

ASSIGNMENT-7.5

NAME:B.SAI TEJASWI

HALL TICKET-2303A51184

BATCH-03

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
   return abs((0.1 + 0.2) - 0.3) < 1e-10 # Use a small tolerance for floating point comparison
4 print(check_sum())
5
```

```
# 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)
   if n == 0:
       return
4  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

```

4)INPUT:

```
ass_7.5.py > ...
2 def get_value():
3     data = {"a": 1, "b": 2}
4 → return data["c"]
   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  < >  Ai coding
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.
```

```
Go Run Terminal Help  ← →  Q Ai cod

Welcome  # AI-Generated Logic Without Modularizat.py  ass_3.2.py

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

OUTPUT:

```
1 2
```

7)INPUT:

```
# Bug: Mixed indentation
```

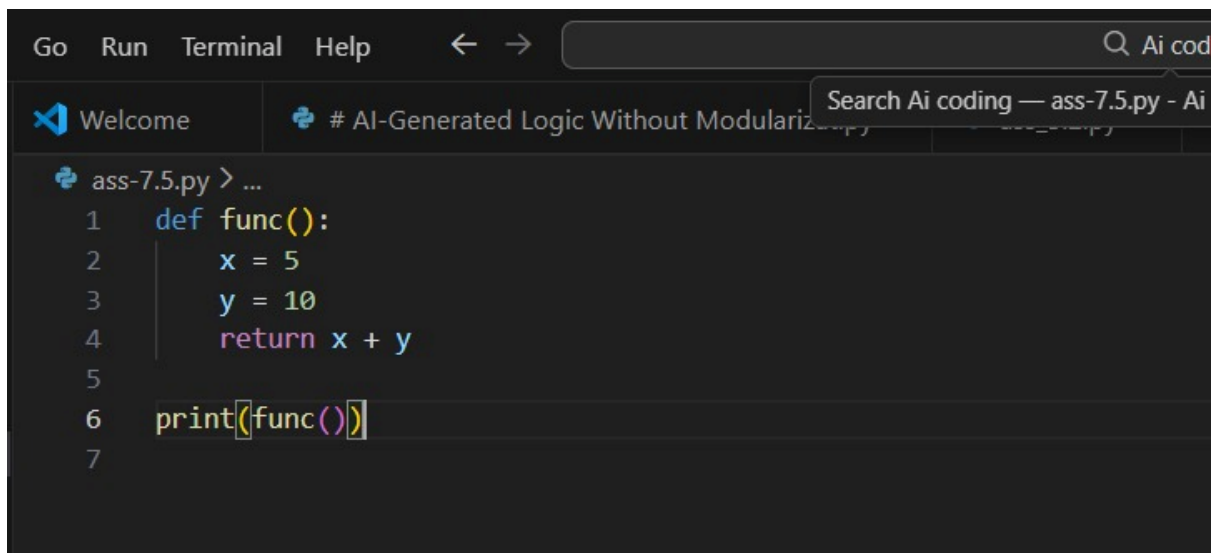
```
def func():
```

```
    x = 5
```

```
        y = 10
```

```
    return x+y
```

```
Expected Output : Consistent indentation applied.
```



A screenshot of a code editor interface. The top menu bar includes 'Go', 'Run', 'Terminal', and 'Help'. A search bar on the right contains 'Ai cod'. The editor has two tabs: 'Welcome' and '# AI-Generated Logic Without Modulariz...'. The active file is 'ass-7.5.py'. The code in the editor is as follows:

```
1 def func():
2     x = 5
3     y = 10
4     return x + y
5
6 print(func())
7
```

OUTPUT:



A screenshot of the output terminal. It shows the result of the function call: 15.

```
15
```

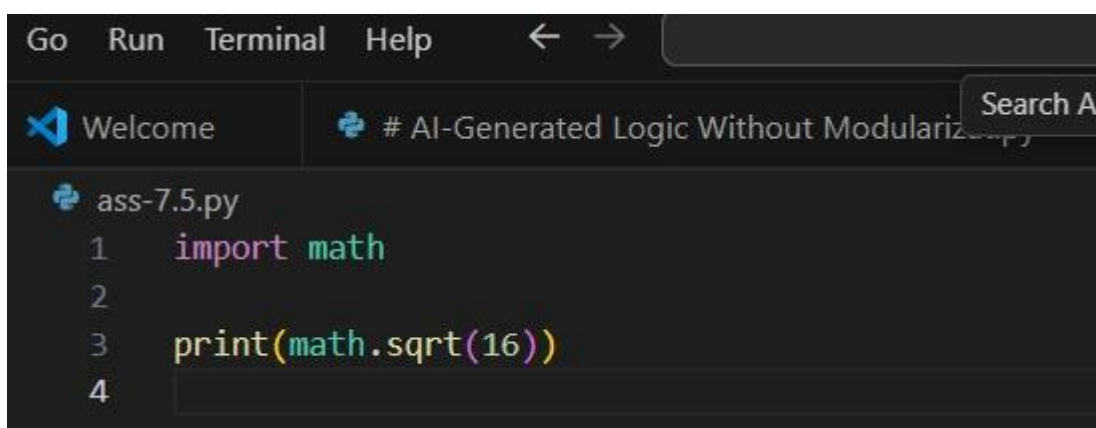
8)INPUT:

Bug: Wrong import

```
import maths
```

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

Expected Output: Corrected to import math



A screenshot of a code editor interface. The top menu bar includes 'Go', 'Run', 'Terminal', and 'Help'. A search bar on the right contains 'Search Ai'. The editor has two tabs: 'Welcome' and '# AI-Generated Logic Without Modulariz...'. The active file is 'ass-7.5.py'. The code in the editor is as follows:

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

OUTPUT:

4.0