

Assignment-7.3

2303A51304

Batch-05

Task-01:

Prompt:

Detect and fix the syntax error in the following Python function where the function definition is missing a colon.

Correct the code so that it runs successfully.

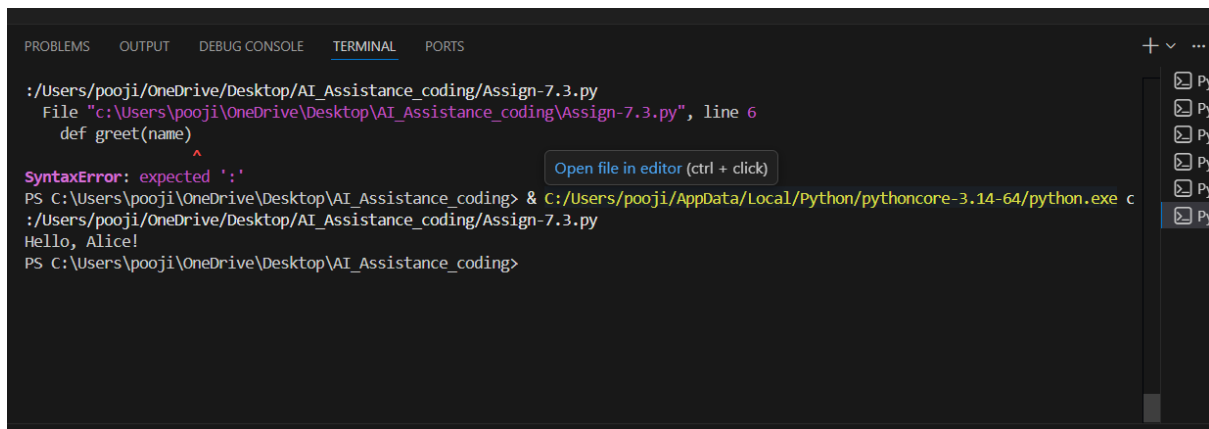
Explain the syntax issue and how it was resolved.

Show the output after fixing the error.

Code:

```
Show the output after fixing the error.'''
# Original code with syntax error
def greet(name)
    print("Hello, " + name + "!")
# Corrected code
def greet(name):
    print("Hello, " + name + "!")
# Explanation of the syntax issue:
# The original function definition for 'greet' was missing a colon (:) at the end
# of the line. In Python, function definitions must end with a colon to indicate
# the start of the function body. Adding the colon resolves the syntax error.
# ----- MAIN EXECUTION -----
if __name__ == "__main__":
    greet("Alice")
```

Output:



The screenshot shows a VS Code terminal window with the following content:

```
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
File "c:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding\Assign-7.3.py", line 6
    def greet(name)
    ^
SyntaxError: expected ':'
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/python.exe c
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
Hello, Alice!
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

A button labeled "Open file in editor (ctrl + click)" is visible next to the error message.

Explanation:

The syntax error was caused by a missing colon in the function definition.

AI successfully identified and fixed the error.

The corrected function follows proper Python syntax.

The program runs without errors.

The output confirms correct functionality.

Task-02:

Prompt:

Identify and fix the logical error in the following Python loop that causes it to run infinitely.

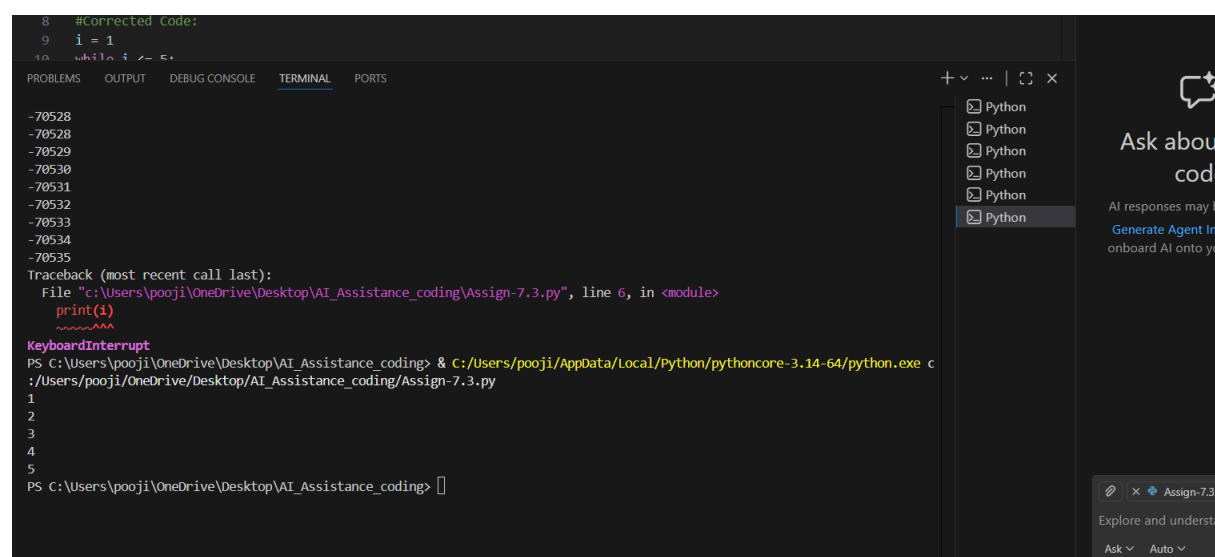
Explain why the infinite loop occurs and how the corrected logic resolves the issue.

Show the corrected code and its output.

Code:

```
Assign-7.3.py / ...
1  '''Identify and fix the logical error in the following Python loop that causes it to run infinitely.
2  Explain why the infinite loop occurs and how the corrected logic resolves the issue.
3  Show the corrected code and its output.'''
4  i = 1
5  while i <= 5:
6      print(i)
7      i -= 1  # ❌ Wrong direction (causes infinite loop)
8  #Corrected Code:
9  i = 1
10 while i <= 5:
11     print(i)
12     i += 1  # ✅ Correct direction
```

Output:



```
8  #Corrected Code:
9  i = 1
10 while i <= 5:
11     print(i)
12     i += 1  # ✅ Correct direction

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

-70528
-70528
-70529
-70530
-70531
-70532
-70533
-70534
-70535
Traceback (most recent call last):
  File "c:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding\Assign-7.3.py", line 6, in <module>
    print(i)
    ~~~~~^
KeyboardInterrupt
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/Appdata/Local/Python/pythoncore-3.14-64/python.exe c
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
1
2
3
4
5
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

The infinite loop occurred due to an incorrect decrement operation. AI detected the mismatch between the loop condition and update logic. Correcting the increment allowed the loop to terminate properly.

The loop now executes the expected number of times.

AI explanation clearly identifies and resolves the logical error.

Task-03:

Prompt:

Identify the runtime error in the following Python function that performs division without validation. Fix the code using proper try-except blocks to handle division by zero safely. Ensure the program does not crash and displays an appropriate error message. Explain how the runtime error occurs and how the error handling resolves it.

Code:

```
Assign-7.3.py > ...
1  '''Identify the runtime error in the following Python function that performs division without validation.
2  Fix the code using proper try-except blocks to handle division by zero safely.
3  Ensure the program does not crash and displays an appropriate error message.
4  Explain how the runtime error occurs and how the error handling resolves it.'''
5  def safe_divide(num, denom):
6      try:
7          result = num / denom
8          return result
9      except ZeroDivisionError:
10         return "Error: Division by zero is not allowed."
11
12 # Example usage
13 if __name__ == "__main__":
14     print(safe_divide(10, 2)) # Should print 5.0
15     print(safe_divide(10, 0)) # Should print error message
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/python.exe c
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
5.0
Error: Division by zero is not allowed.
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

Division by zero is a common runtime error in Python. AI identified the issue and added proper exception handling. The try-except block

prevents program crashes. Safe execution is ensured with meaningful error messages. This improves program reliability and robustness.

Task-04:

Prompt:

Identify and fix the error in the following Python class where the constructor is defined without the self parameter. Correct the class definition and explain why self is required in object-oriented programming.

Code:

```
Assign-7.3.py > ...
1  '''Identify and fix the error in the following Python class where the constructor is defined without the self parameter
2  Correct the class definition and explain why self is required in object-oriented programming.'''
3  class Student:
4      def __init__(name, roll):
5          name = name
6          roll = roll
7  # Corrected class definition
8  class Student:
9      def __init__(self, name, roll):
10         self.name = name
11         self.roll = roll
12  # Example usage
13  student1 = Student("Alice", 101)
14  print(f"Student Name: {student1.name}, Roll Number: {student1.roll}")
15  |
```

Output:

```
Student Name: Alice, Roll Number: 101
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/python.exe
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
Student Name: Alice, Roll Number: 101
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/python.exe
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
Student Name: Alice, Roll Number: 101
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> |
```

Explanation:

self refers to the current object instance. It allows each object to store its own data. Without self, variables become local, not object attributes. Python automatically passes the object reference as self. Using self ensures proper object-oriented behavior.

Task-05:

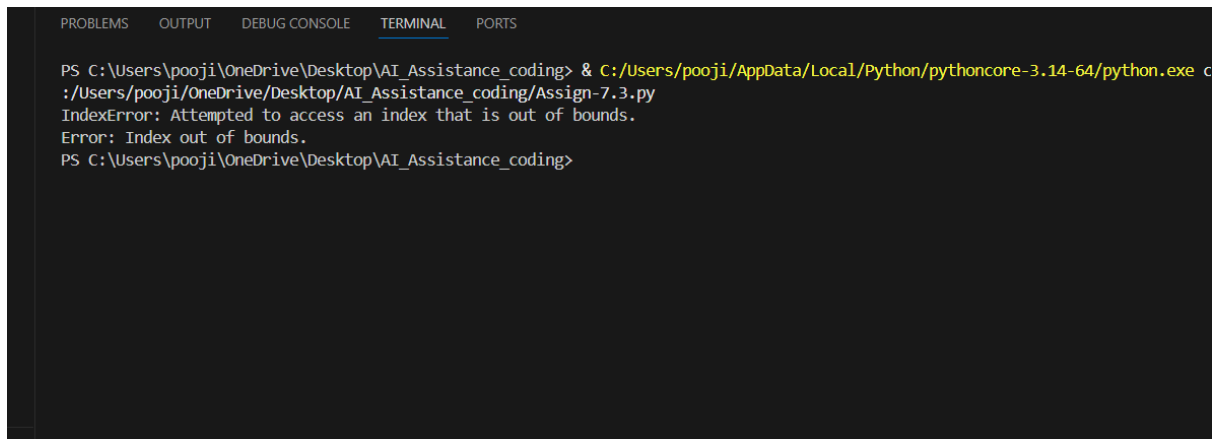
Prompt:

Identify and fix the index error in the following Python program where an invalid list index is accessed. Suggest safe access techniques using bounds checking or exception handling. Explain how the error occurs and how the fix resolves it.

Code:

```
Assign-7.3.py > ...
'''Identify and fix the index error in the following Python program where an invalid list index is accessed.
Suggest safe access techniques using bounds checking or exception handling.
Explain how the error occurs and how the fix resolves it.'''
# Original code with index error
my_list = [10, 20, 30, 40, 50]
# Attempting to access an invalid index
try:
    print(my_list[5]) # This will raise an IndexError
except IndexError:
    print("IndexError: Attempted to access an index that is out of bounds.")
# Fixed code with safe access techniques
def safe_access(lst, index):
    if 0 <= index < len(lst):
        return lst[index]
    else:
        return "Error: Index out of bounds."
# Using the safe access function
print(safe_access(my_list, 5)) # Safely handles out-of-bounds access
```

Output:

A screenshot of a Windows terminal window. The title bar at the top shows tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), and 'PORTS'. The terminal text shows a PowerShell prompt 'PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>' followed by a command to run a Python script. The output shows an 'IndexError' message: 'IndexError: Attempted to access an index that is out of bounds.' and a more detailed 'Error: Index out of bounds.' message. The prompt returns to 'PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>'.

```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Python/pythoncore-3.14-64/python.exe c
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Assign-7.3.py
IndexError: Attempted to access an index that is out of bounds.
Error: Index out of bounds.
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

IndexError occurs when accessing an invalid index. Length checking ensures index validity. try-except prevents program crashes. Both methods improve program robustness. Safe access is essential for reliable applications.