

## ASSIGNMENT – 7.3

2303A51396

### Task 1: Fixing Syntax Errors

#### Scenario

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

#### Requirements

##### Python

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

```
    return a + b
```

- 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 a code editor window with a dark theme. At the top, there's a tab bar with 'ASSIGNMENT 7.3.py X' and a navigation bar with 'ASSSIGNMENT 7.3 > ASSIGNMENT 7.3.py > ...'. Below the tabs, the code editor displays the following Python code:

```
1 def add(a, b):
2     return a + b
3 def subtract(a, b):
```

Below the code editor, there's a navigation bar with 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is underlined), and '...'. To the right of the terminal tab, there are icons for Python, a plus sign, a dropdown arrow, a square, a trash can, a three-dot ellipsis, and a close button.

The terminal output area shows the following text:

```
"nts/AI ASSISTED/ASSSIGNMENT 7.3/ASSIGNMENT 7.3.py"
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

## 2. 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 top-left corner, there's a tab labeled "ASSIGNMENT 7.3.py". Below the tabs, a breadcrumb navigation bar shows "ASSSIGNMENT 7.3 > ASSIGNMENT 7.3.py > ...". The main editor area contains the following Python code:

```
1
2
3 count = 1
4 while count <= 5:
5     print(count)
6     count += 1 # FIX: Increment the counter to avoid infinite loop
7
8
```

Below the editor, a horizontal toolbar has tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is currently selected), and "PORTS". To the right of the terminal tab is a dropdown menu with options "+", "Python", and "Python Deb...". The "Python" option is highlighted.

The terminal window displays the following command and its output:

```
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED> & 'c:\Users\PRANAY\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\PRANAY\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60242' '--' 'C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED\ASSSIGNMENT 7.3\ASSIGNMENT 7.3.py'
```

```
1
2
3
4
5
```

```
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED>
```

### 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 VS Code interface. The left pane displays a Python script named 'ASSIGNMENT 7.3.py' with the following code:

```

1 def divide(a, b):
2     try:
3         return a / b
4     except ZeroDivisionError:
5         return "Cannot divide by zero"
6
7 print("\nTask 3 Output:", divide(10, 0))

```

The right pane shows the terminal output:

```

PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED> & 'c:\Users\PRANAY\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\PRANAY\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53658' '--' 'C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED\ASSIGNMENT 7.3\ASSIGNMENT 7.3.py'

Task 3 Output: Cannot divide by zero
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED>

```

The terminal sidebar shows two entries: 'Python' and 'Python Deb...', with 'Python' currently selected.

#### 4. 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 VS Code interface with the following details:

- File:** ASSSIGNMENT 7.3.py
- Code Content:**

```

1  class Student:
2      def __init__(self, name, age):
3          self.age = age
4
5
6      # creating object
7      s1 = Student("Pranay", 20)
8
9      # accessing attributes
10     print("Name:", s1.name)
11     print("Age:", s1.age)
12

```
- Terminal Output:**

```

documents\AI ASSISTED\ASSSIGNMENT 7.3\ASSIGNMENT 7.3.py'
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED> ^C
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED>
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED> c:; cd 'c:\Users\PRANAY\OneDrive\Documents\AI ASSISTED'; & 'c:\Users\PRANAY\Ap
pData\Local\Programs\Python\Python311\python.exe' 'c:\Users\PRANAY\OneDrive\Documents\AI ASSISTED\ASSSIGNMENT 7.3\ASSIGNMENT 7.3.py'
Name: Pranay
Age: 20
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED>

```
- Debug Bar:** A dropdown menu is open, showing options for Python and Python Debug.

## 5. 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

- AI suggestion using length checks or exception handling

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

- Editor:** The file `ASSIGNMENT 7.3.py` is open. The code defines a list `numbers` and prints its 5th element. A conditional block checks if the index is within the list's bounds.
- Terminal:** The terminal window shows the command to run the script and the resulting output: "Index out of range".
- Bottom Bar:** The "TERMINAL" tab is selected. A tooltip "Open file in editor (ctrl + click)" points to the terminal tab.

```
ASSIGNMENT 7.3 > 🐍 ASSIGNMENT 7.3.py > ...
1 numbers = [10, 20, 30]
2
3 print("\nTask 5 Output:")
4 index = 5
5
6 if index < len(numbers):
7     print(numbers[index])
8 else:
9     print("Index out of range")
10

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ... | ↻
Open file in editor (ctrl + click)
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED> & 'c:\Users\PRANAY\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\PRANAY\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59490' '--' 'c:\Users\PRANAY\OneDrive\Documents\AI ASSISTED\ASSIGNMENT 7.3\ASSIGNMENT 7.3.py'

Task 5 Output:
Index out of range
PS C:\Users\PRANAY\OneDrive\Documents\AI ASSISTED>
```