# ASSIGNMENT -7.5

Perumalla Sushwanth

batch 29

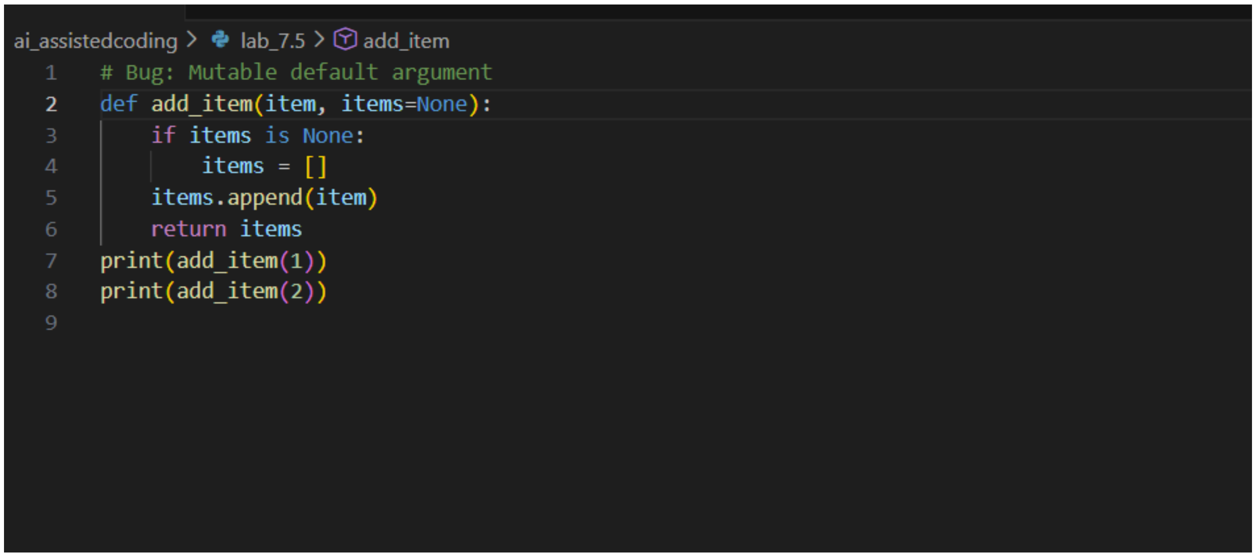
2303a51567

**Task 1 (Mutable Default Argument – Function Bug)**

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

**GIVEN:**

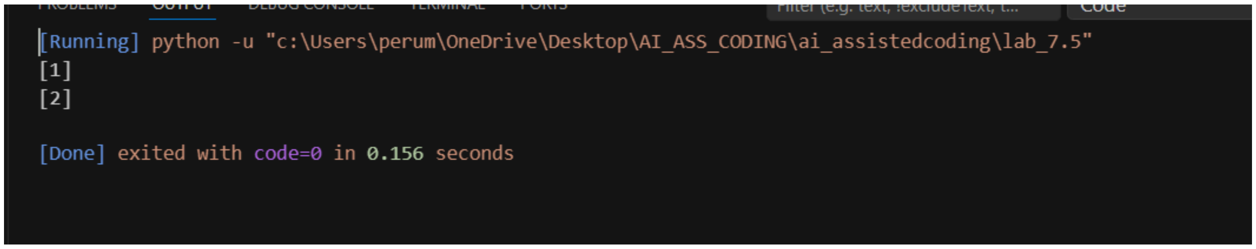
```
# Bug: Mutable default argument
def add_item(item, items=[]):
items.append(item)
return items
print(add_item(1))
print(add_item(2))
```



ai_assistedcoding > lab_7.5 > add_item

```python
# Bug: Mutable default argument
def add_item(item, items=None):
    if items is None:
        items = []
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]

[Done] exited with code=0 in 0.156 seconds
```

**Task 2 (Floating-Point Precision Error)**

Task: Analyze given code where floating-point comparison fails.  Use AI to correct with tolerance.

GIVEN:

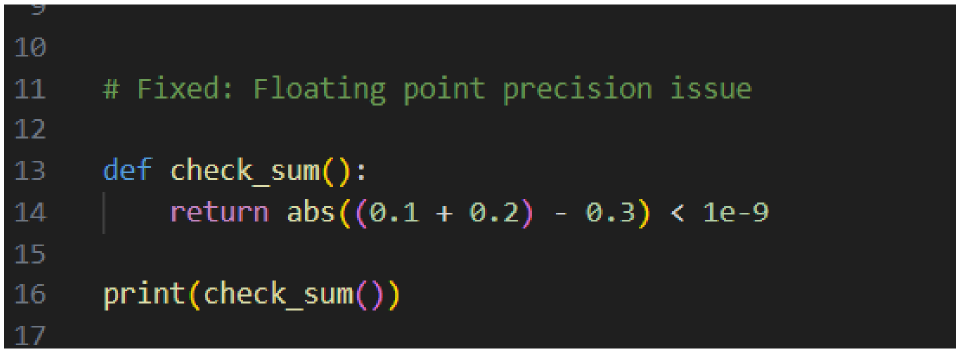
# Bug: Floating point precision issue

def check_sum():

return (0.1 + 0.2) == 0.3

print(check_sum())

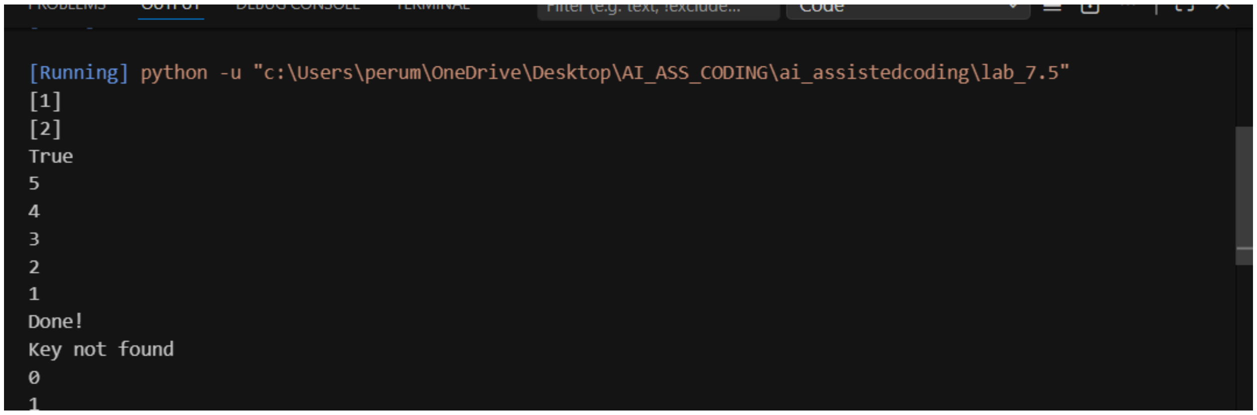
FIXED CODE:


```
# Fixed: Floating point precision issue

def check_sum():
    return abs((0.1 + 0.2) - 0.3) < 1e-9

print(check_sum())
```

OUTPUT:  True


```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
```

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

Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix.

GIVEN CODE:

# Bug: No base case

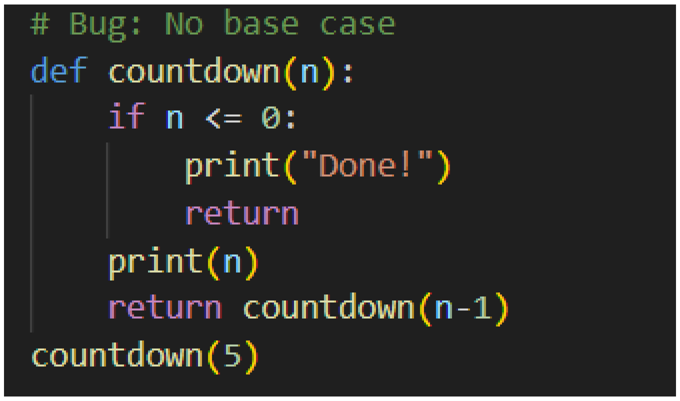
def countdown(n):

print(n)

return countdown(n-1)

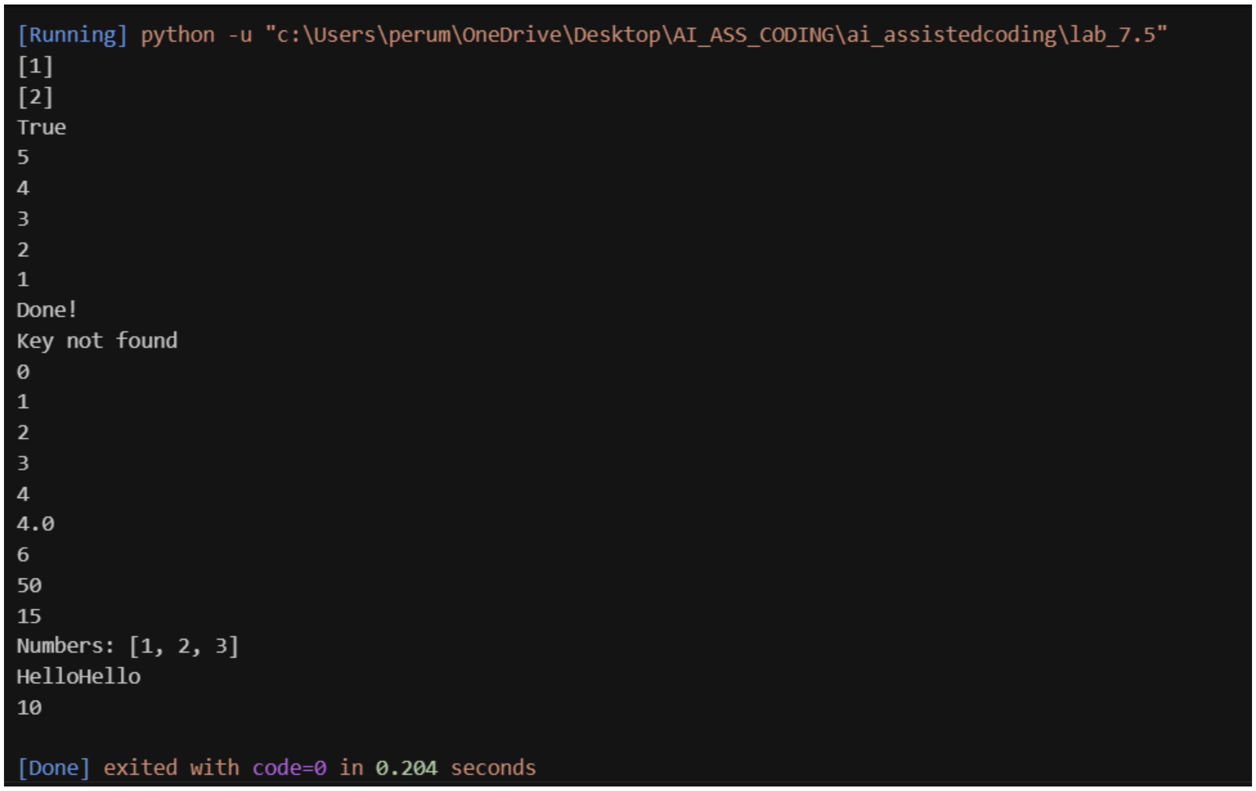
countdown(5)

FIXED CODE:



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

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

## Task 4 (Dictionary Key Error)

Task: Analyze given code where a missing dictionary key causes error. Use AI to fix it.

GIVEN CODE:

# Bug: Accessing non-existing key

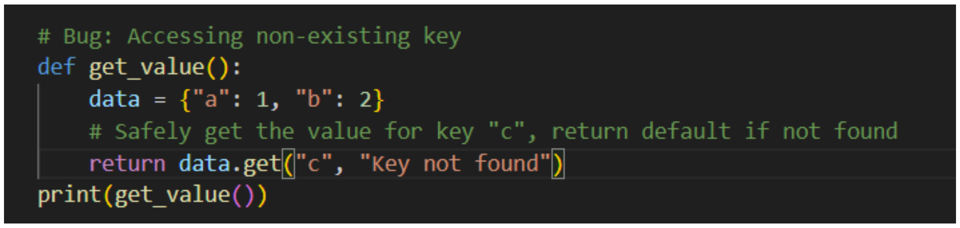
def get_value():

data = {"a": 1, "b": 2}

return data["c"]

print(get_value())

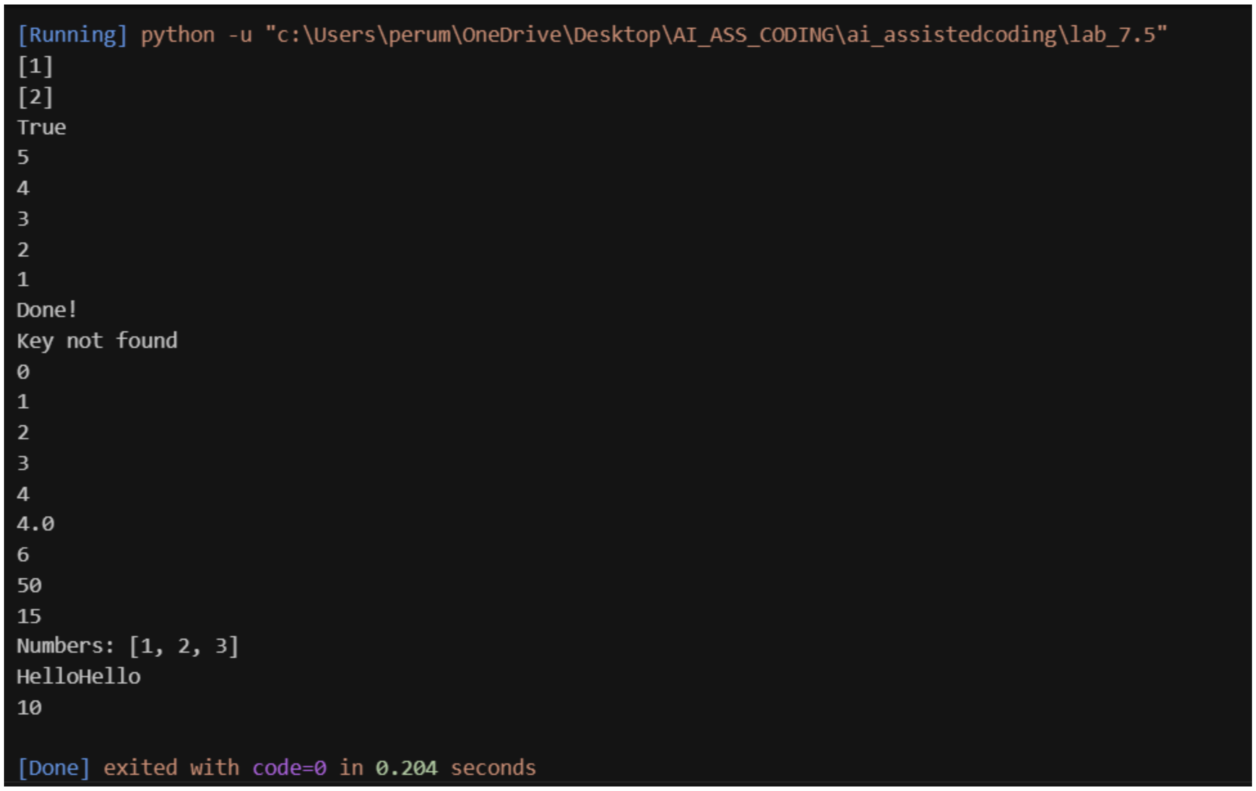
FIXED CODE:



```python
# Bug: Accessing non-existing key
def get_value():
    data = {"a": 1, "b": 2}
    # Safely get the value for key "c", return default if not found
    return data.get("c", "Key not found")
print(get_value())
```

OUTPUT

:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

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

Task: Analyze given code where loop never ends. Use AI to detect and fix it.

GIVEN CODE:

# Bug: Infinite loop

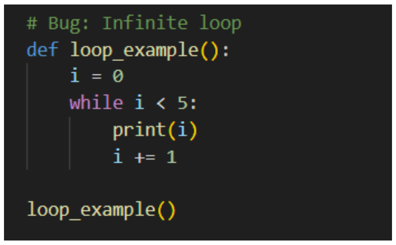
def loop_example():

i = 0

while i < 5:

print(i)

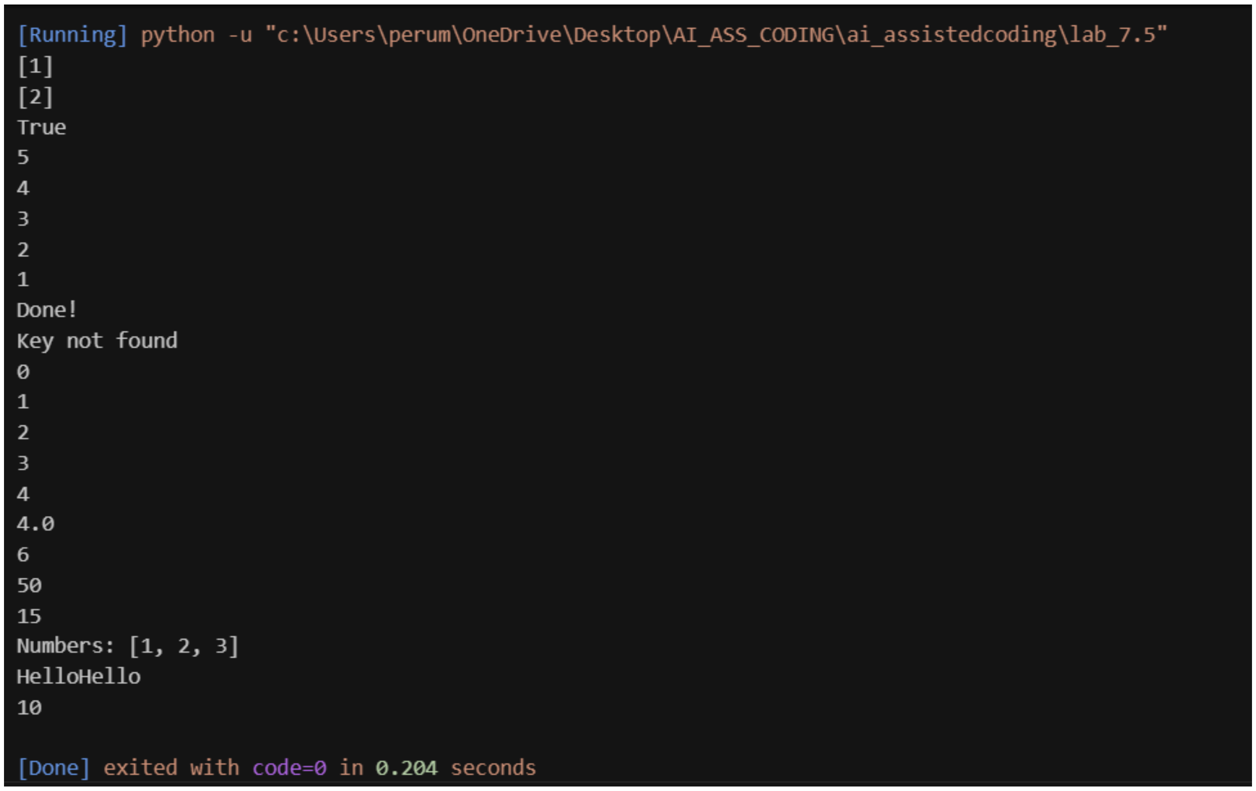
FIXED CODE:



```python
# Bug: Infinite loop
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1

loop_example()
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 6 (Unpacking Error – Wrong Variables)**

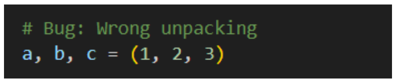
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.

GIVENCODE:

# Bug: Wrong unpacking

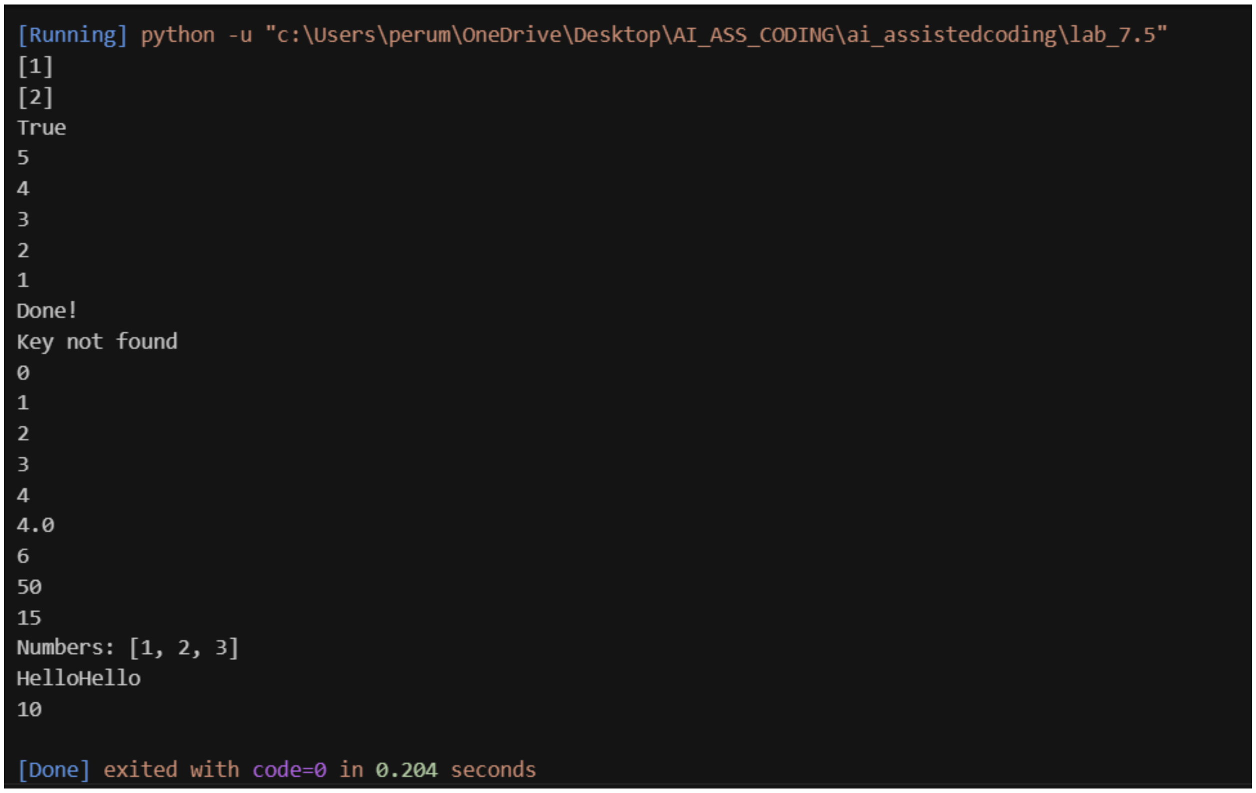
a, b = (1, 2, 3)

FIXED CODE:



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

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 7 (Mixed Indentation – Tabs vs Spaces)**

**Task:** Analyze given code where mixed indentation breaks execution. Use AI to fix it.

GIVEN CODE:

# Bug: Mixed indentation

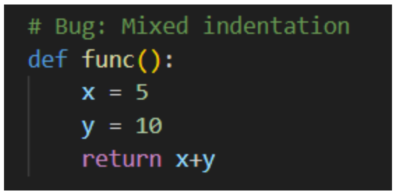
def func():

x = 5

y = 10

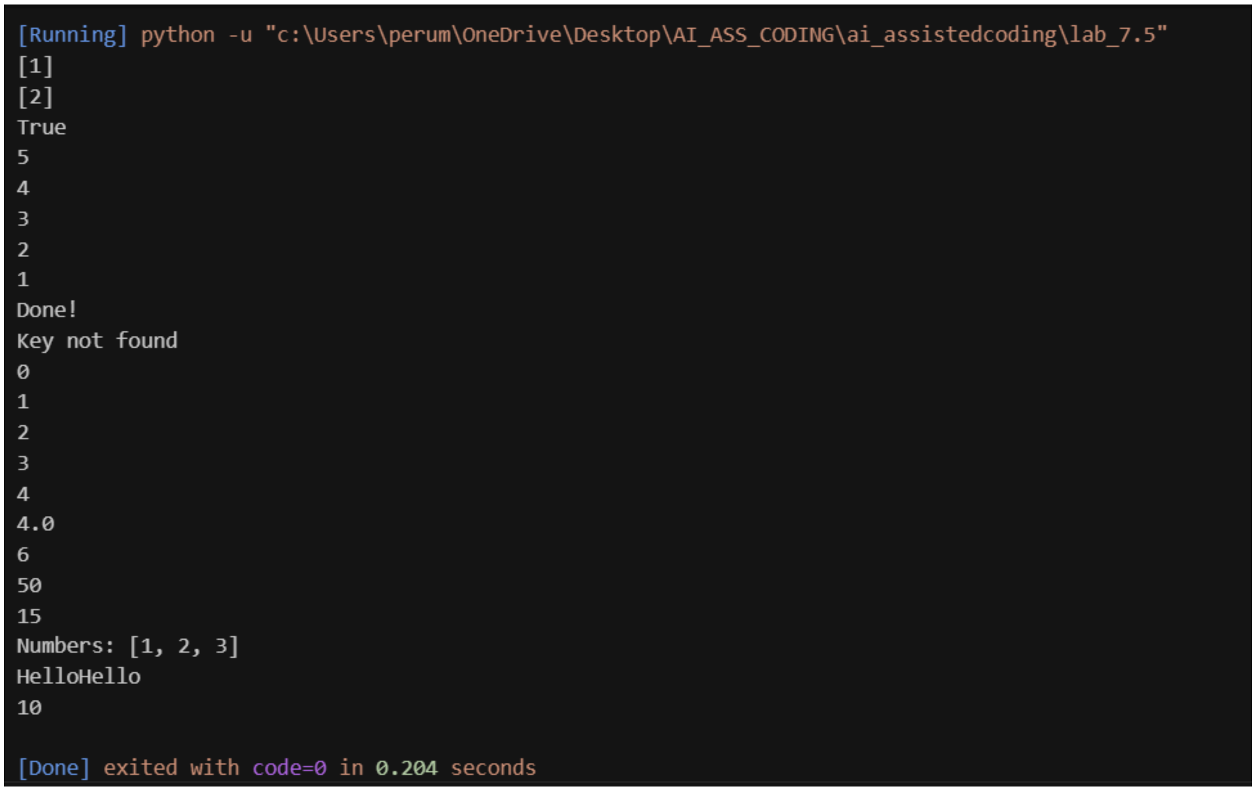
return x+y;

FIXED CODE:



```python
# Bug: Mixed indentation
def func():
    x = 5
    y = 10
    return x+y
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 8 (Import Error – Wrong Module Usage)**

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

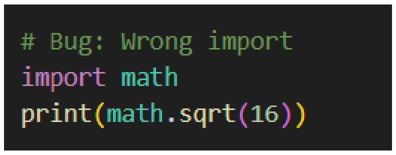
GIVEN CODE:

# Bug: Wrong import

import maths

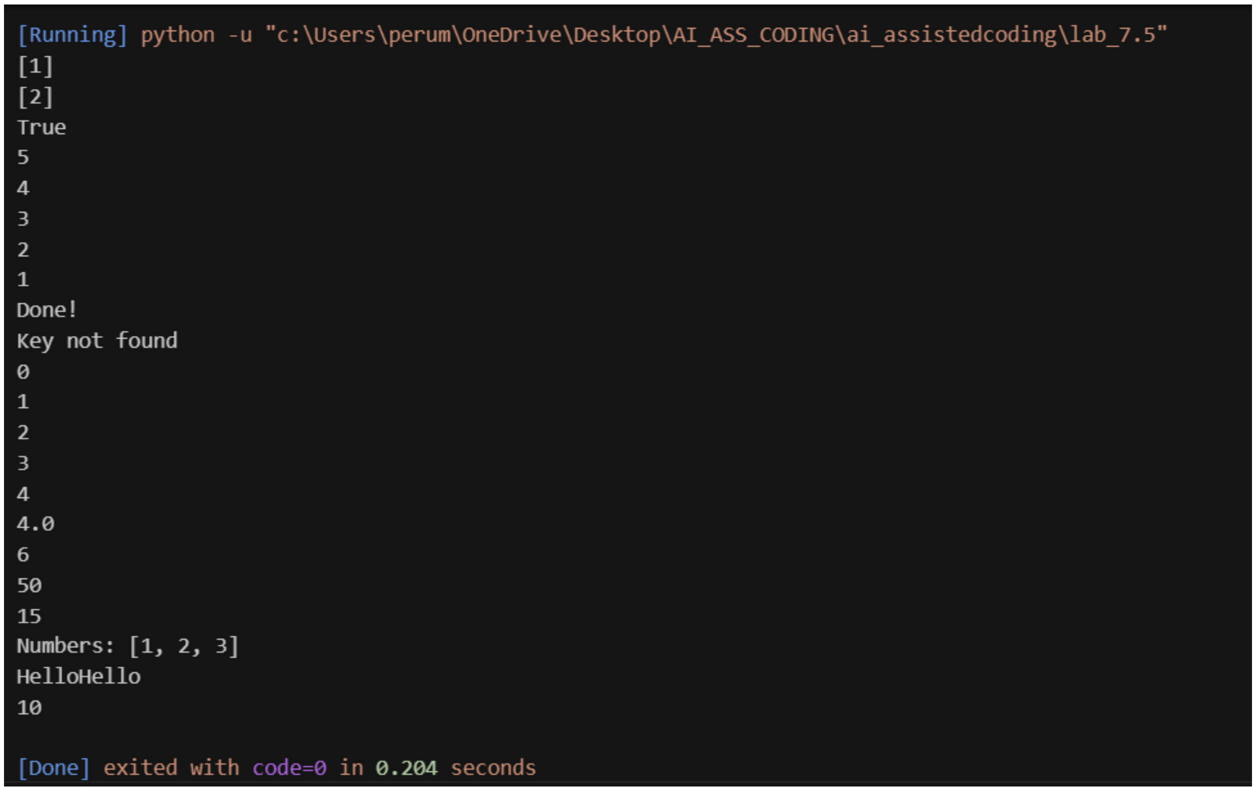
print(maths.sqrt(16))

FIXED CODE:



```python
# Bug: Wrong import
import math
print(math.sqrt(16))
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

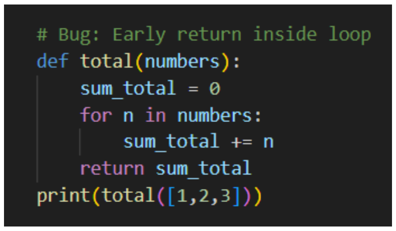
**Task 9 (Unreachable Code – Return Inside Loop)**

**Task:** Analyze given code where a return inside a loop prevents full iteration. Use AI to fix it.

GIVEN CODE:

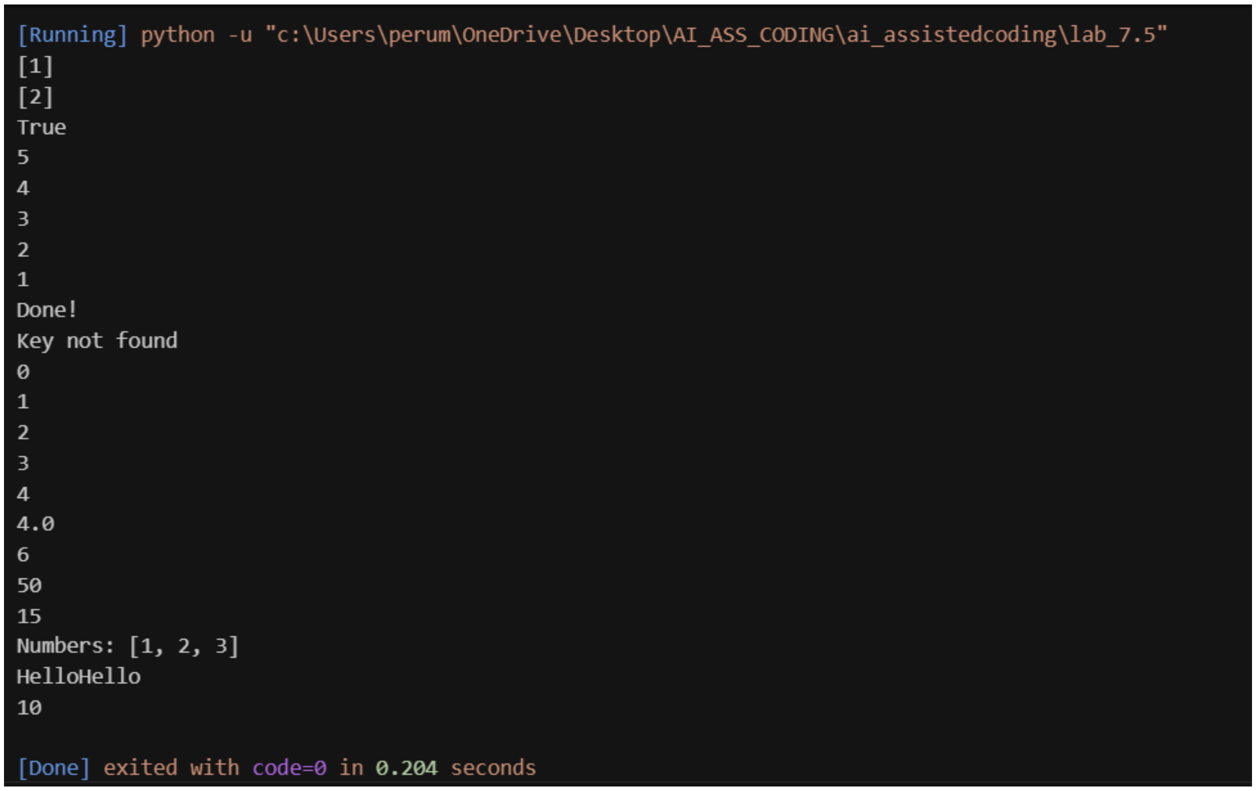
```
# Bug: Early return inside loop
def total(numbers):
for n in numbers:
return n
print(total([1,2,3]))
```

FIXED CODE:



```python
# Bug: Early return inside loop
def total(numbers):
    sum_total = 0
    for n in numbers:
        sum_total += n
    return sum_total
print(total([1,2,3]))
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 10 (Name Error – Undefined Variable)**

Task: Analyze given code where a variable is used before being defined. Let AI detect and fix the error.

GIVEN CODE:

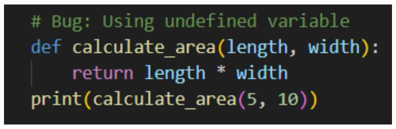
# Bug: Using undefined variable

def calculate_area():

    return length * width

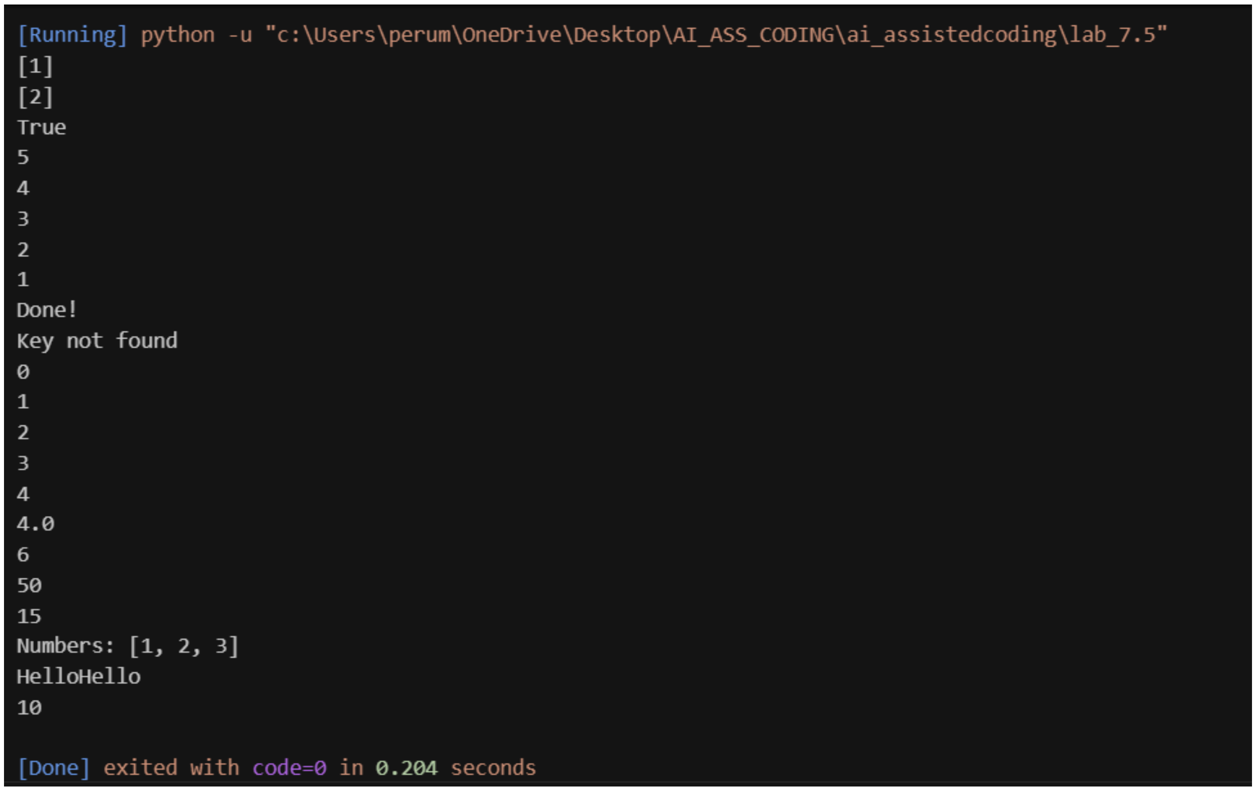
print(calculate_area())

FIXED CODE:



```
# Bug: Using undefined variable
def calculate_area(length, width):
    return length * width
print(calculate_area(5, 10))
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 11 (Type Error – Mixing Data Types Incorrectly)**

Task: Analyze given code where integers and strings are added incorrectly. Let AI detect and fix the error.

GIVEN CODE:

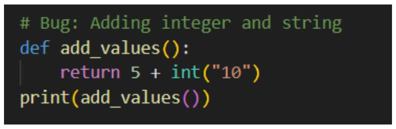
# Bug: Adding integer and string

def add_values():

return 5 + "10"

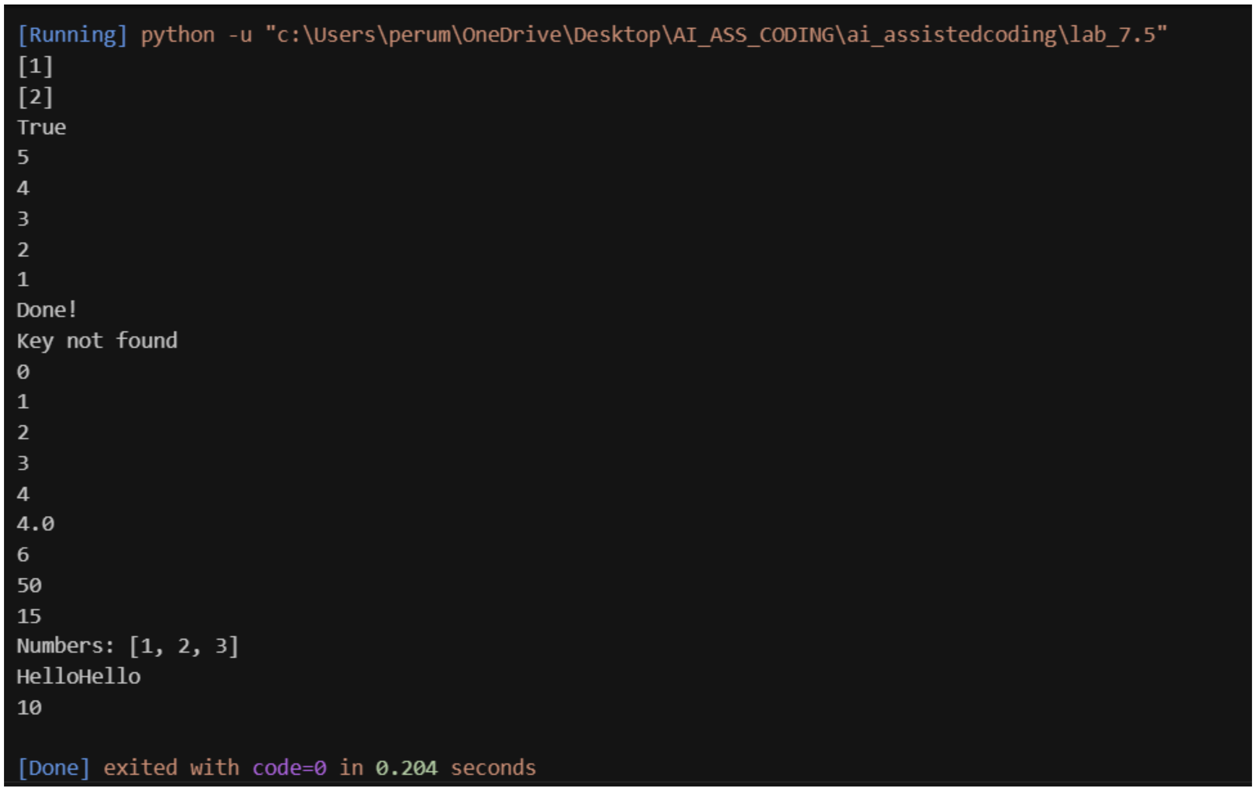
print(add_values())

FIXED CODE:



```
# Bug: Adding integer and string
def add_values():
    return 5 + int("10")
print(add_values())
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 12 (Type Error – String + List Concatenation)**

Task: Analyze code where a string is incorrectly added to a list

GIVEN CODE:

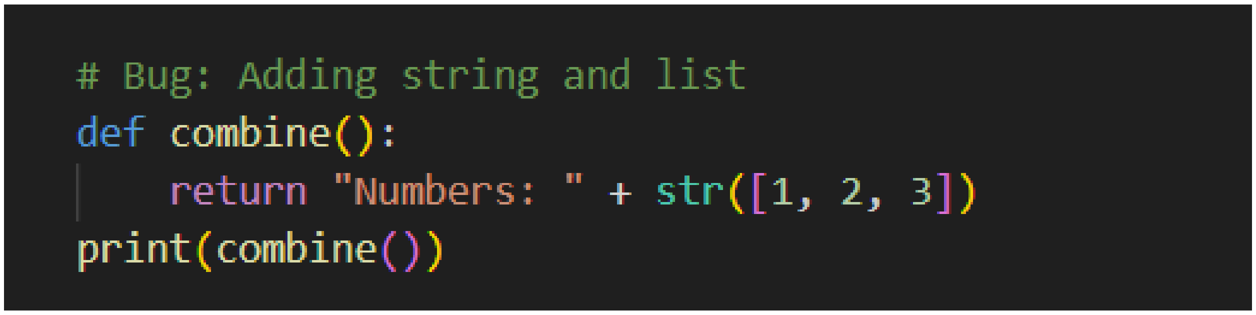
# Bug: Adding string and list

def combine():

return "Numbers: " + [1, 2, 3]

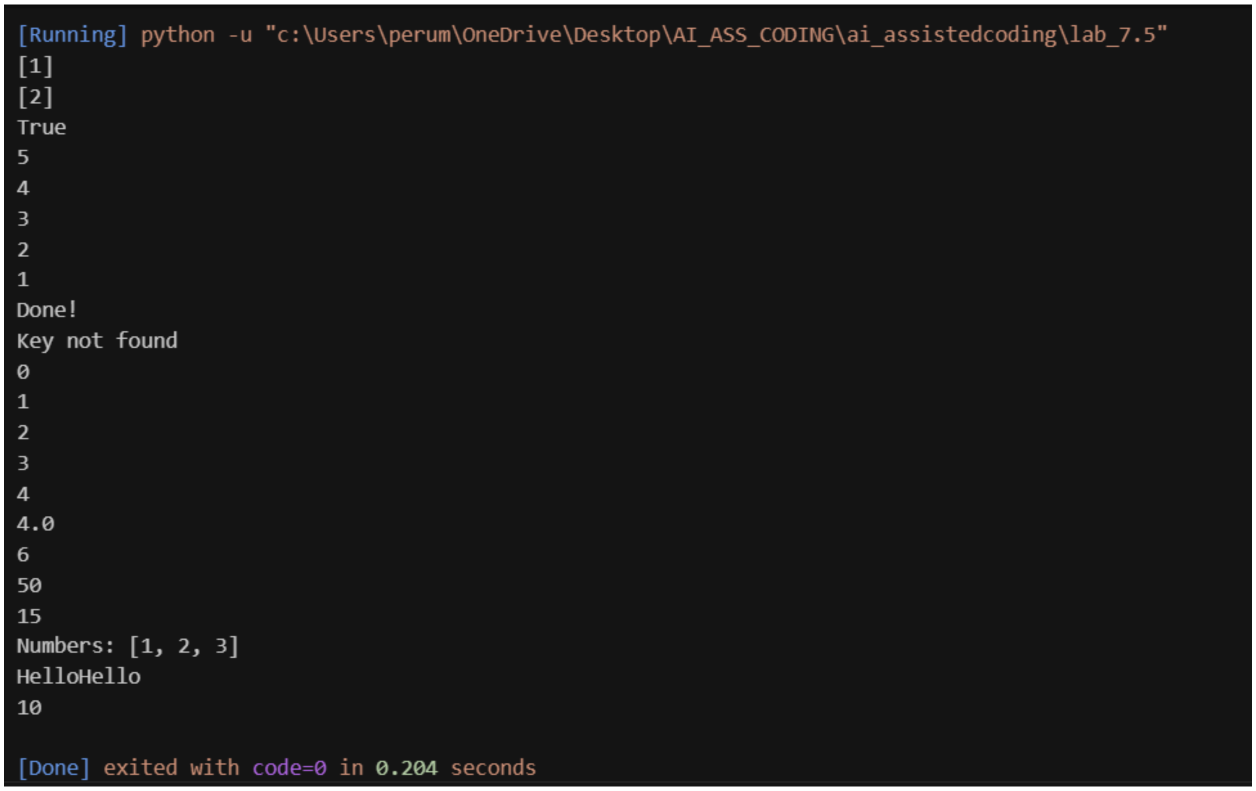
print(combine())

FIXED CODE:



```
# Bug: Adding string and list
def combine():
    return "Numbers: " + str([1, 2, 3])
print(combine())
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 13 (Type Error – Multiplying String by Float)**

Task: Detect and fix code where a string is multiplied by a float.

GIVEN CODE:

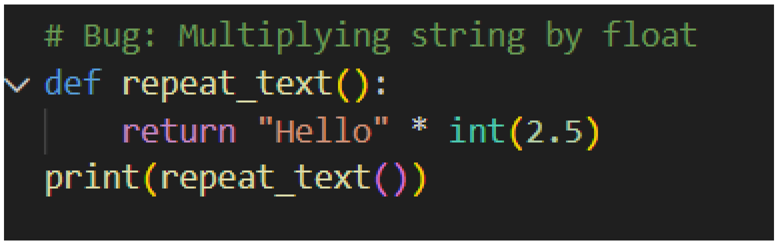
# Bug: Multiplying string by float

def repeat_text():

return "Hello" * 2.5

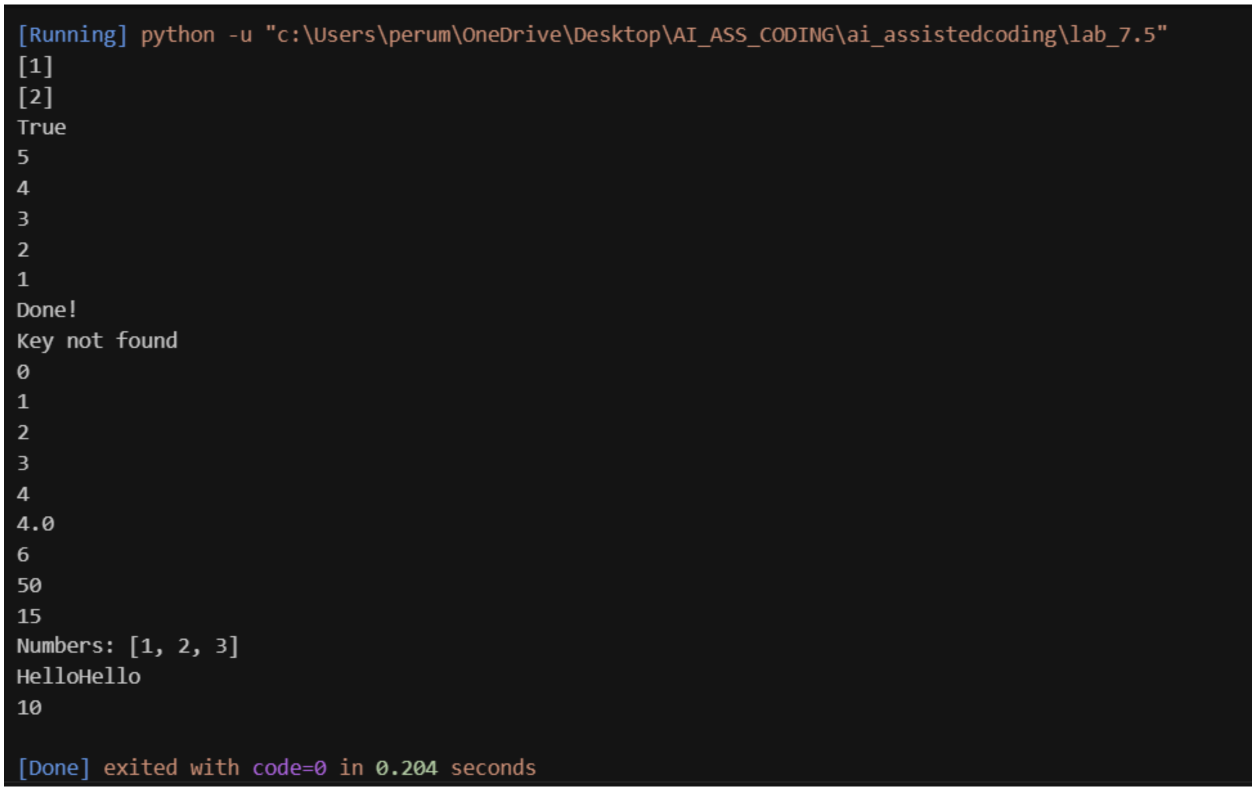
print(repeat_text())

FIXED CODE:



```python
# Bug: Multiplying string by float
def repeat_text():
    return "Hello" * int(2.5)
print(repeat_text())
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 14 (Type Error – Adding None to Integer)**

Task: Analyze code where None is added to an integer.

GIVEN CODE:

# Bug: Adding None and integer

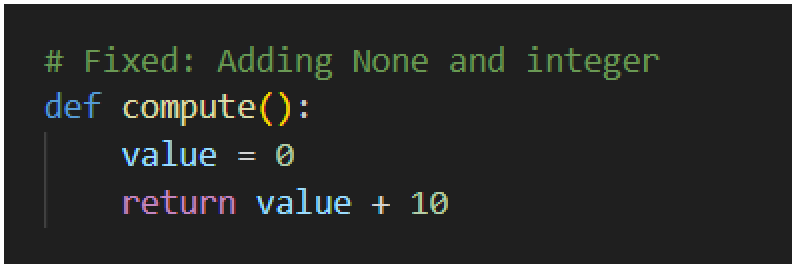
def compute():

value = None

return value + 10

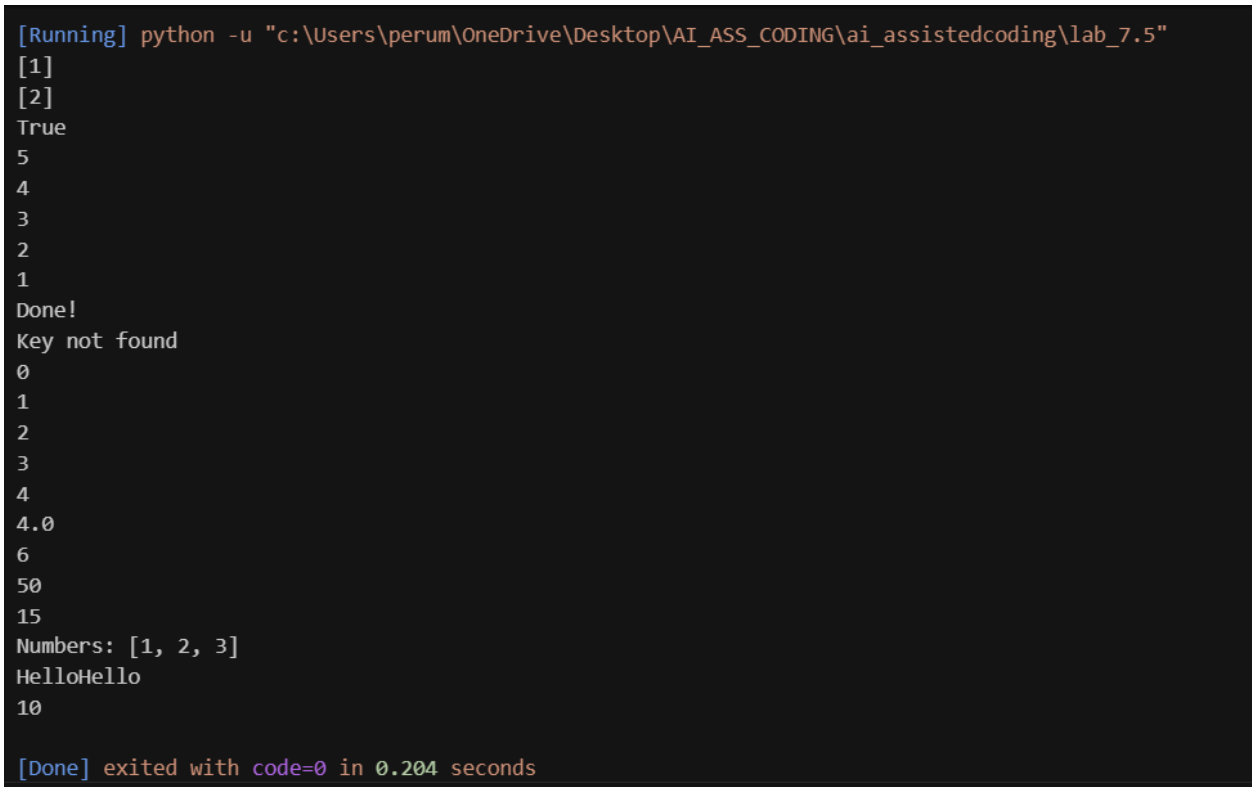
print(compute())

FIXED CODE:



```python
# Fixed: Adding None and integer
def compute():
    value = 0
    return value + 10
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```

**Task 15 (Type Error – Input Treated as String Instead of Number)**

Task: Fix code where user input is not converted properly.

GIVEN CODE:

# Bug: Input remains string

def sum_two_numbers():

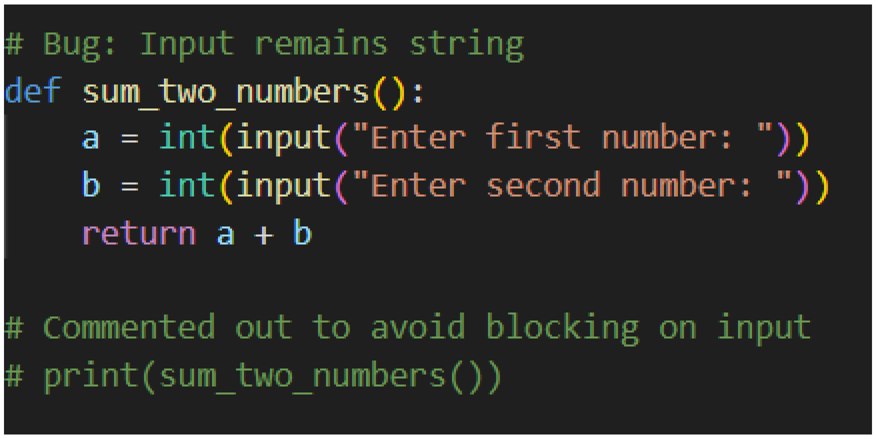
a = input("Enter first number: ")

b = input("Enter second number: ")

return a + b

print(sum_two_numbers())

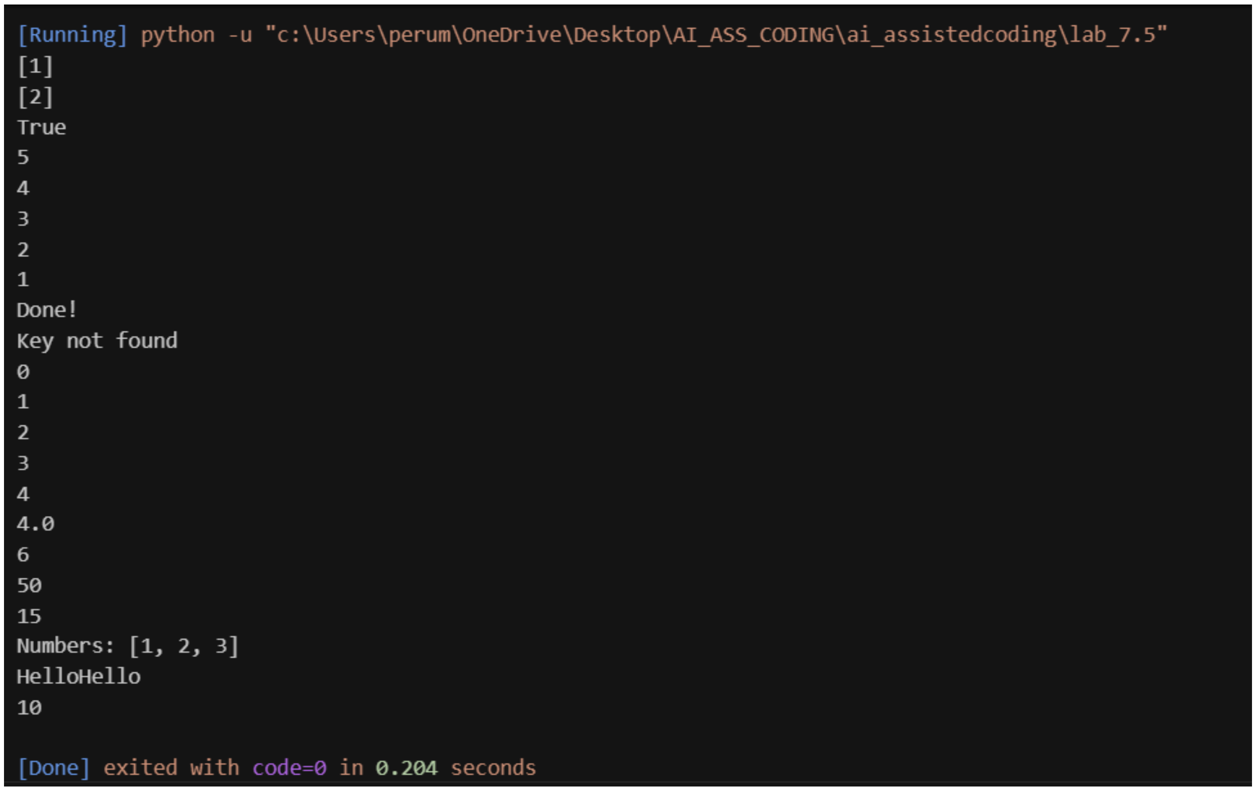
FIXED CODE:



```python
# Bug: Input remains string
def sum_two_numbers():
    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))
    return a + b

# Commented out to avoid blocking on input
# print(sum_two_numbers())
```

OUTPUT:



```
[Running] python -u "c:\Users\perum\OneDrive\Desktop\AI_ASS_CODING\ai_assistedcoding\lab_7.5"
[1]
[2]
True
5
4
3
2
1
Done!
Key not found
0
1
2
3
4
4.0
6
50
15
Numbers: [1, 2, 3]
HelloHello
10

[Done] exited with code=0 in 0.204 seconds
```