

SR-UNIVERSITY

ASSIGNMENT-7.3

HTNO:2303A51477

B.NO:10

Task 1: Fixing Syntax Errors

Scenario

You are reviewing a Python program where a basic function definition contains a syntax error.

```
python

def add(a, b)
    return a + b
```

Requirements

- Provide a Python function `add(a, b)` with a missing colon
- Use an AI tool to detect the syntax error
- Allow AI to correct the function definition
- Observe how AI explains the syntax issue

Expected Output

- Corrected function with proper syntax
- Syntax error resolved successfully
- AI-generated explanation of the fix

Prompting

I am reviewing a Python program and found a syntax error in the following function:

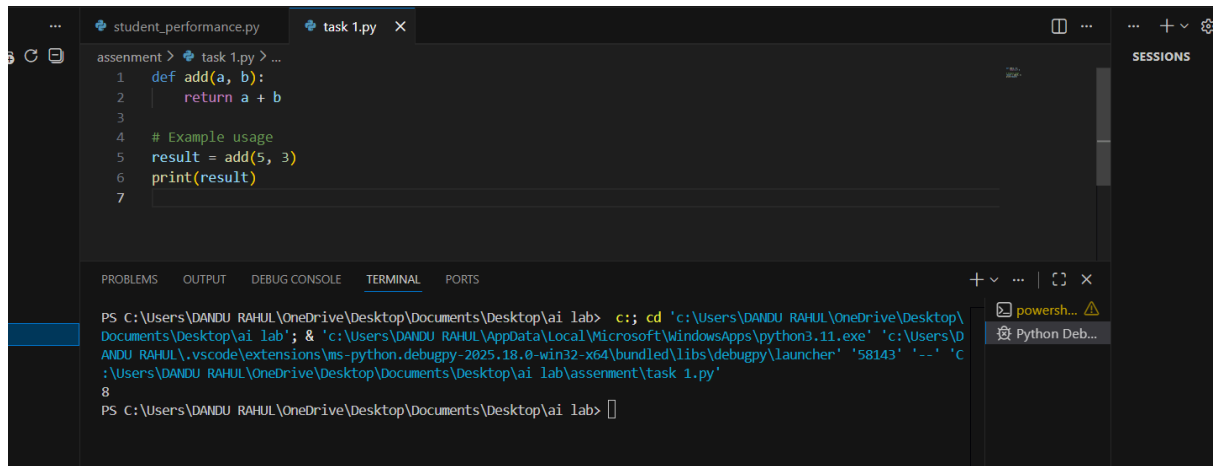
```
def add(a, b)
    return a + b
```

1. Detect the syntax error in this code.
2. Explain what the issue is.
3. Correct the function definition.
4. Show the corrected version of the code.

Code:

```
def add(a, b):
    return a + b

result = add(5, 3)
print(result)
```



```
student_performance.py task 1.py X
assenment > task 1.py > ...
1 def add(a, b):
2     return a + b
3
4 # Example usage
5 result = add(5, 3)
6 print(result)
7
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> c::; cd 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab'; & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab\assenment\task 1.py'
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

analysis :

The error happened because the function was missing a colon `:`.

Python needs a colon after `def add(a, b)` to start the function body.

Adding the colon fixes the syntax error.

Task 2: Debugging Logic Errors in Loops

Scenario

You are debugging a loop that runs infinitely due to a logical mistake.



```
python
def count_down(n):
    while n >= 0:
        print(n)
        n += 1 # Should be n -= 1
```

Requirements

- Provide a loop with an increment or decrement error
- Use AI to identify the cause of infinite iteration
- Let AI fix the loop logic
- Analyze the corrected loop behavior

Expected Output

- Infinite loop issue resolved
- Correct increment/decrement logic applied
- AI explanation of the logic error

Prompt :

I am debugging a Python loop that seems to run infinitely.

Here is the code:

```
i = 1  
  
while i <= 5:  
  
    print(i)  
  
    i -= 1
```

1. Identify why this loop runs infinitely.
2. Explain the logical mistake.
3. Correct the loop logic.
4. Show the corrected version of the code.

Code:

```
i = 1  
  
while i <= 5:  
  
    print(i)  
  
    i += 1
```

The screenshot shows a Python IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor contains a Python script named `task2.py` with the following code:

```
1 i = 1
2
3 while i <= 5:
4     print(i)
5     i -= 1
6
```

The terminal at the bottom shows the command prompt running the script, and the output shows the numbers 1 through 5 being printed, indicating the loop is running.

analysis :

The loop runs forever because `i` is decreasing instead of increasing. Since `i` keeps getting smaller, the condition `i <= 5` is always true. Changing it to `i += 1` stops the loop correctly.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

A Python function crashes during execution due to a division by zero error.

Requirements

```
# Debug the following code
def divide(a, b):
    return a / b

print(divide(10, 0))
```

- Provide a function that performs division without validation
- Use AI to identify the runtime error
- Let AI add try-except blocks for safe execution
- Review AI's error-handling approach

Expected Output

- Function executes safely without crashing
- Division by zero handled using try-except
- Clear AI-generated explanation of runtime error handling

Prompt:

I have a Python function that performs division but it crashes when dividing by zero.

Here is the code:

```
def divide(a, b):
```

```
    return a / b
```

```
print(divide(10, 0))
```

1. Identify the runtime error in this code.
2. Explain why the error occurs.
3. Modify the function using try-except to handle the error safely.
4. Show the corrected code.
5. Explain how the error handling works.

Code:

```
def divide(a, b):  
  
    try:  
  
        return a / b  
  
    except ZeroDivisionError:  
  
        return "Error: Cannot divide by zero"  
  
print(divide(10, 0))
```



```
task3.py
1 def divide(a, b):
2     try:
3         return a / b
4     except ZeroDivisionError:
5         return "Error: Cannot divide by zero"
6
7 print(divide(10, 0))
8
```

```
PS C:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> & 'c:\Users\DANDU\RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU\RAHUL\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55487'
'...' 'c:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab\assessment\task2.py'
1
2
3
4
5
PS C:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> ^C
PS C:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
PS C:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> c;; cd 'c:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab'; & 'c:\Users\DANDU\RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU\RAHUL\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58380' '...' 'c:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab\assessment\task3.py'
Error: Cannot divide by zero
PS C:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
```

analysis :

The program crashes because we are dividing by zero.

Python does not allow division by zero, so it shows a **ZeroDivisionError**.

Using **try-except** prevents the crash and handles the error safely.

Task 4: Debugging Class Definition Errors

Scenario

You are given a faulty Python class where the constructor is incorrectly defined.

You are given a faulty Python class where

python

```
class Rectangle:
    def __init__(length, width):
        self.length = length
        self.width = width
```

Requirements

- Provide a class definition with missing self-parameter
- Use AI to identify the issue in the __init__() method
- Allow AI to correct the class definition
- Understand why self is required

Expected Output

- Corrected __init__() method
- Proper use of self in class definition
- AI explanation of object-oriented error

I am reviewing a Python class that is not working correctly.

Prompt:

The following Python class is not working properly:

```
class Student:
```

```
    def __init__(name, age):
```

```
        name = name
```

```
        age = age
```

Find the error in the __init__() method, fix it, and explain why 'self' is needed.

Code:

```
class Student:

    def __init__(self, name, age):

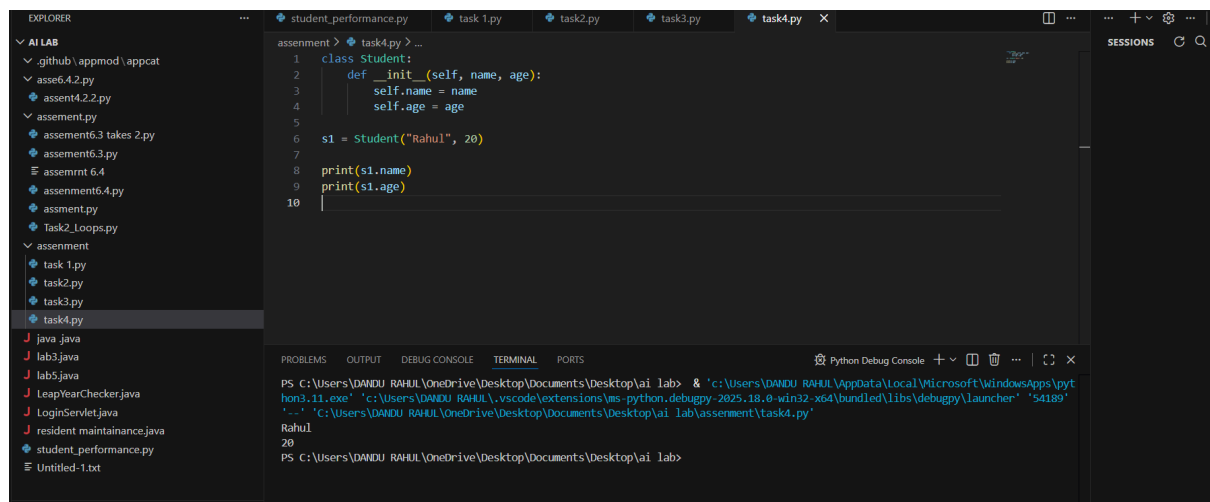
        self.name = name

        self.age = age

s1 = Student("Rahul", 20)

print(s1.name)

print(s1.age)
```



analysis:

The error is because **self** is missing in the constructor.

self is needed to refer to the current object.

Without **self**, the object cannot store its values correctly.

Task 5: Resolving Index Errors in Lists

```
python

numbers = [1, 2, 3]
print(numbers[5])
```

Scenario

A program crashes when accessing an invalid index in a list.

Requirements

- Provide code that accesses an out-of-range list index
- Use AI to identify the Index Error
- Let AI suggest safe access methods
- Apply bounds checking or exception handling

Expected Output

- Index error resolved
- Safe list access logic implemented

Prompt:

I have a Python program that crashes when accessing a list element.

Here is the code:

```
numbers = [10, 20, 30]

print(numbers[5])
```

1. Identify the error in this code.
2. Explain why it occurs.
3. Fix the code using bounds checking or try-except.
4. Show the corrected version.

Code:

```
numbers = [10, 20, 30]
```

```

index = 5

if index < len(numbers):

    print(numbers[index])

else:

    print("Error: Index out of range")

```

The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'AI LAB' with various files. The code editor displays the following Python code:

```

1 numbers = [10, 20, 30]
2 index = 5
3
4 if index < len(numbers):
5     print(numbers[index])
6 else:
7     print("Error: Index out of range")
8

```

The terminal at the bottom shows the command prompt output:

```

PS C:\Users\DAVIDU\OneDrive\Desktop\ai_lab> & 'c:\Users\DAVIDU\OneDrive\Desktop\ai_lab\assessment\task4.py'
Rahul
20
PS C:\Users\DAVIDU\OneDrive\Desktop\ai_lab> ^C
PS C:\Users\DAVIDU\OneDrive\Desktop\ai_lab>
PS C:\Users\DAVIDU\OneDrive\Desktop\ai_lab> cd 'c:\Users\DAVIDU\OneDrive\Desktop\ai_lab'; & 'c:\Users\DAVIDU\OneDrive\Desktop\ai_lab\assessment\task5.py'
Error: Index out of range
PS C:\Users\DAVIDU\OneDrive\Desktop\ai_lab>

```

Analysis:

The error occurs because the program is trying to access an index that does not exist in the list.

The list has only 3 elements, but index 5 is requested.

Using bounds checking or **try-except** prevents the program from crashing

