SR UNIVERSITY

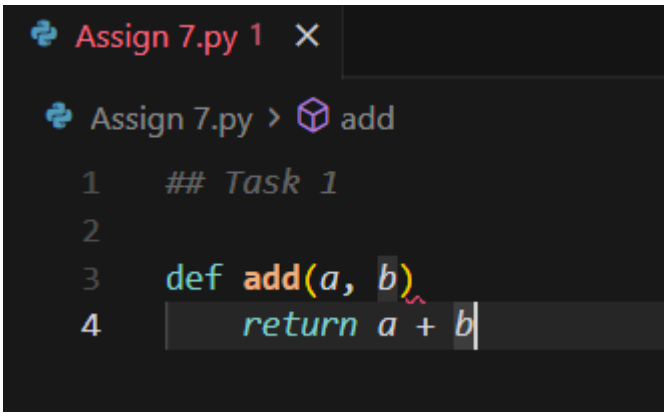
AI Assistant Coding
Assignment-7.3

Name of Student : V. Abhinav **Batch : 41**
Enrollment No. : 2303A52486

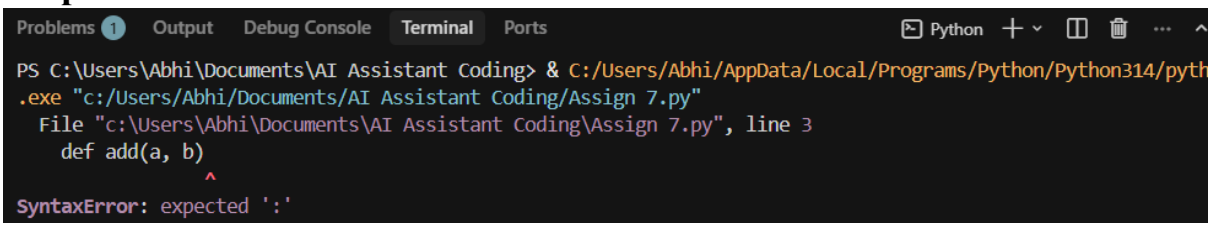
Task 1: Fixing Syntax Errors

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

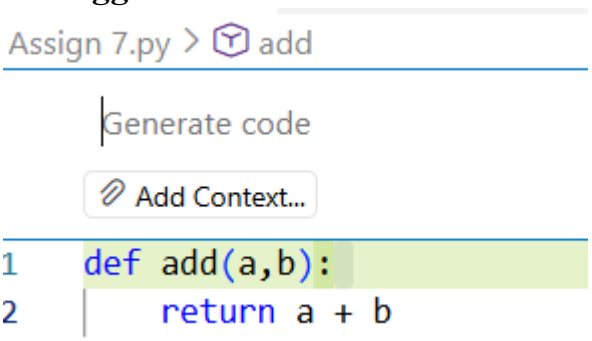
Code with error:



Output:



AI Suggested Correction:



Explanation:

The AI detected a syntax error because the function declaration line did not end with a colon. In Python, the colon indicates the start of an indented code block. Without it, the interpreter cannot parse the function body.

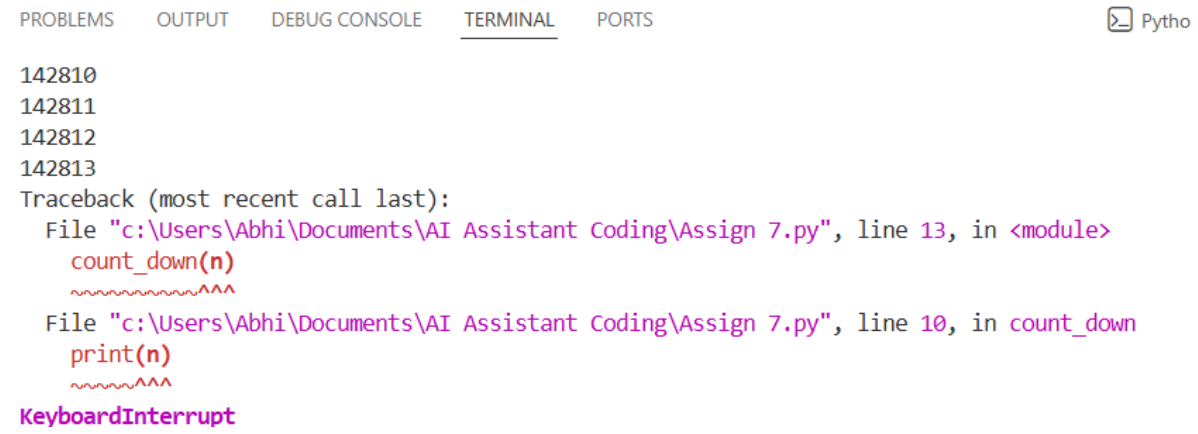
Task 2: Debugging Logic Errors in Loops

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

Buggy Code:

```
# Task-2: Find the logic error and debug it
n = int(input("Enter a number to count down from: "))
def count_down(n):
    while n > 0:
        print(n)
        n += 1
```

Output:



Error Identified by AI

Type of Error: Logical Error
Reason: The loop variable is decremented instead of incremented, so the condition i <= n is always true.

AI-Corrected Code:

```
5
6   # Task-2: Find the logic error and debug it
7   n = int(input("Enter a number to count down from: "))
8   def count_down(n):
9       while n > 0:
10          print(n)
11          n += 1
12              -
13  count_down(n)
```

Output:

```
PS C:\Users\Abhi\Documents\AI Assistant Coding> & C:/Users/Abhi
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Enter a number to count down from: 5
5
4
3
2
1
```

Explanation:

The loop condition depends on i increasing toward a stopping point. Decrementing i causes it to move away from the termination condition, resulting in an infinite loop.

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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

Buggy Code:

```
def divide_numbers(a, b):
    return a/b

print(divide_numbers(10, 0))
```

Output:

```
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Traceback (most recent call last):
  File "c:\Users\Abhi\Documents\AI Assistant Coding\Assign 7.py", line 21, in <module>
    print(divide_numbers(10, 0))
          ~~~~~~~~~~~~~~~^^^^^^^
  File "c:\Users\Abhi\Documents\AI Assistant Coding\Assign 7.py", line 19, in divide_numbers
    return a/b
           ~^~
ZeroDivisionError: division by zero
```

Error Identified by AI:

Type of Error: Runtime Error (ZeroDivisionError)

Reason: Division by zero is mathematically undefined.

AI-Corrected Code (Using try-except):

```
# Debug the following code to handle division by zero error.
def divide_numbers(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
    else:
        return result
```

Output:

```
PS C:\Users\Abhi\Documents\AI Assistant Coding> & C:/User
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Enter the numerator: 10
Enter the denominator: 0
Error: Division by zero is not allowed.
```

Explanation:

The AI added exception handling using try-except to prevent program termination. This ensures graceful handling of invalid input instead of crashing the program.

Task 4: Debugging Class Definition Errors

Scenario:

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

Buggy Code:

```
class Student:
    def __init__(name, age):
        name = name
        age = age

# Create an instance of the Student class
student1 = Student("Alice", 20)
```

Output:

```
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Traceback (most recent call last):
  File "c:\Users\Abhi\Documents\AI Assistant Coding\Assign 7.py", line 38, in <module>
    student1 = Student("Alice", 20)
TypeError: Student.__init__() takes 2 positional arguments but 3 were given
```

Error Identified by AI

Type of Error: TypeError / Object-Oriented Error
Reason: The **self** parameter is mandatory for instance variable access.

AI-Corrected Code:

```
31   #  Task-4: Class Definition Error
32   class Student:
33       def __init__(self, name, age):
34           name = name    self.name = name
35           age = age      self.age = age
36
37   # Create an instance of the Student class
38   student1 = Student("Alice", 20)
```

Output:

```
PS C:\Users\Abhi\Documents\AI Assistant Coding> & C:/User
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Alice 20
```

Explanation:

The self parameter represents the current object instance. Without it, instance variables cannot be properly assigned or accessed.

Task 5: Resolving Index Errors in Lists
Scenario:
A program crashes when accessing an invalid index in a list.

Buggy Code:

```
# Task-5: Resolving Index Errors in Lists

numbers = [10, 20, 30]
print(numbers[5])
```

Output:

```
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Traceback (most recent call last):
  File "c:\Users\Abhi\Documents\AI Assistant Coding\Assign 7.py", line 44, in <module>
    print(numbers[5])
          ~~~~~~~^^^
IndexError: list index out of range
```

Error Identified by AI:

Type of Error: IndexError
Reason: Index is outside the valid range of the list.

AI-Corrected Code (Bounds Checking):

```
43  # resolve the index error in the following code by Bounds Checking.
44  numbers = [10, 20, 30]
45  print(numbers[5])
```

```
if 5 < len(numbers):
    print(numbers[5])
else:
    print("Index out of range.")
```

Output:

```
PS C:\Users\Abhi\Documents\AI Assistant Coding> & C:/U
/Users/Abhi/Documents/AI Assistant Coding/Assign 7.py"
Index out of range.
```

Explanation:

This lab demonstrated how AI can be effectively used as a debugging assistant. Through multiple scenarios, AI successfully identified syntax, logic, runtime, object-oriented, and index-related errors. The explanations provided by AI helped in understanding not only what went wrong, but also why it went wrong.

AI-assisted debugging improves productivity, reduces development time, and reinforces best programming practices. However, human understanding remains essential to evaluate and apply AI suggestions correctly.