

ASSIGNMENT 7.5

Name: A.Sai Varshith

Roll Number : 2303A51831

Batch – 04

Task 1 (Mutable Default Argument – Function Bug)

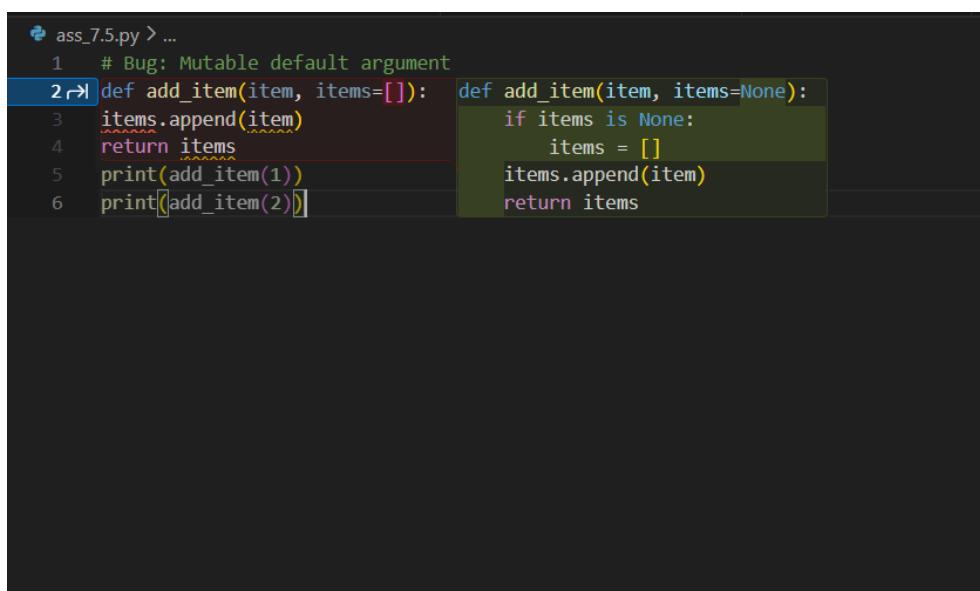
Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it.

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.

INPUT:



```
ass_7.5.py > ...
1  # Bug: Mutable default argument
2  def add_item(item, items=[]):  def add_item(item, items=None):
3      items.append(item)          if items is None:
4      return items                items = []
5      print(add_item(1))        items.append(item)
6      print(add_item(2))        return items
```

```
ass_7.5.py > add_item
1 # Bug: Mutable default argument
2 def add_item(item, items=None):
3     if items is None:
4         items = []
5     items.append(item)
6     return items
7 print(add_item(1))
8 print(add_item(2))
```

OUTPUT:

```
ai_coding/ass_7.5.py
[1]
[2]
```

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

INPUT:

```
1 # Bug: Floating point precision issue
2 def check_sum():
3     return (0.1 + 0.2) == 0.3
4     return abs((0.1 + 0.2) - 0.3) < 1e-10 # Use a small tolerance for floating point comparison
5
6
```

```
# Bug: Floating point precision issue
def check_sum():
    return abs(0.1 + 0.2) - 0.3 < 1e-10 # Use a small tolerance for floating point comparison
print(check_sum())
```

OUTPUT:

```
True
```

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

INPUT:

```
ass_7.5.py > ...
3     print(n)
r→ 4     if n == 0:
            return
5     return countdown(n-1)
5     countdown(5)
```

```
# Bug: No base case
def countdown(n):
    if n == 0:
        return
    print(n)
    countdown(n-1)
countdown(5)
```

OUTPUT:

```
5
4
3
2
1
```

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.

INPUT:

```
⚡ ass_7.5.py > ...
1 def get_value():
2     data = {"a": 1, "b": 2}
3
4 →| return data["c"]
5     return data.get("c", "Key not found")
6
7 print(get_value())
```

```
⚡ ass_7.5.py > ⌂ get_value
1 # Bug: Accessing non-existing key
2 def get_value():
3     data = {"a": 1, "b": 2}
4     return data.get("c", "Key not found")
5
6 print(get_value())
```

OUTPUT:

```
| Key not found
```

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.

INPUT:

```
ass_7.5.py > [i]
1 # Bug: Infinite loop
2 def loop_example():
3     i = 0
4     while i < 5:
5         print(i)
→       i += 1
```

```
Go Run Terminal Help ← → Q Ai codin
Welcome # AI-Generated Logic Without Modularizat.py ass_3.2.py
ass-7.5.py > ...
1 def loop_example():
2     i = 0
3     while i < 5:
4         print(i)
5         i += 1    # Increment added
6
7 loop_example()
8 |
```

OUTPUT:

```
0
1
2
3
4
```

Task 6 (Unpacking Error – Wrong Variables)

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

0# Bug: Wrong unpacking

a, b = (1, 2, 3)

Expected Output: Correct unpacking or using _ for extra values.

INPUT:

```
# Bug: Wrong unpacking
```

```
a, b = (1, 2, 3)
```

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

```
1 def func():
2     x = 5
3     y = 10
4     print(x + y)
```

OUTPUT:

```
1 2
```

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
```

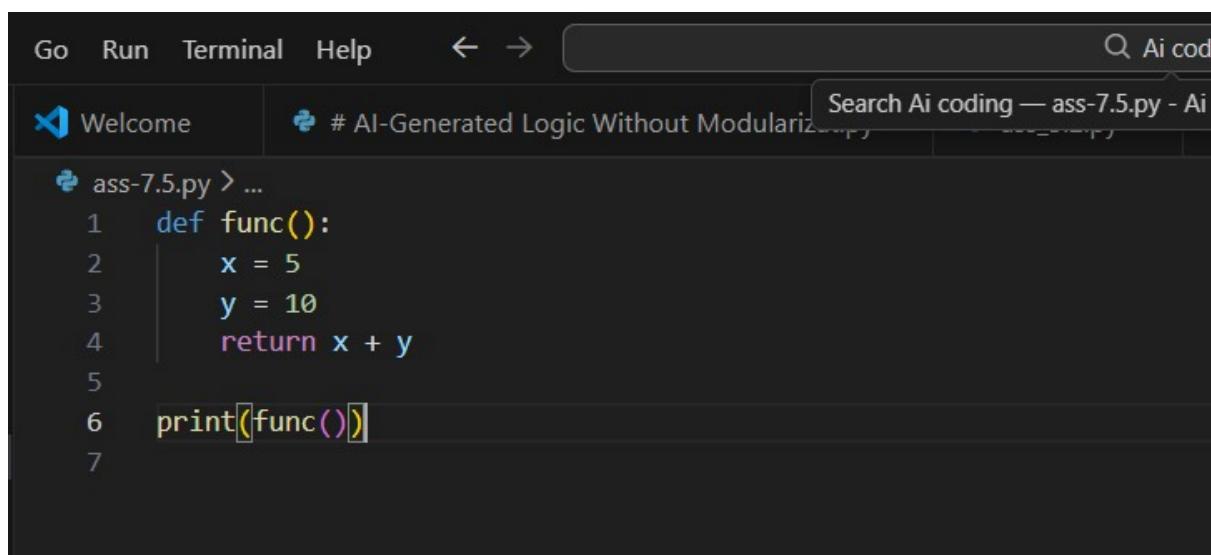
Expected Output : Consistent indentation applied.

INPUT:

```
# Bug: Mixed indentation

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

Expected Output : Consistent indentation applied.



The screenshot shows a code editor interface with a dark theme. The top menu bar includes 'Go', 'Run', 'Terminal', and 'Help'. Below the menu is a toolbar with navigation icons. A search bar on the right contains the text 'Search Ai coding — ass-7.5.py - Ai'. The main workspace displays the following Python code:

```
ass-7.5.py > ...
1 def func():
2     x = 5
3     y = 10
4     return x + y
5
6 print(func())
7
```

OUTPUT:



The screenshot shows a terminal window with a dark theme. The command prompt shows the path 'hyd-Moto/OneDrive/DESKTOP/AI Coding' followed by a blue circular progress indicator and the number '15'.

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))
```

Expected Output: Corrected to import math

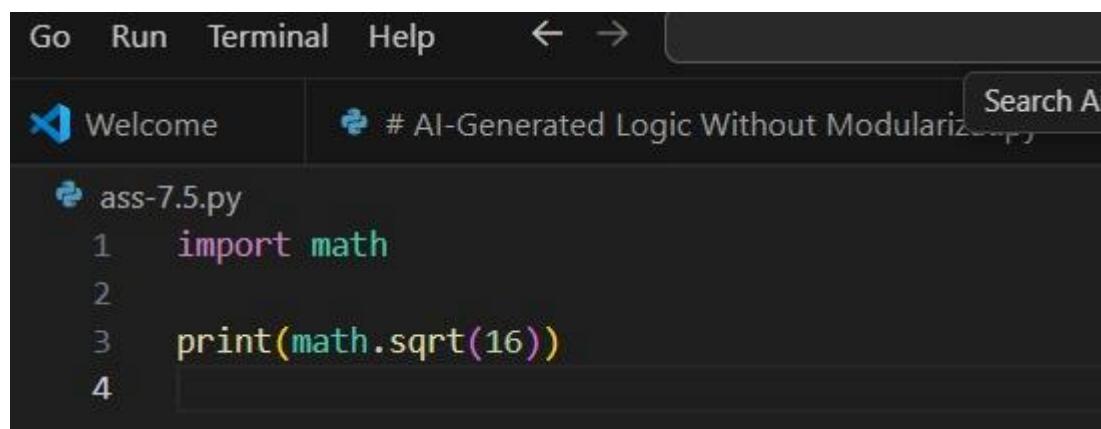
INPUT:

```
# Bug: Wrong import
```

```
import maths
```

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

Expected Output: Corrected to import math



The screenshot shows a code editor interface with a dark theme. At the top, there is a menu bar with 'Go', 'Run', 'Terminal', and 'Help'. Below the menu is a toolbar with left and right arrow buttons. On the left side, there is a sidebar with a 'Welcome' icon and a file list containing 'ass-7.5.py'. The main workspace shows the following Python code:

```
1 import math
2
3 print(math.sqrt(16))
4
```

OUTPUT:

```
4.0
```