

AI Assisted Coding-7.3

P.SIDDARTHA || Batch-09 || 2303A51693

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:

```
% PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> & "C:/Program Files/Python313/python.exe" c:/Users/SIDDARTHA/OneDrive/Desktop/ai/lab7.py  
File "c:/Users/SIDDARTHA/OneDrive/Desktop/ai/lab7.py", line 2  
    def add(a,b)  
          ^  
SyntaxError: expected ':'  
% PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai>
```

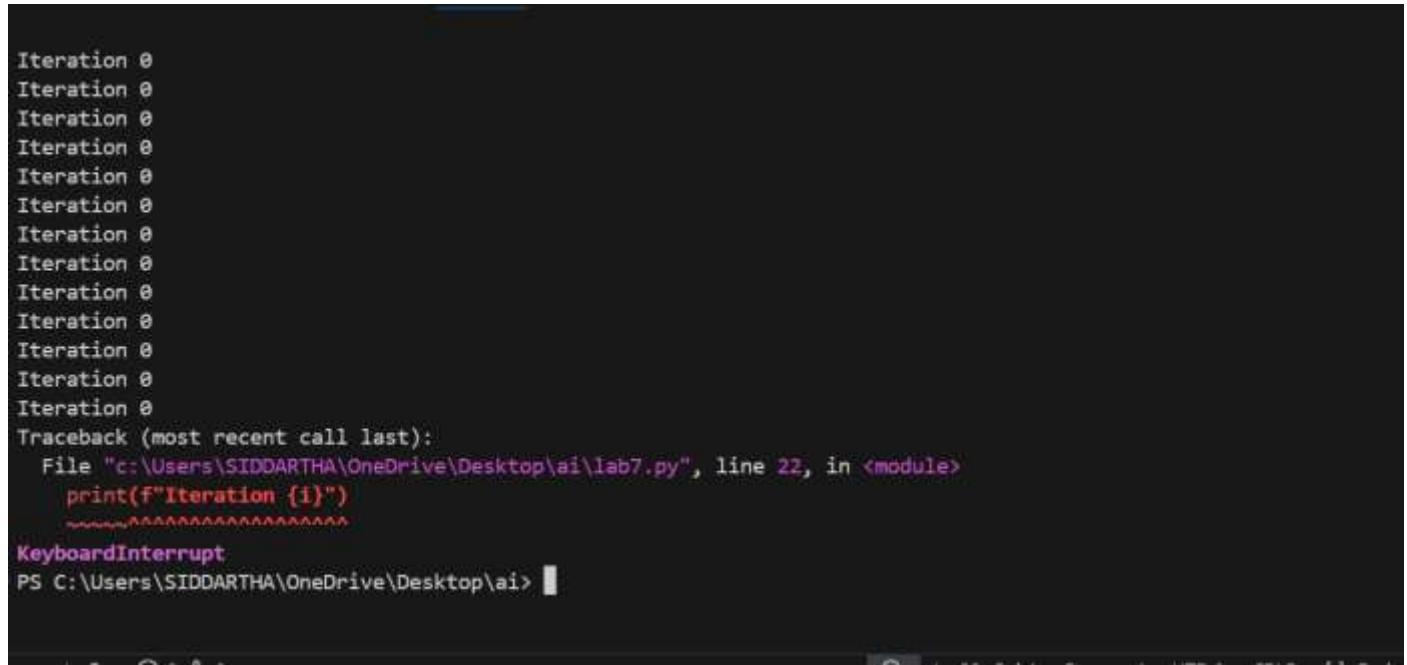
Task 2: Debugging Logic Errors in Loops

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

Code:

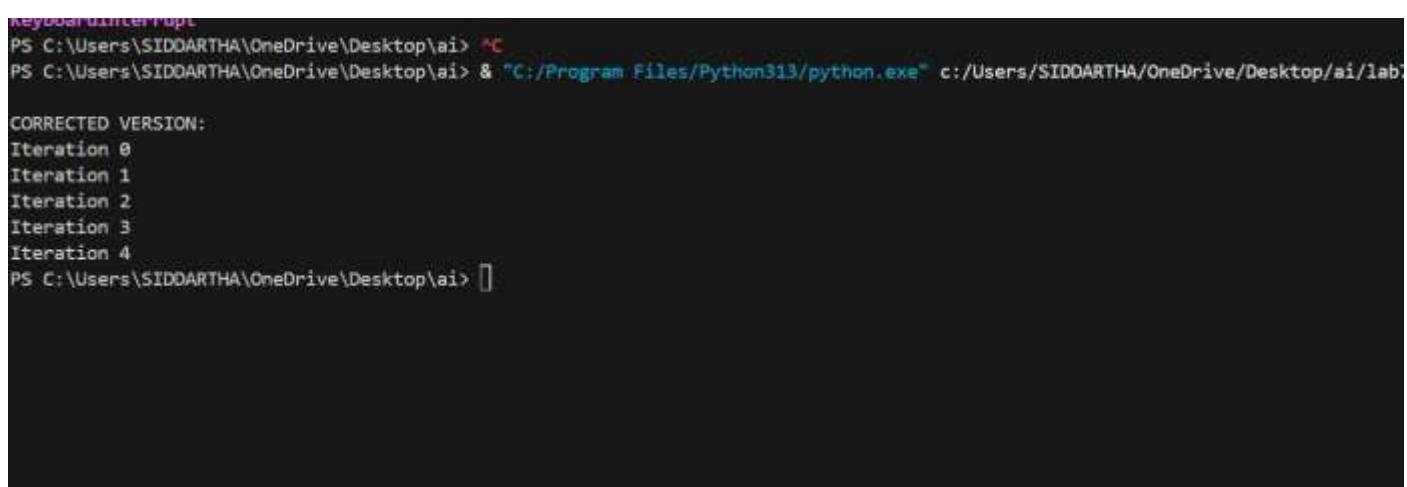
```
#Task-02:  
# Infinite Loop - ERROR VERSION  
print("ERROR VERSION - Infinite Loop:")  
i = 0  
while i < 5:  
    print(f"Iteration {i}")  
    # Problem: i is never incremented, so the loop never exits  
    # The condition i < 5 is always True  
  
# CORRECTED CODE:  
print("\nCORRECTED VERSION:")  
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 a terminal window with a black background and white text. It displays the output of the first version of the code, which results in an infinite loop. The text shows multiple iterations of "Iteration 0" followed by a traceback:

```
Iteration 0  
Traceback (most recent call last):  
  File "c:\Users\SIDDARTHA\OneDrive\Desktop\ai\lab7.py", line 22, in <module>  
    print(f"Iteration {i}")  
    ~~~~~^~~~~~  
KeyboardInterrupt  
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai>
```



The screenshot shows a terminal window with a black background and white text. It displays the output of the corrected code, which prints five iterations of "Iteration 0" to "Iteration 4" and then exits:

```
KeyboardInterrupt  
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> ^C  
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> & "C:/Program Files/Python313/python.exe" c:/Users/SIDDARTHA/OneDrive/Desktop/ai/lab7.py  
  
CORRECTED VERSION:  
Iteration 0  
Iteration 1  
Iteration 2  
Iteration 3  
Iteration 4  
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> ]
```

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:

```
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> & "C:/Program Files/Python313/python.exe" c:/Users/SIDDARTHA/OneDrive/Desktop/ai/lab7.py
WITHOUT VALIDATION:
ERROR: Cannot divide by zero!

WITH VALIDATION:
Result: 5.0
Error: Cannot divide by zero. Denominator must be non-zero.
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai>
```

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:

```

PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> & "C:/Program Files/Python311/python.exe" c:/Users/SIDDARTHA/OneDrive/Desktop/ai/lab7.py
FAULTY CLASS DEFINITION:

CORRECTED CLASS DEFINITION:
Name: Alice, Age: 30
Name: Bob, Age: 25
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> []

```

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("\nWITHOUT 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:

```
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> & "C:/Program Files/Python313/python.exe" c:/Users/SIDDARTHA/OneDrive/Desktop/ai/lab7.py
WITHOUT VALIDATION:
ERROR: List index out of range!

WITH VALIDATION:
Value at index 2: 30
Error: Index 10 is out of range. List has only 5 elements.
PS C:\Users\SIDDARTHA\OneDrive\Desktop\ai> []
```