LAB ASSIGNMENT – 7.5

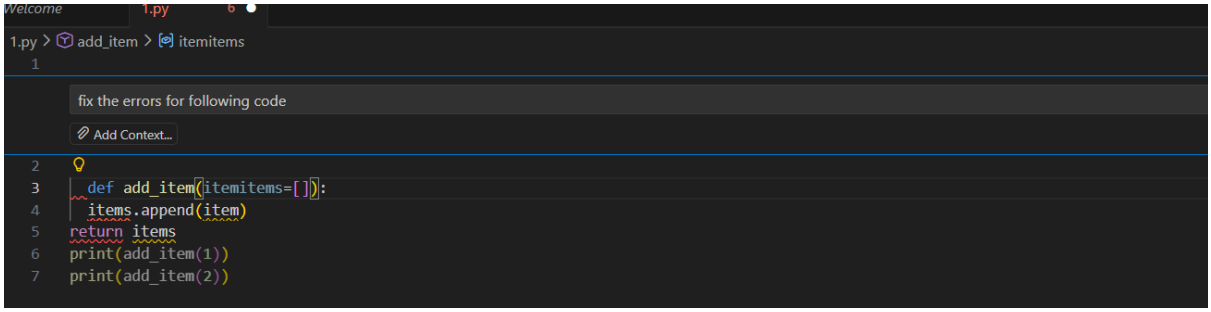
NAME:- CH VISHWANATH RAO

BATCH:02

HT NO:- 2303A51095

Task 1 (Mutable Default Argument – Function Bug)

Error code:-



CORRECTED CODE:

CODE:

Output:-

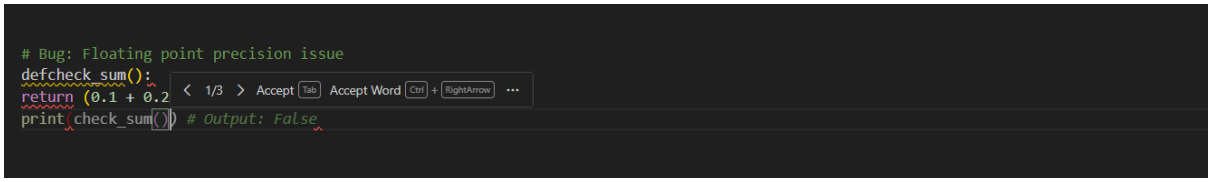
```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDriv
[1]
[2]
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDriv
[1]
[2]
[3]
[4]
```

Task 2 (Floating-Point Precision Error)

Task: Analyse the given code where floating-point comparison fails.

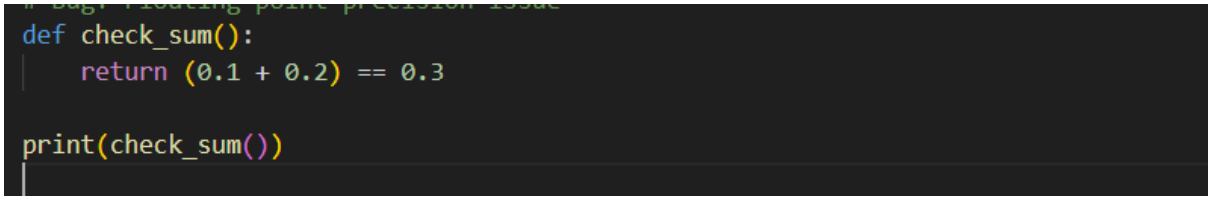
Use AI to correct with tolerance.

ERROR CODE:

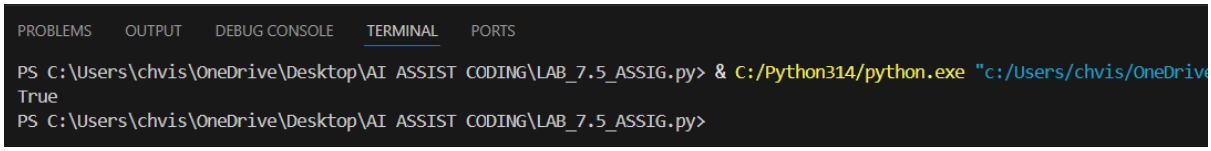


CORRECTED CODE:

CODE:



OUTPUT:

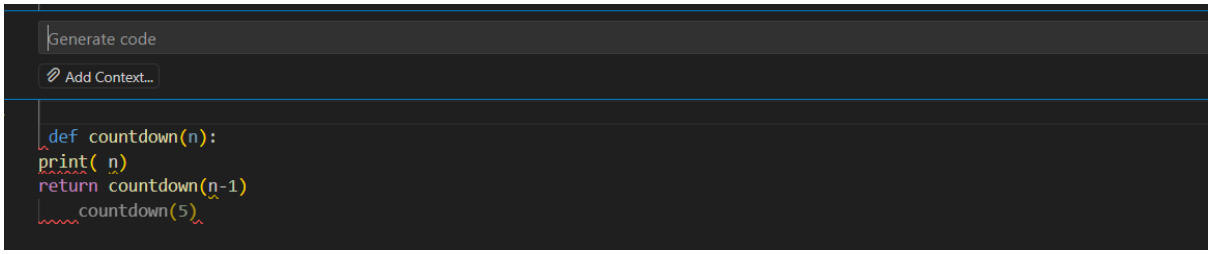


Task 3 (Recursion Error – Missing Base Case)

Task: Analyse the given code where the recursion runs infinitely due to

a missing base case. Use AI to fix.

ERROR CODE:



CORRECTED CODE:

```
def countdown(n):
    if n == 0:
        return
    print(n)
    return countdown(n-1)

countdown(5)
```

CODE:

```
def countdown(n):
    if n == 0:
        return
    print(n)
    return countdown(n-1)

countdown(10)
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users
10
9
8
7
6
5
4
3
2
1
```

Task 4 (Dictionary Key Error):

ERROR CODE:

```
    def get_value():
data = {"a": 1"b": 2}
return data["c"]
print(get_value()
```

Corrected code:-

```
def get_value(data, key):
    if key in data:
        return data[key]
    else:
        return None

data = {"a": 1, "b": 2}
print(get_value(data, "c"))
#Output: None
data = {"a": 1, "b": 2, "c": 3}
print(get_value(data, "c"))
```

Code:

```
def get_value(data, key):
    if key in data:
        return data[key]
    else:
        return None

data = {"a": 1, "b": 2}
print(get_value(data, "c"))
#Output: None
data = {"a": 1, "b": 2, "c": 3}
print(get_value(data, "c"))
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/D
None
3
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

Task 5 (Infinite Loop – Wrong Condition)

Task: Analyse the given code where the loop never ends. Use AI to detect

and fix it.

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.

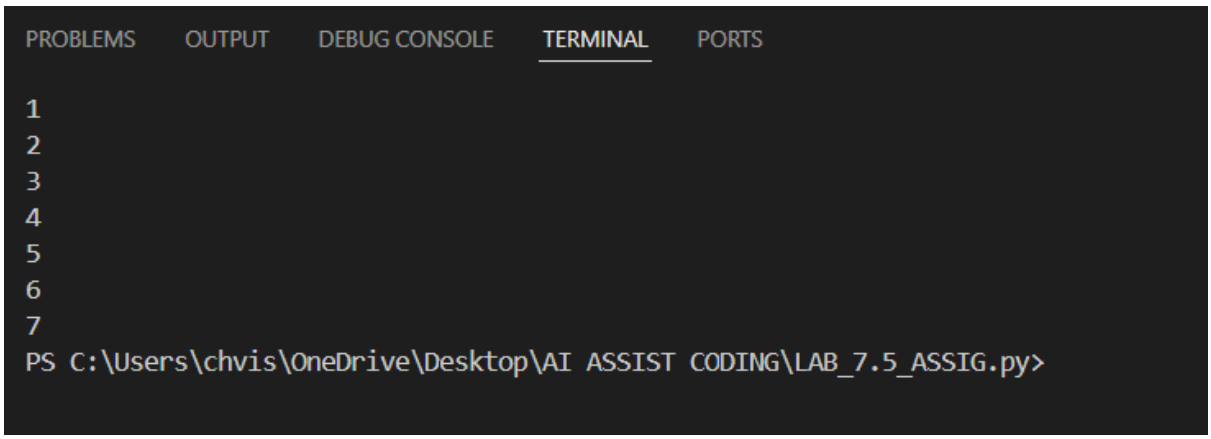
ERROR CODE:

```
defloop_example():
i = 0
while i < 5:
print(i)
```

Corrected code:

```
def loop_expl():
    m = 0
    while m < 8:
        print(m)
        m += 1
loop_expl()
```

OUTPUT:



Task 6 (Unpacking Error – Wrong Variables)

Task: Analyse the 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.

ERROR CODE:

```
#wrong unpacking correct the code below code unpacking or using _ for extra values

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

CORRECTED CODE:

```
36
37    a, b, _ = (1, 2, 3)
38    print(a, b)
```

CODE:

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

print(a, b)

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/
G.py/1.py"
1 2 [3, 4, 5]
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

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.

ERROR CODE:

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

CORRECTED CODE:

```
def func():
    x = 5
    y = 10
    return x+y
print(func())
def func2(x, y):
    return x+y
print(func2(5, 10))
```

CODE:

```
def func():
   x = 5
   y = 10
   return x+y
print(func())
def func2(x, y):
   return x+y
print(func2(5, 10))
```

OUTPUT:

```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/Desktop/AI ASSIST CODING/LAB_7
G.py/1.py"
15
15
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

Task 8 (Import Error – Wrong Module Usage):

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

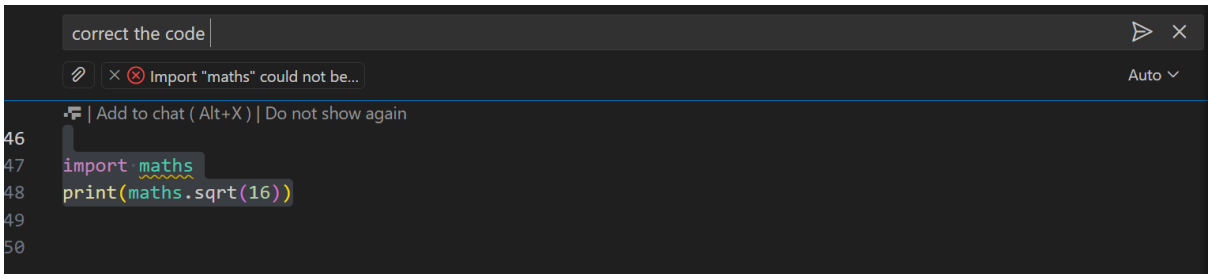
 # Bug: Wrong import

import maths

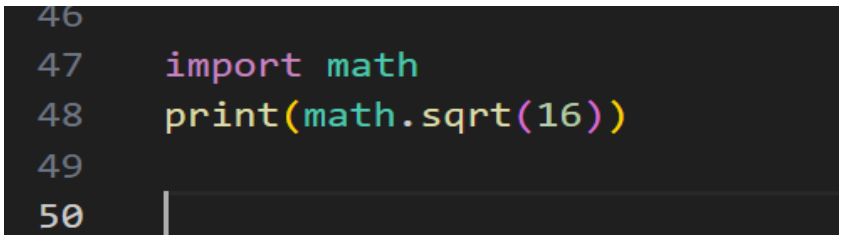
print(maths.sqrt(16))

Expected Output: Corrected to import math

ERROR CODE:



CORRECTED CODE:

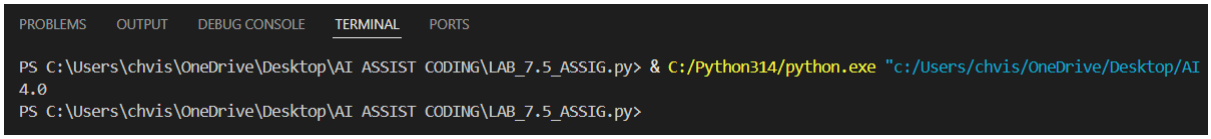


CODE:

import math

print(math.sqrt(16))

OUTPUT:



Task 9 (Unreachable Code – Return Inside Loop)

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

 # Bug: Early return inside loop

def total(numbers):

for n in numbers:

return n

print(total([1,2,3]))

Expected Output: Corrected code accumulates the sum and returns after the loop.

ERROR CODE:

```
deftotal(numbers)
for n in numbers:
return n
print(total([1,2,3]))
```

CORRECTED CODE:

```
def total(numbers):
    sum_total = 0
    for n in numbers:
        sum_total += n
    return sum_total

print(total([1,2,3]))
```

Output:

```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/Desktop/AI
6
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

Task 10 (Name Error – Undefined Variable)

Task: Analyse the given code where a variable is used before being

defined. Let AI detect and fix the error.

Bug: Using undefined variable

def calculate_area():

return length * width

print(calculate_area())

Requirements:

• Run the code to observe the error.

• Ask AI to identify the missing variable definition.

• Fix the bug by defining length and width as parameters.

• Add 3 assert test cases for correctness.

Expected Output :

• Corrected code with parameters.

• AI explanation of the bug.

Successful execution of assertions.

ERROR CODE:

```
def calculate_area ()
return length width
print(calculate_area())
```

CORRECTED CODE:

```
#fix the above code errors and give
def calculate_area(length, width):
    return length * width
print(calculate_area(5, 6))
```

Output:

```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/Desktop/AI ASSIST CODING/LAB_7.5_ASSIG.py/1.py"
30
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

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.

Bug: Adding integer and string

def add_values():

return 5 + "10"

print(add_values())

Requirements:

• Run the code to observe the error.

• AI should explain why int + str is invalid.

• Fix the code by type conversion (e.g., int("10") or str(5)).

• Verify with 3 assert cases.

Expected Output #6:

• Corrected code with type handling.

• AI explanation of the fix.

Successful test validation.

ERROR CODE:

```
def add_values()
return 5 + "10"
print(add_values()
```

CORRECTED CODE:

```
def add_values(a, b):
    return a + b
# Function call
print(add_values(5, 10))
```

Output:

```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/Desktop/AI ASSIST CODING/LAB_7.5_ASSIG.py/1.
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/Desktop/AI ASSIST CODING/LAB_7.5_ASSIG.py/1.
15
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

Task 12 (Type Error – String + List Concatenation)

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

Bug: Adding string and list

def combine():

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

print(combine())

Requirements:

• Run the code to observe the error.

• Explain why str + list is invalid.

• Fix using conversion (str([1,2,3]) or " ".join()).

• Verify with 3 assert cases.

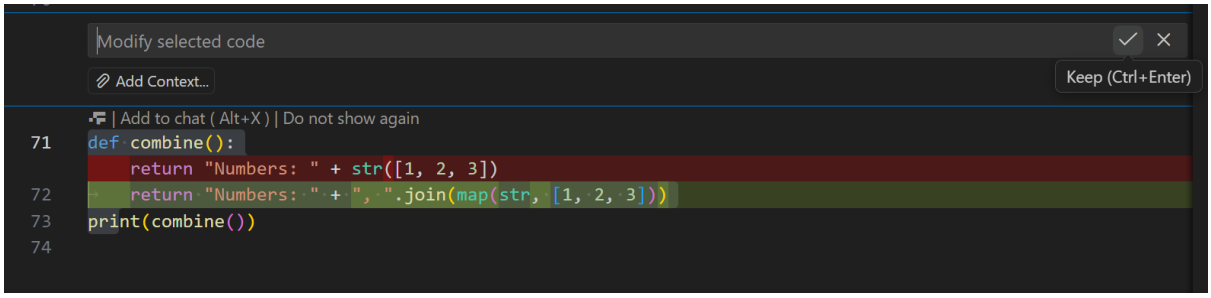
Expected Output:

• Corrected code

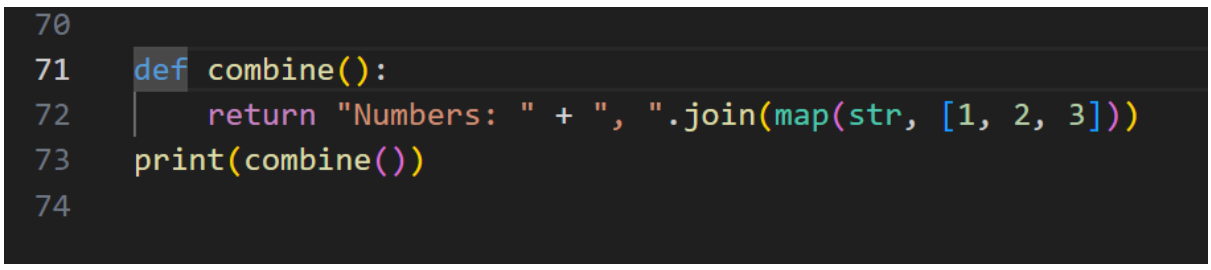
• Explanation

• Successful test validation

ERROR CODE:



CORRECTED CODE:



OUTPUT:

```
Numbers: 1, 2, 3
```

Task 13 (Type Error – Multiplying String by Float)

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

Bug: Multiplying string by float

def repeat_text():

return "Hello" * 2.5

print(repeat_text())

Requirements:

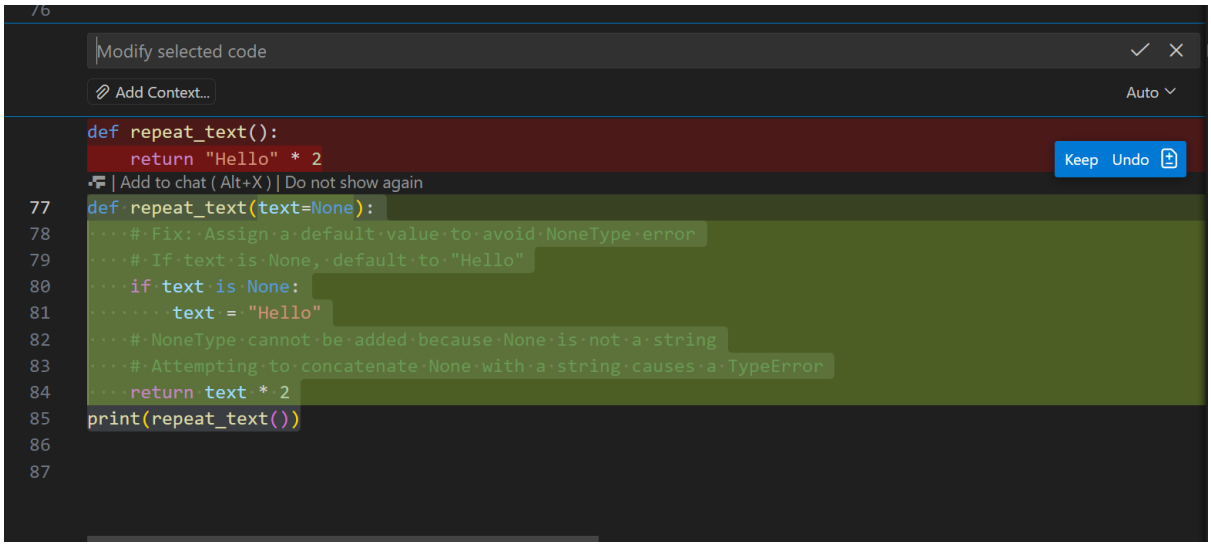
• Observe the error.

• Explain why float multiplication is invalid for strings.

• Fix by converting float to int.

• Add 3 assert test cases.

ERROR CODE:



CORRECTED CODE:

```
def repeat_text(msg=None):

    if msg is None:
        msg = "Hello"
    print(msg)
    return msg * 2
print(repeat_text())
```

Output:

```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive
Hello
HelloHello
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```

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

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

 # 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())

Requirements:

- **Explain why input is always string.**
- **Fix using int() conversion.**
- **Verify with assert test cases.**

ERROR CODE:

```
def sum_two_numbers()
a = input("Enter first number: ")
b = input("Enter second number")
return a + b
print(sum_two_numbers()
```

CORRECTED CODE:

```
#fix the above errors of the code and give the right one
def two_numbers_sum():
    a = int(input("Enter 1st number: "))
    b = int(input("Enter 2nd number: "))
    return a + b
print(two_numbers_sum())
```

OUTPUT:

```
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py> & C:/Python314/python.exe "c:/Users/chvis/OneDrive/Desktop/AI ASSI
Enter 1st number: 20
Enter 2nd number: 45
65
PS C:\Users\chvis\OneDrive\Desktop\AI ASSIST CODING\LAB_7.5_ASSIG.py>
```