

Assingment-7.1

Name: Nandini Reddy

Hall Ticket No:2303A51896

Batch-08

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.

The screenshot shows a VS Code interface. On the left, there's a code editor with a file named 'AIAC.py'. The code contains a function 'greet()' that prints a string. There is a missing closing parenthesis on line 3. The terminal tab at the bottom shows the command 'python AIAC.py' being run, which results in an 'IndentationError' because the print statement is not properly indented under the function definition. The terminal also shows the corrected code where the print statement is properly indented.

```
AIAC.py 2 ×
AIAC.py > ...
1 # Bug: Missing parentheses in print statement
2 def greet():
3     print "Hello, AI Debugging Lab!"
4     greet()
5
6

PROBLEMS 2 TERMINAL OUTPUT DEBUG CONSOLE PORTS POSTMAN CONSOLE Python Debug Console
PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:/Users/yaraV/.vscode/extensions/ms-python.python-2025.18.0-win32-x64/bundled\libs\debugpy\launcher' '52029' '--'
y'
File "d:\python_dsa\AIAC.py", line 3
    print "Hello, AI Debugging Lab!"
    ^^^^
IndentationError: expected an indented block after function definition on line 2
PS D:\python_dsa>
```

Corrected Code

A screenshot of the Visual Studio Code interface. The left pane shows a Python file named 'AIAC.py' with the following code:

```
1 # Bug: Missing parentheses in print statement
2 def greet():
3     print("Hello, AI Debugging Lab!")
4 greet()
```

The terminal at the bottom shows the following output:

```
PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53869' '--' 'd:\python_dsa\AIAC.py'
Hello, AI Debugging Lab!
```

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.

A screenshot of the Visual Studio Code interface. The left pane shows a Python file named 'AIAC.py' with the following code:

```
1 # Bug: Using assignment (=) instead of comparison (==)
2 def check_number(n):
3     if n = 10:
4         return "Ten"
5     else:
6         return "Not Ten"
```

The terminal at the bottom shows the following output:

```
PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53869' '--' 'd:\python_dsa\AIAC.py'
Hello, AI Debugging Lab!
PS D:\python_dsa> ^C
PS D:\python_dsa> d;; cd 'd:\python_dsa'; & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\la
File "d:\python_dsa\AIAC.py", line 3
  if n = 10:
      ^^^^^^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
PS D:\python_dsa>
```

Corrected Code

The screenshot shows the VS Code interface with the Python extension installed. The code editor has a file named 'AIAC.py' open, containing the following code:

```
# Fixed Version: Using comparison operator (==)
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"

# Test Cases
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(-10) == "Not Ten"

print("All tests passed successfully!")
```

The terminal below shows the command being run and the resulting output:

```
PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57860'
y
All tests passed successfully!
PS D:\python_dsa>
```

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.

The screenshot shows the VS Code interface with the Python extension installed. The code editor has a file named 'AIAC.py' open, containing the following code:

```
# 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"))
```

The terminal below shows the command being run and the resulting output, which includes a stack trace for a FileNotFoundError:

```
All tests passed successfully!
PS D:\python_dsa> ^C
PS D:\python_dsa> d:; cd 'd:\python_dsa'; & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57860'
PS D:\python_dsa> AIAC.py
Traceback (most recent call last):
  File "d:\python_dsa\AIAC.py", line 6, in <module>
    print(read_file("nonexistent.txt"))
               ~~~~~
  File "d:\python_dsa\AIAC.py", line 3, in read_file
    with open(filename, 'r') as f:
               ~~~~~
FileNotFoundError: [Errno 2] No such file or directory: 'nonexistent.txt'
```

Safe Version with Exception Handling

The screenshot shows the VS Code interface with the AIAC.py file open. The code implements exception handling for reading files. It includes test cases for existing files, missing files, and invalid paths. The terminal at the bottom shows the command run and the successful execution of all test cases.

```
AIAC.py
1 def read_file(filename):
2     try:
3         with open(filename, 'r') as f:
4             return f.read()
5     except FileNotFoundError:
6         return "Error: File not found."
7     except OSError:
8         return "Error: Invalid file path or unable to access file."
9
10 # -----
11 # Test Scenarios
12 # -----
13
14 # 1 File Exists
15 with open("testfile.txt", "w") as f:
16     f.write("Sample content")
17
18 assert read_file("testfile.txt") == "Sample content"
19
20 # 2 File Missing
21 assert read_file("nonexistent.txt") == "Error: File not found."
22
23 # 3 Invalid Path
24 assert read_file("invalid:/path/test.txt") == "Error: Invalid file path or unable to access file."
25
26 print("All test cases passed successfully!")
27
```

TERMINAL OUTPUT DEBUG CONSOLE PORTS POSTMAN CONSOLE Python Debug Console

```
PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:\Users\yarav\vscode\extensions\ms-python\debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '49701' '--' 'd:\python_dsa\AIAC.py'
y
All test cases passed successfully!
PS D:\python_dsa>
```

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.

The screenshot shows the VS Code interface with the AIAC.py file open. The code defines a Car class with a start() method but calls drive() on an instance. The terminal shows the resulting AttributeError.

```
AIAC.py
1 # Bug: Calling an undefined method
2 class Car:
3     def start(self):
4         return "Car started"
5
6 my_car = Car()
7 print(my_car.drive()) # drive() is not defined
8
```

TERMINAL OUTPUT DEBUG CONSOLE PORTS POSTMAN CONSOLE Python Debug Console

```
y
Traceback (most recent call last):
File "d:\python_dsa\AIAC.py", line 7, in <module>
    print(my_car.drive()) # drive() is not defined
          ^^^^^^^^^^
AttributeError: 'Car' object has no attribute 'drive'
PS D:\python_dsa>
```

Corrected Code

A screenshot of the Visual Studio Code interface. The left pane shows a Python file named AIAC.py with the following code:

```
6     return "Car is driving"
7
8
9 # Create object
10 my_car = Car()
11
12 # Test calls
13 print(my_car.drive())
14
15
16 # Assert Test Cases
17 assert my_car.start() == "car started"
18 assert my_car.drive() == "Car is driving"
19 assert isinstance(my_car, Car)
20
21 print("All tests passed successfully!")
22
23
24
25
26
27
```

The right pane shows the Python Debug Console with the following output:

```
PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:\Users\yarav\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50613' '--' 'd:\python_dsa\AIAC.py'
y
Car is driving
All tests passed successfully!
PS D:\python_dsa>
```

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

A screenshot of the Visual Studio Code interface. The left pane shows a Python file named AIAC.py with the following code:

```
1 # Bug: TypeError due to mixing string and integer
2 def add_five(value):
3     return value + 5
4
5 print(add_five("10"))
6
```

The right pane shows the Python Debug Console with the following output, indicating a TypeError:

```
PS D:\python_dsa> ^C
PS D:\python_dsa> d; cd 'd:\python_dsa'; & 'c:\Python314\python.exe' 'c:\Users\yarav\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62802' '--' 'd:\python_dsa\AIAC.py'
Traceback (most recent call last):
  File "d:\python_dsa\AIAC.py", line 5, in <module>
    print(add_five("10"))
           ~~~~~~^~~~~~
  File "d:\python_dsa\AIAC.py", line 3, in add_five
    return value + 5
           ~~~~~~^~~~~~
TypeError: can only concatenate str (not "int") to str
PS D:\python_dsa>
```

Solution 1: Type Casting (Convert to Integer)

A screenshot of the Visual Studio Code interface. The left pane shows the code editor with a file named 'AIAC.py'. The code contains a function 'add_five' that adds 5 to an integer input. It includes three test cases using assert statements. The right pane shows the terminal window with the output of running the script, which shows that all test cases passed.

```
AIAC.py > ...
1 # Solution 1: Type Casting
2 def add_five(value):
3     return int(value) + 5
4
5
6 # Test Cases
7 assert add_five("10") == 15
8 assert add_five(20) == 25
9 assert add_five("0") == 5
10
11 print("Solution 1 tests passed!")
12 |
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS POSTMAN CONSOLE

PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:/Users/yarav/.vscode/extensions/ms-python.debugpy-2025.18.0-win32-x64/bundled/libs\debugpy\launcher' '59738' '--' 'd:\pyt
y'
Solution 1 tests passed!
PS D:\python_dsa>

Solution 2: String Concatenation

A screenshot of the Visual Studio Code interface. The left pane shows the code editor with a file named 'AIAC.py'. The code contains a function 'add_five' that concatenates '5' to the end of a string input. It includes three test cases using assert statements. The right pane shows the terminal window with the output of running the script, which shows that all test cases passed.

```
AIAC.py > ...
1 # Solution 2: String Concatenation
2 def add_five(value):
3     return str(value) + "5"
4
5
6 # Test Cases
7 assert add_five("10") == "105"
8 assert add_five(20) == "205"
9 assert add_five("") == "5"
10
11 print("Solution 2 tests passed!")
12 |
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS POSTMAN CONSOLE

PS D:\python_dsa> & 'c:\Python314\python.exe' 'c:/Users/yarav/.vscode/extensions/ms-python.debugpy-2025.18.0-win32-x64/bundled/libs\debugpy\launcher' '63944' '--' 'd:\pyt
y'
Solution 2 tests passed!
PS D:\python_dsa>