

2303A51504

BATCH-25

ASSIGNMENT-7.3

Task 1: Fixing Syntax Errors

Scenario

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

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

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has icons for Explorer, Search, and Timeline. The main area shows a code editor with the file 'ass 7.3.py' open, containing the following code:

```
def add(a, b):
    return a + b
print(add(3, 5))
```

Below the code editor is a terminal window showing the output of running the script:

```
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 7.3.py"
8
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
```

The status bar at the bottom indicates 'Ln 5, Col 1' and 'Spaces: 4'. The taskbar at the bottom of the screen shows various application icons.

Task 2: Debugging Logic Errors in Loops

Scenario

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

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

The screenshot shows a dark-themed instance of Visual Studio Code. In the center-left, the code editor displays a file named 'ass 7.3.py' containing the following code:

```
6
7     #Buggy loop (infinite loop due to missing increment)
8     #i = 1
9     #while i <= 5:
10    #print(i)
11    # i is never incremented + infinite loop"""
12
13
14    # Fixed loop (logic corrected)
15    i = 1
16    while i <= 5:
17        print(i)
18        i += 1
19    # increment added to stop infinite iteration
20
21
```

Below the code editor is the terminal pane, which shows the output of running the script:

```
>> 2
>> 3
>> 4
>> 5
>>
1
2
3
4
5
```

The status bar at the bottom indicates the path 'PS C:\Users\parva\OneDrive\Desktop\AI Assted>' and the terminal settings: 'Ln 10, Col 10', 'Spaces: 4', 'UTF-8', 'CRLF', and '() Py'.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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

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

The screenshot shows the Visual Studio Code interface. The left sidebar has icons for Explorer, Search, Outline, and Timeline. The main editor window displays a Python file named 'ass 7.3.py'. The code contains two functions: a buggy one that divides by zero, and a safe one using try-except. The terminal below shows the output of running the script, which catches the ZeroDivisionError and prints an error message. The status bar at the bottom indicates the file is saved.

```
File Edit Selection View Go Run Terminal Help < > AI Assted
```

```
EXPLORER ... ass 7.3.py X ass 7.3.py > ...
```

```
20
21 # Buggy function (no validation → runtime error)
22 def divide(a, b):
23     return a / b
24
25 # This will crash if b = 0
26 # print(divide(10, 0))
27
28
29 # Fixed function using try-except (safe execution)
30 def safe_divide(a, b):
31     try:
32         return a / b
33     except ZeroDivisionError:
34         return "Error: Division by zero is not allowed"
35
36
37 # Testing
38 print(safe_divide(10, 2))
39 print(safe_divide(10, 0))
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Users\parva\OneDrive\Desktop\AI Assted> 5.0
>> Error: Division by zero is not allowed
>> |
```

```
Ln 40, Col 1 Spaces: 4 UTF-8 CRLF {} Python
```

```
29°C Sunny
```

```
Search
```

```
CHAT ID: AI Assted
```

```
Ident follow Explain com
```

```
Y n F a e G k
```

```
+ ... |
```

```
Python Python powershell powershell powershell powershell powershell powershell powershell powershell Agent
```

Task 4: Debugging Class Definition Errors

Scenario

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

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

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

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** AI Assisted.
- Explorer:** Shows 'AI ASSISTED' as the active item under 'EXPLORER'.
- Code Editor:** Displays a Python file named 'ass 7.3.py'. The code contains two versions of a 'Student' class: one with a missing 'self' parameter and one corrected. A comment indicates testing with 'Mouna' at age 20.
- Terminal:** Shows a PowerShell prompt in the background, indicating the code was run in a terminal.
- Status Bar:** Shows the current file is 'ass 7.3.py', line 58, column 1, with 4 spaces, using UTF-8 encoding, and CR LF line endings.

Task 5: Resolving Index Errors in Lists

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

The screenshot shows the GitHub Copilot AI Assistant integrated into the Visual Studio Code (VS Code) interface. The main area displays a Python file named `ass 7.3.py` with code related to handling list index errors. A sidebar on the right provides AI assistance, including a prompt to identify and fix syntax errors and a message about reaching a monthly message quota.

```
# Buggy code (IndexError: list index out of range)
numbers = [10, 20, 30]
print(numbers[5]) # invalid index

# Fixed code using exception handling (safe access)
numbers = [10, 20, 30]
try:
    print(numbers[5])
except IndexError:
    print("Error: Index out of range")

# Fixed code using bounds checking
index = 2
if index < len(numbers):
    print(numbers[index])
else:
    print("Error: Index out of range")
```

TERMINAL

```
PS C:\Users\parva\OneDrive\Desktop\AI Assted> Error: Index out of range
>> 30
>> 
```