

Assignment-7.1

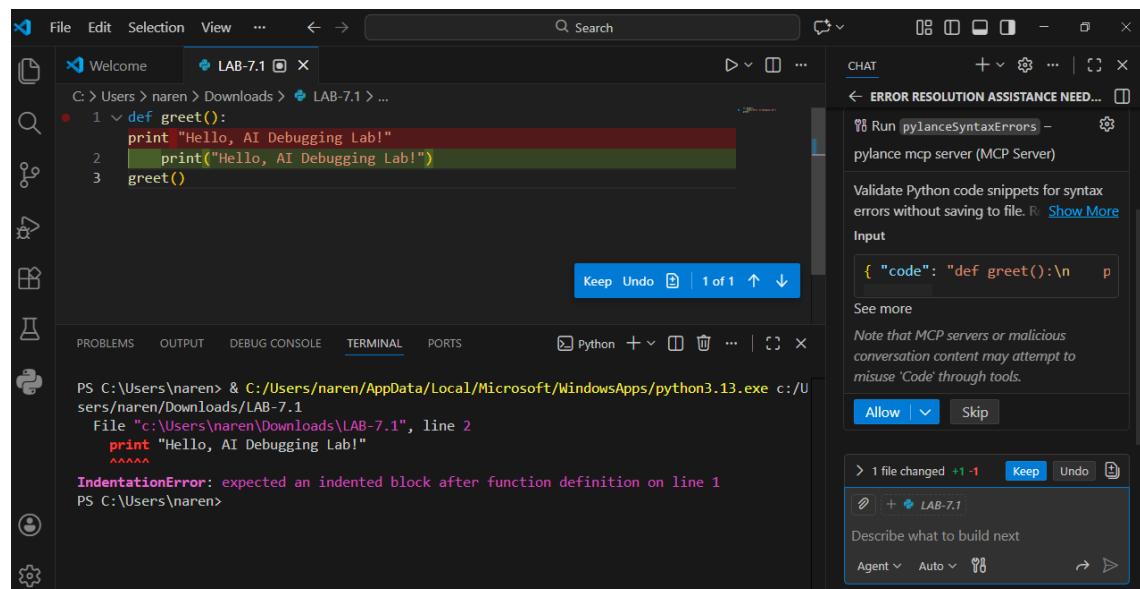
N. SOURISH

2303A52360

Batch-44

Task 1 description: (Syntax Errors – Missing Parentheses in Print Statement)

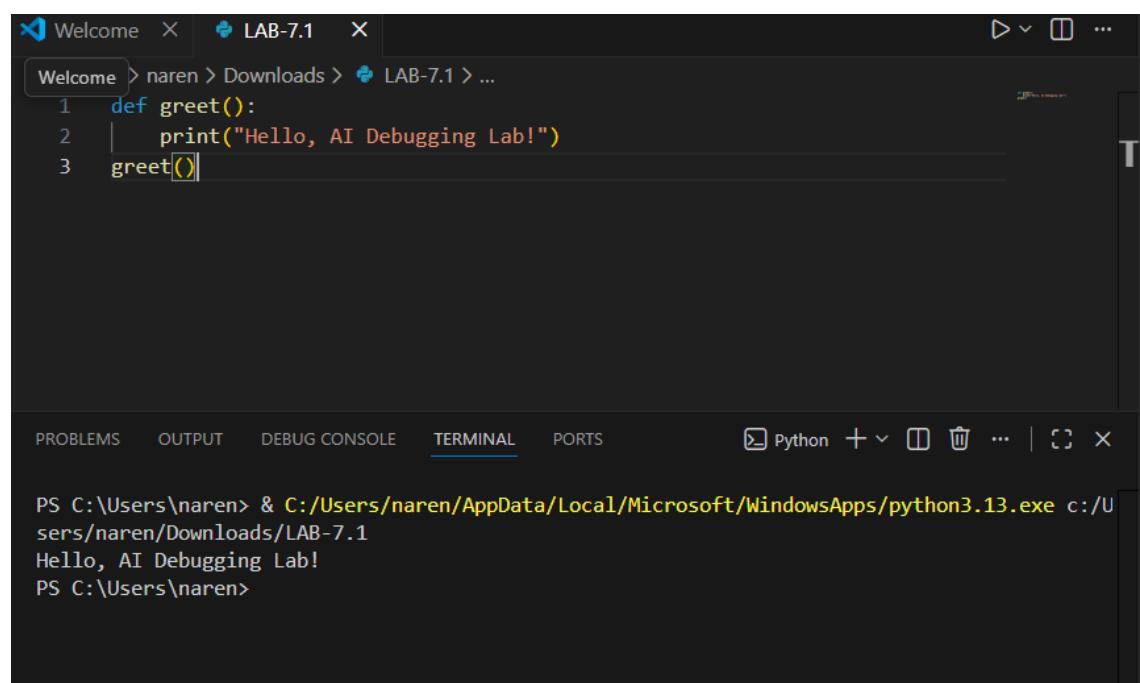
Task: Provide a Python snippet with a missing parenthesis in a print statement (e.g., print "Hello"). Use AI to detect and fix the syntax error.



The screenshot shows a Python file named LAB-7.1 with the following code:

```
1 def greet():
2     print("Hello, AI Debugging Lab!"
3     greet()
```

A red squiggly underline appears under the opening parenthesis of the second print statement. The status bar at the bottom indicates an `IndentationError` on line 1. A right-click context menu is open over the third line, showing options like "Run", "Run All", "Run Selection", and "Run Selection All".



The screenshot shows the same Python file LAB-7.1 with the code now fixed:

```
1 def greet():
2     print("Hello, AI Debugging Lab!")
3     greet()
```

The red squiggly underline is gone. The terminal below shows the output of running the script, which prints "Hello, AI Debugging Lab!" to the console.

Task 2 description: (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.

The screenshot shows the Visual Studio Code interface with a Python file named `check_number.py` open. The code contains a syntax error: `n = 10` instead of `n == 10`. The code editor highlights this line in red. A tooltip on the right side of the screen provides a code fix suggestion: `{ "code": "def check_number(n):\n if n = 10:\n return "Ten"\n else:\n return "Not Ten"\n"}`. Below the code editor, the terminal window shows the command `python3.13.exe` running and outputting a `SyntaxError` message about the invalid syntax. The status bar at the bottom indicates `1 file changed +1 -1`.

The screenshot shows the Visual Studio Code interface with the same Python file now corrected. The code editor displays the fixed code: `n == 10`. The terminal window shows the command `python3.13.exe` running and outputting the expected result `Ten`. The status bar at the bottom indicates `1 file changed +1 -1`.

Task 3 description: (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

The screenshot shows a Python script named `LAB-7.1` in a VS Code editor. The code defines a function `read_file` that attempts to open a file and return its contents. It includes a try-except block to handle `FileNotFoundException`. The terminal below shows the script being run and crashing with a `FileNotFoundError` because the file `"nonexistent.txt"` does not exist.

```
C: > Users > naren > Downloads > LAB-7.1 > ...
1 def read_file(filename):
2     with open(filename, 'r') as f:
3         return f.read()
4     try:
5         with open(filename, 'r') as f:
6             return f.read()
7     except FileNotFoundError:
8         return "File not found"
9 print(read_file("nonexistent.txt"))

PS C:\Users\naren> & C:/Users/naren/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/naren/Downloads/LAB-7.1
Traceback (most recent call last):
  File "c:/Users/naren/Downloads/LAB-7.1", line 4, in <module>
    print(read_file("nonexistent.txt"))
               ^^^^^^^^^^^^^^^^^^
  File "c:/Users/naren/Downloads/LAB-7.1", line 2, in read_file
    with open(filename, 'r') as f:
               ^^^^^^^^^^^^^^
FileNotFoundException: [Errno 2] No such file or directory: 'nonexistent.txt'
PS C:\Users\naren>
```

A floating panel titled "ERROR RESOLUTION ASSISTANCE NEEDED..." suggests running `pylanceRunCodeSnippet` to execute Python code snippets directly in the workspace environment. A button labeled "Allow" is present in the panel.

The screenshot shows the same Python script in VS Code after applying the suggested fix. The code now uses a standard `try-except` block without the nested `with` statement. The terminal output shows the script running successfully and printing the expected "File not found" message.

```
C: > Users > naren > Downloads > LAB-7.1 > ...
1 def read_file(filename):
2     try:
3         with open(filename, 'r') as f:
4             return f.read()
5     except FileNotFoundError:
6         return "File not found"
7 print(read_file("nonexistent.txt"))

PS C:\Users\naren> & C:/Users/naren/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/naren/Downloads/LAB-7.1
File not found
PS C:\Users\naren>
```

Task 4 description: (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.

The screenshot shows a Python file named `LAB-7.1` in the VS Code editor. The code defines a `Car` class with a `start` method and prints the result of calling `drive` and `start` on an instance of `Car`. The line `print(my_car.start())` is highlighted in red, indicating a syntax error. The terminal below shows the execution of the script and the resulting `AttributeError`.

```
C: > Users > naren > Downloads > LAB-7.1 > ...
1  class Car:
2      def start(self):
3          return "Car started"
4  my_car = Car()
5  print(my_car.drive())
6  print(my_car.start())

PS C:\Users\naren> & C:/Users/naren/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/naren/Downloads/LAB-7.1
Traceback (most recent call last):
  File "c:/Users/naren/Downloads/LAB-7.1", line 5, in <module>
    print(my_car.start())
                  ^^^^^^^^^^
AttributeError: 'Car' object has no attribute 'start'
PS C:\Users\naren>
```

The screenshot shows the same Python file in VS Code. The AI assistance panel on the right suggests a fix for the error. It proposes changing `start` to `drive` and provides a link to learn more about the error.

```
C: > Users > naren > Downloads > LAB-7.1 > ...
1  class Car:
2      def start(self):
3          return "Car started"
4  my_car = Car()
5  print(my_car.start())

PS C:\Users\naren> & C:/Users/naren/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/naren/Downloads/LAB-7.1
Car started
PS C:\Users\naren>
```

CHAT

← ERROR RESOLUTION ASSISTANCE NEED...

Run `pylanceSyntaxErrors` – pylance mcp server (MCP Server)

Validate Python code snippets for syntax errors without saving to file. [Show More](#)

Input

{ "code": "class Car:\n def

See more

Note that MCP servers or malicious conversation content may attempt to misuse 'Code' through tools.

Allow | Skip

> 1 file changed +1 -1 Keep Undo

LAB-7.1

Describe what to build next

Agent Auto

SOLVE THE ERROR

✓ Read LAB-7.1

✓ Edited LAB-7.1 +0 -0

✓ Ran `pylanceSyntaxErrors` – pylance mcp server (MCP Server)

The error in your code was an `AttributeError` because the `Car` class doesn't have a `drive()` method—you defined `start()` instead.

Task 5 Description (Type Error – Mixing Strings and Integers in Addition)

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

The screenshot shows a VS Code interface with a Python file named `LAB-7.1`. The code defines a function `add_five` that returns the sum of its argument and 5. Two calls to this function are made: `print(add_five("10"))` and `print(add_five(10))`. The second call is highlighted in red, indicating a syntax error. The terminal below shows the execution of the script and the resulting `TypeError`:

```
C:\> Users > naren > Downloads > LAB-7.1 > ...
1 def add_five(value):
2     return value + 5
3 print(add_five("10"))
3 print(add_five(10))

PS C:\Users\naren> & C:/Users/naren/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/naren/Downloads/LAB-7.1
Traceback (most recent call last):
  File "c:/Users/naren/Downloads/LAB-7.1", line 3, in <module>
    print(add_five("10"))
           ~~~~~~^~~~~~
  File "c:/Users/naren/Downloads/LAB-7.1", line 2, in add_five
    return value + 5
           ~~~~~~^~~~~~
TypeError: can only concatenate str (not "int") to str
PS C:\Users\naren>
```

The right-hand sidebar displays an "ERROR RESOLUTION ASSISTANCE NEED..." panel. It suggests running `pylanceSyntaxErrors` to validate the code. A tooltip provides a note about MCP servers or malicious content attempting to misuse the 'Code' tool. Buttons for "Allow" and "Skip" are present. Below the panel, a message indicates 1 file changed and a "Keep" button.

The screenshot shows the same VS Code interface after the code has been corrected. The `print(add_five(10))` line now uses parentheses instead of square brackets, resolving the type error. The terminal output is identical to the previous screenshot.

The right-hand sidebar now displays a message from the AI stating that the user was trying to add an integer to a string and that the corrected code is now:

```
def add_five(value):
    return value + 5
print(add_five(10))
```

The AI also notes that this will run without errors and print `15`, but suggests handling string inputs by converting them if needed.