

AI Assisted Coding-Lab 7.3

Shivani Pambidi

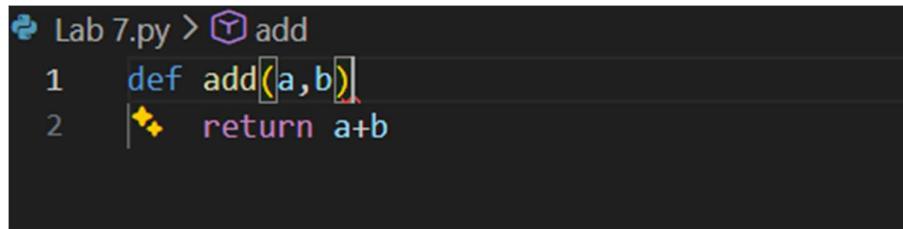
2303A52172

Batch – 41

Question 1:

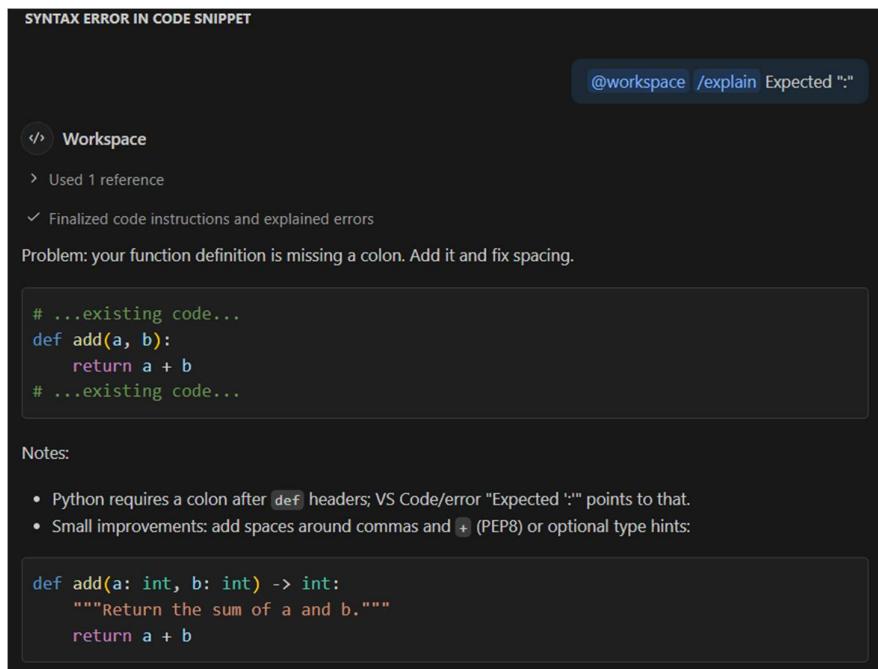
TASK 1 : Fixing Syntax Errors

CODE:



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

EXPLANATION:



JUSTIFICATION:

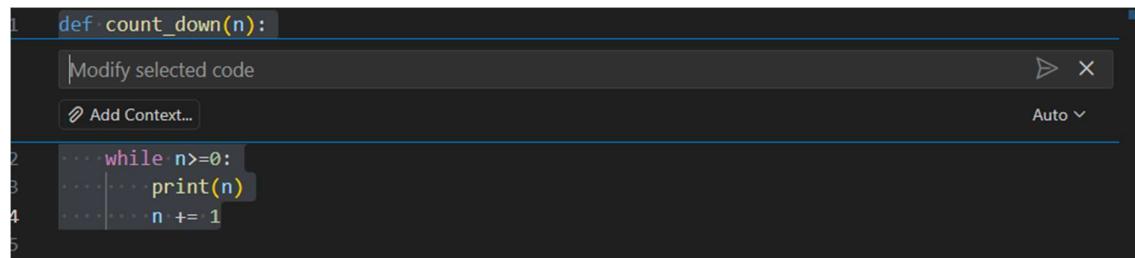
The function had a syntax error because the colon was missing after the function definition. Python requires a colon to start the function body. An AI

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:

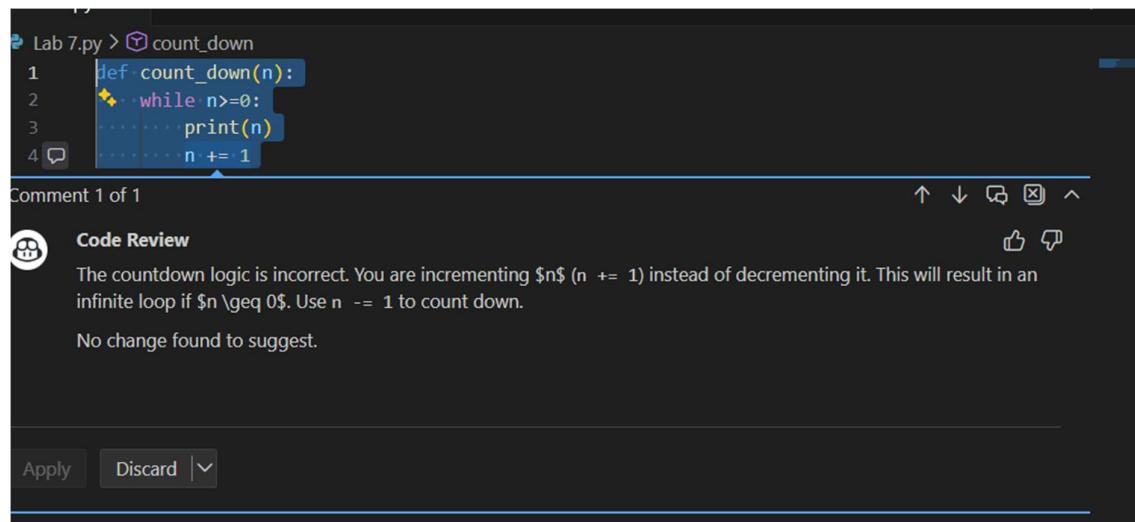
TASK 2: Debugging Logic Errors in Loops

MODIFY:



```
1 def count_down(n):
    Modify selected code
    Add Context...
    Auto ▾
2     while n>=0:
3         print(n)
4         n += 1
5
```

REVIEW:



Lab 7.py > count_down

```
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

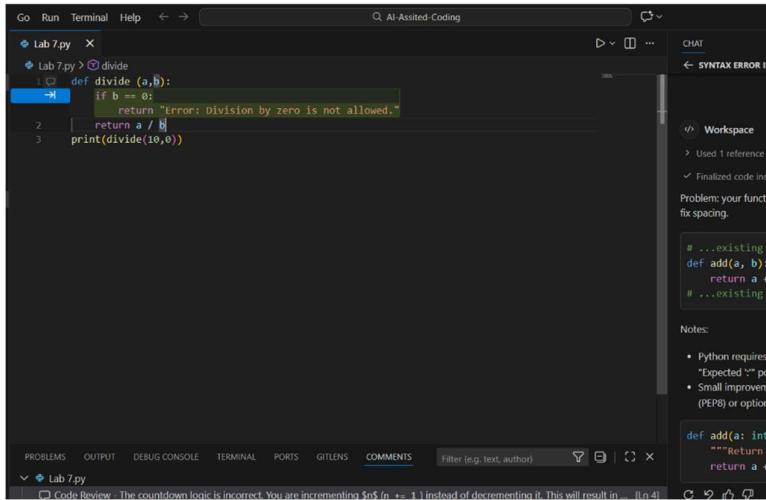
JUSTIFICATION:

The loop was running infinitely because the value of n was increasing instead of decreasing. The condition while $n \geq 0$ 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:

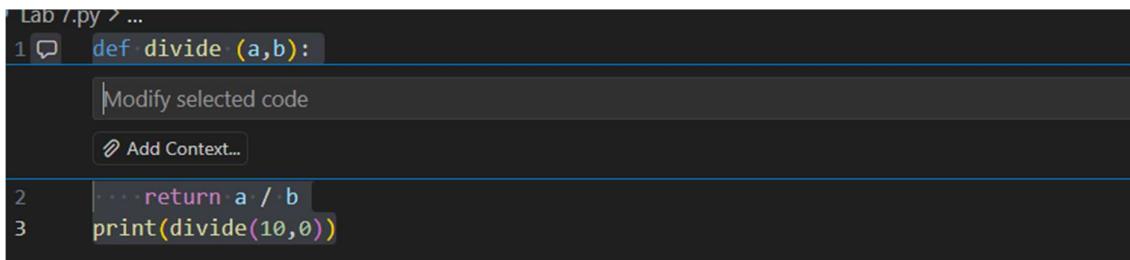


A screenshot of the VS Code interface. The top bar shows 'Go', 'Run', 'Terminal', 'Help', and other standard menu items. The main editor window displays a Python file named 'Lab 7.py' with the following code:

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

An error message is shown in the status bar: "SYNTAX ERROR IN Lab 7.py". The right-hand sidebar includes a 'CHAT' section, a 'Workspace' tree, and a 'Notes' section containing bullet points about Python's requirements for division.

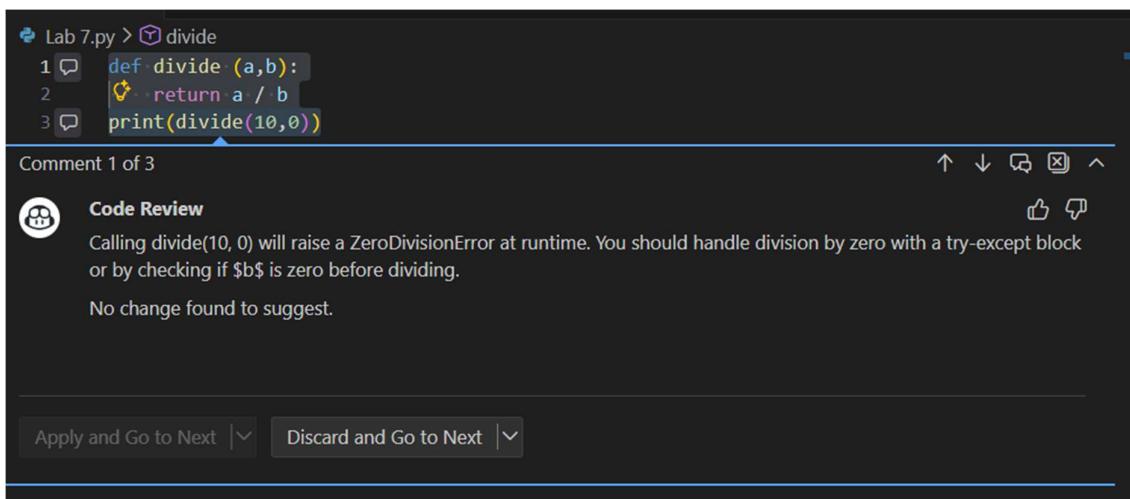
MODIFY:



A screenshot of the VS Code interface showing the 'Modify selected code' feature. The code editor has a modal open over it, prompting the user to 'Modify selected code'. The code being modified is the same as in the previous screenshot:

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

REVIEW:



A screenshot of the VS Code interface showing a 'Code Review' comment. The comment is from 'Comment 1 of 3' and reads:

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.

At the bottom of the review panel, there are two buttons: 'Apply and Go to Next' and 'Discard and Go to Next'.

The screenshot shows a code editor window titled "Lab 7.py > divide". The code contains a function definition:`1 def divide(a,b):
2 return a / b
3 print(divide(10,0))`A blue arrow points from the line "return a / b" to a comment at the bottom of the screen: "Comment 1 of 1". The comment reads: "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." Below the comment are two buttons: "Apply" and "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:

The screenshot shows a code editor window titled "Lab 7.py > Rectangle > __init__". The code defines a class "Rectangle" with an __init__ method:`1 class Rectangle:
2 def __init__(length,width):
3 self.length=length
4 self.width=width
5`An AI tool has suggested a fix: "def __init__(self,length,width):". A modal dialog box titled "Modify selected code" contains this suggestion. The "Add Context..." button is visible below the input field. The "Auto" dropdown menu is open.

EXPLANATION:

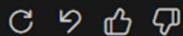
• Resolved Python coding issues

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
```

```
Lab 7.py > Rectangle
1     class Rectangle:
2         def __init__(length,width):
Comment 1 of 2
Code Review
self.length=length will raise an error because self is not defined due to the missing self parameter in __init__.
No change found to suggest.

Apply and Go to Next | Discard and Go to Next | 

4 💬 self.length=length
5 💬 self.width=width

Lab 7.py > Rectangle
1     class Rectangle:
2         def __init__(length,width):
3             self.length=length
4             self.width=width
Comment 1 of 1
Code Review
self.width=width will raise an error for the same reason as above.
No change found to suggest.

Apply | Discard | 
```

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:

A screenshot of a code editor window titled "Lab 7.py > ...". The code is as follows:

```
1 numbers=[1,2,3]
2 print(numbers[5])
3
4
```

The line "print(numbers[5])" is highlighted with a yellow selection bar. At the top of the editor, there is a toolbar with a "Modify selected code" button, a "Save" icon, and a "X" icon. Below the toolbar, there is a "Add Context..." button and a dropdown menu set to "Auto".

REVIEW:

A screenshot of a code review interface. The code is the same as above:

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

Comment 1 of 1: 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.

At the bottom, there are "Apply" and "Discard" buttons.

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.