

## ASSIGNMENT-7.1

Name: P . Samanvith

HTNO:2303A52090

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 terminal window with the following content:

```
1
2     h="hello world"
3     print|h
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/sravan/AppData/Local/Programs/Py
eDrive/Desktop/AI_assisstant_coding/lab07.py
File "c:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding\lab07.py", line 3
    print|h
                                         ^
SyntaxError: '(' was never closed
PS C:\Users\sravan\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.

The screenshot shows a code editor interface with a dark theme. A Python script named `lab07.py` is open. The code contains a function `check_number` with a bug in the if statement. The error is highlighted with a yellow starburst icon. The terminal below shows the command `PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding & C:/Users/sravan/AppData/Local/Programs/Python/Python313/pyt` followed by the error message: `rs/sravan/OneDrive/Desktop/AI_assisstant_coding/lab07.py`. File "c:\Users\sravan\OneDrive\Desktop\AI\_assisstant\_coding\lab07.py", line 4  
if n=10:  
^\_\_\_\_\_  
SyntaxError: invalid syntax. Maybe you meant '==' or '!=' instead of '='?. The terminal ends with `PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding>`.

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.

The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical scroll bar. At the top, there are tabs for 'PROBLEMS' (with a blue circle icon), 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected. In the terminal window, the following text is displayed:

```
PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/sravan/AppData/Local/OneDrive/Desktop/AI_assisstant_coding/lab07.py
  File "c:/Users/sravan/OneDrive/Desktop/AI_assisstant_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\sravan\OneDrive\Desktop\AI_assisstant_coding>
```

The code in the editor is:

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

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 interface with a Python file named `lab07.py`. The code contains a `Car` class with a `start` method and a `my_car` variable. A cursor is at the end of the line `print(my_car.drive())`, which is highlighted in red, indicating a syntax error. A tooltip "start" appears over the `start` method name. Below the code editor is a terminal window titled "Terminal (Ctrl+`)" showing the execution of `lab07.py`. The terminal output shows an `AttributeError`: "'Car' object has no attribute 'drive'" at line 5, which corresponds to the highlighted line in the code editor.

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

Terminal (Ctrl+`)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

S C:\Users\saravanan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/saravanan/AppData/Local/Programs/Desktop/AI_assisstant_coding/lab07.py
raceback (most recent call last):
  File "c:\Users\saravanan\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\saravanan\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.

The screenshot shows a code editor interface with a dark theme. At the top, there's a status bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. Below the status bar is a code editor window containing the following Python code:

```
lab07.py > add_five
1 def add_five(value):
2     return value + 5
3 → print(add_five("18"))  print(add_five(18))
```

The code consists of three lines. The first line defines a function `add_five` that takes a parameter `value` and returns its sum with 5. The second line contains a call to `print` with a string argument "18". The third line contains another call to `print` with an integer argument 18. A blue arrow points to the first `print` statement, indicating it is the current line of interest.

Below the code editor, the terminal window displays the following output:

```
PS C:\Users\sravan\OneDrive\Desktop\AI_assisstant_coding> & C:/Users/sravan/AppData/Local/Programs/Python/Python37-32/eDrive/Desktop/AI_assisstant_coding/lab07.py
Traceback (most recent call last):
  File "c:/Users/sravan/OneDrive/Desktop/AI_assisstant_coding/lab07.py", line 3, in <module>
    print(add_five("18"))
           ^~~~~~
  File "c:/Users/sravan/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\sravan\OneDrive\Desktop\AI_assisstant_coding>
```

The terminal shows a `TypeError` occurring at line 3, where the string "18" is being concatenated with the result of the `add_five` function call. The error message specifies that you can only concatenate strings, not integers.