

# Ai Assisted Coding

## Assignment-7.5

Name: C. Vaishnav Reddy

HT No:2303A2203

Batch:45

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=[]):
```

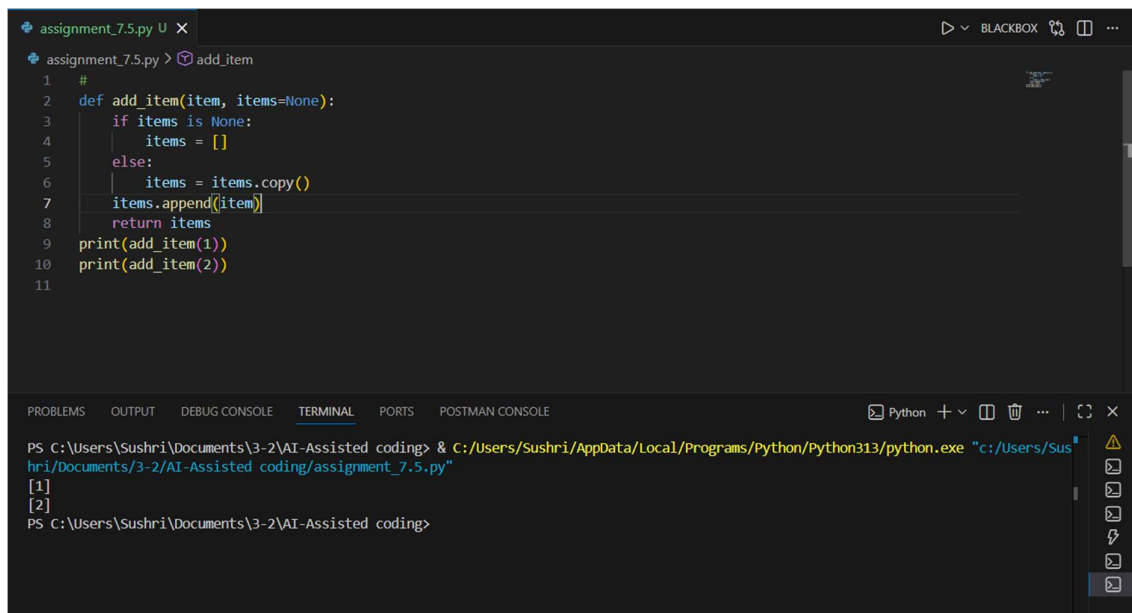
```
    items.append(item)
```

```
    return items
```

```
print(add_item(1))
```

```
print(add_item(2))
```

Expected Output: Corrected function avoids shared list bug.



The screenshot shows a code editor with a file named 'assignment\_7.5.py'. The code defines a function 'add\_item' that takes an 'item' and a default 'items' list. The function checks if 'items' is None and creates a new list, or else it copies the existing list and appends the new item. This prevents the shared list bug. The code then prints the results of calling 'add\_item(1)' and 'add\_item(2)'. The terminal output shows the execution of the script, resulting in the printed lists [1] and [2].

```
1 #
2 def add_item(item, items=None):
3     if items is None:
4         items = []
5     else:
6         items = items.copy()
7     items.append(item)
8     return items
9 print(add_item(1))
10 print(add_item(2))
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

PS C:\Users\Sushri\Documents\3-2\AI-Assisted coding> & C:/Users/Sushri/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Sushri/Documents/3-2/AI-Assisted coding/assignment\_7.5.py"

[1]  
[2]  
PS C:\Users\Sushri\Documents\3-2\AI-Assisted coding>

## Task 2 (Floating-Point Precision Error)

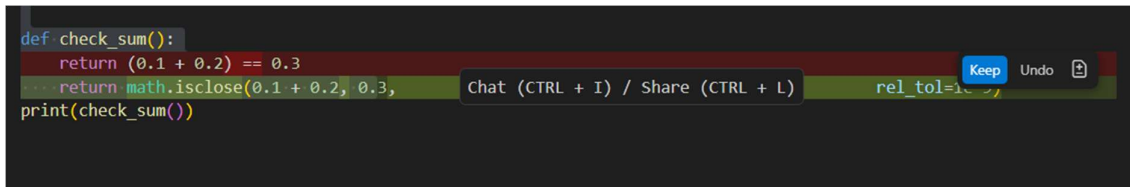
Task: Analyze given code where floating-point comparison fails.

Use AI to correct with tolerance.

# Bug: Floating point precision issue

```
def check_sum():  
    return (0.1 + 0.2) == 0.3  
  
print(check_sum())
```

Expected Output: Corrected function

A screenshot of a code editor with a dark background. The code is as follows:

```
def check_sum():  
    return (0.1 + 0.2) == 0.3  
... return math.isclose(0.1 + 0.2, 0.3,  
print(check_sum())
```

On the right side of the editor, there are buttons for 'Keep', 'Undo', and a share icon. Below the editor, there is a tooltip that says 'Chat (CTRL + I) / Share (CTRL + L)' and a status bar on the far right that says 'rel\_tol=1e-09'.

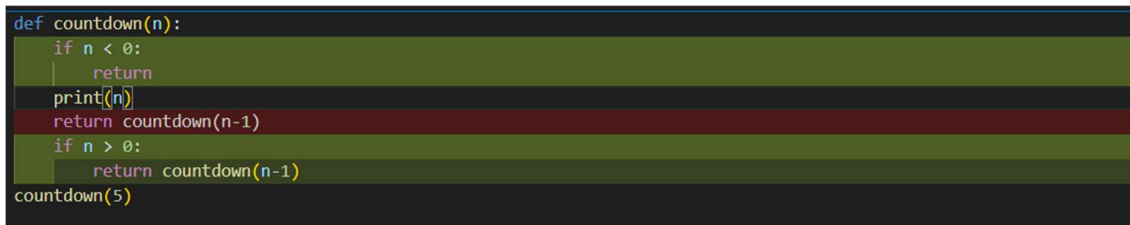
## Task 3 (Recursion Error – Missing Base Case)

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)
```

Expected Output : Correct recursion with stopping condition.

A screenshot of a code editor with a dark background. The code is as follows:

```
def countdown(n):  
    if n < 0:  
        return  
    print(n)  
    return countdown(n-1)  
    if n > 0:  
        return countdown(n-1)  
countdown(5)
```


## 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())
```

Expected Output: Corrected with .get() or error handling.



The screenshot shows a code editor with a dark theme. At the top, there is a title bar that says "fix missing key bug in get\_value function using .get function". Below the title bar, there is a search bar with the text "Add Context...". The main code area contains the following Python code:

```
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", "Key not found")
print(get_value())
```

At the bottom of the code area, there is a button that says "Chat (CTRL + I) / Share (CTRL + L)".

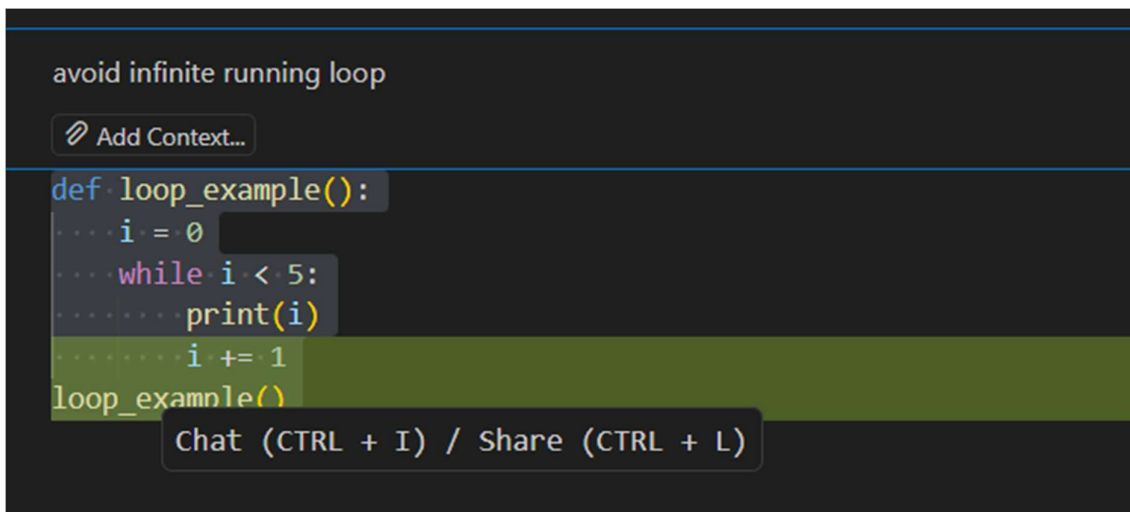
### Task 5 (Infinite Loop – Wrong Condition)

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)
```

Expected Output: Corrected loop increments i.



The screenshot shows a code editor with a dark theme. At the top, there is a title bar that says "avoid infinite running loop". Below the title bar, there is a search bar with the text "Add Context...". The main code area contains the following Python code:

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

At the bottom of the code area, there is a button that says "Chat (CTRL + I) / Share (CTRL + L)".

### 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)
```

Expected Output: Correct unpacking or using \_ for extra values`

```
a, b, c = 1, 2, 3
```

Task 7 (Mixed Indentation – Tabs vs Spaces)

Task: Analyze given code where mixed indentation breaks

execution. Use AI to fix it.

# Bug: Mixed indentation

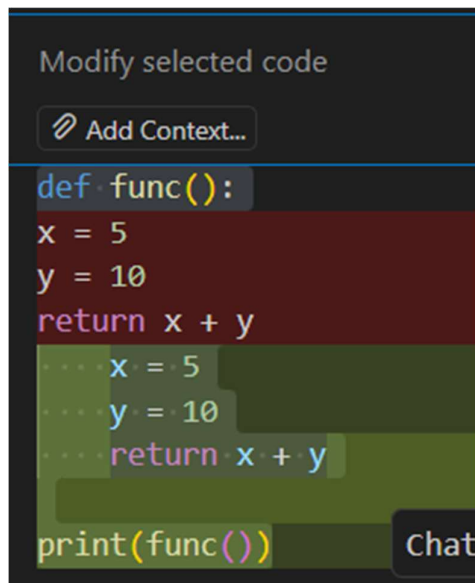
```
def func():
```

```
    x = 5
```

```
    y = 10
```

```
    return x+y
```

Expected Output : Consistent indentation applied.



```
def func():  
    x = 5  
    y = 10  
    return x + y  
  
print(func())
```

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

# Bug: Wrong import

```
import maths
```

```
print(maths.sqrt(16))
```

Expected Output: Corrected to import math

```
import maths
print(maths.sqrt(16))
print([math.sqrt(16)])
```