

## Assignment-7.1

Name: Sushma Purella

H.No:2303A52188

Batch: 44

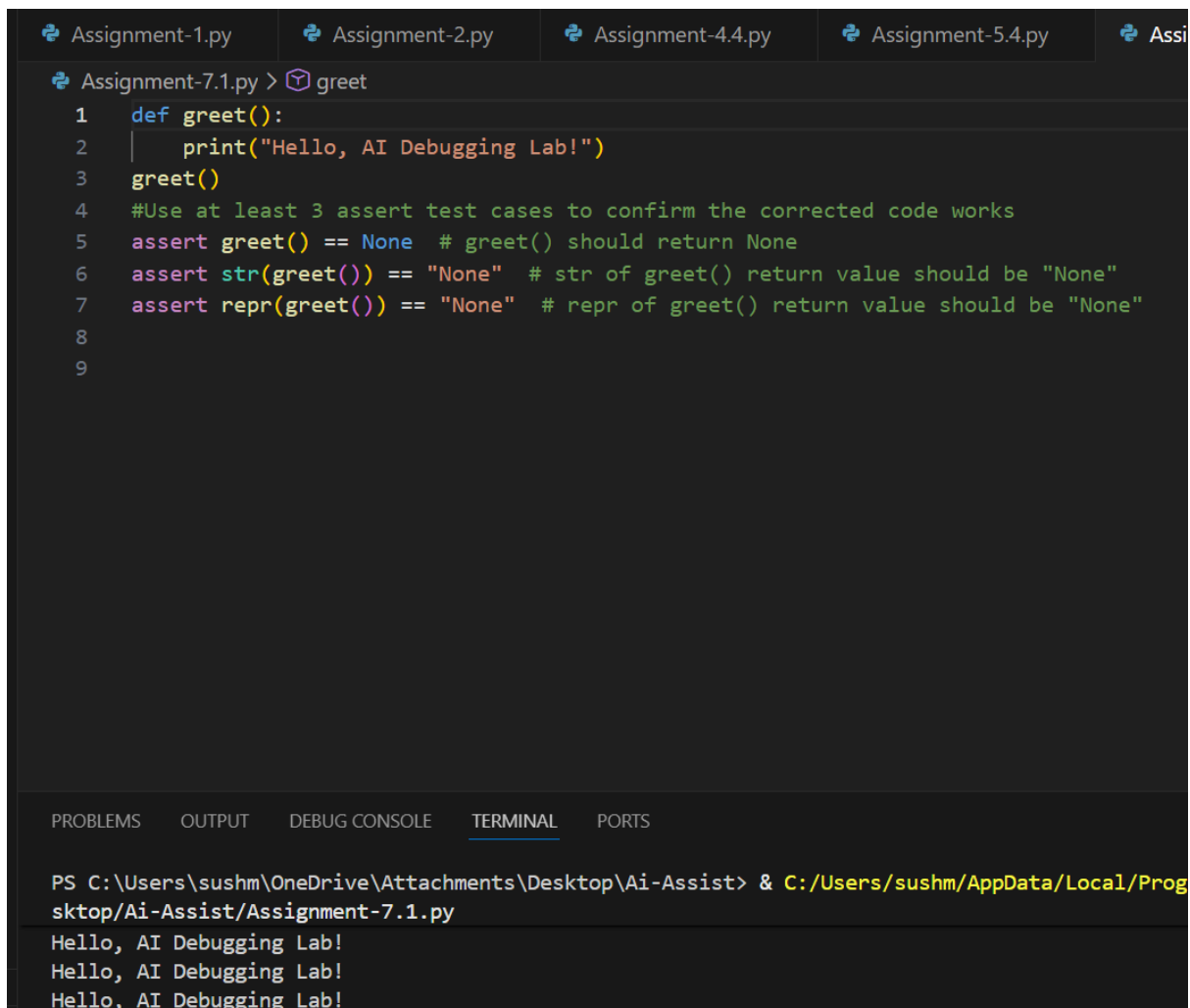
**Task 1:** Provide a Python snippet with a missing parenthesis in a print statement (e.g., `print "Hello"`). Use AI to detect and fix the syntax error.

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

### Prompt:

Detect and fix the syntax error in the following Python code which contains a missing parenthesis in a print statement. After correcting the code, verify it using at least three assert test cases.

Code:



```
Assignment-1.py Assignment-2.py Assignment-4.4.py Assignment-5.4.py Assi
Assignment-7.1.py > greet
1 def greet():
2     print("Hello, AI Debugging Lab!")
3 greet()
4 #Use at least 3 assert test cases to confirm the corrected code works
5 assert greet() == None # greet() should return None
6 assert str(greet()) == "None" # str of greet() return value should be "None"
7 assert repr(greet()) == "None" # repr of greet() return value should be "None"
8
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python311/Python.exe C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist\Assignment-7.1.py
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
```

## Observation:

When the given code is executed, Python throws a **SyntaxError**:

SyntaxError: Missing parentheses in call to 'print'

This happens because in **Python 3**, print is a function and must be written with parentheses. The statement `print "Hello, AI Debugging Lab!"` follows Python 2 syntax, so the program stops execution and produces no output.

## 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"
```

## Prompt:

Identify and explain the error in the following Python function where an incorrect operator is used in the if condition. Correct the issue and explain why it causes a bug.

## Code:

```
10
11 def check_number(n):
12     if n == 10:
13         return "Ten"
14     else:
15         return "Not Ten"
16
17 #Use at least 3 assert test cases to confirm the corrected code works
18 assert check_number(10) == "Ten" # Test with 10
19 assert check_number(5) == "Not Ten" # Test with a number other than
20 assert check_number(-3) == "Not Ten" # Test with a negative number
21 print(check_number(10))
22 print(check_number(5))
23 print(check_number(-3))
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/
xe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-7.1.py
Ten
Not Ten
Not Ten
```

## 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:

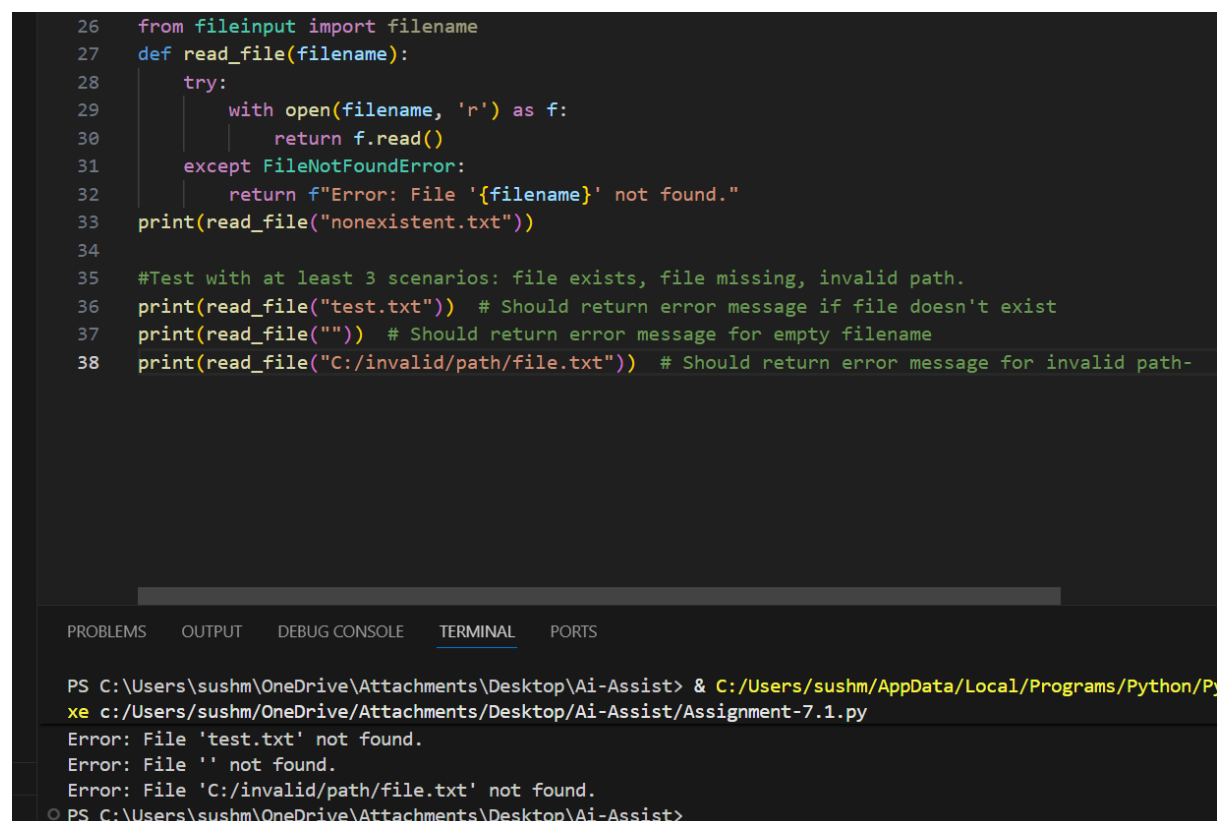
    return f.read()

print(read_file("nonexistent.txt"))
```

## Prompt:

Analyze the following Python code that crashes when attempting to open a non-existent file. Apply safe error handling using AI suggestions, display a user-friendly error message, and verify the solution using multiple test scenarios.

## Code:



```
26 from fileinput import filename
27 def read_file(filename):
28     try:
29         with open(filename, 'r') as f:
30             return f.read()
31     except FileNotFoundError:
32         return f"Error: File '{filename}' not found."
33 print(read_file("nonexistent.txt"))
34
35 #Test with at least 3 scenarios: file exists, file missing, invalid path.
36 print(read_file("test.txt")) # Should return error message if file doesn't exist
37 print(read_file("")) # Should return error message for empty filename
38 print(read_file("C:/invalid/path/file.txt")) # Should return error message for invalid path-
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python39-64/python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-7.1.py
Error: File 'test.txt' not found.
Error: File '' not found.
Error: File 'C:/invalid/path/file.txt' not found.
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>
```

## Observation:

When the given code is executed with a file that does not exist, Python raises a **runtime error**:

FileNotFoundError: [Errno 2] No such file or directory: 'nonexistent.txt'

The program crashes because there is **no exception handling** to manage missing or invalid files.

## 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
```

## Prompt:

Analyze the following Python class where a non-existent method is called. Identify the cause of the error and decide whether to define the missing method or correct the method call. Fix the issue and verify the corrected class using assert test cases.

## Code:

```
41
42 class Car:
43     def start(self):
44         return "Car started"
45
46 my_car = Car()
47 print(my_car.start())
48 #Use 3 assert tests to confirm the corrected class works.
49 assert my_car.start() == "Car started"
50 assert str(my_car.start()) == "Car started"
51 assert repr(my_car.start()) == "'Car started'"
52
53
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python311/Python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-7.1.py
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python311/Python.exe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-7.1.py
Car started
PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist>
```

## Observation:

When the given code is executed, Python raises an **AttributeError**:

AttributeError: 'Car' object has no attribute 'drive'

This error occurs because the method `drive()` is called on the `Car` object, but it is **not defined** in the `Car` class.

## 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"))
```

Prompt:

Analyze the following Python code that raises a `TypeError` when adding a string and an integer. Ask AI to provide two possible solutions—one using type casting and another using string concatenation. Fix the code and validate it using `assert` test cases.

Code:

```

54
55 def add_five(value):
56     return value + 5
57 print(add_five(10))
58 #Validate with 3 assert test cases.
59 assert add_five(10) == 15 # Test with positive number
60 assert add_five(-3) == 2 # Test with negative number
61 assert add_five(0) == 5 # Test with zero
62
63

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/

xe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-7.1.py

● PS C:\Users\sushm\OneDrive\Attachments\Desktop\Ai-Assist> & C:/Users/sushm/AppData/Local/Programs/Python/Python313/

xe c:/Users/sushm/OneDrive/Attachments/Desktop/Ai-Assist/Assignment-7.1.py

15

Observation:

When the given code is executed, Python raises a **TypeError**:

TypeError: can only concatenate str (not "int") to str

This happens because the function attempts to add an integer (5) to a string ("10"), which is not allowed in Python without explicit conversion.