

Assignment-7.5

The image shows a Visual Studio Code editor window with a Python file named `assignment_7.5.py`. The script defines a function `add_item` that appends an item to a list. The terminal shows the command to run the script, which outputs the list `[1, 2]`.

```
assignment_7.5.py X
assignment_7.5.py > add_item
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
```

TERMINAL

```
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.

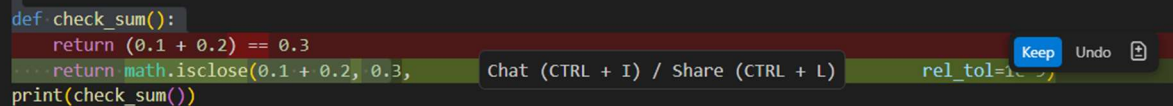
Bug: Floating point precision issue

```
def check_sum():
```

```
    return (0.1 + 0.2) == 0.3
```

```
print(check_sum())
```

Expected Output: Corrected function



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

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.



```
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, a status bar indicates 'fix missing key bug in get_value function using .get function'. Below this, there is a search bar with 'Add Context...' and an 'Auto' dropdown. The main code area shows 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 prompt: '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, a status bar indicates 'avoid infinite running loop'. Below this, there is a search bar with 'Add Context...'. The main code area shows 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 prompt: 'Chat (CTRL + I) / Share (CTRL + L)'.

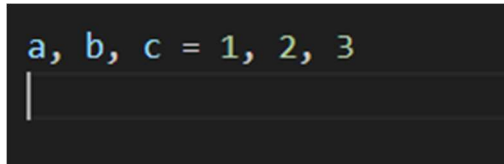
Task 6 (Unpacking Error – Wrong Variables)

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

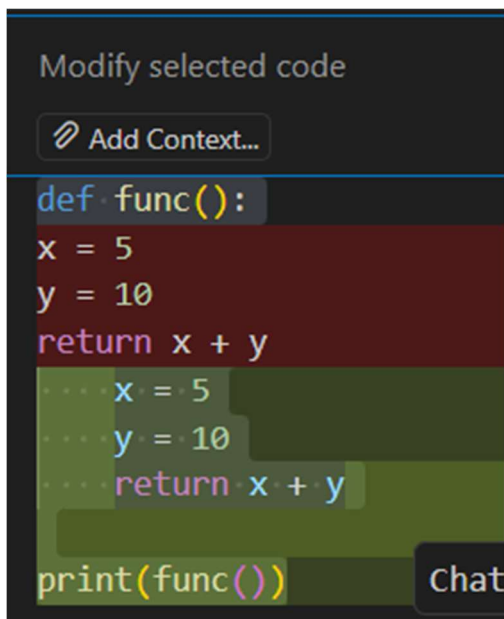
```
def func():
```

```
    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))
```