

AI Assisted Coding-Lab 7.3

Shivani Pambidi

2303A52172

Batch – 41

Question 1:

TASK 1 : Fixing Syntax Errors

CODE:

```
Lab 7.py > add
1 def add(a,b)
2     return a+b
```

EXPLANATION:

SYNTAX ERROR IN CODE SNIPPET

@workspace /explain Expected ":"

Workspace

> Used 1 reference

✓ Finalized code instructions and explained errors

Problem: your function definition is missing a colon. Add it and fix spacing.

```
# ...existing code...
def add(a, b):
    return a + b
# ...existing code...
```

Notes:

- Python requires a colon after `def` headers; VS Code/error "Expected ':'" points to that.
- Small improvements: add spaces around commas and `PEP8` or optional type hints:

```
def add(a: int, b: int) -> int:
    """Return the sum of a and b."""
    return a + b
```

JUSTIFICATION:

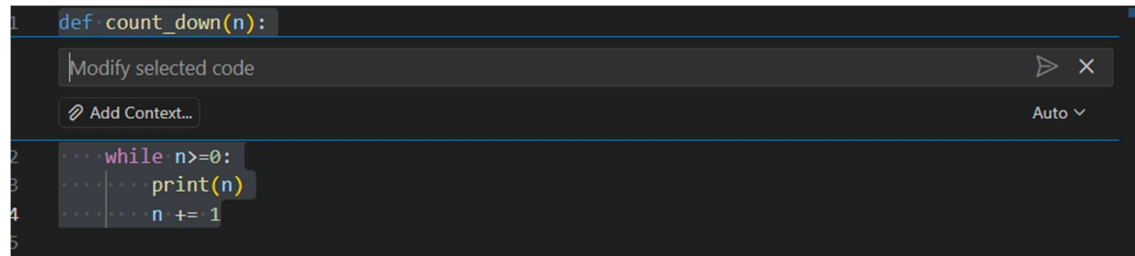
The function had a syntax error because the colon was missing after the function definition. Python requires a colon to start the function body. An AI

tool identified this error and fixed it by adding the missing colon. After the correction, the function executed properly and the syntax error was resolved.

Question 2:

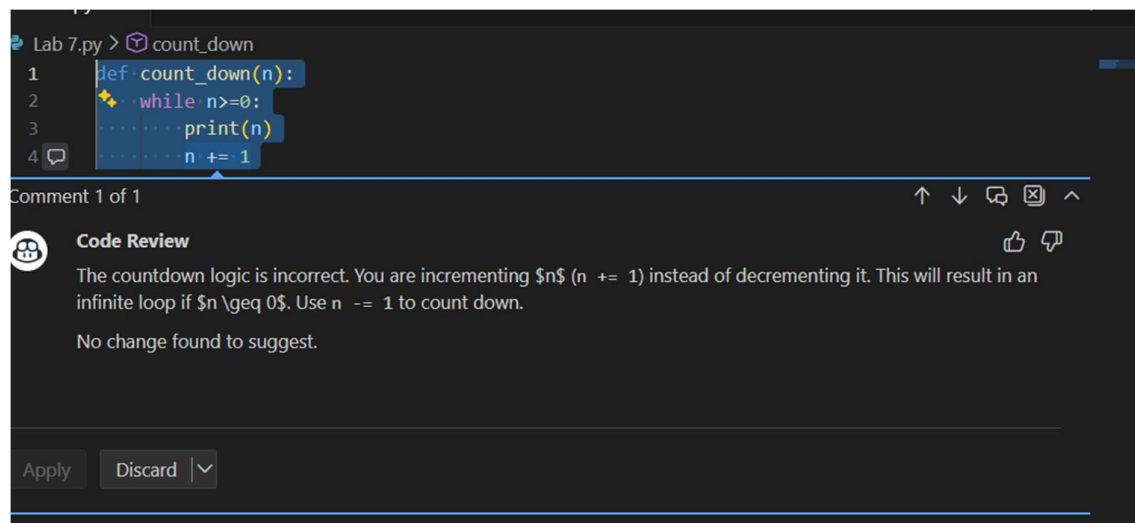
TASK 2: Debugging Logic Errors in Loops

MODIFY:



```
1 def count_down(n):  
2     while n>=0:  
3         print(n)  
4         n += 1  
5
```

REVIEW:



```
1 def count_down(n):  
2     while n>=0:  
3         print(n)  
4         n += 1
```

Comment 1 of 1

Code Review

The countdown logic is incorrect. You are incrementing n ($n += 1$) instead of decrementing it. This will result in an infinite loop if $n \geq 0$. Use $n -= 1$ to count down.

No change found to suggest.

Apply Discard

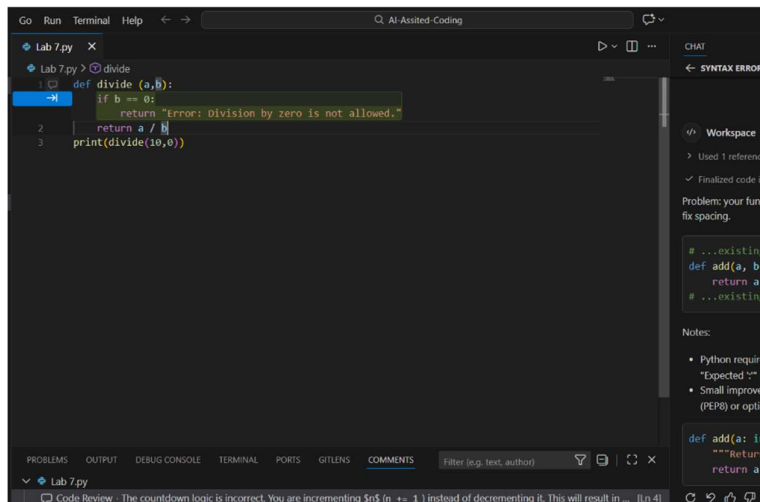
JUSTIFICATION:

The loop was running infinitely because the value of n was increasing instead of decreasing. The condition $n \geq 0$ always remained true since n never moved toward zero. An AI tool identified this logical mistake by analyzing the loop condition and update statement. The AI corrected the logic by changing the increment to a decrement so that n reduces in each iteration. After this fix, the loop stops correctly at zero and the infinite loop issue is resolved.

Question 3:

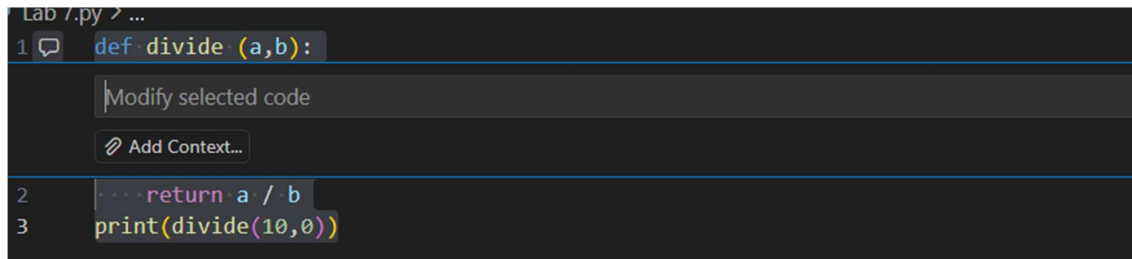
TASK 3 : Handling Runtime Errors (Division by Zero)

EXTRACT:



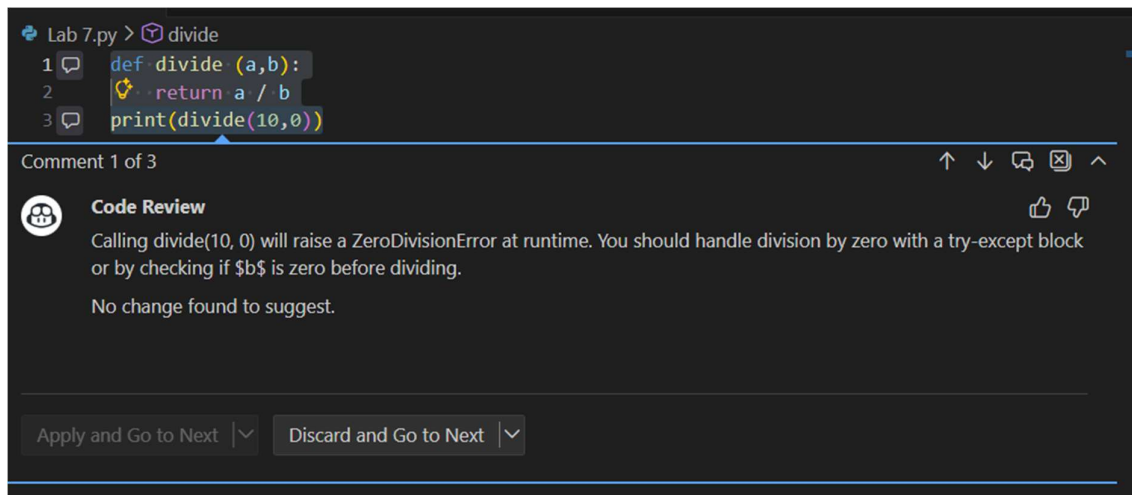
```
1 def divide(a,b):
2     try:
3         return a / b
4     except ZeroDivisionError:
5         return "Error: Division by zero is not allowed."
6 print(divide(10,0))
```

MODIFY:



```
1 def divide(a,b):
2     return a / b
3 print(divide(10,0))
```

REVIEW:



```
1 def divide(a,b):
2     return a / b
3 print(divide(10,0))
```

Comment 1 of 3

Code Review

Calling divide(10, 0) will raise a ZeroDivisionError at runtime. You should handle division by zero with a try-except block or by checking if \$b\$ is zero before dividing.

No change found to suggest.

Apply and Go to Next | Discard and Go to Next

```
Lab 7.py > divide
1 def divide (a,b):
2     return a / b
3     print(divide(10,0))
```

Comment 1 of 1

Code Review

Calling divide(10, 0) will raise a ZeroDivisionError at runtime because division by zero is not allowed in Python. Consider adding error handling to manage this case gracefully.

No change found to suggest.

Apply Discard

JUSTIFICATION:

The program crashes because it tries to divide a number by zero, which is not allowed in Python. An AI tool identifies this as a runtime error that occurs during execution. The AI fixes the issue by adding a try-except block to catch the division by zero error. With this handling, the program runs safely without crashing and shows a proper message instead.

Question 4:

TASK 4: Debugging Class Definition Errors

FIX:

```
Lab 7.py > Rectangle > __init__
1 class Rectangle:
2     def __init__(length,width):
3     def __init__(self,length,width):
4         self.length=length
5         self.width=width
```

EXPLANATION:

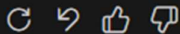
Resolved Python coding issues

Problem: the instance method is missing the required first parameter (conventionally named `self`), so references to `self` in the body are undefined. Fix by adding `self` as the first parameter.

```
# ...existing code...
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width
# ...existing code...
```

Notes:

- Methods must accept the instance as the first parameter (use `self` by convention).
- Optional improvements: add type hints (`def __init__(self, length: float, width: float) -> None:`), validation, or helper methods like `area()` or `__repr__()`.



REVIEW:

Lab 7.py > Rectangle

```
1 class Rectangle:
2     def __init__(length,width):
```

Comment 1 of 4

Code Review

The `__init__` method is missing the `self` parameter. It should be `def __init__(self, length, width):`.

No change found to suggest.

Apply and Go to Next | Discard and Go to Next

```
3         self.length=length
4         self.width=width
5
```

Lab 7.py > Rectangle

```
1 class Rectangle:
2     def __init__(length,width):
```

Comment 1 of 3

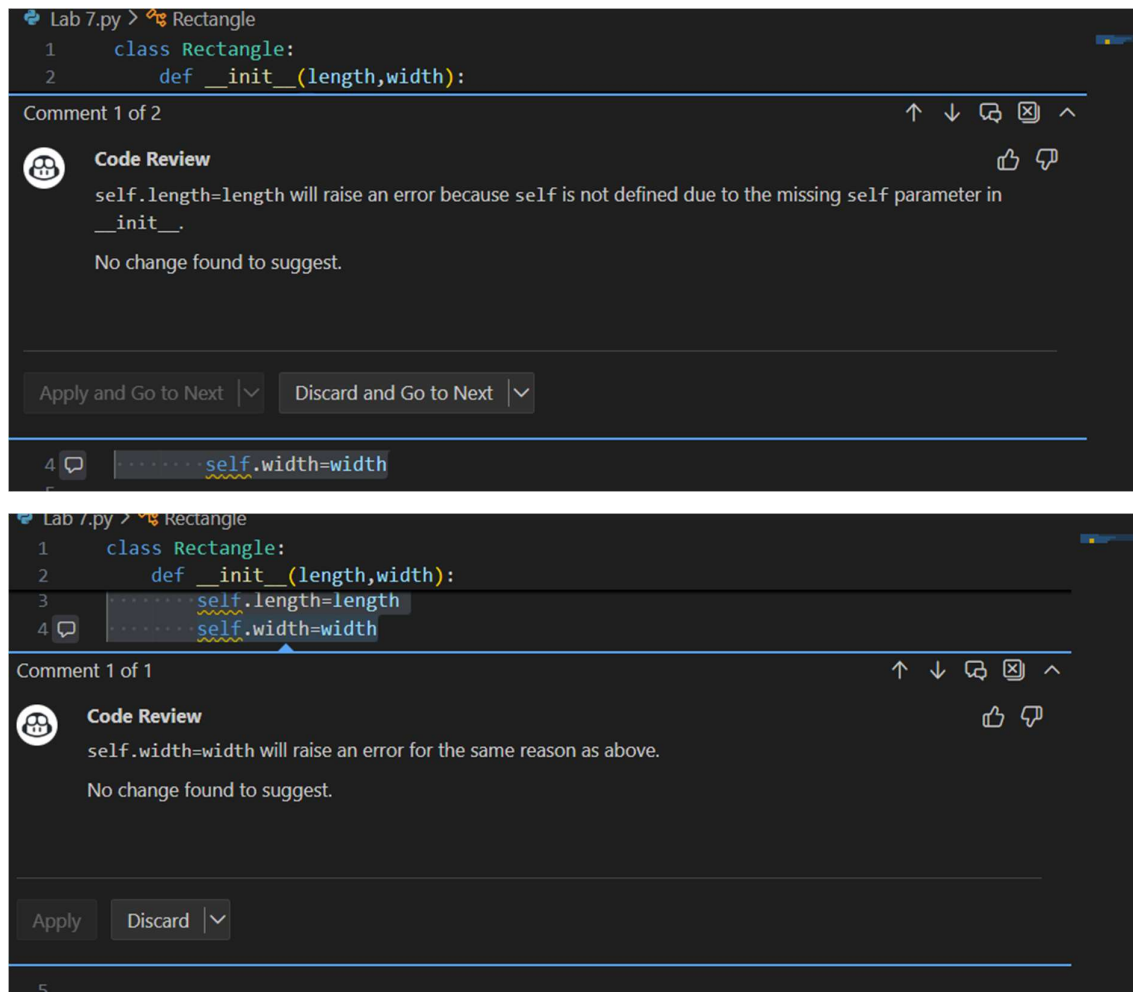
Code Review

There should be a space after the comma in the parameter list for readability (`length, width`).

No change found to suggest.

Apply and Go to Next | Discard and Go to Next

```
3         self.length=length
4         self.width=width
5
```



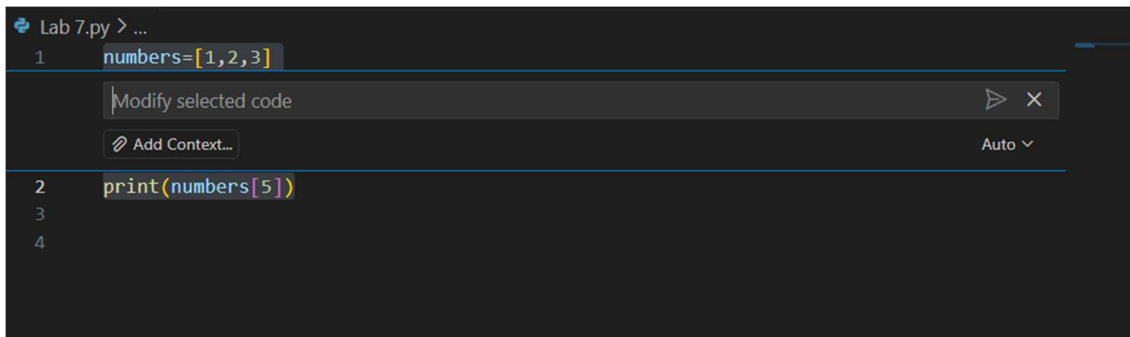
JUSTIFICATION:

The class had an error because the `__init__` method was missing the `self` parameter. In Python, `self` is required to refer to the current object inside the class. An AI tool detected this object-oriented mistake and corrected the constructor by adding `self` as the first parameter. After the fix, object attributes are assigned correctly and the class works as expected.

Question 5:

TASK 5: Resolving Index Errors in Lists

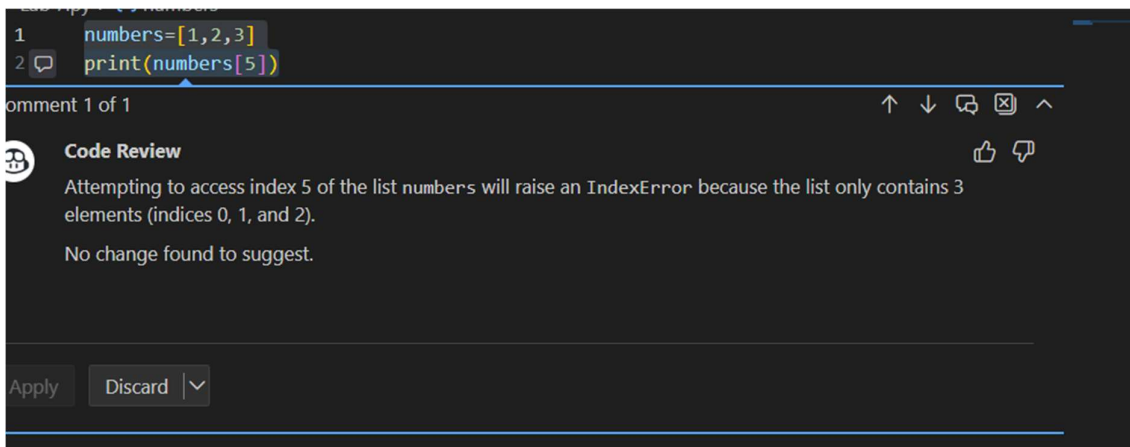
MODIFY:



```
Lab 7.py > ...  
1 numbers=[1,2,3]  
2 print(numbers[5])  
3  
4
```

The screenshot shows a code editor with a dark theme. The file is named 'Lab 7.py'. Line 1 contains the code `numbers=[1,2,3]`. Line 2 contains `print(numbers[5])`. A context menu is open over line 2, showing options like 'Modify selected code', 'Add Context...', and a play button. The list 'numbers' has 3 elements, but the code attempts to access index 5, which is out of range.

REVIEW:



```
1 numbers=[1,2,3]  
2 print(numbers[5])
```

Comment 1 of 1

Code Review

Attempting to access index 5 of the list `numbers` will raise an `IndexError` because the list only contains 3 elements (indices 0, 1, and 2).

No change found to suggest.

Apply Discard

The screenshot shows a code review interface. The code from the previous screenshot is visible at the top. A comment is shown, stating that accessing index 5 of the list `numbers` will raise an `IndexError` because the list only contains 3 elements (indices 0, 1, and 2). The comment also states 'No change found to suggest.' At the bottom, there are buttons for 'Apply' and 'Discard'.

JUSTIFICATION:

The program crashes because it tries to access an index that does not exist in the list. An AI tool identifies this as an `IndexError` caused by an out-of-range index value. The AI suggests fixing it by checking the list length or using exception handling before accessing the element. After applying safe access logic, the program runs without crashing and the index error is resolved.