Debugging and Algorithm Optimization – Finding the Minimum Value

Project Description

This project is a debugging exercise where I identified and fixed logic errors in a function designed to find the smallest number in a list without using Python's built-in min() function. The improved version handles empty lists, avoids incorrect assumptions, and includes clear, maintainable code. The project showcases problem analysis, algorithm improvement, and robust coding.

Problem Analysis

The goal of the function was to find the minimum value in a list without using Python's built-in min() function. However, the code below does not always work correctly:

```
def minimum(some_list):
    a = 0
    for x in range(1, len(some_list)):
        if some_list[x] < a:
            a = some_list[x]
    return a
```

What Is Wrong?

1. Incorrect initial value:

   The variable a is initialized to 0. This assumes that 0 is the lowest value in the list, which is not always true. For example, in the list [5, 6, 7], the correct minimum is 5, but this function will return 0, which isn't even in the list.

2. Skipping the first element:

   The loop starts at index 1, so the function never compares the first element of the list (some_list[0]). If the first item is the smallest, it will be ignored.

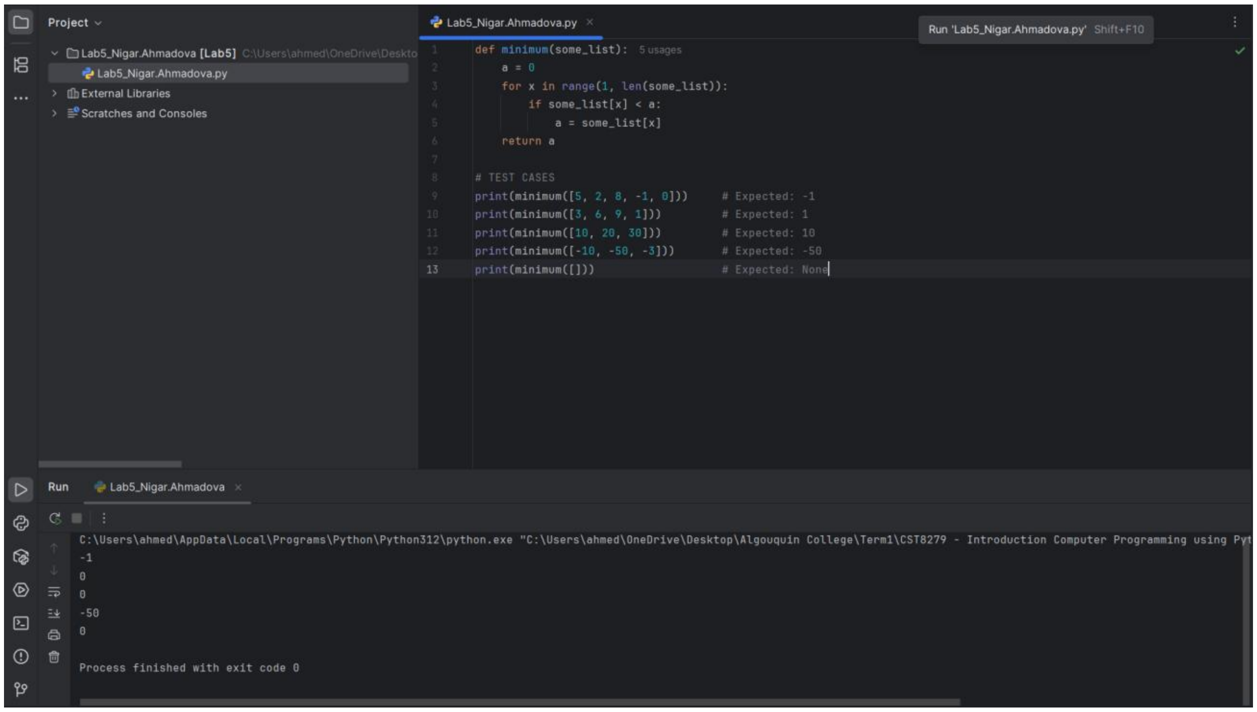
What Needs to be Corrected:

1. a = some_list[0]: The function now uses the first actual element of the list as the starting value, which avoids incorrect assumptions.
2. Starts comparing from index 1 (which is now fine because index 0 is already stored in a).
3. if not some_list: handles empty list cases gracefully and avoids errors.

Corrected Code:

```
def minimum(some_list):
    if not some_list:
        return None          # Handle empty list
    a = some_list[0]         # Start with the first element
    for x in range(1, len(some_list)):
        if some_list[x] < a:
            a = some_list[x]
    return a
```

Incorrect Code:



Lab5_Nigar.Ahmadova.py

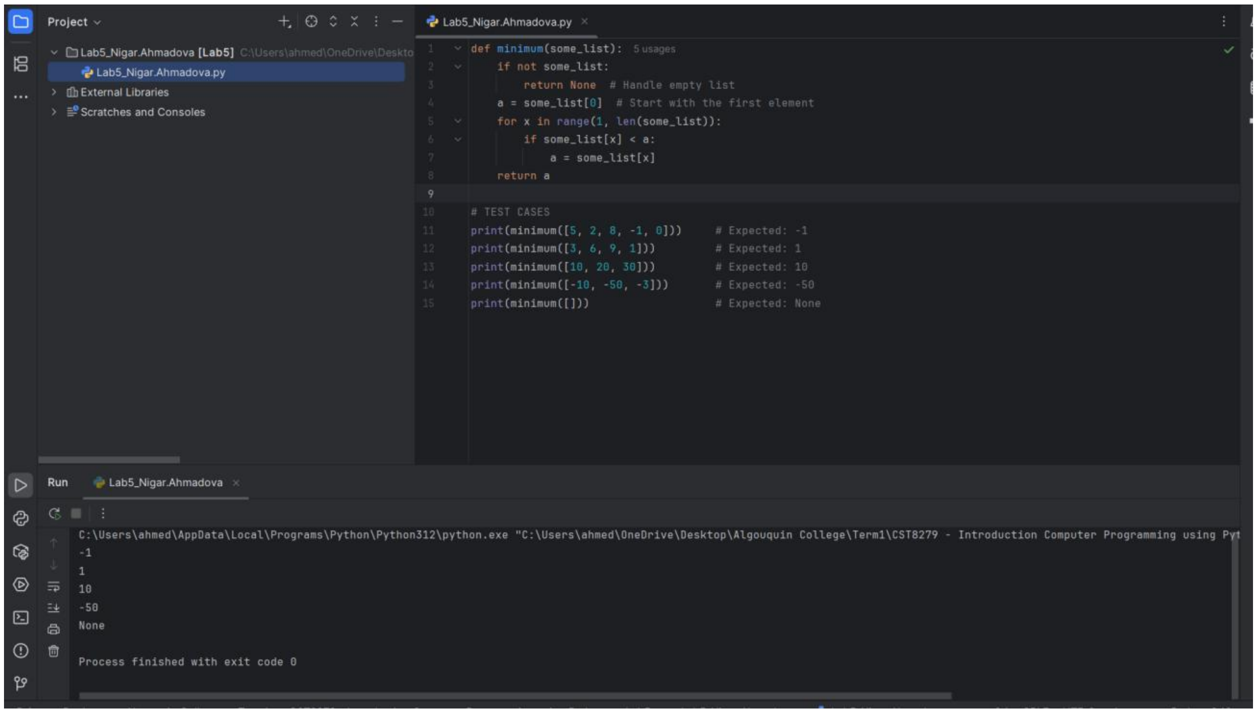
```python
def minimum(some_list):  5 usages
    a = 0
    for x in range(1, len(some_list)):
        if some_list[x] < a:
            a = some_list[x]
    return a

# TEST CASES
print(minimum([5, 2, 8, -1, 0]))      # Expected: -1
print(minimum([3, 6, 9, 1]))          # Expected: 1
print(minimum([10, 20, 30]))          # Expected: 10
print(minimum([-10, -50, -3]))        # Expected: -50
print(minimum([]))                    # Expected: None
```

```
C:\Users\ahmed\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\ahmed\OneDrive\Desktop\Algouquin College\Term1\CST8279 - Introduction Computer Programming using Pyt
-1
0
0
-50
0

Process finished with exit code 0
```

Correct Code:



Lab5_Nigar.Ahmadova.py

```python
def minimum(some_list):  5 usages
    if not some_list:
        return None  # Handle empty list
    a = some_list[0]  # Start with the first element
    for x in range(1, len(some_list)):
        if some_list[x] < a:
            a = some_list[x]
    return a

# TEST CASES
print(minimum([5, 2, 8, -1, 0]))      # Expected: -1
print(minimum([3, 6, 9, 1]))          # Expected: 1
print(minimum([10, 20, 30]))          # Expected: 10
print(minimum([-10, -50, -3]))        # Expected: -50
print(minimum([]))                    # Expected: None
```

```
C:\Users\ahmed\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\ahmed\OneDrive\Desktop\Algouquin College\Term1\CST8279 - Introduction Computer Programming using Pyt
-1
1
10
-50
None

Process finished with exit code 0
```

References

1. Python Software Foundation. (n.d.). *Built-in Functions*. Python.org. https://docs.python.org/3/library/functions.html
2. Purdue OWL. (n.d.). *APA Formatting and Style Guide*. OWL at Purdue. https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/general_format.html
3. JetBrains. (n.d.). *PyCharm*. https://www.jetbrains.com/pycharm/