

### LabAssignment#7.1

Course Title : **AI Assistant Coding**

Name of Student : **B. Sai Varsha**

Enrollment No. : **2303A54065**

BatchNo. : **48**

### Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs

#### 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.

#### Expected Output #1:

- Corrected code with proper syntax and AI explanation.

#### Output Screenshot:

The screenshot shows a code editor interface with several tabs open. The active tab is 'task1.py' which contains the following code:

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

The code has syntax errors: a missing closing parenthesis in the first print statement and a missing closing parenthesis in the greet() call. Below the code editor, the terminal window shows the output of running the script:

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "C:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task1.py"
KeyboardInterrupt
Hello, AI Debugging Lab!
Task 1 tests passed!
```

The terminal also shows the current working directory as 'C:\Users\Lenovo\Desktop\AI Coding' and the Python environment as '.venv'. The status bar at the bottom indicates the line number (Ln 13), column (Col 1), spaces (Spaces: 4), encoding (UTF-8), and file type (Python).

**Explanation:** In Python 3, `print` is defined as a built-in function. A function call requires parentheses. Using the Python 2 `print` statement format violates Python 3 syntax rules, resulting in a `SyntaxError`. The correction is to use the functional form of `print()`.

### 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.

### Output Screenshot:

The screenshot shows a code editor interface with several files listed in the Explorer sidebar. The current file is `task2.py`, which contains the following code:

```
def check_number(n):  
    if n == 10:  
        return "Ten"  
    else:  
        return "Not Ten"  
  
# Assert Test Cases  
assert check_number(10) == "Ten"  
assert check_number(5) == "Not Ten"  
assert check_number(-10) == "Not Ten"  
  
print("Task 2 tests passed!")
```

In the bottom right corner, there is a terminal window showing command-line output:

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task1.py"  
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task2.py"  
Task 2 tests passed!
```

### Explanation:

The operator `=` is an assignment operator used to store a value in a variable. Conditional statements require a boolean expression, which is formed using comparison operators such as `==`. Using `=` in an if-condition is syntactically invalid in Python and produces a `SyntaxError`. The correction is to replace `=` with `==`.

### 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, invalidpath.

## **Expected Output #3:**

- Safe file handling with exception management.

## **Output Screenshot:**

The screenshot shows a VS Code interface with the following details:

- File Explorer:** On the left, it lists several Python files under the "ass5" folder, including "task1.py", "task2.py", and "task3.py".
- Terminal:** At the bottom, the terminal window shows the command "vo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task3.py" and the message "All tests passed!".
- Code Editor:** The main area displays the code for "task3.py". The code defines a function "read\_file" that reads a file and handles various exceptions like FileNotFoundError and OSError. It also includes an assert block to verify the function's correctness.

**EXPLANATION :** File operations depend on the existence and validity of the file path. When `open()` is executed with a missing file, Python raises a `File Not Found Error` at runtime. Exception handling using `try-except` prevents abrupt termination and enables controlled execution by returning a meaningful error message.

## Task Description #4 (Calling a Non-Existant 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

```
g. Calling an uninitialised 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.

## Expected Output #4:

- Corrected class with clear AI explanation.

## Output Screenshot:

```

File Edit Selection View Go Run ... < > Q AI Coding task1.py ...assignment 7.1.py task2.py ...assignment 7.1.py task3.py ...assignment 7.1.py task4.py ...assignment 7.1.py
EXPLORER
> OPEN EDITORS
AI CODING
> .venv
ass5.py
  > assignment 7.1.py
    task1.py
    task2.py
    task3.py
    task4.py
    .env
    ass_5 task1.py
    ass_5 task2.py
    ass_5 task3.py
    ass_5 task4.py
    ass_5 task5.py
    ass4.3 task1.py
    ass4.3 task2.py
    ass4.3 task3.py
    ass4.3 task4(4.1).py
    ass4.3 task4(4.2).py
    ass4.3 task5.py
    ass6.3 task1.py
    ass6.3 task2.py
    ass6.3 task3.py
    ass6.3 task4.py
    ass6.3 task5.py
> OUTLINE
> TIMELINE

```

```

1 class Car:
2     def start(self):
3         return "Car started"
4
5     def drive(self):
6         return "Car is driving"
7
8 # Object
9 my_car = Car()
10
11 # Output
12 print(my_car.drive())
13
14 # Assert Test Cases
15 assert my_car.start() == "Car started"
16 assert my_car.drive() == "Car is driving"
17 assert isinstance(my_car.drive(), str)
18
19 print("Task 4 tests passed!")
20

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

● (.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py" assignment 7.1.py/task4.py
Car is driving
Task 4 tests passed!
● (.venv) PS C:\Users\Lenovo\Desktop\AI Coding>

```

Ln 20, Col 1 Spaces: 4 UTF-8 CRLF () Python .venv (3.12.10)

**Explanation:** In object-oriented programming, a method must be defined within a class before it can be invoked by an object of that class. Calling an undefined method results in an `AttributeError` because the object does not contain the requested attribute. The correction requires either defining the missing method in the class or modifying the call to an existing method.

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

**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.

### Expected Output #5:

- Corrected code that runs successfully for multiple inputs.

## Output Screenshot:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure under "AI CODING". The "ass5.py" folder contains files: assignment 7.1.py, task1.py, task2.py, task3.py, task4.py, and task5.py. There are also several ".env" and "ass\_x\_taskN.py" files in the root directory.
- Code Editor (Top Right):** The active file is "task5.py" which contains the following code:

```
def add_five_cast(value):
    return int(value) + 5

# Assert Test Cases
assert add_five_cast("10") == 15
assert add_five_cast(0) == 5
assert add_five_cast("25") == 30

print("Task 5 (casting) tests passed!")
```
- Terminal (Bottom):** The terminal shows the command being run and its output:

```
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding> & "C:\Users\Lenovo\Desktop\AI Coding\.venv\Scripts\python.exe" "c:/Users/Lenovo/Desktop/AI Coding/ass5.py/assignment 7.1.py/task5.py"
Task 5 (casting) tests passed!
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

**EXPLANATION :** Python enforces strict type rules for arithmetic operations. Addition between a string and an integer is not supported because the operands are of incompatible types. This produces a `TypeError`. The correction is performed by explicit type conversion, either converting the string to an integer for numeric addition or converting the integer to a string for concatenation.