

## AI-Assignment-7.3

**Name:**G.Harika

**Ht.No:**2303A51612

**Batch:**05

**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

**Code:** `result = add(5, 3)`

```
print(f"The result of add(5, 3) is: {result}")
```

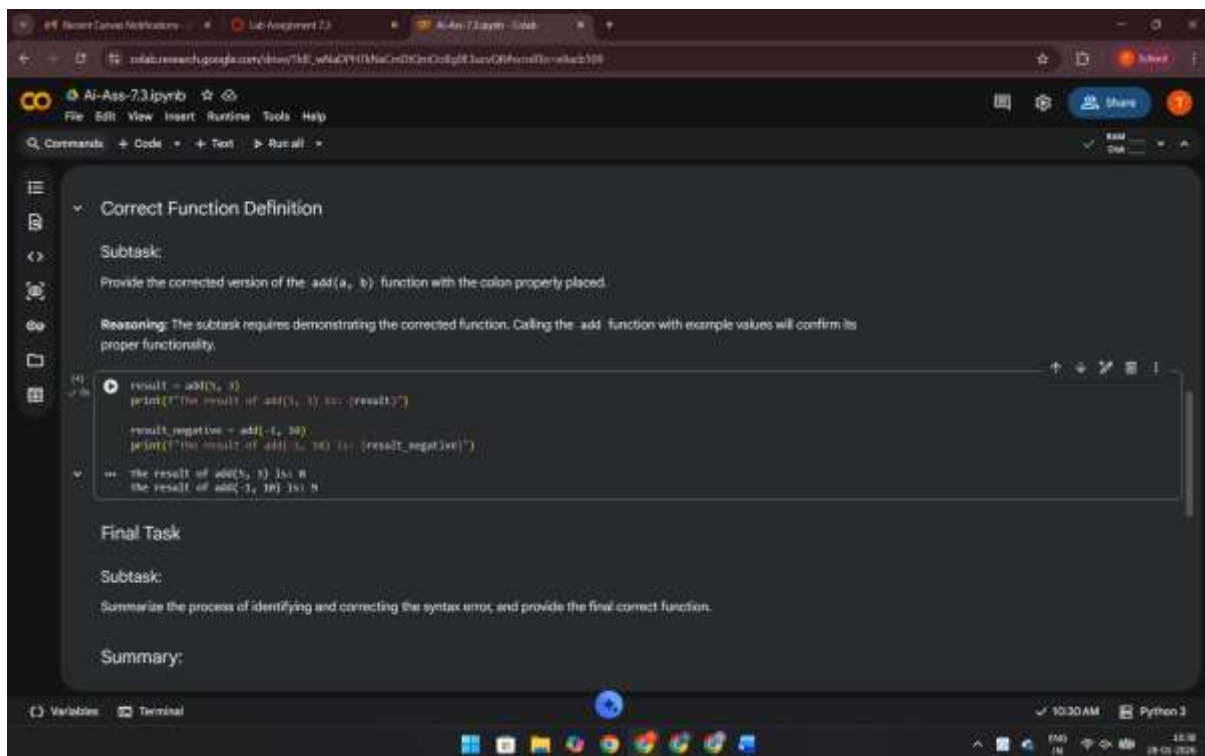
```
result_negative = add(-1, 10)
```

```
print(f"The result of add(-1, 10) is: {result_negative}")
```

**Output:**The result of add(5, 3) is: 8

The result of add(-1, 10) is: 9

1. **Explanation: Intentional Error:** A Python function `add(a, b)` was initially defined without the required colon, causing a `SyntaxError: expected ':'` when executed.
2. **Error Detection:** The Python interpreter explicitly reported the `SyntaxError`, clearly indicating the missing colon.
3. **Correction:** The function definition was corrected by adding the colon (`def add(a, b):`).
4. **Verification:** The corrected function was then successfully executed and tested with example values.



## 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

**Code:** `count = 0`

`while count < 5:`

`print(f"Current count: {count}")`

`count += 1`

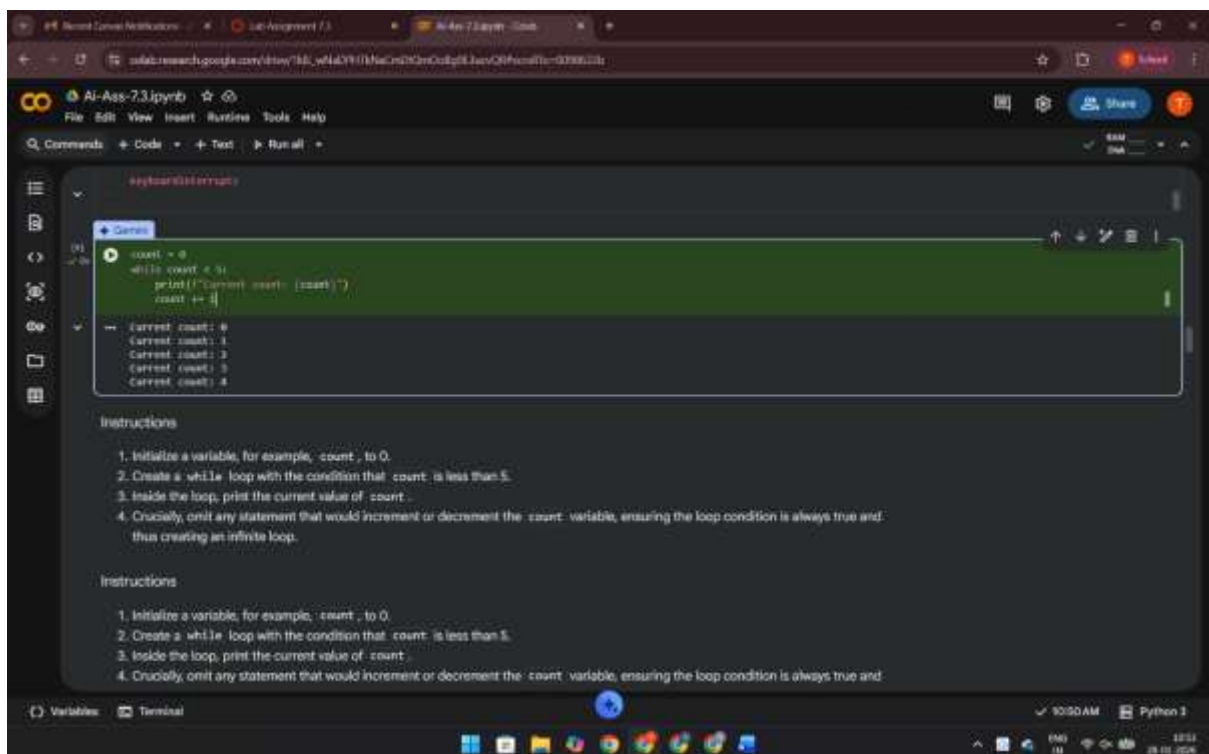
**Output:** `Current count: 0`

`Current count: 1`

`Current count: 2`

`Current count: 3`

`Current count: 4`



**Explanation:** We introduced an infinite loop by forgetting to increment the count variable in a while loop. This caused the loop condition (count < 5) to always be true, printing "Current count: 0" repeatedly. We then corrected this by adding count += 1 inside the loop, which made it terminate correctly and print counts from 0 to 4. This highlights the importance of modifying loop control variables to ensure termination.

### **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

**Code:** `def divide_safe(a, b):`

`try:`

`result = a / b`

`return result`

`except ZeroDivisionError:`

`return "Error: Cannot divide by zero"`

`except TypeError:`

`return "Error: Invalid input type"`

# Example calls

```
print(divide_safe(10, 2))
```

```
print(divide_safe(10, 0))
```

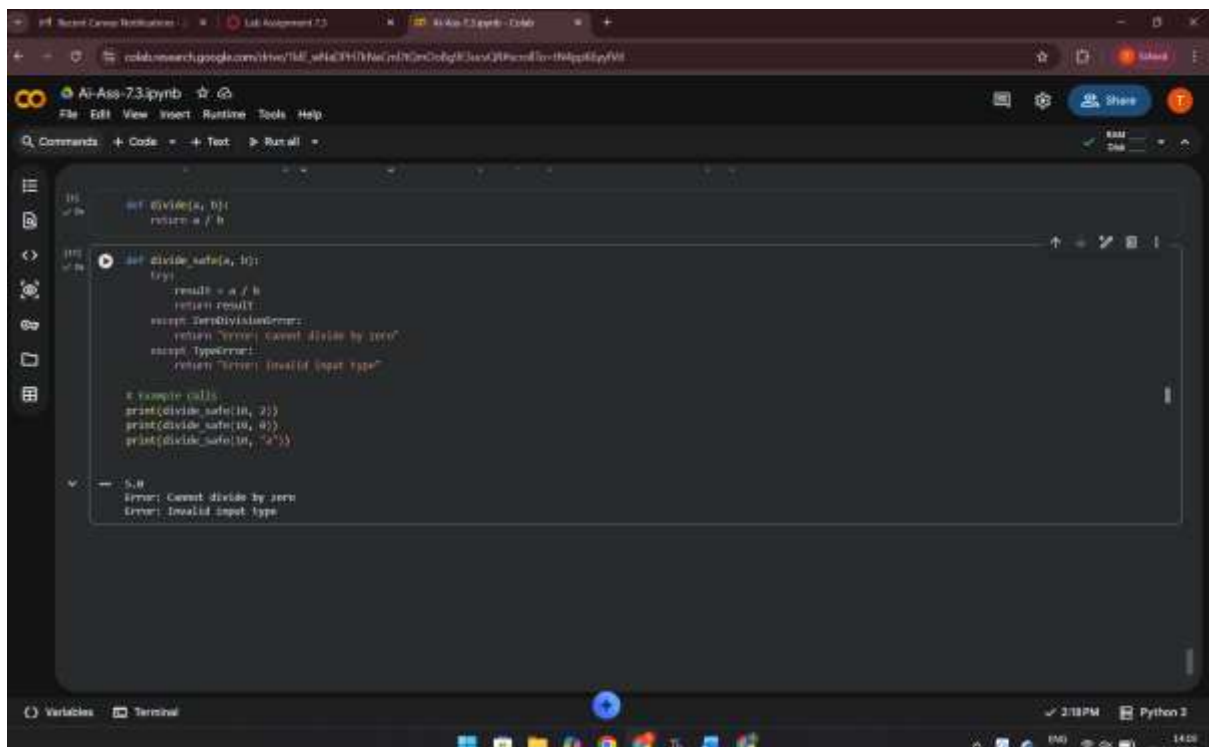
```
print(divide_safe(10, "a"))
```

**Output:**

5.0

Error: Cannot divide by zero

Error: Invalid input type

A screenshot of a Jupyter Notebook interface. The top bar shows the file name 'AI-Ass-7.3.ipynb' and various icons for file operations, settings, and sharing. The main area contains a code cell with the following Python code:

```
def divide(a, b):  
    return a / b  
  
def divide_safe(a, b):  
    try:  
        result = a / b  
        return result  
    except ZeroDivisionError:  
        return "Error: Cannot divide by zero"  
    except TypeError:  
        return "Error: Invalid input type"  
  
# Example calls  
print(divide_safe(10, 2))  
print(divide_safe(10, 0))  
print(divide_safe(10, "a"))
```

Below the code cell, the output is displayed:

```
5.0  
Error: Cannot divide by zero  
Error: Invalid input type
```

The bottom of the interface shows a 'Variables' tab and a 'Terminal' tab, along with a status bar indicating 'Python 3' and the time '2:18 PM'.

**Explanation:**

The original division function crashes when dividing by zero.

- AI identifies the runtime error as **ZeroDivisionError**.
- AI fixes it using try-except to prevent program failure.
- Error handling makes the code safer and more reliable.
- Using specific exceptions improves clarity and control.

## **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

### Code:

```
class Student:
```

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

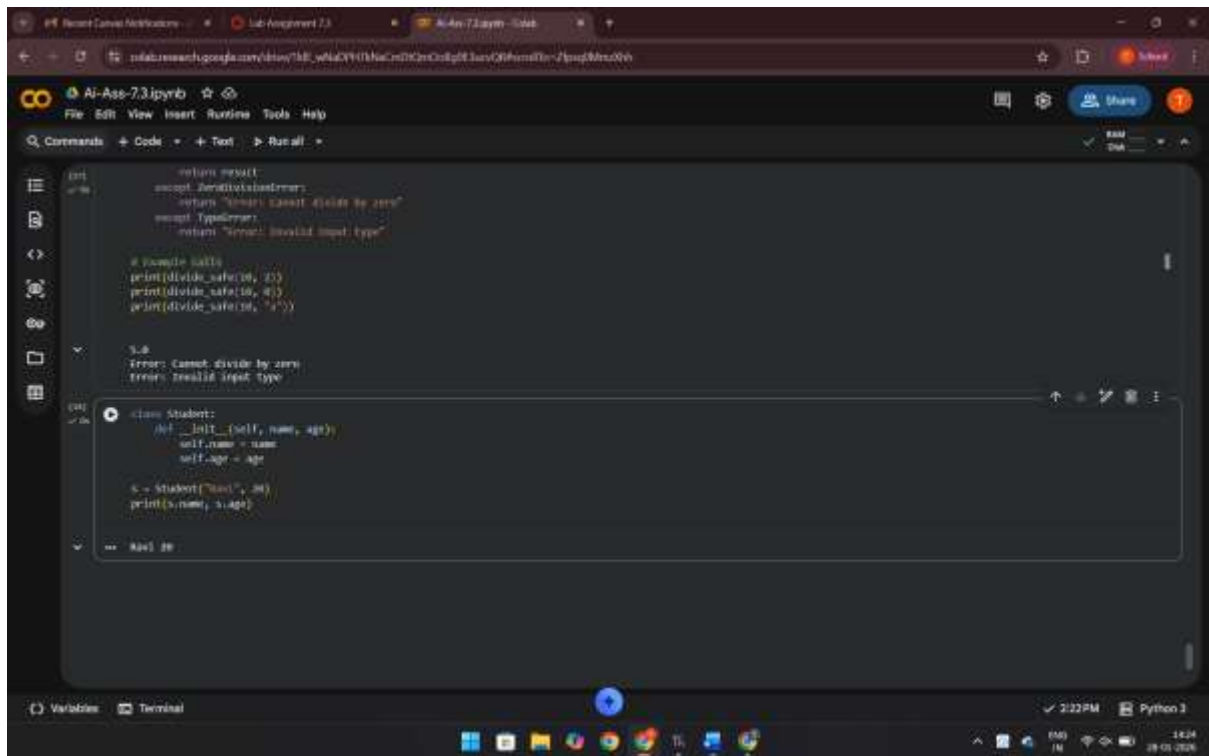
```
        self.name = name
```

```
        self.age = age
```

```
s = Student("Ravi", 20)
```

```
print(s.name, s.age)
```

**Output:**Ravi 20



**Explanation:** self refers to the current object.

Without self, Python cannot store data inside the object.

AI identifies the missing self and fixes the constructor.

Using self.variable correctly binds data to the object.

## Task 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

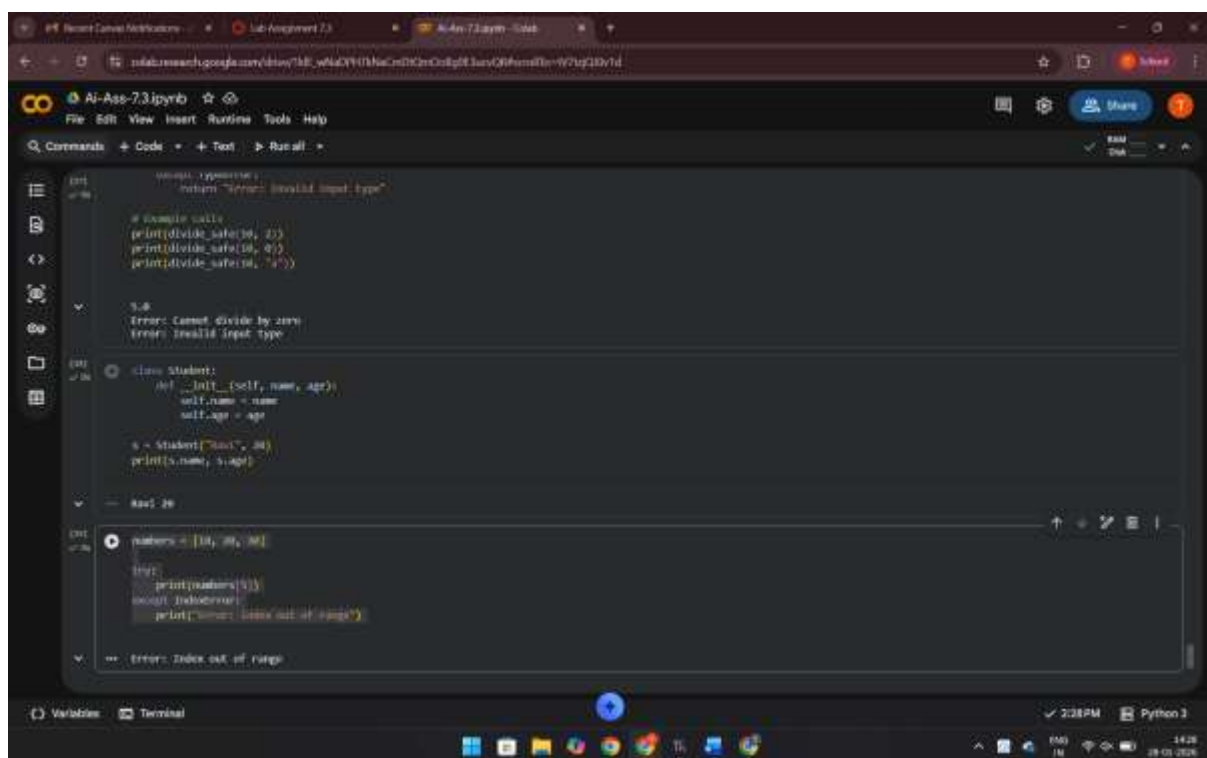
**Code:** numbers = [10, 20, 30]

try:

```
print(numbers[5])
```

except IndexError:

```
print("Error: Index out of range")
```



**Output:**Error: Index out of range

**Explanation:** Accessing an invalid index causes IndexError.

AI detects the error and prevents program crash.

try-except or bounds checking ensures safe list access.

This makes the program stable and error-free



