

Lab Assignment- 7.1

NAME: A.Harshita

ID.NO:2303A52211

Subject: AI Assisted Coding

Task #1 – Syntax Error (Missing Parentheses in print)

Prompt: Use AI to detect the syntax error caused by missing parentheses in the print statement, correct it, and validate the fix using at least 3 assert test cases.

The screenshot shows the BLACKBOX AI Agent - Coding Copilot interface. On the left, there's a sidebar with icons for file, search, task, and help. The main area has a dark theme with white text. On the right, there's a code editor window titled "Extension: BLACKBOXAI Agent - Coding Copilot" containing Python code. Below the code editor is a terminal window showing command-line output.

```
# Task #1 - Syntax Error (Missing Parentheses in print)
def greet():
    return "Hello, AI Debugging Lab!"

print(greet())

# Validation with assert test cases
assert greet() == "Hello, AI Debugging Lab!"
assert len(greet()) > 0
assert "Hello" in greet()
```

Terminal output:

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/Windows/Start Menu/Programs/Python 3.9/Python.exe "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py"
Hello, AI Debugging Lab!
PS C:\Users\kotas\OneDrive\AI assist>
```

AI Explanation

Python 3 requires parentheses in `print()`. Without them, it raises a **SyntaxError**.

Code Explanation

The syntax was corrected by adding parentheses to the `print` function.

Task #2 – Incorrect If Condition (= vs ==)

Prompt

Ask AI to explain why using `=` instead of `==` causes a bug, correct the code, and verify the fix using 3 assert test cases.

The screenshot shows a code editor interface with a sidebar titled 'BLACKBOX'. A task named 'correct the given code' is selected. The code editor displays a Python file 'Lab7.1_2256.py' containing the following code:

```
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"

# Validation with assert test cases for check_number
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(8) == "Not Ten"
```

A red highlight covers the line 'if n == 10:' and a green highlight covers the line 'if n == 10:'. A tooltip above the code states: 'Fix the task by marking AI to provide who <operator> instead of <operator>'.

The terminal window shows the command 'PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps e "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py"' followed by several lines of output from the AI debugger.

AI Explanation

= is an assignment operator, not comparison. Conditions require ==.

Code Explanation

Using == ensures proper comparison instead of assignment.

Task #3 – Runtime Error (File Not Found)

Prompt

Use AI to add safe error handling using try-except, provide a user-friendly message, and validate with 3 test scenarios and assert checks.

The screenshot shows a code editor interface with a sidebar titled 'BLACKBOX'. A task named 'correct the given code' is selected. The code editor displays a Python file 'Lab7.1_2256.py' containing the following code:

```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()

# Task #3 - Runtime Error (File Not Found)
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return "Error: The file '{}' was not found. Please check the file path and try again.".format(filename)

# Validation with assert test cases for read_file
assert "Error:" in read_file("nonexistent.txt")
assert "not found" in read_file("nonexistent.txt")
assert read_file("nonexistent.txt") == "Error: The file 'nonexistent.txt' was not found. P
```

A red highlight covers the line 'with open(filename, 'r') as f:' and a green highlight covers the line 'with open(filename, 'r') as f:'. A tooltip above the code states: 'Fix the task by marking AI to provide who <operator> instead of <operator>'.

The terminal window shows the command 'PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py"' followed by several lines of output from the AI debugger.

AI Explanation

The error occurs because the program tries to open a non-existent file, so AI suggests using a try–except block to handle the exception safely.

Code Explanation

try-except prevents crashes and handles runtime errors gracefully.

Task #4 – Calling a Non-Existent Method

Prompt

Use AI to analyze the bug, either define the missing method or correct the method call, and validate using 3 assert tests.

The screenshot shows the AI assist interface in a code editor. On the left, there's a sidebar with icons for tasks, files, and a message box containing the task description. The main area shows a Python script named `Lab7.1_2256.py`. The code defines a `Car` class with a `start` method that returns "Car started". It then calls the `drive` method on an instance of `Car`, which is undefined. A tooltip on the `drive` call says "Method 'drive' not found". Below the code, a message box provides instructions: "First, the task is to use AI to analyze the bug, either define the missing method or correct the method call, and validate using 3 assert tests." The bottom right shows a terminal window with command-line history and output from running the script.

```
#Task #4 - Calling a Non-Existent Method
class Car:
    def start(self):
        return "Car started"
    + def start(self):
    +     return "Car started"
    +
    + def drive(self):
    +     return "Car is driving"

my_car = Car()
print(my_car.drive())

# Validation with assert test cases for Car class
assert my_car.start() == "Car started"
```

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py"
Error: The file 'nonexistent.txt' was not found. Please check the file path and try again.
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py"
File "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py", line 42
    def start(self):
    ^
IndentationError: expected an indented block after class definition on line 41
PS C:\Users\kotas\OneDrive\AI assist> ~C
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab7.1_2256.py"
Hello, AI Debugging Lab!
Error: The file 'nonexistent.txt' was not found. Please check the file path and try again.
Car is driving
PS C:\Users\kotas\OneDrive\AI assist>
```

AI Explanation

`drive()` does not exist, causing **AttributeError**.

Code Explanation

The missing method was defined to eliminate the **AttributeError**.

Task Description #5 (TypeError – Mixing Strings and Integers in Addition)

Prompt

Ask AI for two solutions (type casting and string concatenation), fix the bug, and validate using 3 assert test cases.

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with icons for file operations, a search bar, and a 'TASK' section containing a task titled 'Lab7.1.2256.py'. The main area displays a Python script named 'Lab7.1.2256.py'. The code is as follows:

```
Lab7.1.2256.py > ...
1 # Task Description #8 (TypeError - Mixing Strings and Integers in Addition)
2 # Solution 1: Type casting - convert string to int if necessary, then add 5
3 def add_five(value):
4     if isinstance(value, str):
5         value = int(value)
6     return value + 5
7
8 # Solution 2: String concatenation - if string, append "5"; else convert to string and app
9 def add_five_concat(value):
10    if isinstance(value, str):
11        return value + "5"
12    else:
13        return str(value) + "5"
14
15 print(add_five("10")) # Uses type casting
16 print(add_five_concat("10")) # uses string concatenation
17
18 # Validation with assert test cases for add_five (type casting)
19 assert add_five(10) == 15
20 assert add_five("10") == 15
21 assert add_five(5) == 10
22
23 # Validation with assert test cases for add_five_concat (string concatenation)
24 assert add_five_concat("10") == "105"
25 assert add_five_concat(10) == "105"
26 assert add_five_concat("Hello") == "Hello5"
27
```

The code editor highlights certain parts of the code in red and green, indicating AI suggestions or changes. A tooltip on the right side of the editor says: 'First, the task is to ask AI for two solutions (type casting and string concatenation), fix the bug, and validate using 3 assert test cases.' Below the editor, there's a message: 'Type a message (ctrl+shift+I) to add item...'. At the bottom, there's a terminal window showing a command-line interface with some errors.

AI Explanation

The error happens due to adding incompatible data types, so AI suggests type casting or string concatenation to fix it.

Code Explanation

The error occurred due to mixing data types. Fixes include converting to integer or concatenating strings.