

Assigmnet-7

K.SAI ROHAN

2303A52025

Batch-38

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

Corrected code:

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

Task 2: Debugging Logic Errors in Loops

Scenario

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

```
def count_down(n):  
    while n>=0:  
        print(n)  
        n+=1
```

Requirements

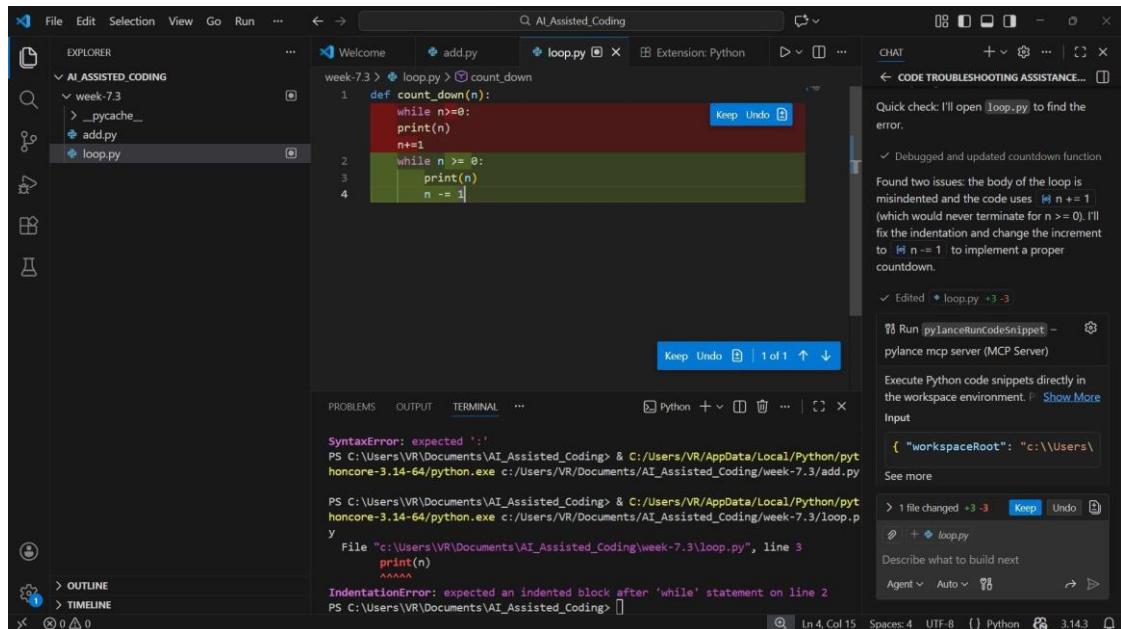
- Provide a loop with an increment or decrement error
- Use AI to identify the cause of infinite iteration
- Let AI fix the loop logic
- Analyze the corrected loop behavior

Expected Output

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

Corrected code:

```
def count_down(n):  
    while n >= 0:  
        print(n)  
        n -= 1
```



Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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

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

```
    return a/b
```

```
print(divide(10/0))
```

Requirements

- 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

Corrected Code:

```

def divide(a,
          b): if b ==
          0:
            raise ValueError("Cannot divide by zero")
        return a / b
if __name__ == "__

```

```

# demonstrate error handling for division by
zero try:
    print(divide(10, 0))
except ValueError as e:
    print("Error:", e)

```

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the workspace: 'divide.py' is the active file.
- Code Editor:** Displays the code above. A tooltip on the right says: "Fixed division functionality and crafted new content".
- Terminal:** Shows command-line output related to the code execution.
- Right Panel:** Includes sections for 'CHAT', 'CODE TROUBLESHOOTING ASSISTANCE...', 'PROBLEMS', and 'OUTPUT'.

Task 4: Debugging Class Definition Errors

Scenario

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

class Rectangle:

```

def __init__(length,width):
    self.length=length
    self.width=width

```

Requirements

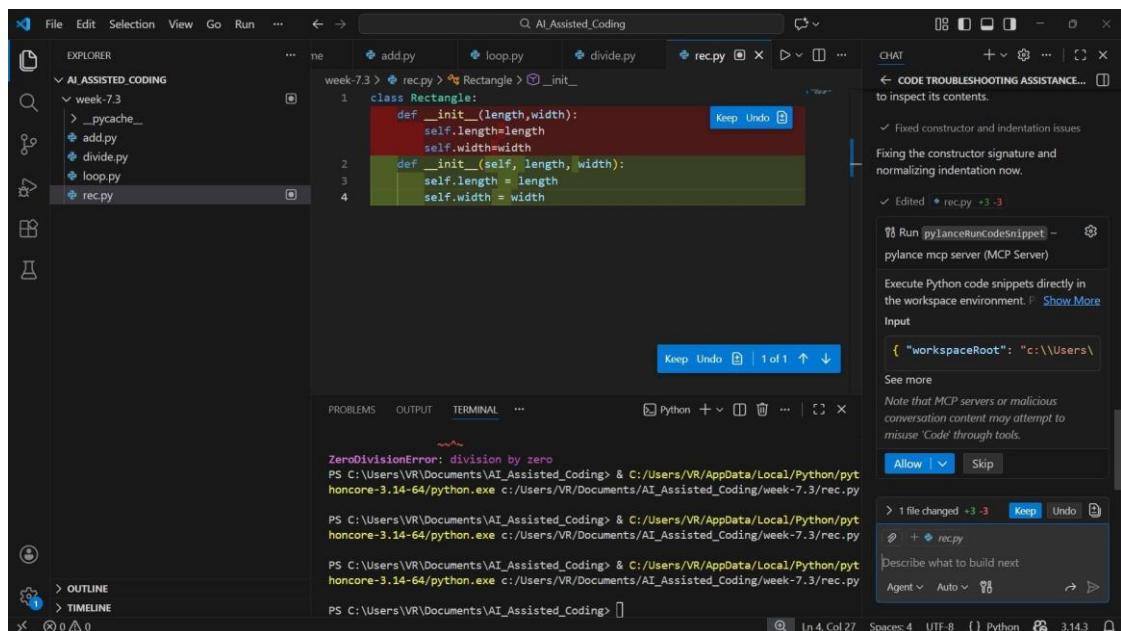
- Provide a class definition with missing self-parameter
- Use AI to identify the issue in the `__init__()` method
- Allow AI to correct the class definition
- Understand why self is required Expected Output

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

```
class Rectangle:
    def _____ init_(self,
                    length, width): self.length
```

oriented error Corrected code:

```
= length self.width = width
```



Task 5: Resolving Index Errors in Lists

Scenario

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

Requirements

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

- 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
- AI suggestion using length checks or exception handling Corrected code:

```

numbers = [1, 2, 3]

def get_number(nums, idx):
    try:
        return nums[idx]
    except IndexError:
        raise IndexError(f"Index {idx} out of

```

```

if __name__ == "__main__":
    print(get_number(numbers, 2)) # outputs: 3
    try:
        print(get_number(numbers,
5)) except IndexError as e:

```

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

- Code Editor:** Displays the Python script with the error at line 16: `print("Error:", e)`.
- Terminal:** Shows the command `honcore-3.14-64/python.exe` being run, followed by the error message: `IndexError: list index out of range`.
- AI Assistant Panel:** A sidebar titled "CODE TROUBLESHOOTING ASSISTANCE..." provides a suggestion: "I'll add safe index access and a small test to prevent an IndexError and show expected output." It also shows a snippet of code: `# outputs: 3` and `print("Error:", e)`.
- Bottom Status Bar:** Shows the file path `C:\Users\VR\Documents\AI_Assisted_Coding\week-7.3\num.py`, line 17, column 1, and other status information like "Spaces: 4" and "UTF-8".