

ASSIGNMENT-7.1

Name:M Varshith Reddy

HTNO:2303A52087

Batch:40

Task Description #1 (Syntax Errors – Missing Parentheses in Print Statement)

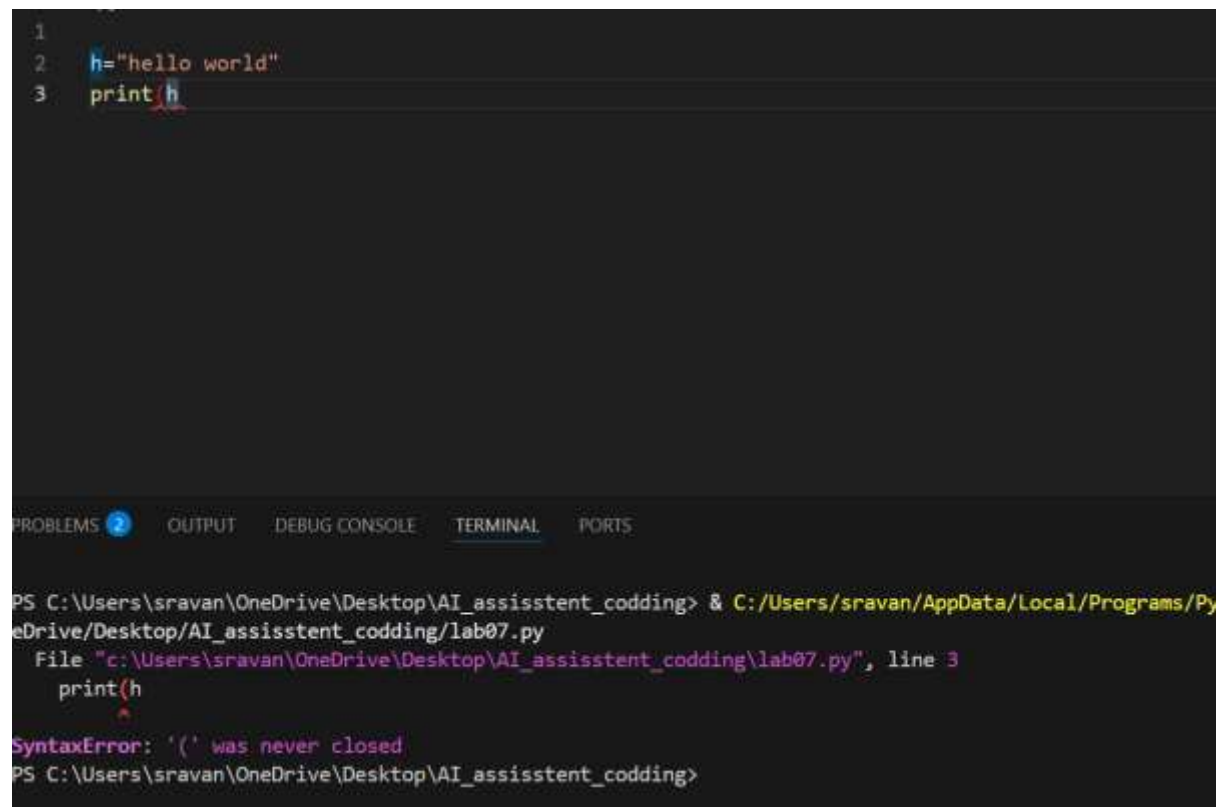
Task: Provide a Python snippet with a missing parenthesis in a print statement (e.g., print "Hello"). Use AI to detect and fix the syntax error.

Bug: Missing parentheses in print statement

```
def greet():
```

```
    print "Hello, AI Debugging Lab!"
```

```
greet()
```



The screenshot shows a code editor with a dark theme. The code is as follows:

```
1
2 h="hello world"
3 print h
```

Below the code editor, the terminal window is open, showing the command to run the script and the resulting error message:

```
PS C:\Users\srajan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/srajan/AppData/Local/Programs/Python/Python39-64/python.exe C:/Users/srajan/OneDrive/Desktop/AI_assisstant_coding/lab07.py
File "c:\Users\srajan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 3
    print(h
          ^
SyntaxError: '(' was never closed
PS C:\Users\srajan\OneDrive\Desktop\AI_assisstant_coding>
```

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:
```

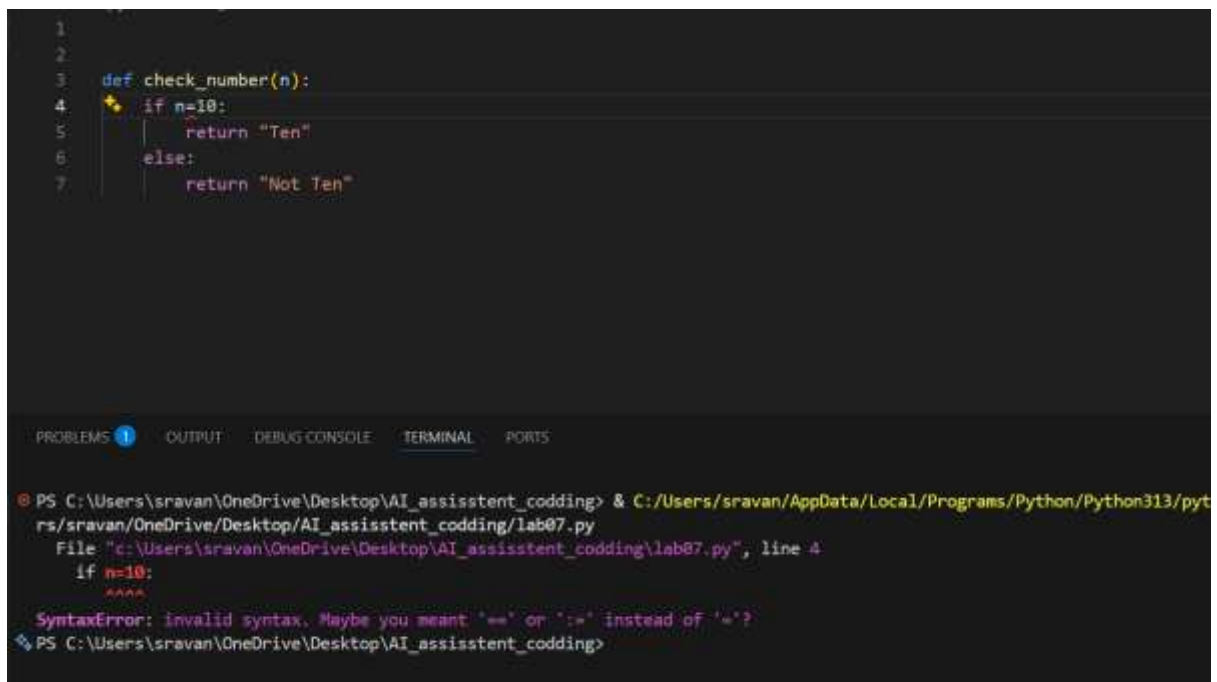
```
        return "Not Ten"
```

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.



```
1
2
3 def check_number(n):
4     if n=10:
5         return "Ten"
6     else:
7         return "Not Ten"
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
@ PS C:\Users\srajan\OneDrive\Desktop\AI_assisntent_coddng> & C:/Users/srajan/AppData/Local/Programs/Python/Python313/pyt
rs/srajan/OneDrive/Desktop/AI_assisntent_coddng/lab07.py
File "c:\Users\srajan\OneDrive\Desktop\AI_assisntent_coddng\lab07.py", line 4
    if n=10:
        ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

PS C:\Users\srajan\OneDrive\Desktop\AI_assisntent_coddng>

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

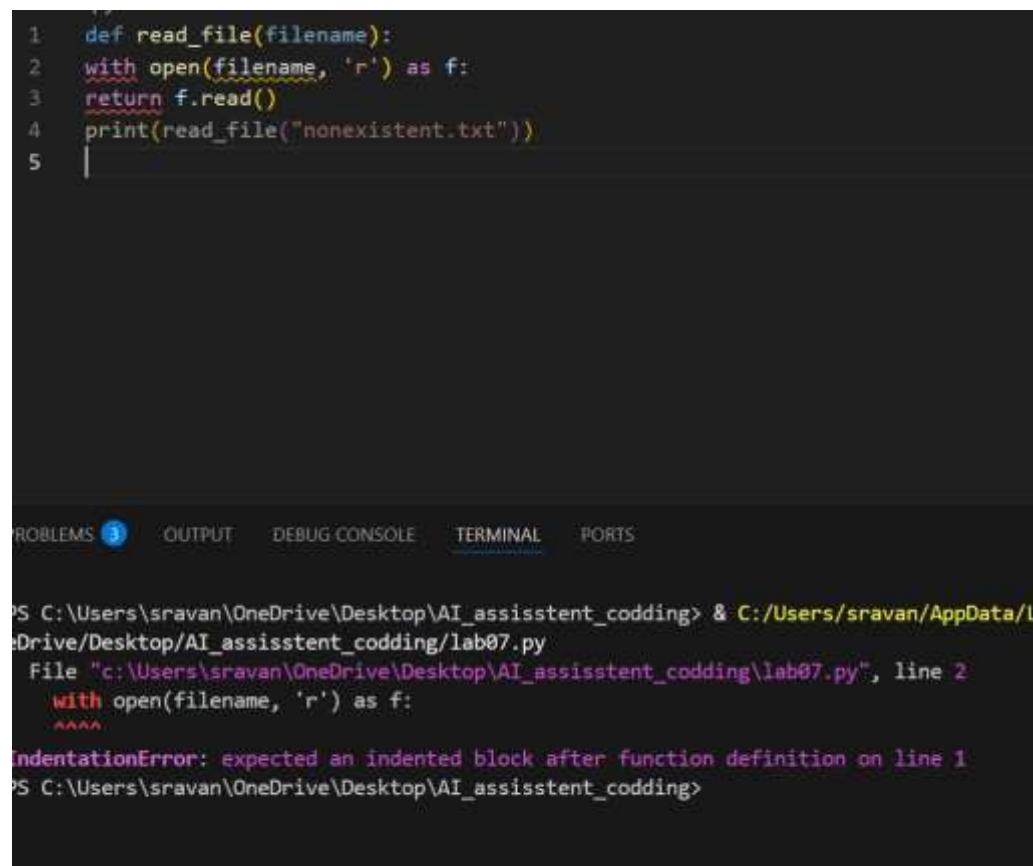
Task: Provide code that attempts to open a non-existent file and crashes. Use AI to apply safe error handling.

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.



```
1 def read_file(filename):  
2     with open(filename, 'r') as f:  
3         return f.read()  
4     print(read_file("nonexistent.txt"))  
5
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\srajan\OneDrive\Desktop\AI_assisntent_coding> & C:/Users/srajan/AppData/Local/Programs/Python/Python311/Python.exe C:\Users\srajan\OneDrive\Desktop\AI_assisntent_coding\lab07.py  
File "c:\Users\srajan\OneDrive\Desktop\AI_assisntent_coding\lab07.py", line 2  
    with open(filename, 'r') as f:  
    ^^^^^  
IndentationError: expected an indented block after function definition on line 1  
PS C:\Users\srajan\OneDrive\Desktop\AI_assisntent_coding>
```

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

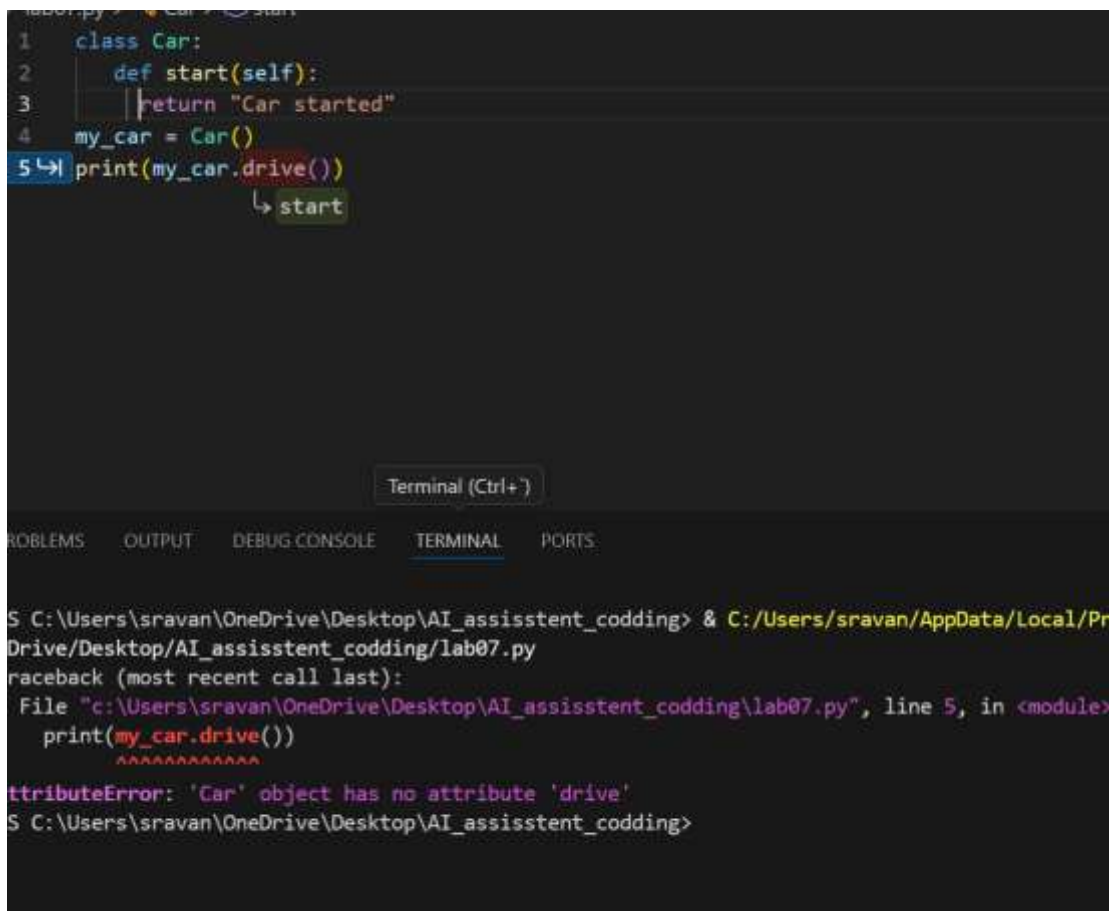
```
class Car:
    def start(self):
        return "Car started"

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.



The screenshot shows a code editor with the following Python code:

```
1 class Car:
2     def start(self):
3         return "Car started"
4 my_car = Car()
5 print(my_car.drive())
```

Below the code editor is a terminal window titled "Terminal (Ctrl+`)" showing the execution of the script. The terminal output is as follows:

```
S C:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/sraavan/AppData/Local/Programs/Python/Python310/python.exe C:/Users/sraavan/AppData/Local/Programs/Python/Python310/python.exe C:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding\lab07.py
Traceback (most recent call last):
  File "c:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 5, in <module>
    print(my_car.drive())
AttributeError: 'Car' object has no attribute 'drive'
S C:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding>
```

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

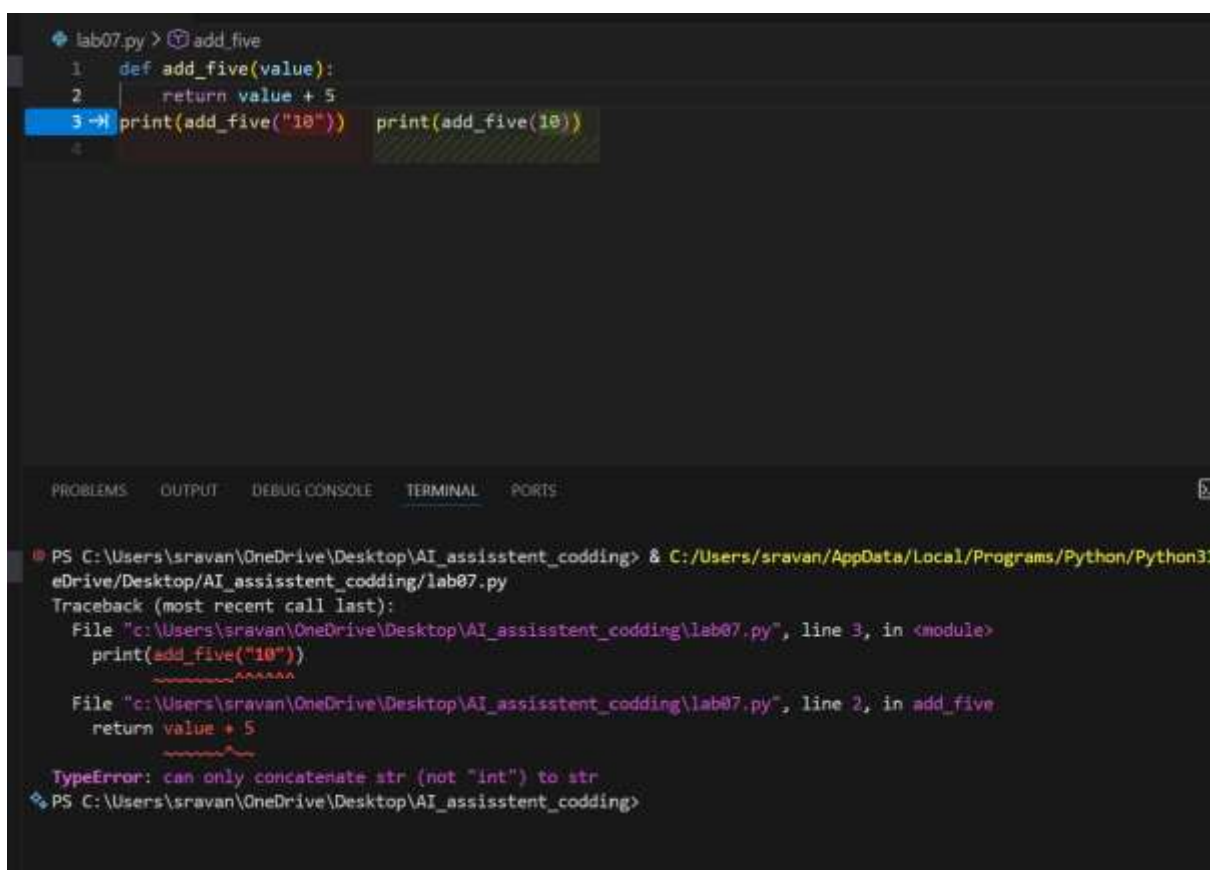
```
def add_five(value):
```

```
    return value + 5
```

```
print(add_five("10"))
```

Requirements:

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



```
lab07.py > add_five
1 def add_five(value):
2     return value + 5
3 → print(add_five("10")) print(add_five(10))
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/sraavan/AppData/Local/Programs/Python/Python311/Python311.exe C:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding\lab07.py
Traceback (most recent call last):
  File "c:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 3, in <module>
    print(add_five("10"))
           ~~~~~~^~~~~~
  File "c:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 2, in add_five
    return value + 5
           ~~~~~~^~
TypeError: can only concatenate str (not "int") to str
PS C:\Users\sraavan\OneDrive\Desktop\AI_assisstant_coding>
```