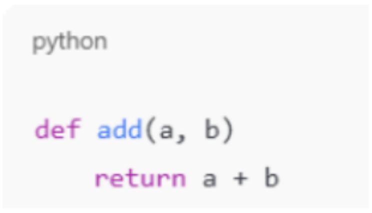
AI ASSISTANT CODING - ASSIGNMENT-7.3

Name: Valaboju Harshavardhanu **Roll No:** 2303A52354 **Batch:** 41

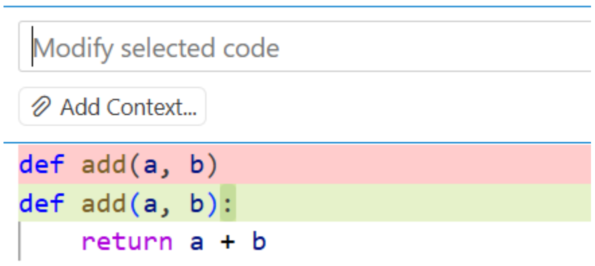
Task 1: Fixing Syntax Errors

Scenario:



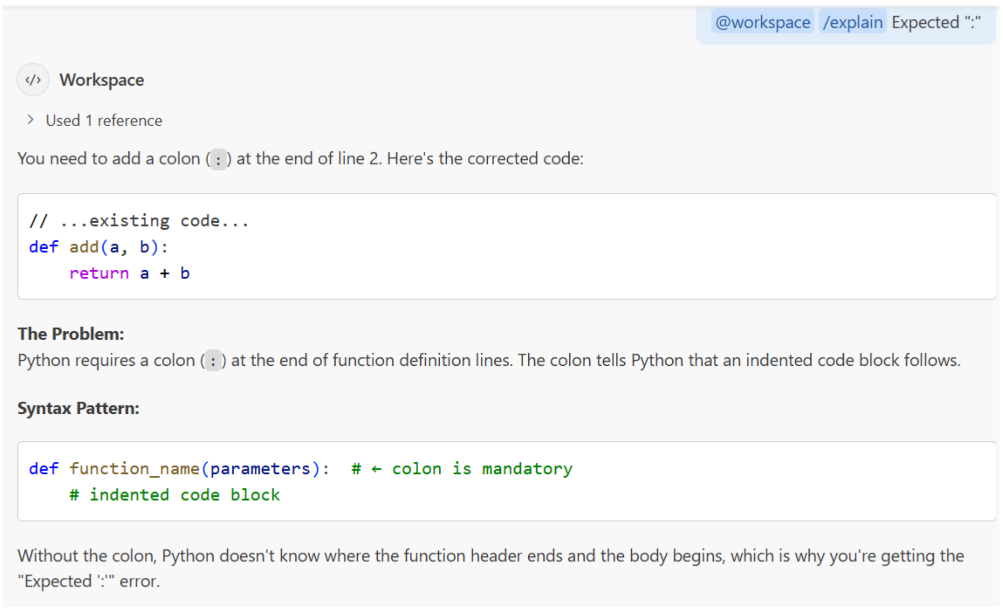
```python
def add(a, b)
    return a + b
```

Error Fix:



```python
def add(a, b)
def add(a, b):
    return a + b
```

Error Explanation:



You need to add a colon (`:`) at the end of line 2. Here's the corrected code:

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

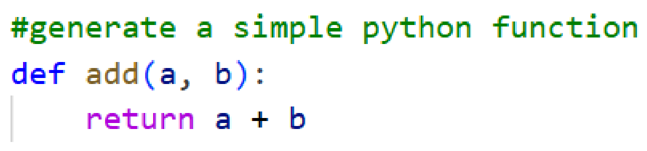
The Problem:
Python requires a colon (`:`) at the end of function definition lines. The colon tells Python that an indented code block follows.

Syntax Pattern:

```
def function_name(parameters):  # ← colon is mandatory
    # indented code block
```

Without the colon, Python doesn't know where the function header ends and the body begins, which is why you're getting the "Expected ':'" error.

AI Corrected code:



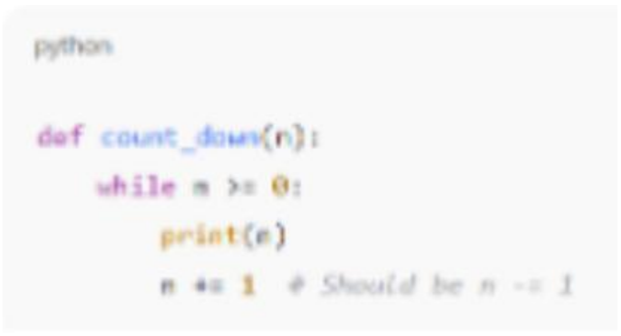
```python
#generate a simple python function
def add(a, b):
    return a + b
```

Justification:

In the above task using AI tool corrected the function with proper syntax and it solved successfully and accurately with the AI and AI also generated the explanation of how it is fixed the Error.

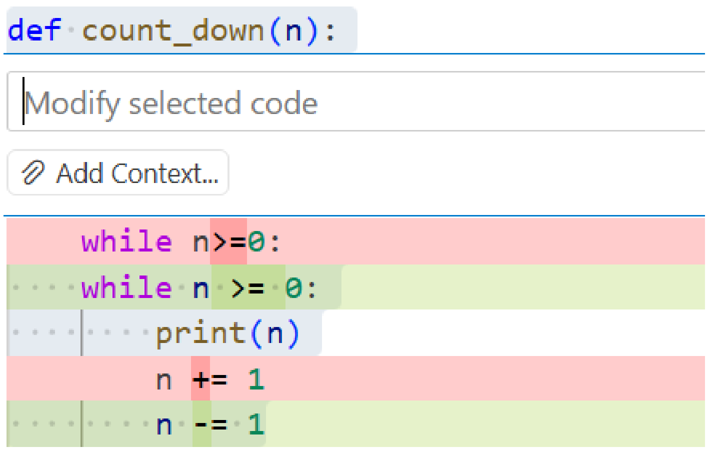
Task 2: Debugging Logic Errors in Loops

Scenario:



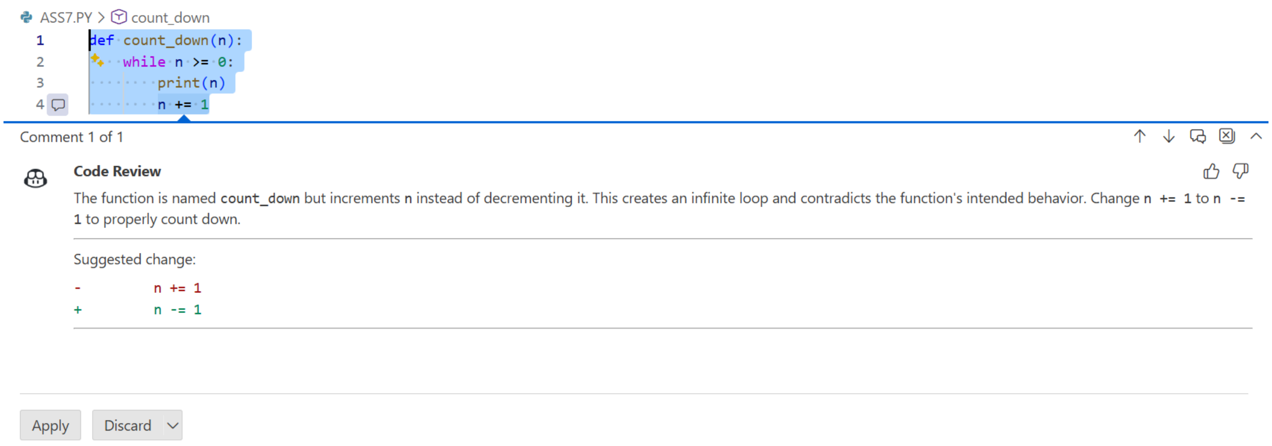
```
python

def count_down(n):
    while n >= 0:
        print(n)
        n += 1  # Should be n -= 1
```

Error Fix:



```python
def count_down(n):
    while n>=0:
    while n >= 0:
        print(n)
        n += 1
        n -= 1
```

Error Explanation:



ASS7.PY > count_down

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

Comment 1 of 1

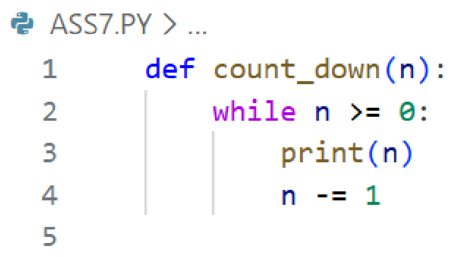
Code Review

The function is named `count_down` but increments `n` instead of decrementing it. This creates an infinite loop and contradicts the function's intended behavior. Change `n += 1` to `n -= 1` to properly count down.

Suggested change:

```
-        n += 1
+        n -= 1
```

AI Corrected code:



ASS7.PY > ...

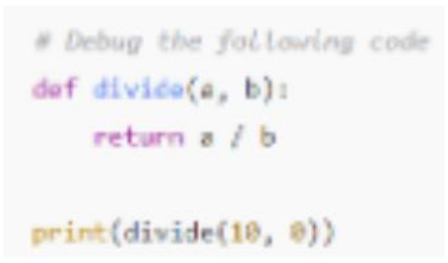
```
def count_down(n):
    while n >= 0:
        print(n)
        n -= 1

```

Justification:

In the above task using the AI tool we have identified the error and the AI have resolved the error which is infinite loop issue and corrected the iteration logic and explanation of the fixed error as well

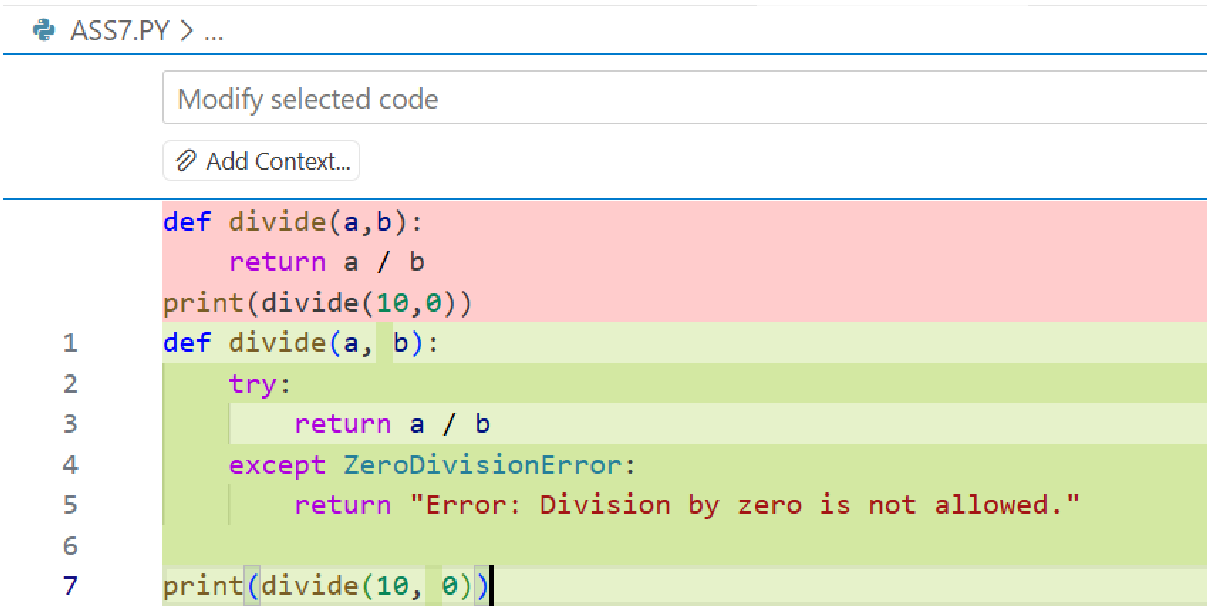
Task 3: Handling Runtime Errors (Division by Zero)

Scenario:



```
# Debug the following code
def divide(a, b):
    return a / b

print(divide(10, 0))
```

Error Fix:



ASS7.PY > ...

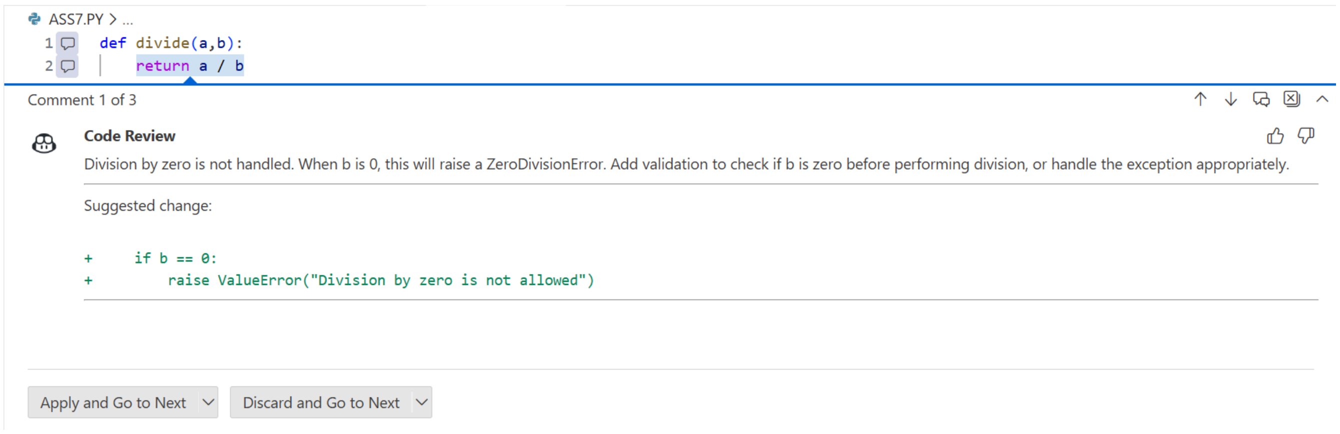
Modify selected code

Add Context...

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

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

print(divide(10, 0))
```

Error Explanation:



ASS7.PY > ...

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

Comment 1 of 3

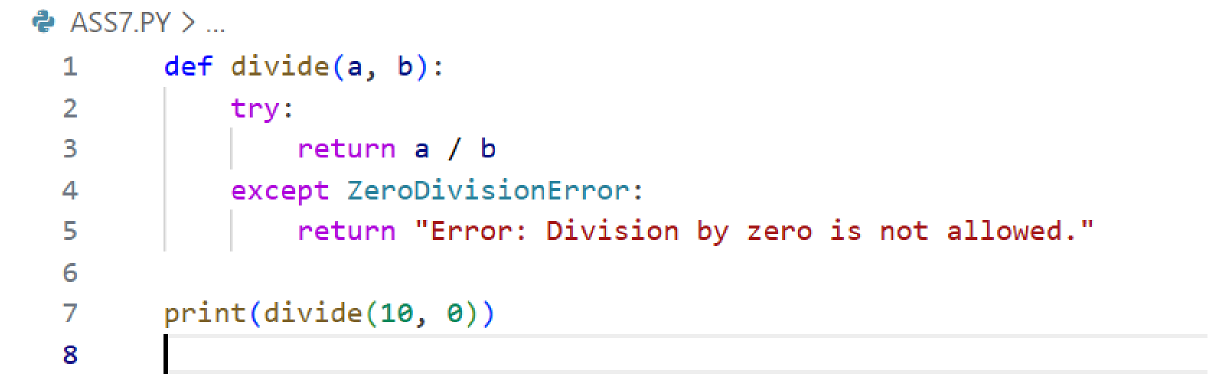
Code Review

Division by zero is not handled. When b is 0, this will raise a ZeroDivisionError. Add validation to check if b is zero before performing division, or handle the exception appropriately.

Suggested change:

```
+    if b == 0:
+        raise ValueError("Division by zero is not allowed")
```

AI Corrected code:



ASS7.PY > ...

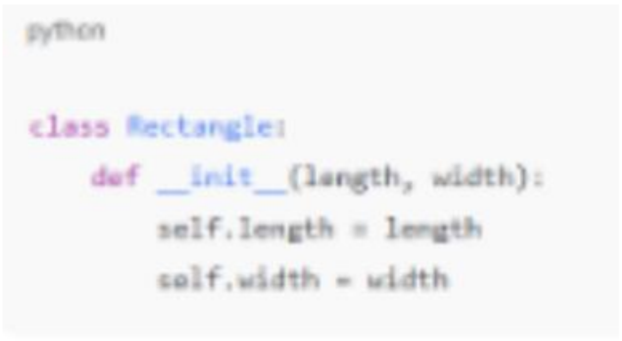
```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

print(divide(10, 0))

```

Justification:

In the above task the error identified by the AI and that is a runtime error and the AI fixed the error of the division by zero handled using the try-except and a the clear explanation of the resolved error using the exception handling.

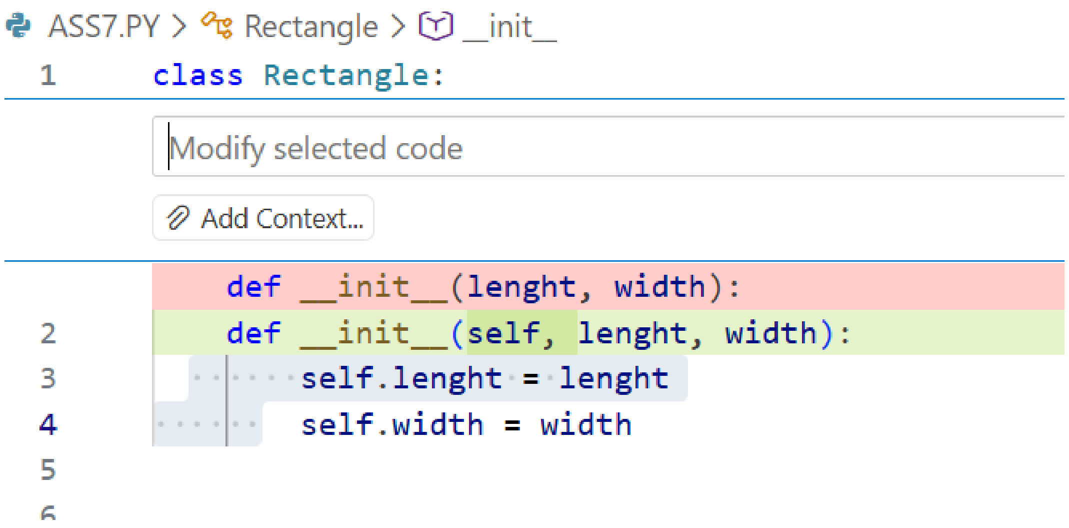
Task 4: Debugging Class Definition Errors

Scenario:



```python
class Rectangle:
    def __init__(length, width):
        self.length = length
        self.width = width
```

Error Fix:

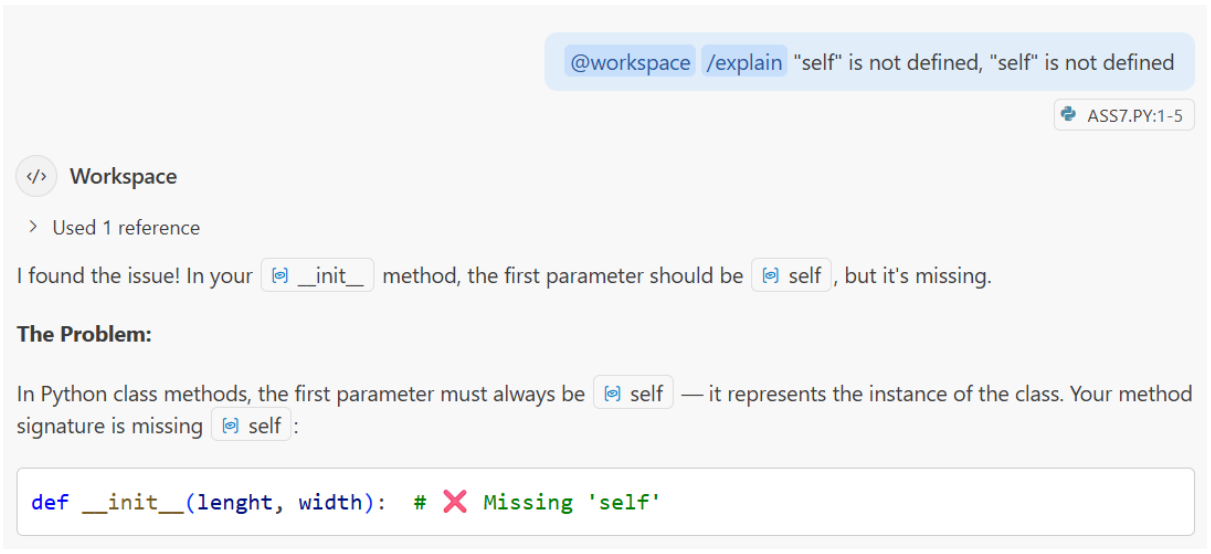


ASS7.PY > Rectangle > __init__

```python
class Rectangle:
```

```python
    def __init__(lenght, width):
    def __init__(self, lenght, width):
        self.lenght = lenght
        self.width = width
```

Error Explanation:


@workspace /explain "self" is not defined, "self" is not defined

ASS7.PY:1-5

Workspace

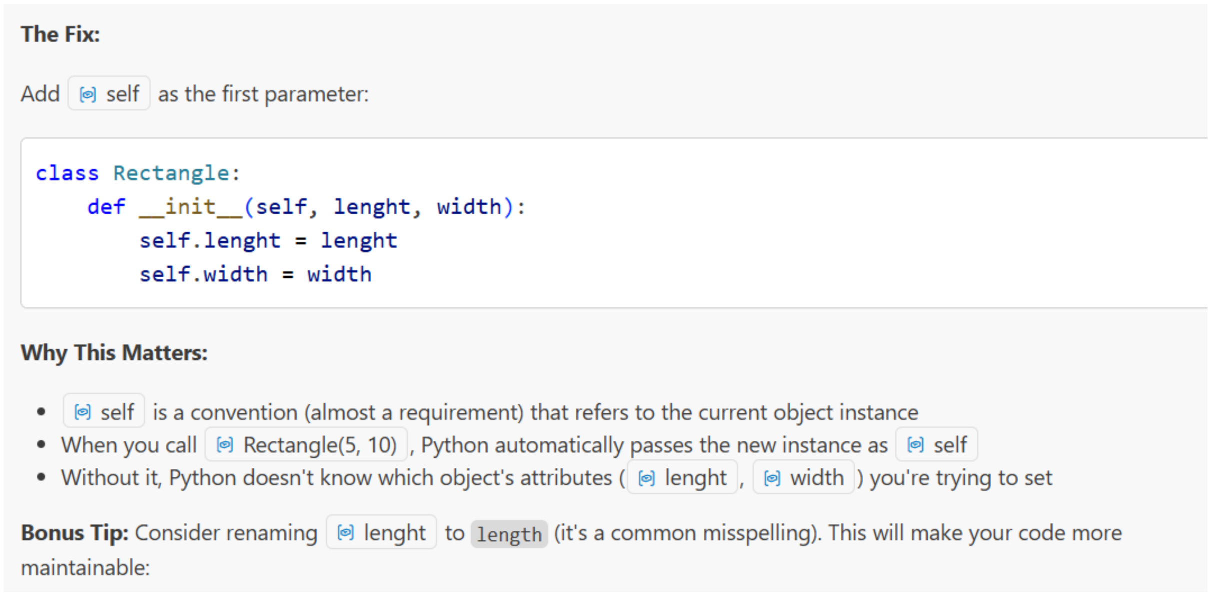
Used 1 reference

I found the issue! In your __init__ method, the first parameter should be self, but it's missing.

The Problem:

In Python class methods, the first parameter must always be self — it represents the instance of the class. Your method signature is missing self:

```
def __init__(lenght, width):  # ❌ Missing 'self'
```


The Fix:

Add self as the first parameter:

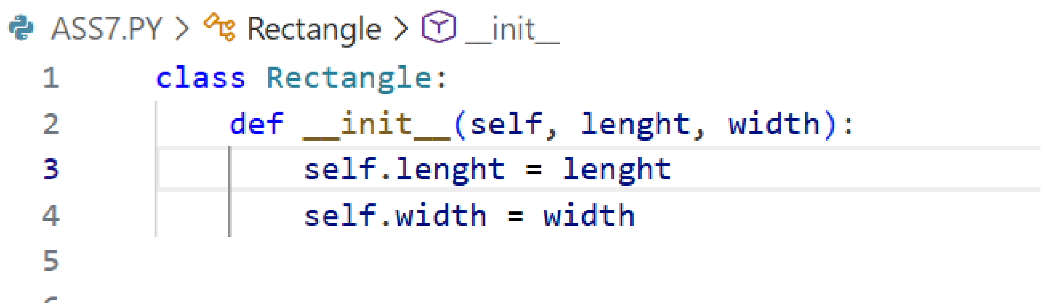
```
class Rectangle:
    def __init__(self, lenght, width):
        self.lenght = lenght
        self.width = width
```

Why This Matters:

- self is a convention (almost a requirement) that refers to the current object instance
- When you call Rectangle(5, 10), Python automatically passes the new instance as self
- Without it, Python doesn't know which object's attributes (lenght, width) you're trying to set

Bonus Tip: Consider renaming lenght to `length` (it's a common misspelling). This will make your code more maintainable:

AI Corrected code:


ASS7.PY > Rectangle > __init__

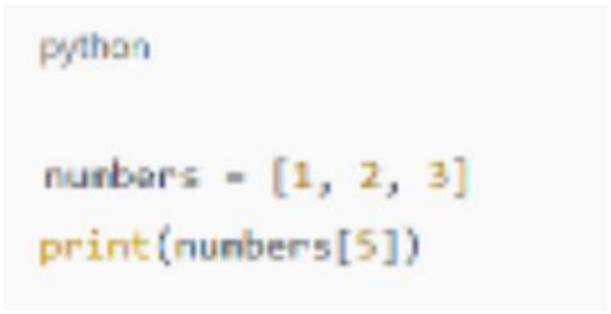
```
class Rectangle:
    def __init__(self, lenght, width):
        self.lenght = lenght
        self.width = width
```

Justification:

In the scenario, we can observe that there is a missing of the self parameter in the init method and the AI fixed the error and the generated the explanation of where the error is occurred and the how the error is impact the whole code and how to resolve it is explained by the AI.

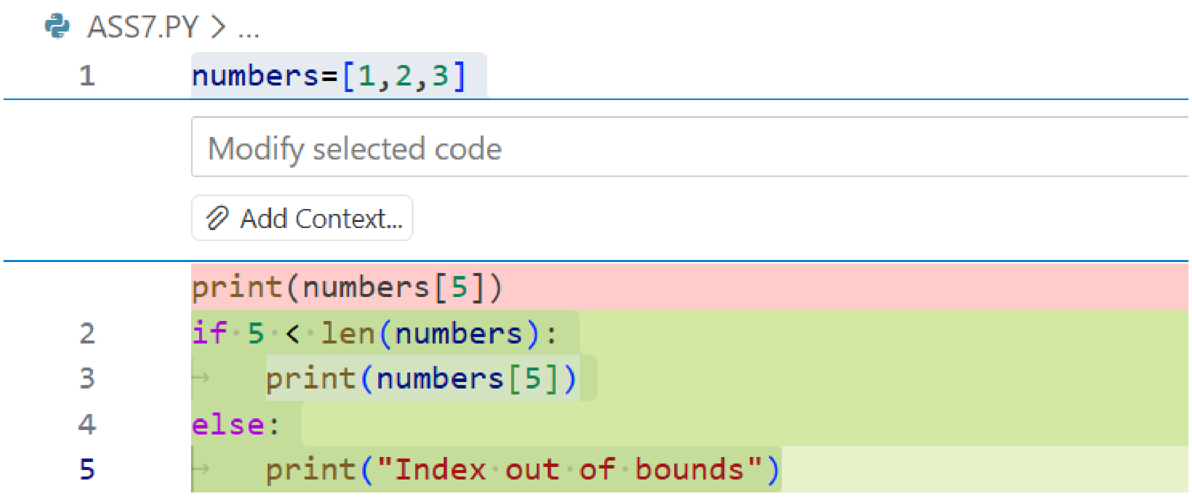
Task 5: Resolving Index Errors in Lists

Scenario:



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

Error Fix:



ASS7.PY > ...

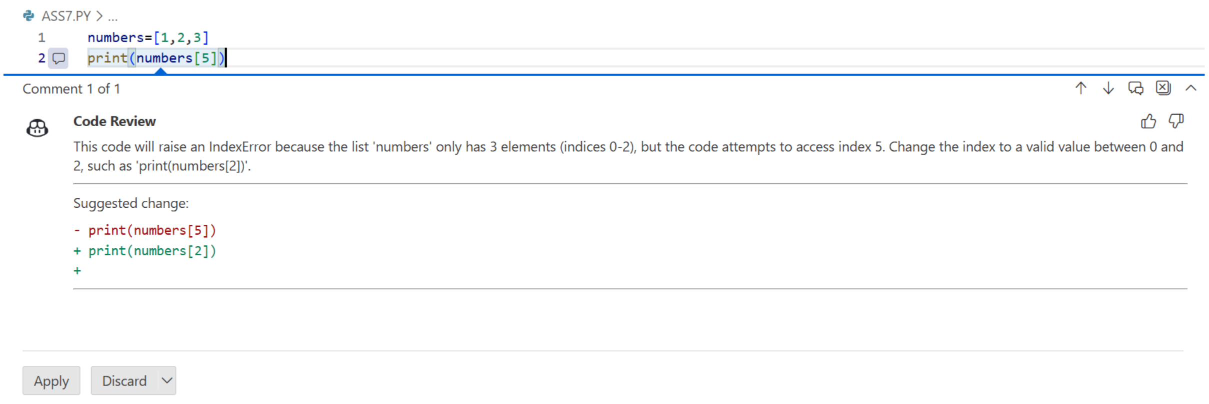
```python
numbers=[1,2,3]
```

Modify selected code

Add Context...

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

Error Explanation:



ASS7.PY > ...

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

Comment 1 of 1

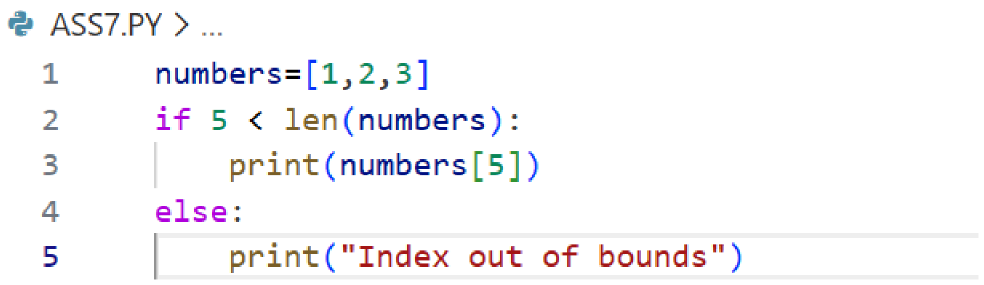
Code Review

This code will raise an IndexError because the list 'numbers' only has 3 elements (indices 0-2), but the code attempts to access index 5. Change the index to a valid value between 0 and 2, such as 'print(numbers[2])'.

Suggested change:

```
- print(numbers[5])
+ print(numbers[2])
+
```

AI Corrected code:



ASS7.PY > ...

```
numbers=[1,2,3]
if 5 < len(numbers):
    print(numbers[5])
else:
    print("Index out of bounds")
```

Justification:

In the scenario the AI identified the error that is the list index range out of bounds and the fixed the error by applying the index bounds checking and generated the error fixed code along with the explanation of the error.