

# AI Assisted Coding-7.3

K.Raghavendra || Batch-09 || 2303A51897

## Task 1: Fixing Syntax Errors

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

**Code:**

#Task-01:

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

# Function to add two numbers
# Error: Missing colon (:) at the end of function definition
# Syntax Error - def add(a, b) is missing a colon
```

# CORRECTED CODE:

```
def add(a, b):
    """Function that takes two parameters and returns their sum"""
    return a + b
```

# Test the function

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

**Output:**

The screenshot shows a code editor interface with several tabs open. The main tab displays a Python script named '2303AS1897\_5.py' containing three tasks:

- Task 1: Fixing Syntax Errors**: Contains the initial code with errors and the corrected version.
- Task 2: Debugging Logic Errors in Loops**: Contains code with an infinite loop and its corrected version.
- Task 3: Handling Runtime Errors (Division by Zero)**: Contains code that performs division by zero and its corrected version.

The status bar at the bottom indicates the file path 'C:/Python314/python.exe" c:/Users/ragha/Desktop/T.../AI Assistance in coding/2303AS1897\_5.py"' and the terminal output 'Task 1 Output: 8'.

## Task 2: Debugging Logic Errors in Loops

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

**Code:**

```
#Task-02:  
# Infinite Loop - ERROR VERSION  
print("ERROR VERSION - Infinite Loop:")  
i = 0  
while i < 5:  
    print(f"Iteration {i}")  
    # Problem: i is never incremented, so the loop never exits  
    # The condition i < 5 is always True
```

**# CORRECTED CODE:**

```
print("\nCORRECTED VERSION:")  
i = 0 while i < 5:  
    print(f"Iteration {i}")  
    i += 1 # Increment i by 1 each iteration to eventually reach the exit condition
```

**Output:**

The screenshot shows the Visual Studio Code interface. The code editor displays the corrected Python script. The terminal at the bottom shows the output of the script, which prints the numbers 1, 2, 3, 4, and 5, indicating the loop has exited. The sidebar on the right shows session history and AI assistance logs.

```
File Edit Selection View Go Run Terminal Help  
2303AS1897_5.py  
15  
16 # Task 2: Debugging Logic Errors in Loops  
17  
18  
19 # ✘ Code With Error (Infinite Loop)  
20 # i = 1  
21 # while i <= 5:  
22 #     print(i)  
23 #     i -= 1  
24  
25 # ✘ Corrected Code  
26 print("\nTask 2 Output:")  
27 i = 1  
28 while i <= 5:  
29     print(i)  
30     i += 1  
31  
32  
33 # Task 3: Handling Runtime Errors (Division by Zero)  
34  
35  
36  
37 # ✘ Code With Error  
38 # def divide(a, b):  
39 #     return a / b  
40 # print(divide(10, 0))  
41  
42 # ✘ Corrected Code  
43 def divide(a, b):  
44     try:  
45         return a / b  
46     except ZeroDivisionError:  
47         return "Cannot divide by zero"  
48  
49 print("\nTask 3 Output:", divide(10, 0))  
50  
51  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\raghā\Desktop\TempVAI Assistance in coding> & C:/Python314/python.exe "c:/Users/raghā/Desktop/Temp/AI Assistance in coding/2303AS1897_5.py"  
Task 2 Output:  
1  
2  
3  
4  
5
```

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

```
#Task 3: Handling Runtime Errors (Division by Zero) #
Function WITHOUT validation (causes runtime error)
def divide_without_validation(a, b):
    """Division function with no error handling - will crash if b is 0"""
    return a / b

# Test - this will crash
print("WITHOUT VALIDATION:")
try:
    result = divide_without_validation(10, 0)
print(f"Result: {result}") except
ZeroDivisionError:    print("ERROR:
Cannot divide by zero!")

# Function WITH try-except blocks (safe execution)
def divide_with_validation(a, b):
    """Division function with error handling using try-except"""
try:
    # Attempt the division operation      result = a / b      return result
except ZeroDivisionError:      # Catch division by zero error
print("Error: Cannot divide by zero. Denominator must be non-zero.")
return None    except TypeError:
    # Catch type errors (non-numeric values)
print("Error: Both arguments must be numbers.")
return None

# Test - safe execution
print("\nWITH VALIDATION:") result
= divide_with_validation(10, 2) if
result is not None:    print(f"Result:
{result}") result =
divide_with_validation(10, 0)
```

### Output:

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

- File Explorer:** Shows a file named "2303A51897\_5.py".
- Editor:** Displays Python code for Task 3, which includes a loop from 1 to 5 and a division operation that results in an error. A comment indicates the task is about handling runtime errors for division by zero.
- Terminal:** Shows the command "C:/Python314/python.exe" running the script, with the output "Task 3 Output: Cannot divide by zero".
- Output:** Shows three Python processes listed under "SESSIONS".
- Status Bar:** Shows "AI Assistance in coding".

## Task 4: Debugging Class Definition Errors Code:

```
#Task 4: Debugging Class Definition Errors
# FAULTY CODE - Missing 'self' parameter in __init__()
print("\nFAULTY CLASS DEFINITION:")
class Person:
    """Class definition with ERROR in constructor"""
    def __init__(name, age): # ERROR: Missing 'self' as first parameter
        """Constructor without self parameter - causes TypeError"""
        name = name      age = age

    # This will cause an error when trying to create an instance
    # TypeError: __init__() takes 2 positional arguments but 3 were given
    # try:
    #     person1 = Person("Alice", 30)
    # except TypeError as e:
    #     print(f"ERROR: {e}")

# CORRECTED CODE - Proper class definition with 'self' parameter
print("\nCORRECTED CLASS DEFINITION:")
class Person:
    """Class definition with proper constructor including 'self' parameter"""
    def __init__(self, name, age):
```

```

"""Constructor with 'self' parameter - allows proper object creation
self: represents the instance of the class      name: parameter for
person's name      age: parameter for person's age""""      self.name
= name # Store name as instance variable      self.age = age #  

Store age as instance variable

```

```

def display_info(self):
    """Method to display person's information"""
    print(f"Name: {self.name}, Age: {self.age}")

```

# Test - safe execution with corrected class

```

try:
    person1 = Person("Alice", 30)
    person1.display_info()

```

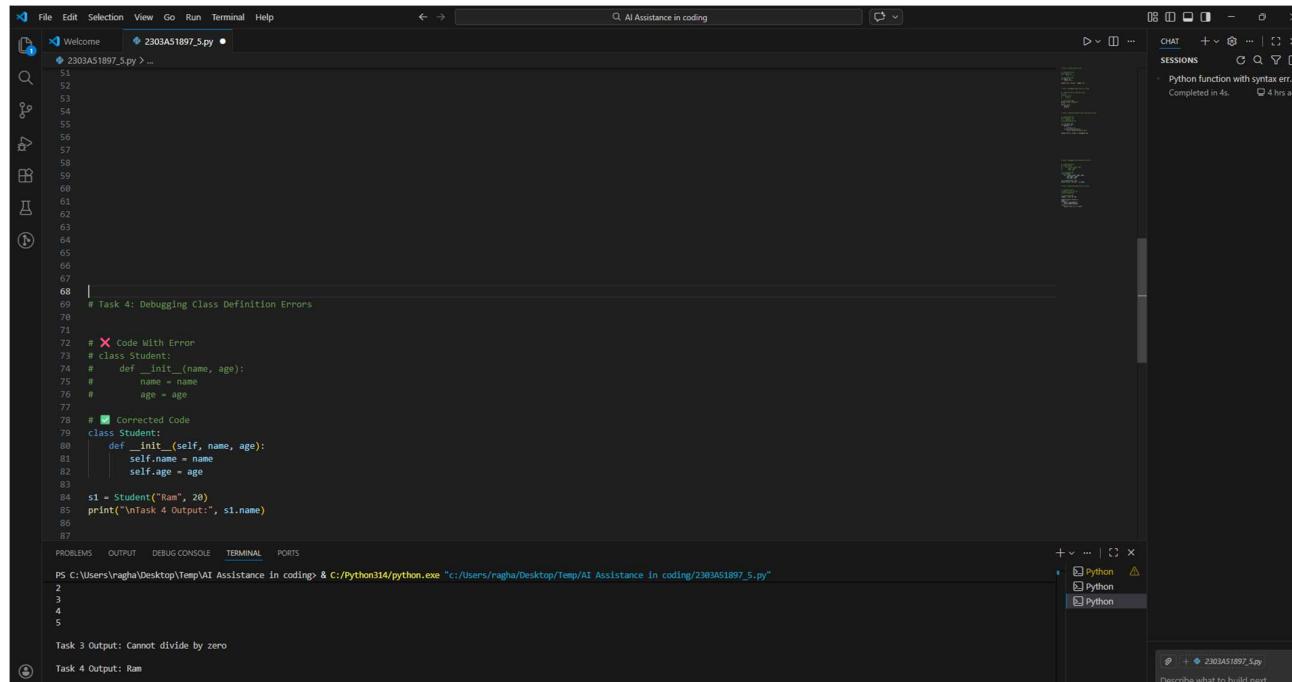
```

    person2 = Person("Bob", 25)
    person2.display_info() except

```

TypeError as e:

print(f"ERROR: {e}") Output:



```

# Task 4: Debugging Class Definition Errors
# X Code with Error
# class Student:
#     def __init__(name, age):
#         name = name
#         age = age
#
#     # Corrected Code
# class Student:
#     def __init__(self, name, age):
#         self.name = name
#         self.age = age
#
# s1 = Student("Ram", 28)
# print("\nTask 4 Output:", s1.name)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\raghav\Desktop\Temp\AI Assistance in coding> & C:/Python314/python.exe "c:/Users/raghav/Desktop/Temp/AI Assistance in coding/2303A51897_5.py"
2
3
4
5
Task 3 Output: Cannot divide by zero
Task 4 Output: Ram

```

## Task 5: Resolving Index Errors in Lists Code:

### #Task-05: Handling Index Errors (Out-of-Range List Access)

```
# Function WITHOUT validation (causes runtime error)
def access_list_without_validation(lst, index):
    """Function that accesses list without bounds checking - will crash if index is out
    of range"""
    return lst[index]

# Test - this will crash
print("WITHOUT VALIDATION:")
try:
    my_list = [10, 20, 30, 40, 50]    result = access_list_without_validation(my_list,
    10) # Index 10 doesn't exist (list has only 5 elements)    print(f"Value at index 10:
{result}") except IndexError:    print("ERROR: List index out of range!")

# Function WITH try-except blocks (safe execution)
def access_list_with_validation(lst, index):
    """Function that accesses list with error handling using try-except"""
try:
    # Attempt to access the list at given index
    result = lst[index]    return result    except
    IndexError:
        # Catch index out of range error    print(f"Error: Index {index} is out of
        range. List has only {len(lst)} elements.")    return None    except TypeError:
        # Catch type errors (non-numeric index)
    print("Error: Index must be an integer.")
    return None

# Test - safe execution
print("\nWITH VALIDATION:")
my_list = [10, 20, 30, 40, 50] result
=
access_list_with_validation(my_list,
2) if result is not None:
print(f"Value at index 2: {result}")

result = access_list_with_validation(my_list, 10)
```

**Output:**

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Welcome, 2303A51897\_5.py, AI Assistance in coding.
- Code Editor:** Displays Python code for Task 5, which includes error handling for index out-of-range errors.
- Terminal:** Shows command-line output for tasks 3, 4, and 5, along with AI assistance logs.
- Output Panel:** Shows multiple Python sessions and a message from AI assistance.
- Status Bar:** Chat, Sessions, Python function with syntax error... Completed in 4s, 4 hrs ago.

```
88 # Task 5: Resolving Index Errors in Lists
89
90
91 # ✘ Code With Error
92 numbers = [10, 20, 30]
93 # print(numbers[5])
94
95 # ✅ Corrected Code
96 numbers = [10, 20, 30]
97
98 print("Task 5 Output:")
99 index = 5
100 if index < len(numbers):
101     print(numbers[index])
102 else:
103     print("Index out of range")
```

TERMINAL OUTPUT:

```
PS C:\Users\raghav\Desktop\Temp\AI Assistance in coding> & C:/Python314/python.exe "c:/Users/raghav/Desktop/Temp/AI Assistance in coding/2303A51897_5.py"
2
3
4
5

Task 3 Output: Cannot divide by zero
Task 4 Output: Ram
Task 5 Output:
Index out of range
PS C:\Users\raghav\Desktop\Temp\AI Assistance in coding>
```

SESSIONS:

- Python
- Python
- Python

CHAT:

- Python function with syntax error... Completed in 4s, 4 hrs ago