Name:O.ISRAEL H.No:2303A51825 Batch:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**	
Program Name: B. Tech	**Assignment Type: Lab**		**Academic Year:**2025-2026
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju	
Course Code	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week4 – Wednesday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
AssignmentNumber:7.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	**Question**	***Expected Time to complete***
1	**Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs**	Week4 - Wednesday
	Lab Objectives • To identify and correct syntax, logic, and runtime errors in Python programs using AI tools	

Name:O.ISRAEL H.No:2303A51825 Batch:26

- To understand common programming bugs and AI-assisted debugging suggestions
- To evaluate how AI explains, detects, and fixes different types of coding errors
- To build confidence in using AI for structured debugging practices

Lab Outcomes (LOs)

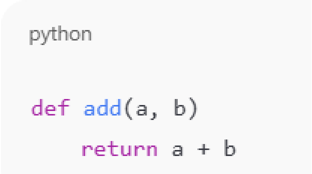
After completing this lab, students will be able to:

- Use AI tools to detect and correct syntax, logic, and runtime errors
- Interpret AI-suggested bug fixes and explanations
- Apply systematic debugging strategies using AI-generated insights
- Refactor buggy code using reliable programming patterns

Task 1: Fixing Syntax Errors

Scenario

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



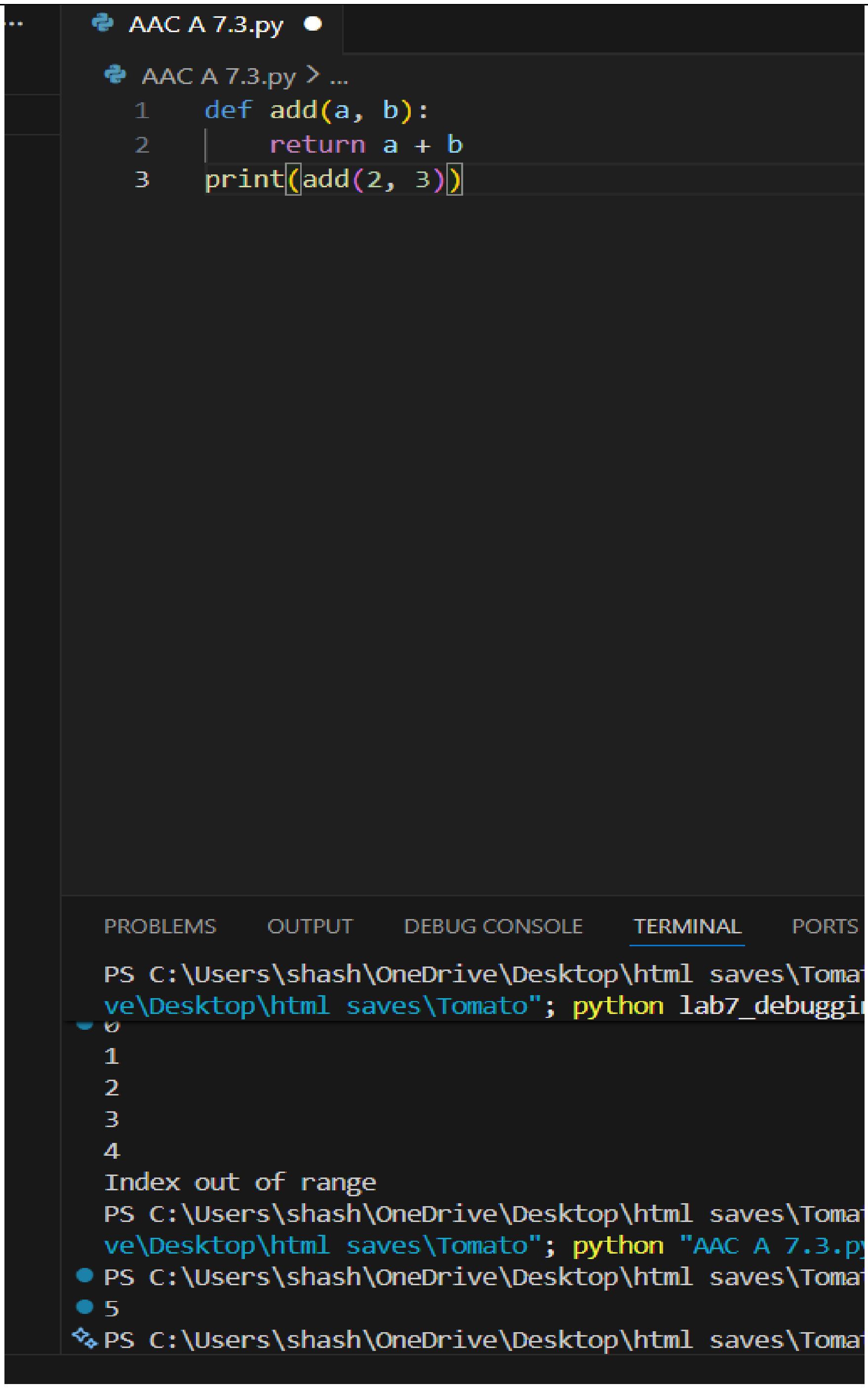
```python
def add(a, b)
    return a + b
```

Requirements

- Provide a Python function add(a, b) with a **missing colon**
- Use an AI tool to detect the syntax error
- Allow AI to correct the function definition
- Observe how AI explains the syntax issue

Expected Output

- Corrected function with proper syntax
- Syntax error resolved successfully
- AI-generated explanation of the fix



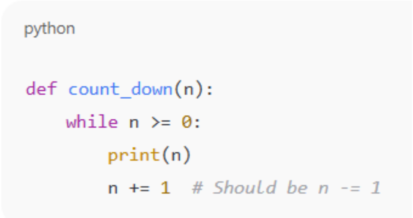
```python
def add(a, b):
    return a + b
print(add(2, 3))
```

```
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
ve\Desktop\html saves\Tomato"; python lab7_debuggi
0
1
2
3
4
Index out of range
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
ve\Desktop\html saves\Tomato"; python "AAC A 7.3.py
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
5
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
```

Task 2: Debugging Logic Errors in Loops

Scenario

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



python

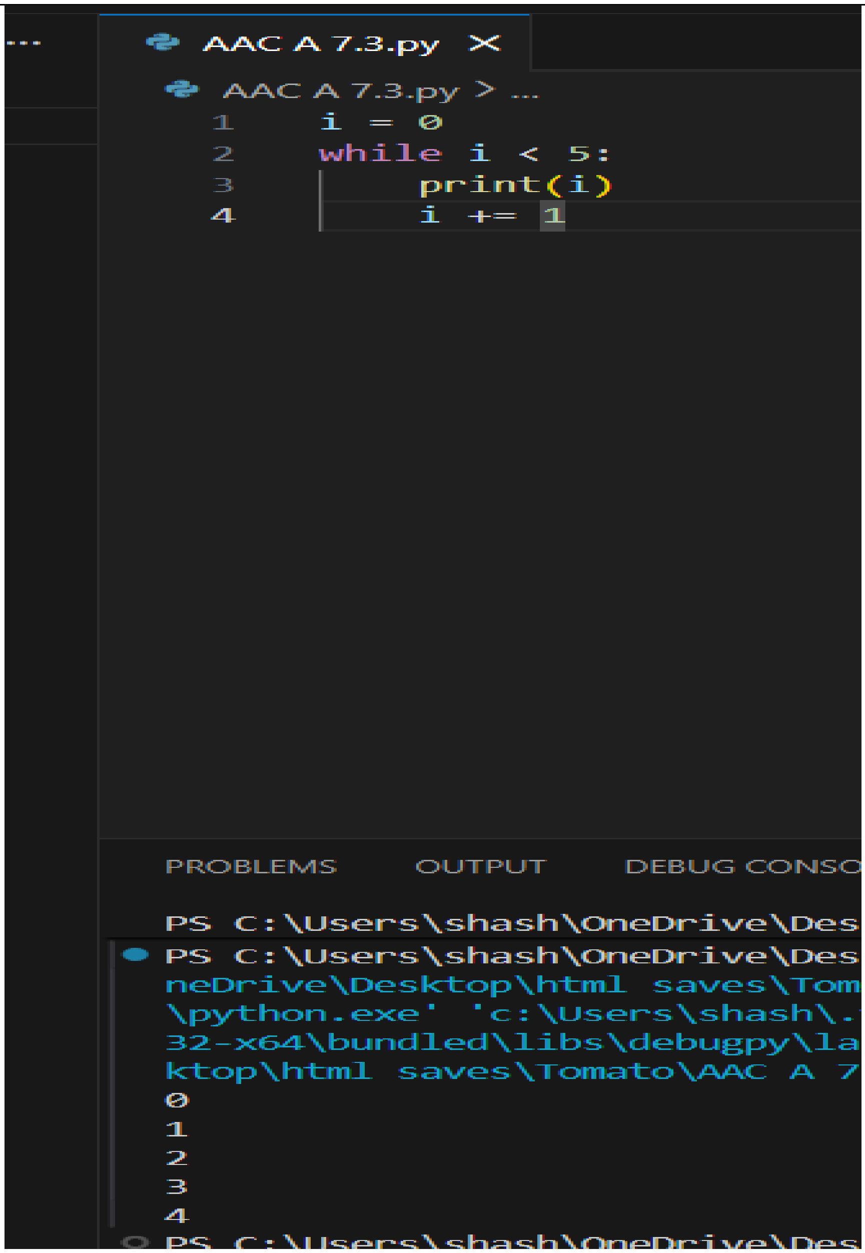
```python
def count_down(n):
    while n >= 0:
        print(n)
        n += 1  # Should be n -= 1
```

Requirements

- Provide a loop with an **increment or decrement error**
- Use AI to identify the cause of infinite iteration
- Let AI fix the loop logic
- Analyze the corrected loop behavior

Expected Output

- Infinite loop issue resolved
- Correct increment/decrement logic applied
- AI explanation of the logic error



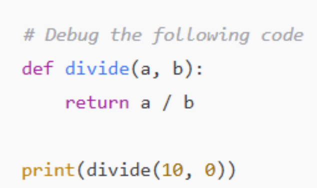
```python
i = 0
while i < 5:
    print(i)
    i += 1
```

```
PS C:\Users\shash\OneDrive\Des
PS C:\Users\shash\OneDrive\Des
neDrive\Desktop\html saves\Tom
\python.exe' 'c:\Users\shash\.
32-x64\bundled\libs\debugpy\la
ktop\html saves\Tomato\AAC A 7
0
1
2
3
4
PS C:\Users\shash\OneDrive\Des
```

Task 3: Handling Runtime Errors (Division by Zero)

Scenario

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



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

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

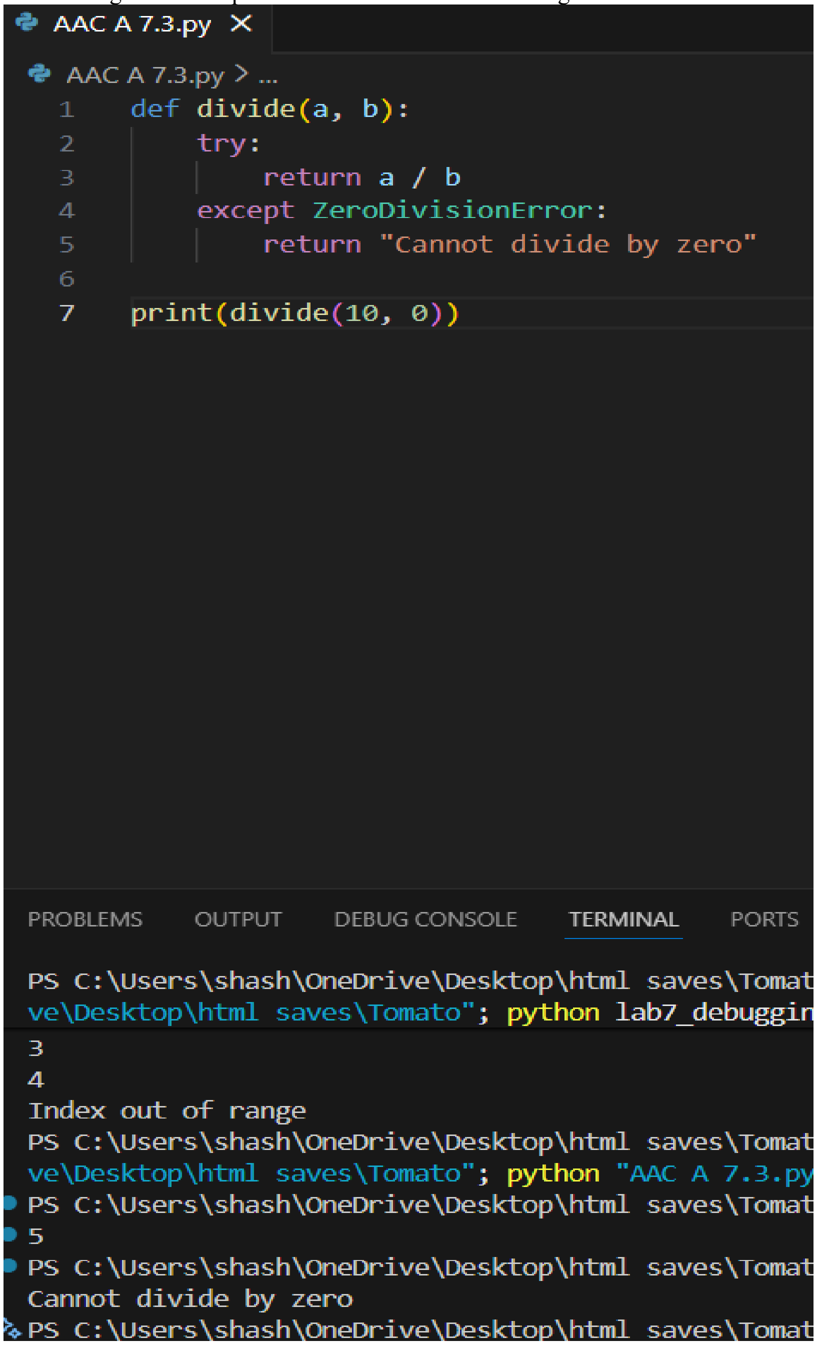
Requirements

- Provide a function that performs division without validation
- Use AI to identify the runtime error
- Let AI add try-except blocks for safe execution

- Review AI’s error-handling approach

Expected Output

- Function executes safely without crashing
- Division by zero handled using try-except
- Clear AI-generated explanation of runtime error handling



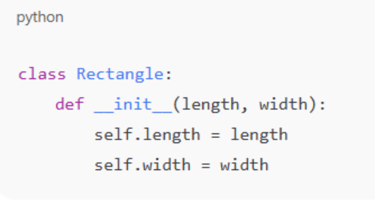
```python
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Cannot divide by zero"

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

```
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
ve\Desktop\html saves\Tomato"; python lab7_debuggin
3
4
Index out of range
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
ve\Desktop\html saves\Tomato"; python "AAC A 7.3.py
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
5
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
Cannot divide by zero
PS C:\Users\shash\OneDrive\Desktop\html saves\Tomat
```

Task 4: Debugging Class Definition Errors

Scenario

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



python

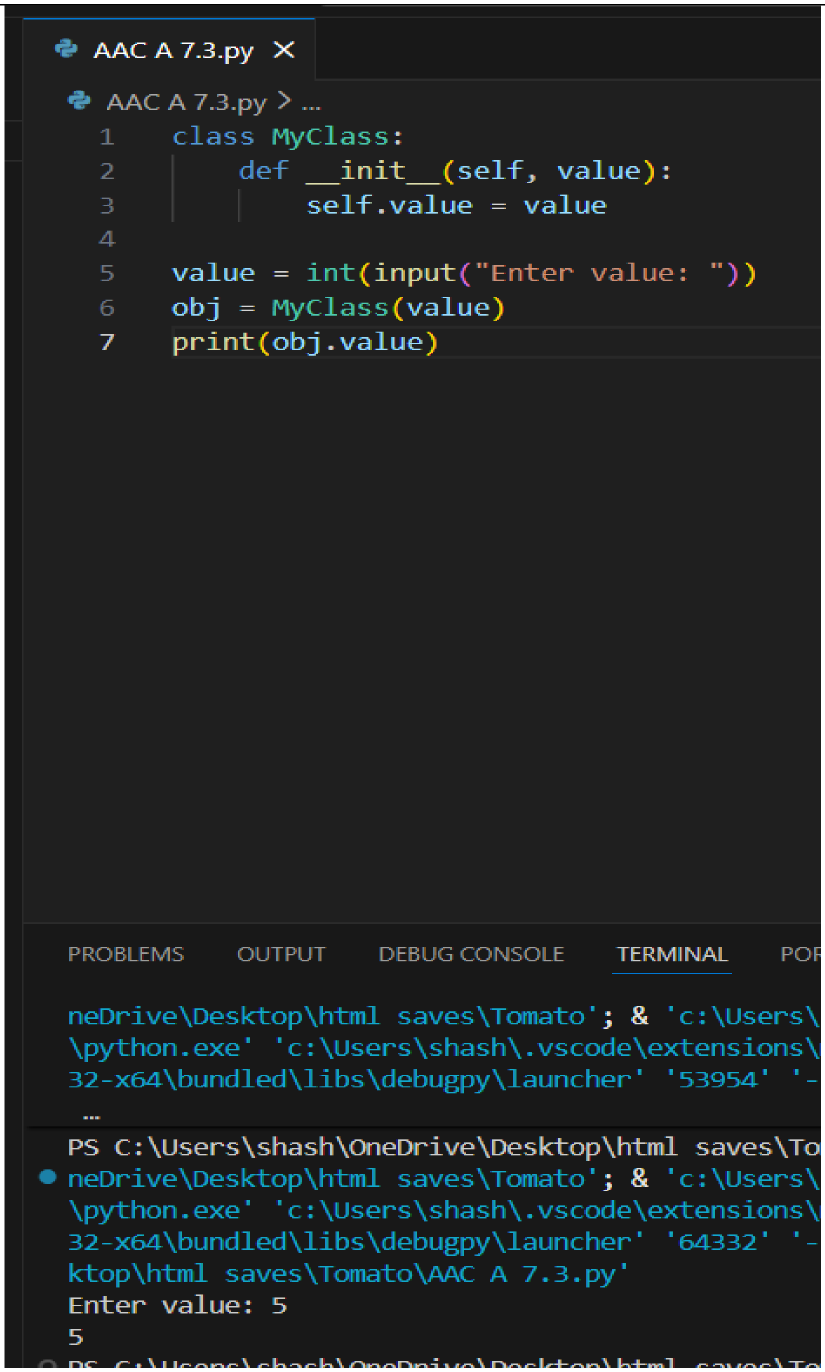
```python
class Rectangle:
    def __init__(length, width):
        self.length = length
        self.width = width
```

Requirements

- Provide a class definition with **missing self-parameter**
- Use AI to identify the issue in the __init__() method
- Allow AI to correct the class definition
- Understand why self is required

Expected Output

- Corrected __init__() method
- Proper use of self in class definition
- AI explanation of object-oriented error



```python
class MyClass:
    def __init__(self, value):
        self.value = value

value = int(input("Enter value: "))
obj = MyClass(value)
print(obj.value)
```

```
neDrive\Desktop\html saves\Tomato'; & 'c:\Users\
\python.exe' 'c:\Users\shash\.vscode\extensions\
32-x64\bundled\libs\debugpy\launcher' '53954' '-
 ...
PS C:\Users\shash\OneDrive\Desktop\html saves\To
neDrive\Desktop\html saves\Tomato'; & 'c:\Users\
\python.exe' 'c:\Users\shash\.vscode\extensions\
32-x64\bundled\libs\debugpy\launcher' '64332' '-
ktop\html saves\Tomato\AAC A 7.3.py'
Enter value: 5
5
PS C:\Users\shash\OneDrive\Desktop\html saves\To
```

Task 5: Resolving Index Errors in Lists

Scenario

A program crashes when accessing an invalid index in a list.

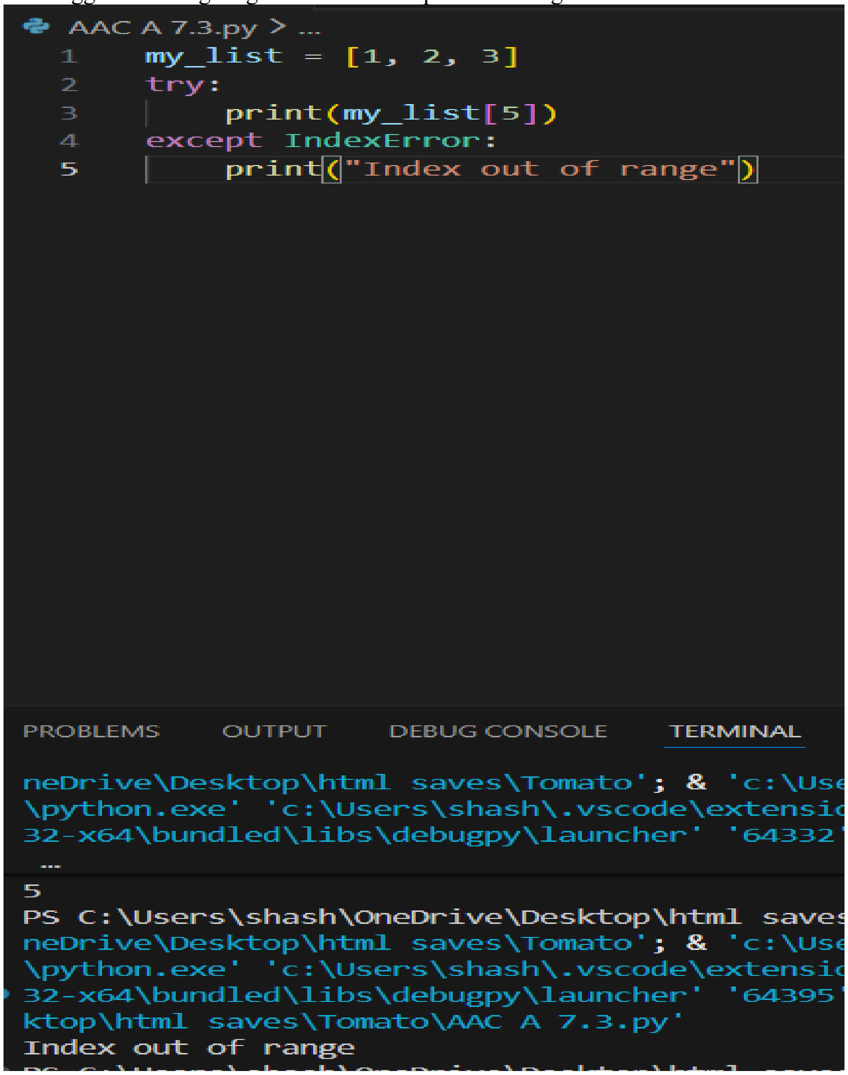
```python
numbers = [1, 2, 3]
print(numbers[5])
```

Requirements

- Provide code that accesses an **out-of-range list index**
- Use AI to identify the Index Error
- Let AI suggest safe access methods
- Apply bounds checking or exception handling

Expected Output

- Index error resolved
- Safe list access logic implemented
- AI suggestion using length checks or exception handling



```python
my_list = [1, 2, 3]
try:
    print(my_list[5])
except IndexError:
    print("Index out of range")
```

```
neDrive\Desktop\html saves\Tomato'; & 'c:\Use
\python.exe' 'c:\Users\shash\.vscode\extensio
32-x64\bundled\libs\debugpy\launcher' '64332'
 ...
5
PS C:\Users\shash\OneDrive\Desktop\html saves
neDrive\Desktop\html saves\Tomato'; & 'c:\Use
\python.exe' 'c:\Users\shash\.vscode\extensio
32-x64\bundled\libs\debugpy\launcher' '64395'
ktop\html saves\Tomato\AAC A 7.3.py'
Index out of range
PS C:\Users\shash\OneDrive\Desktop\html saves
```

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots