

# AI Assisted Coding

## Assignment – 7.1

**K.VAMSHIDHAR | | 2303A51OH7 | | Batch:- 8**

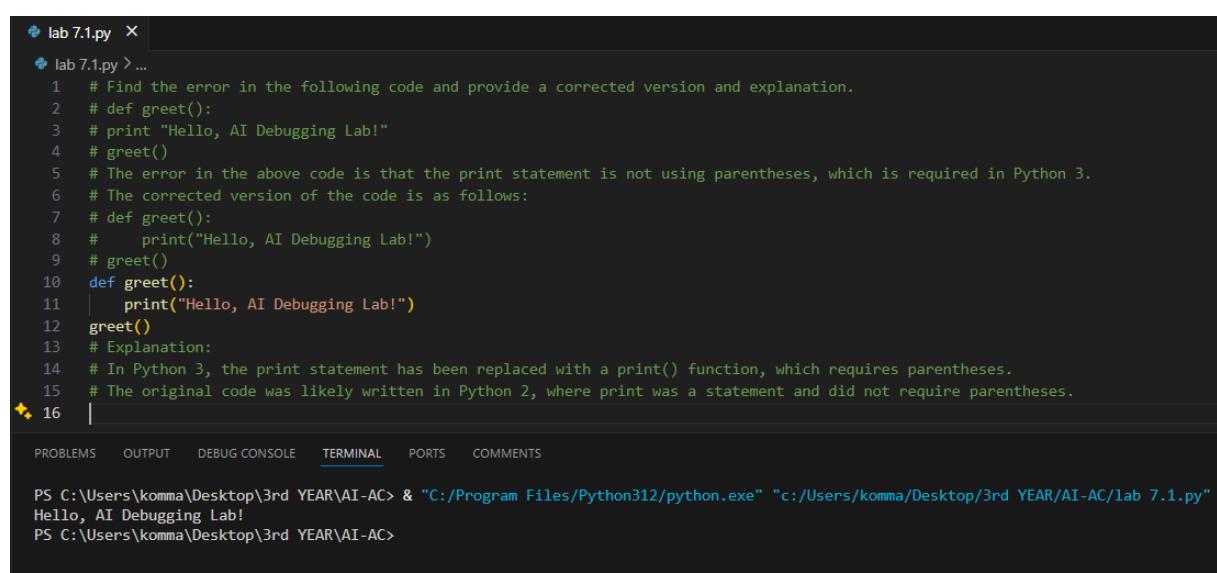
Task Description #1 (Syntax Errors – Missing Parentheses in Print Statement)

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()
```

Requirements:

- Run the given code to observe the error.
- Apply AI suggestions to correct the syntax.
- Use at least 3 assert test cases to confirm the corrected code works.



```
lab 7.1.py > ...
1  # Find the error in the following code and provide a corrected version and explanation.
2  # def greet():
3  #     print "Hello, AI Debugging Lab!"
4  # greet()
5  # The error in the above code is that the print statement is not using parentheses, which is required in Python 3.
6  # The corrected version of the code is as follows:
7  # def greet():
8  #     print("Hello, AI Debugging Lab!")
9  # greet()
10 def greet():
11     print("Hello, AI Debugging Lab!")
12 greet()
13 # Explanation:
14 # In Python 3, the print statement has been replaced with a print() function, which requires parentheses.
15 # The original code was likely written in Python 2, where print was a statement and did not require parentheses.
16 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"
Hello, AI Debugging Lab!
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

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.

The screenshot shows a code editor window titled "lab 7.1.py". The code is a Python function named `check\_number` that checks if a number `n` is equal to 10. In the original code, the assignment operator `=` was used instead of the equality operator `==`. The code also includes explanatory comments about the error and the fix. Below the code editor is a terminal window showing the execution of the script and its output. The terminal shows the script being run with `python.exe`, and the output shows two runs: one where `n` is 10 and one where `n` is not 10, both resulting in the expected outputs "Ten" and "Not Ten" respectively.

```
lab 7.1.py > ...
18 #fix the error in the if condition for the below provided code and explain the error and the fix.
19 # def check_number(n):
20 # if n = 10:
21 #     return "Ten"
22 # else:
23 #     return "Not Ten"
24 # The error in the above code is that the assignment operator (=) is used instead of the equality operator (==) in the if condition.
25 # The corrected version of the code is as follows:
26 def check_number(n):
27     if n == 10:
28         return "Ten"
29     else:
30         return "Not Ten"
31 def check_number(n):
32     if n == 10:
33         return "Ten"
34     else:
35         return "Not Ten"
36 # Explanation:
37 # In the original code, the if condition was using the assignment operator (=) instead of the equality operator (==).
38 # The assignment operator is used to assign a value to a variable, while the equality operator is used to compare two values for equality.
39 # In the corrected code, we use the equality operator (==) to check if the value of n is equal to 10.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"
Hello, AI Debugging Lab!
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"
Ten
Not Ten
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"
Ten
Not Ten
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

### 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.

```

lab 7.1.py
lab 7.1.py > ...
45
46     #Handle the error in the below code by try and except block and explain the error and add the user friendly error message.
47     # def read_file(filename):
48     # with open(filename, 'r') as f:
49     # return f.read()
50     # print(read_file("nonexistent.txt"))
51     # The error in the above code is that it tries to read a file that does not exist, which will raise a FileNotFoundError.
52     # The corrected version of the code with error handling is as follows:
53     # def read_file(filename):
54     #     try:
55     #         with open(filename, 'r') as f:
56     #             return f.read()
57     #     except FileNotFoundError:
58     #         return f"Error: The file '{filename}' was not found. Please check the filename and try again."
59     # print(read_file("nonexistent.txt"))
60
61 def read_file(filename):
62
63     try:
64         with open(filename,
65                  'r') as f:
66             return f.read()
67     except FileNotFoundError:
68         return f"Error: The file '{filename}' was not found. Please check the filename and try again."
69 print(read_file("nonexistent.txt"))
70 # Explanation:
71 # In the original code, when the function tries to read a file that does not exist, it raises a FileNotFoundError, which can crash the program if not handled.
72 # In the corrected code, we use a try-except block to catch the FileNotFoundError. If the file is not found, we return a user-friendly error message that informs the user about the
73

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Python +

```

ps :C:\Users\komma\Desktop\3rd YEAR\AI-AC & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"
Error: The file 'nonexistent.txt' was not found. Please check the filename and try again.
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>

```

### 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.

### 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.

```

74
75
76 #In the below code fix the bug related to the Calling an undefined method and explain the error and the fix.
77 # class Car:
78 #     def start(self):
79 #         return "Car started"
80 #     my_car = Car()
81 #     print(my_car.drive()) # drive() is not defined
82 # The error in the above code is that the method `drive()` is called on the `my_car` object, but it is not defined in the `Car` class.
83 # The corrected version of the code is as follows:
84
85 class Car:
86     def start(self):
87         return "Car started"
88
89     def drive(self):
90         return "Car is driving"
91 my_car = Car()
92 print(my_car.drive()) # Output: Car is driving
93 # Explanation:
94 # In the original code, the `drive()` method was called on the `my_car` object, but it was not defined in the `Car` class, which would result in an AttributeError.
95 # In the corrected code, we defined the `drive()` method within the `Car` class, which allows us to call it on the `my_car` object without any errors. Now, when we call `my_car.dri
96

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Python +

PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"  
Car is driving  
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>

## # 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.

```

98
99
100 #In the below code there is a TypeError due to mixing string and integer. You fix the error and add that numbers.
101 # def add_five(value):
102 #     return value + 5
103 # print(add_five("10"))
104 # The error in the above code is that it tries to add an integer (5) to a string ("10"), which will raise a TypeError.
105 # The corrected version of the code is as follows:
106 def add_five(value):
107     return int(value) + 5
108 print(add_five("10")) # Output: 15
109 # Explanation:
110 # In the original code, the function `add_five` attempts to add an integer (5) to a string ("10"), which is not allowed in Python.
111 # In the corrected code, we convert the input `value` to an integer using the `int()` function before adding 5. This allows us to
112

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Python +

PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 7.1.py"  
15  
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>