

## **AI ASSISTED CODING**

**Name:** P.Swaran Raj

**Roll.No:** 2303A52247

**Batch:** 37

**Assignment- 7.3**

### **Task 1: Fixing Syntax Errors**

#### **Scenario:**

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

#### **Prompt Given for modification:**

**“#Detect the syntax error and correct it and explain”**

```
... AI Assist 1.py ai assit 2.py ai assist6.2.py AI Assist7.3.py ...
AI Assist7.3.py
1 def add(a, b)
2     return a+b

Detect the syntax error and correct it and explain why this error
occured

Add Context... Auto ▾

Ln 2, Col 15 (29 selected) Spaces: 4 UTF-8 CRLF { } Plain Text ⚙ 09:54
IN 04-02-2026
```

The screenshot shows a code editor interface with multiple tabs at the top: "AI Assist 1.py", "ai assit 2.py", "ai assist6.2.py", "AI Assist7.3.py", and "...". The active tab is "AI Assist7.3.py". Below the tabs, there is a toolbar with a search icon, file icons, and other tools. The main area displays the following Python code:

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

The line "def add(a, b):" is highlighted with a green background, indicating it is selected. The status bar at the bottom shows "Ln 2, Col 15 (30 selected)".

### Error Explanation:

The code has a syntax error. In Python, function definitions must end with a colon (:) after the parameter list.

### Output:

The corrected version should be:

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

### Comments:

Ai fixed and explained the syntax error.

## Task 2: Debugging Logic Errors in Loops

## Scenario:

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

## Prompt Given for modification:

"#Fix the loop logic."

The screenshot shows a dark-themed code editor window. At the top, there's a tab labeled "AI Assist7.3.py > ...". Below the tabs, a number "3" is displayed. A task bar is visible with the text "fix the loop logic" and icons for "Run" and "X". There's also a "Add Context..." button and a dropdown menu set to "Auto". The main code area contains the following Python code:

```
4 def count_down(n):
5     while n >= 0:
6         print(n)
7         n += 1 # should be n -= 1
```

A modal dialog titled "Modify selected code" is open over the code editor. It has a "✓" button and an "X" button at the top right. Below it are "Add Context..." and "Auto" buttons. The code in the dialog matches the code in the editor. At the bottom right of the dialog, there are "Keep", "Undo", and a "redo" icon buttons. The "Keep" button is highlighted in blue.

```
4 def count_down(n):
5     while n >= 0:
6         print(n)
7         n -= 1 # should be n -= 1
```

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

- Code:**

```
4 def count_down(n):
5     while n >= 0:
6         print(n)
7         n -= 1 # should be n -= 1
```
- Annotations:**
  - Inside the loop, `n += 1` increments `n` by 1 each time, making it larger (never smaller).
  - Since `n` keeps increasing, it never becomes negative, so the loop never exits.
- Text:** For example, if you call `count_down(5)`, it will print 5, then 6, then 7, forever.
- AI Assistant Panel:** A floating panel titled "AI Assist7.3.py" contains:
  - A text input field: "Describe what to build next".
  - Buttons: "Agent", "Auto", and "Help".
  - System status: "Ln 4, Col 1", "Spaces: 4", "UTF-8", "Python 3.13.3", "10:13", "ENG IN", "04-02-2026".

## Error Explanation:

this code will go into an infinite loop if `n` starts as a non-negative integer (0 or greater).

- The while loop condition is `n >= 0`, so it runs as long as `n` is 0 or positive.
- Inside the loop, `n += 1` increments `n` by 1 each time, making it larger (never smaller).
- Since `n` keeps increasing, it never becomes negative, so the loop never exits.

## Output:

it should be `n -= 1` to decrement instead, which would make it count down properly (e.g., from 5 to 0). If `n` starts negative, the loop wouldn't run at all

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

## Comments:

AI fixed the loop logic ,instead of incrementing it went for decrementing that made count down properly and if negative number starts the loop will end.

### Task 3: Handling Runtime Errors (Division by Zero)

#### Scenario:

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

#### Prompt Given for modification:

**“#identify the run time error and fix it so that function executes safely.”**

The screenshot shows a code editor interface with a dark theme. At the top, there is a task card with the number '8' and a lightbulb icon. The main text area contains the following text:

```
identify the run time error and fix it so that function  
executes safely
```

Below the text area are two buttons: 'Add Context...' and 'Auto ▾'. The code editor window displays the following Python code:9 def divide(a, b):  
10 return a/b  
11  
12 print(divide(10, 0))

A screenshot of a code editor window titled "Modify selected code". The code is as follows:

```
9 def divide(a, b):
10     if b == 0:
11         return "Error: Cannot divide by zero"
12     return a/b
13
14 print(divide(10, 0))
```

The line "return "Error: Cannot divide by zero"" is highlighted with a green background. The status bar at the bottom right shows "Auto".

Error: Cannot divide by zero

None

A screenshot of a code editor window showing the same code as above, but with a modification:

```
9 def divide(a, b):
10     if b == 0:
11         print("Error: Cannot divide by zero")
12         return None
13     return a/b
14
15 print(divide(10, 2))
```

5.0

### Error Explanation:

The code will raise a `ZeroDivisionError` at runtime when you try to divide by zero. In Python, division by zero is mathematically undefined and not allowed, so the interpreter throws an exception instead of returning a result.

### Output:

When you run `print(divide(10, 0))`, you'll see output like:

## **ZeroDivisionError: division by zero**

To handle this safely, you could add error checking in the function:

```
def divide(a, b):
    if b == 0:
        return "Cannot divide by zero"
    return a / b
```

**o/p:** Error: Cannot divide by zero

None

```
def divide(a, b):
    if b == 0:
        print("Error: Cannot divide by zero")
        return None
    return a/b

print(divide(10, 2))
```

**o/p:** 5.0

### **Comments:**

AI fixed the error and explained it , provided why it can't divide by zero and when we add 2 instead of 0 it gave the output 5.0 it handled the runtime error and executed safely

## **Task 4: Debugging Class Definition Errors**

### **Scenario:**

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

### **Prompt Given for modification:**

**“#identify the issue in the `__init__` constructor and correct the class definition.”**

identify the issue in the `__init__` constructor and  
correct the class definition

Add Context... Auto ▾

```
17 class Rectangle:
18     def __init__(self, length, width):
19         self.length = length
20         self.width = width
```

Modify selected code

Add Context... Keep (Ctrl+Enter)

```
17 class Rectangle:
18     def __init__(self, length, width):
19         self.length = length
20         self.width = width
21
22     def __str__(self):
23         return f"Rectangle(length={self.length}, width={self.width})"
```

```
17 class Rectangle:
18     def __init__(self, length, width):
19         self.length = length
20         self.width = width
21
22     def __str__(self):
23         return f"Rectangle(length={self.length}, width={self.width})"
```

## Error Explanation:

The provided code defines a Rectangle class with an `__init__` method that correctly initializes length and width attributes. The AI detected that `self`

was missing in the constructor. It corrected the class definition by adding `self` to properly reference object attributes.

### Output:

However, the class is incomplete for practical use. A typical Rectangle class would include methods to calculate properties like area or perimeter.

For example, you could add:

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

    def __str__(self):
        return f"Rectangle(length={self.length}, width={self.width})"
```

### Comments:

AI fixed the runtime error , `__init__` constructor and class definition and it explained the error.

## Task 5: Resolving Index Errors in Lists

### Scenario:

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

### Prompt Given for modification:

**"#Identify and fix the index error caused by accessing an out-of-range list index."**

The screenshot shows a dark-themed code editor interface. At the top, there is a text input field containing the instruction: "Identify and fix the index error caused by accessing an out-of-range list index." Below the input field are two buttons: "Add Context..." and "Send (Enter)". The code editor displays two lines of Python code:

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

A screenshot of a code editor interface. At the top, there's a toolbar with a search bar labeled "Modify selected code" and buttons for "Keep", "Undo", and a plus sign. Below the toolbar, the code is displayed in two sections:

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

The line "print(numbers[5])" is highlighted in red, indicating an error. The line "print(numbers[2])" is highlighted in green, suggesting a fix. The status bar at the bottom shows "Auto" with a dropdown arrow.

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

### Error Explanation:

Lists in Python are zero-indexed (start at 0), so a list with 3 items goes from index 0 to 2. Asking for index 5 is like asking for a 6th item that isn't there.

The AI identified the index error and suggested bounds checking.

Safe access was implemented using length checks or exception handling.

### Output:

The program accesses list elements safely without crashing.

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

**o/p:3**

### Comments:

AI provides effective solutions for common runtime errors.

Validating list indices improves code reliability.

### Conclusion:

This lab demonstrates how AI tools assist in systematically identifying and fixing syntax, logic, and runtime errors in Python programs.