

ASSIGNMENT-7.5

NAME: Sreeja Thalla

HALLTICKET-2303A51220

BATCH-04

1) INPUT 1:

```
⌚ ass_7.5.py > ...
1  # Bug: Mutable default argument
2 ↵ def add_item(item, items=[]):
3     items.append(item)
4     return items
5     print(add_item(1))
6     print(add_item(2))
```

```
⌚ 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]
```

2) 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     print(check_sum())
```

```
# 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
```

3) INPUT:

```
ass_7.5.py > ...
3     print(n)
4     if n == 0:
5         return
6     return countdown(n-1)
7 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
```

4) INPUT:

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

OUTPUT:

```
Key not found
```

5) 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
```

6) INPUT:

```
# Bug: Wrong unpacking
a, b = (1, 2, 3)
```

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

The screenshot shows a code editor interface with a terminal window. The terminal window has tabs: 'Welcome', '# AI-Generated Logic Without Modularizat.py', and 'ass_3.2.py'. The current tab is 'ass_7.5.py' which contains the following code:

```
ass-7.5.py > ...
1 a, b, _ = (1, 2, 3)
2 print(a, b)
3 |
```

OUTPUT:

```
• 1 2
```

7) 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. At the top, there is a menu bar with 'Go', 'Run', 'Terminal', and 'Help' options. Below the menu is a toolbar with left and right arrow buttons. A search bar on the right contains the text 'Search Ai coding — ass-7.5.py - Ai'. The main workspace shows a file named 'ass-7.5.py' with the following content:

```
# AI-Generated Logic Without Modularization
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 'naya@naya-OptiPlex-DESKTOP:~/AI Coding\$'. The output of the command is '15'.

8) 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, similar to the one above. The main workspace shows a file named 'ass-7.5.py' with the following content:

```
# AI-Generated Logic Without Modularization
ass-7.5.py
1 import math
2
3 print(math.sqrt(16))
4
```

OUTPUT:

4.0