ASSIGNMENT 7.3

Task 1: Fixing Syntax Errors

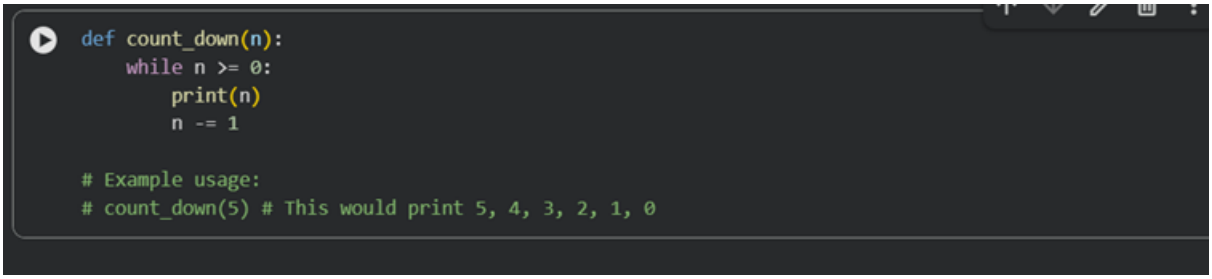
Scenario

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

Task 2: Debugging Logic Errors in Loops

Scenario

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



Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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

```
[20]
✓ 0s
def divide_numbers(x, y):
    """Divides two numbers with error handling for ZeroDivisionError."""
    try:
        return x / y
    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")
        return None  # Or raise a custom error, or return a specific value

# Example usage with error handling
print(f"10 divided by 2 is: {divide_numbers(10, 2)}")
print(f"5 divided by 0 is: {divide_numbers(5, 0)}")
print(f"100 divided by 10 is: {divide_numbers(100, 10)}")

10 divided by 2 is: 5.0
Error: Cannot divide by zero!
5 divided by 0 is: None
100 divided by 10 is: 10.0
```

Task 4: Debugging Class Definition Errors

Scenario

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

```
[21]
✓ 0s
class MyClass:
    def __init__(name, value): # Missing 'self' parameter
        name.name = name
        name.value = value

    def display(self):
        print(f"Name: {self.name}, Value: {self.value}")

# Attempt to create an instance (this will raise an error after definition)
# obj = MyClass("test_name", 10)
```

+ Code + Text

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid index in a list

[23]
✓ 0s

```
my_list = [10, 20, 30]
print(f"The list is: {my_list}")

# Safely accessing list elements

# 1. Using bounds checking (if statement)
index_to_access = 3
if 0 <= index_to_access < len(my_list):
    print(f"Safely accessing index {index_to_access}: {my_list[index_to_access]}")
else:
    print(f"Error: Index {index_to_access} is out of range for a list of length {len(my_list)}")

index_to_access = 1
if 0 <= index_to_access < len(my_list):
    print(f"Safely accessing index {index_to_access}: {my_list[index_to_access]}")
else:
    print(f"Error: Index {index_to_access} is out of range for a list of length {len(my_list)}")

# 2. Using exception handling (try-except block)
try:
    print(f"Attempting to access index 3 using try-except: {my_list[3]}")
except IndexError:
    print("Error: Tried to access an index that is out of range!")
```

```
print("Error: Tried to access an index that is out of range!")

try:
    print(f"Attempting to access index 0 using try-except: {my_list[0]}")
except IndexError:
    print("Error: Tried to access an index that is out of range!")
```

```
The list is: [10, 20, 30]
Error: Index 3 is out of range for a list of length 3
Safely accessing index 1: 20
Error: Tried to access an index that is out of range!
Attempting to access index 0 using try-except: 10
```