LAB ASSIGNMENT_7.5

NAME: ch.vishnu vardhan

Hall Ticket No: 2303A51718

BATCH-11

Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs

Lab Objectives:

• To identify and correct syntax, logic, and runtime errors in

Week4 -Monday:

Python programs using AI tools.

• 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 to perform 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 supported by AI-generated insights.

Refactor buggy code using responsible and reliable programming patterns.

Task 1 (Mutable Default Argument – Function Bug)

Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it. # Bug: Mutable default argument def add_item(item, items=[]):

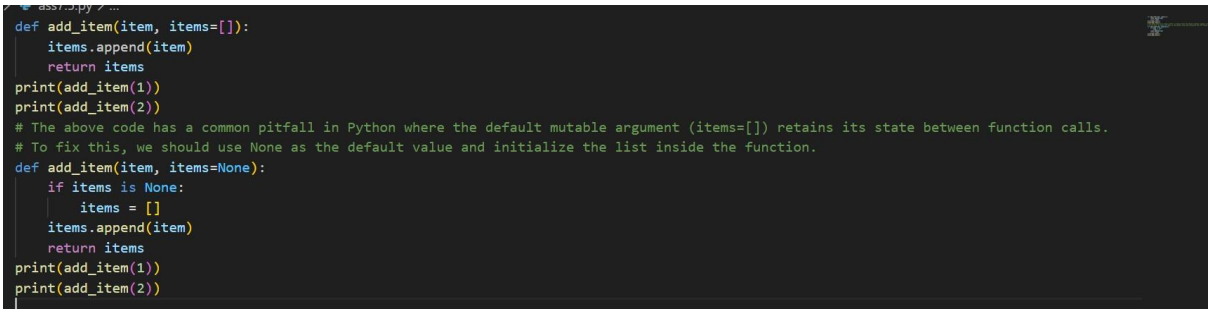
items.append(item) return

items print(add_item(1))

print(add_item(2))

Prompt: Corrected function avoids shared list bug.

Expected Output:



Output:

Task 2 (Floating-Point Precision Error)

Task: Analyze given code where floating-point comparison fails.

Use AI to correct with tolerance. #

Bug: Floating point precision issue

def check_sum(): return (0.1 + 0.2)

== 0.3 print(check_sum())

Prompt : Corrected function

Expected Output:

```
ts > ass7.5.py > ...
def check_sum():
    return (0.1 + 0.2) == 0.3
print(check_sum())
# This will print False due to floating-point precision issues in Python.
# To fix this, we can use the round function to compare the values with a certain precision.
def check_sum_fixed():
    return round(0.1 + 0.2, 10) == round(0.3, 10)
print(check_sum_fixed())
# This will print True as the rounding resolves the precision issue.
```

Output:

```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Asse
False
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Asse
False
True
PS C:\Users\harik\My project>
```

Task 3 (Recursion Error – Missing Base Case)

Task: Analyze given code where recursion runs infinitely due to

missing base case. Use AI to fix.

Bug: No base case def

countdown(n):

print(n) return

countdown(n-1)

countdown(5)

Prompt : Correct recursion with stopping condition **Expected**

Output :

```
#Fixed Code:
def countdown(n):
    if n <= 0:
        print("Countdown finished!")
        return
    print(n)
    return countdown(n-1)
```

Output:

```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Pro
5
4
3
2
1
Countdown finished!
```

Task 4 (Dictionary Key Error)

Task: Analyze given code where a missing dictionary key causes

error. Use AI to fix it.

Bug: Accessing non-existing key

def get_value(): data = {"a": 1, "b":

2} return data["c"]

print(get_value())

Prompt: Corrected with .get() or error handling.

Expected Output:

```
# correct the above code with .get() or error handling.
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", 0)
print(get_value())
```

Output:

```
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
0
0
PS C:\Users\harik\My project> & C:/Users/harik/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/harik/My project/Assets/ass7.5.py"
0
0
0
PS C:\Users\harik\My project>
```

Do you want to install the recommended 'Unity' extension from Microsoft for this repository?

Task 5 (Infinite Loop – Wrong Condition)

Task: Analyze given code where loop never ends. Use AI to detect

and fix it.

Bug: Infinite loop

def loop_example(): i

= 0 while i < 5:

print(i)

Prompt : Corrected loop increments i.

Expected Output:

```
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
#correct the above code and handle the errors
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
loop_example()
#Corrected loop increments i.
def loop_example():
    i = 0
```

```
#Corrected loop increments i.
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
loop_example()
loop_example()
```

Output:

Task 6 (Unpacking Error – Wrong Variables)

Task: Analyze given code where tuple unpacking fails. Use AI to fix

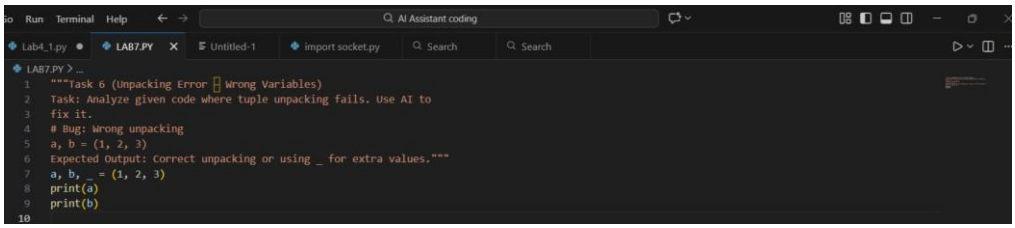
it.

Bug: Wrong unpacking

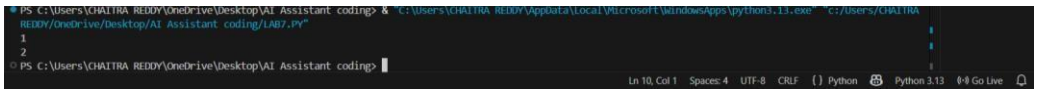
a, b = (1, 2, 3)

Prompt: Correct unpacking or using _ for extra values.

Expected Output:



Output:



Task: Analyze given code where mixed indentation breaks execution. Use AI to fix it.

Bug: Mixed indentation def

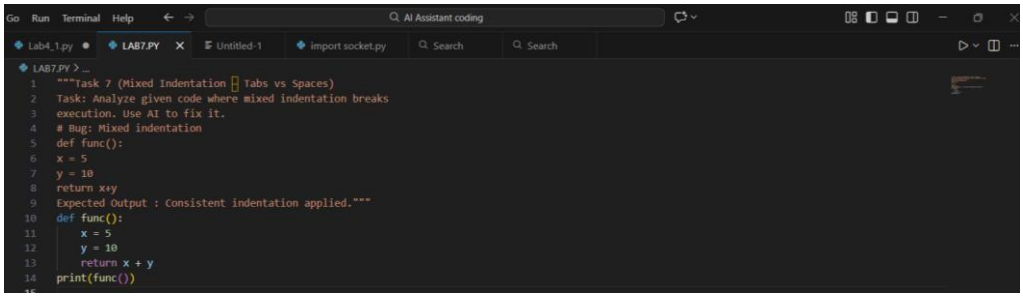
func():

x = 5 y =

10 return

x+y

Expected Output : Consistent indentation applied.



Output:

Task 8 (Import Error – Wrong Module Usage)

Task: Analyze given code with incorrect import. Use AI to fix.

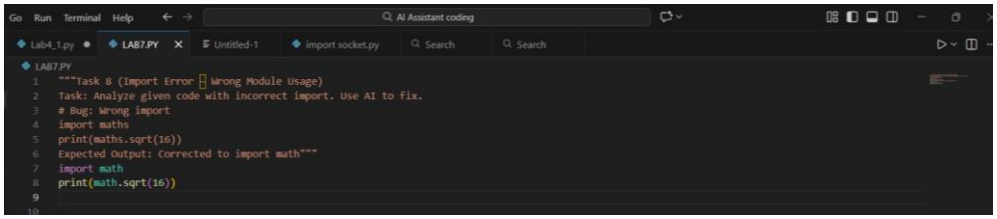
Bug: Wrong import

import maths

print(maths.sqrt(16))

Prompt : Corrected to import math **Expected**

Output:



Output: