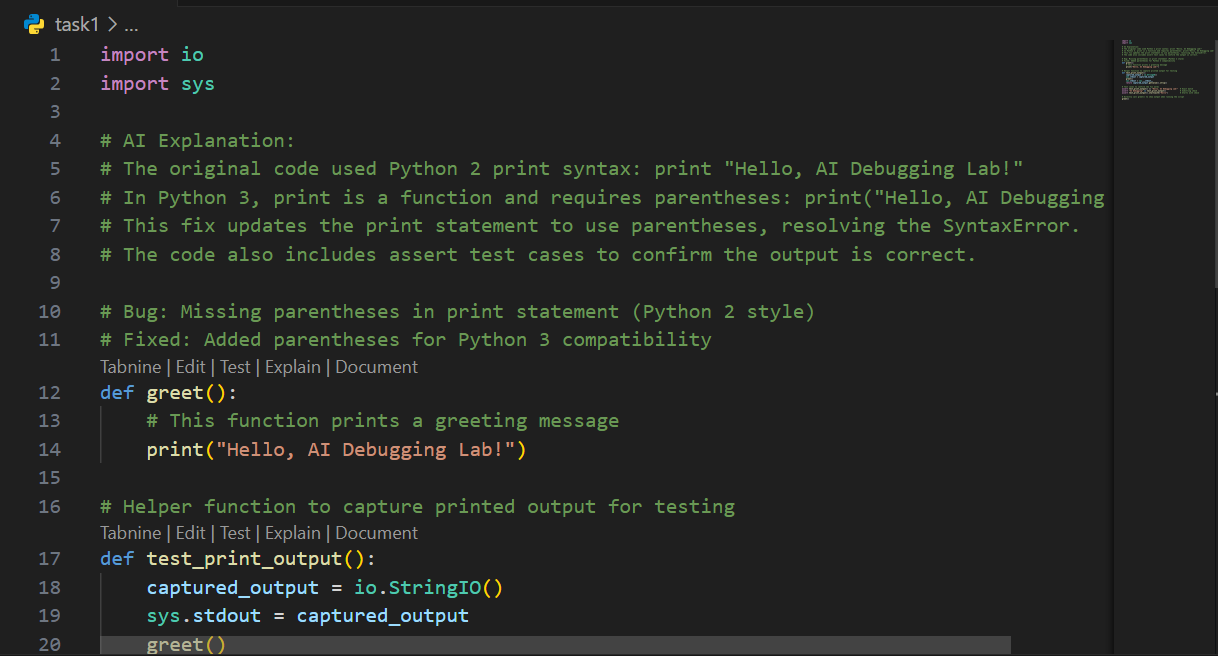
Expected Output #1:  
• Corrected code with proper syntax and AI explanation.

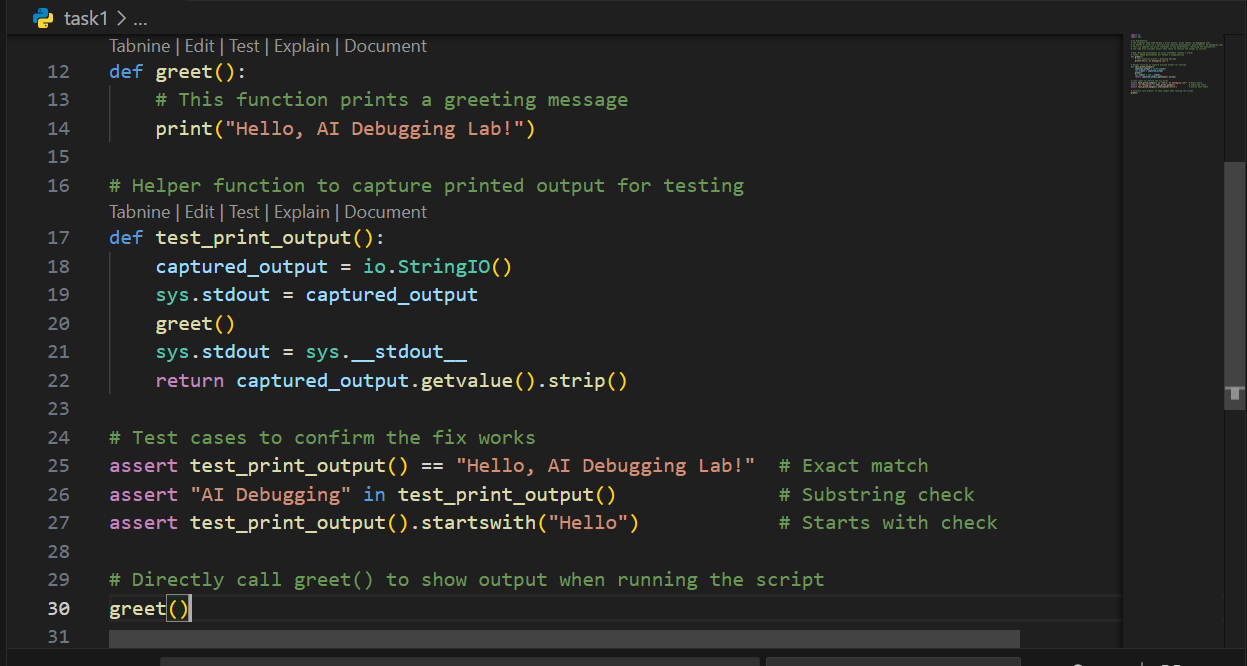
Prompt:

I have a Python program that shows an error. Identify the type of error (syntax, logic, runtime, type, or attribute). Explain why the error occurs.Suggest a corrected version of the code.

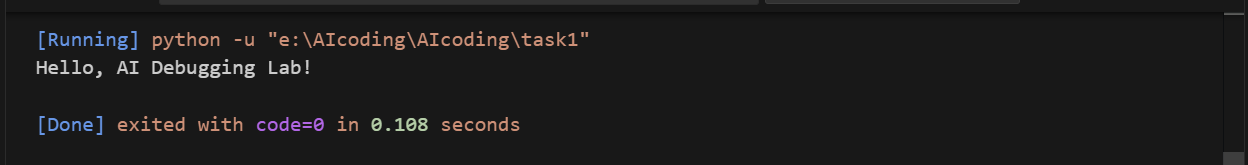
Provide at least 3 assert test cases to confirm the fix works. Show the expected output after correction.

Corrected code:





Ouput:



Task2: Supply a function where an if-condition mistakenly uses = instead  
of ==. Let AI identify and fix the issue.

Buggy code:

def check\_number(n):  
if n = 10:  
return "Ten"  
else:  
return "Not Ten"

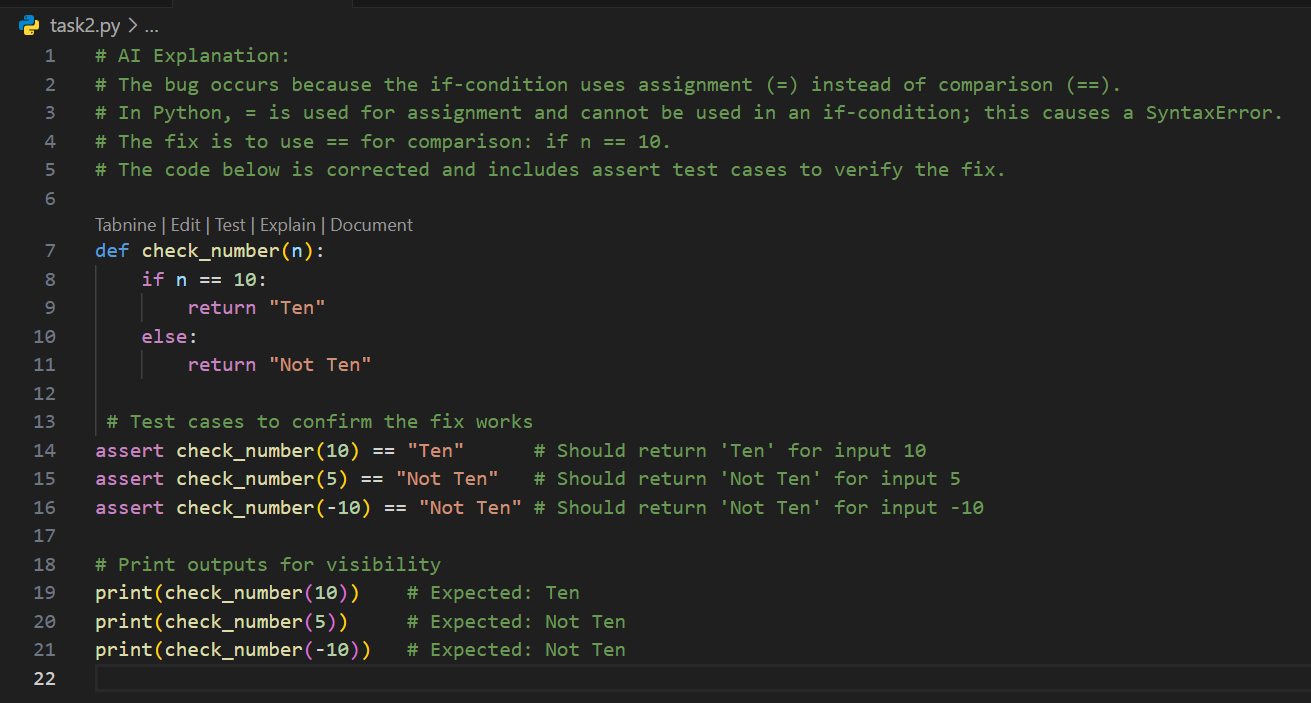
Expected Output #2:  
• Corrected code using == with explanation and successful test  
execution

Prompt:

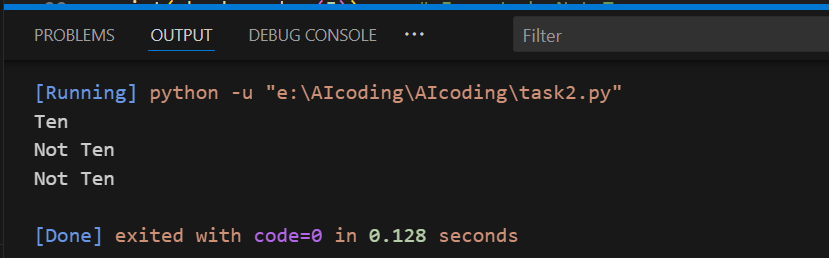
The following code uses = instead of == inside an if statement. Please debug and correct it.

Provide the corrected function code.

Add 3 assert test cases that should all pass.

Corrected code: 

Ouput:



Task3: Provide code that attempts to open a non-existent file and crashes.  
Use AI to apply safe error handling.

Buggy Code:

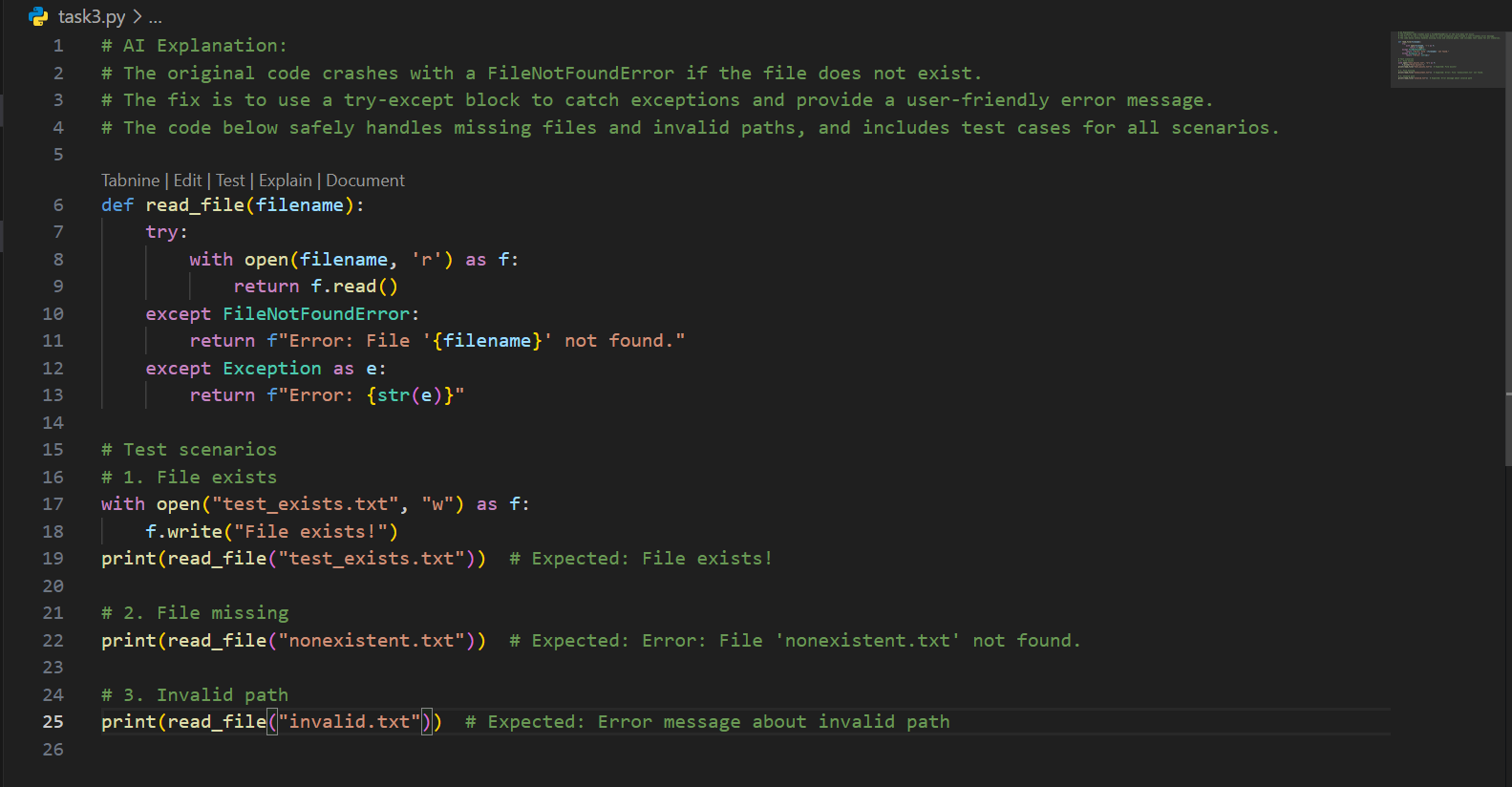
def read\_file(filename):  
with open(filename, 'r') as f:  
return f.read()  
print(read\_file("nonexistent.txt"))

Expected Output #3:  
• Safe file handling with exception management.

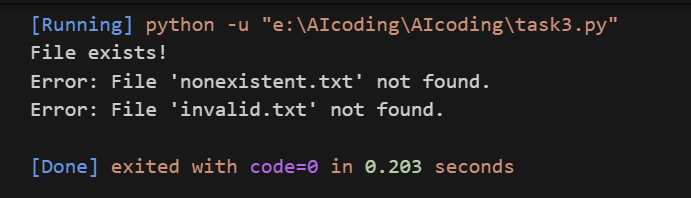
Prompt:

The following code crashes if the file does not exist. Please debug and correct it by adding safe error handling.  
Implement a try-except block with a user-friendly error message.  
Test the function with at least 3 scenarios:

1. File exists
2. File missing
3. Invalid path

Corrected code:  


Output:



Task4: Give a class where a non-existent method is called (e.g.,  
obj.undefined\_method()). Use AI to debug and fix.

Buggy Code:

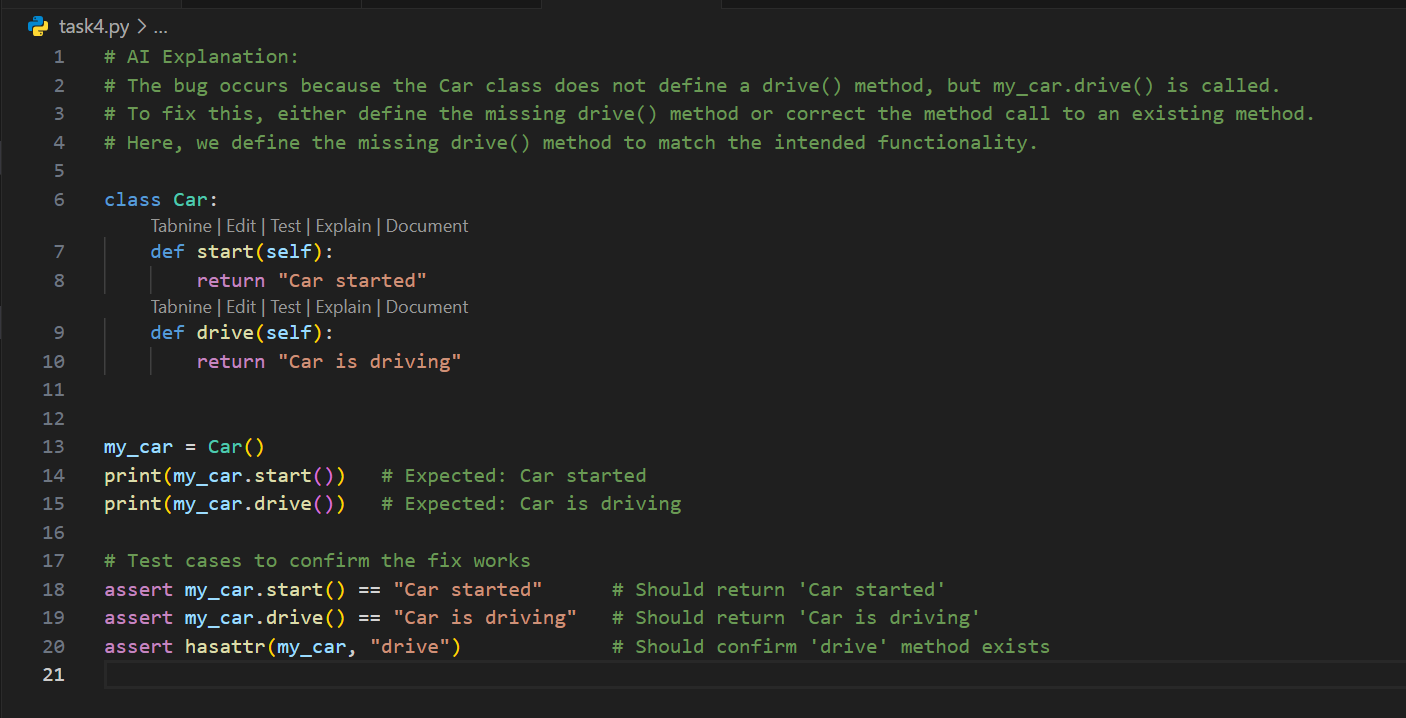
class Car:  
def start(self):  
return "Car started"  
my\_car = Car()  
print(my\_car.drive()) # drive() is not defined

Expected Output #4:  
• Corrected class with clear AI explanation.

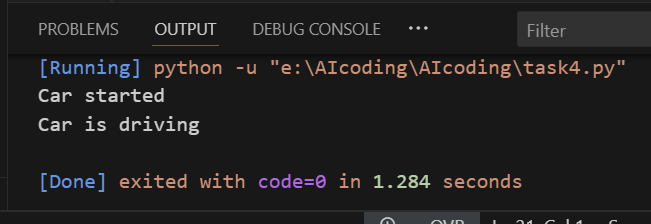
Prompt:

The following code calls a non-existent method and raises an AttributeError. Please debug and correct it.  
Analyze whether to define the missing method or correct the method call.  
Update the class accordingly and provide 3 assert tests to confirm the corrected class works.

Corrected Code:



Output:



Task5: Provide code that adds an integer and string ("5" + 2) causing a  
TypeError. Use AI to resolve the bug.

Buggy code:

def add\_five(value):  
return value + 5  
print(add\_five("10"))

Expected Output #5:  
• Corrected code that runs successfully for multiple inputs.

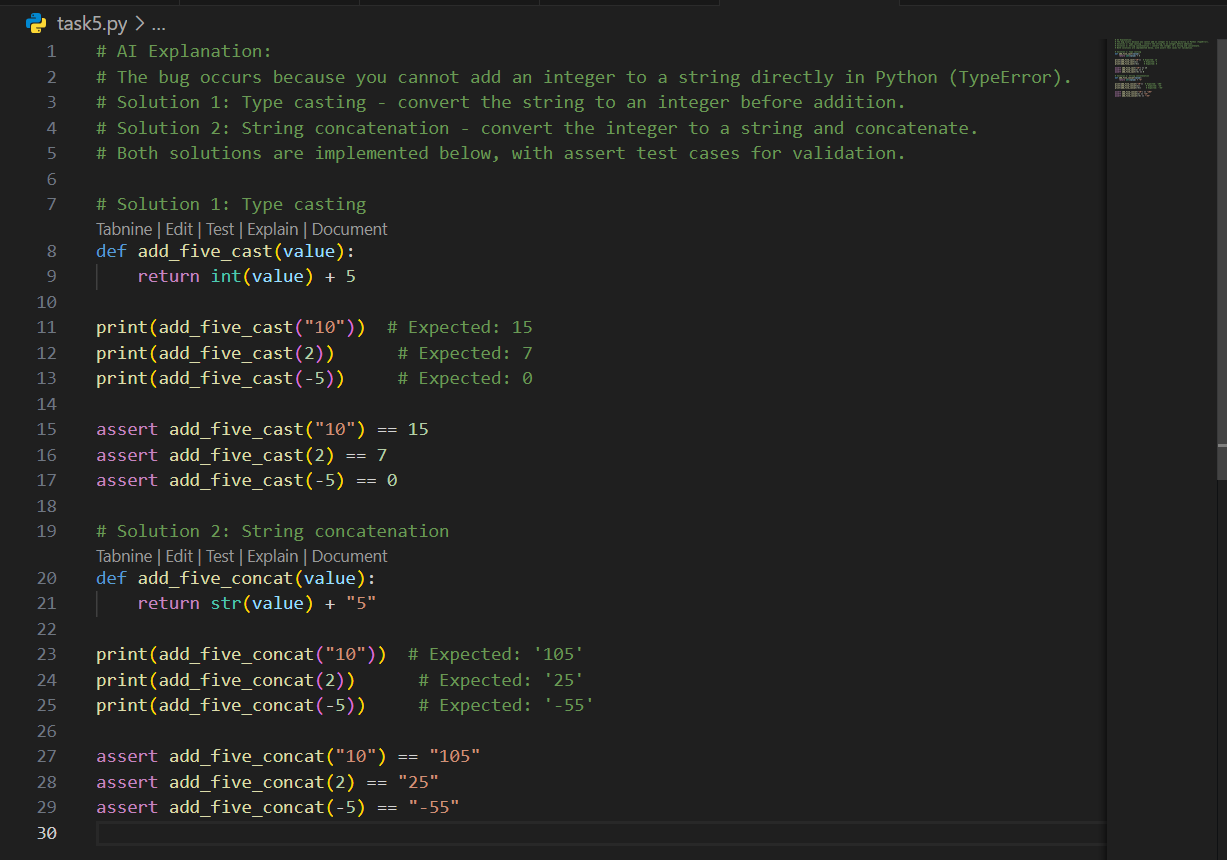
Prompt:

The following code mixes strings and integers, causing a TypeError. Please debug and correct it.  
Provide **two solutions**:

Fix using type casting (convert string to int before addition).

Fix using string concatenation (convert int to string).  
Validate both solutions with 3 assert test cases each.

Corrected code:



Output:

