SR UNIVERSITY

School of Computer Science and Artificial Intelligence

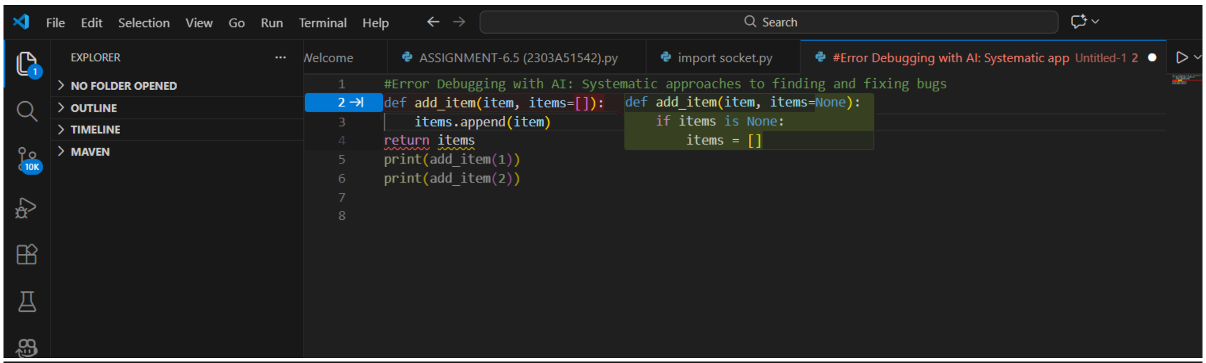
Lab Assignment # 7

Name of Student : R.SHIVASAI REDDY
Enrollment No. : 2303A51542
Batch No. :22

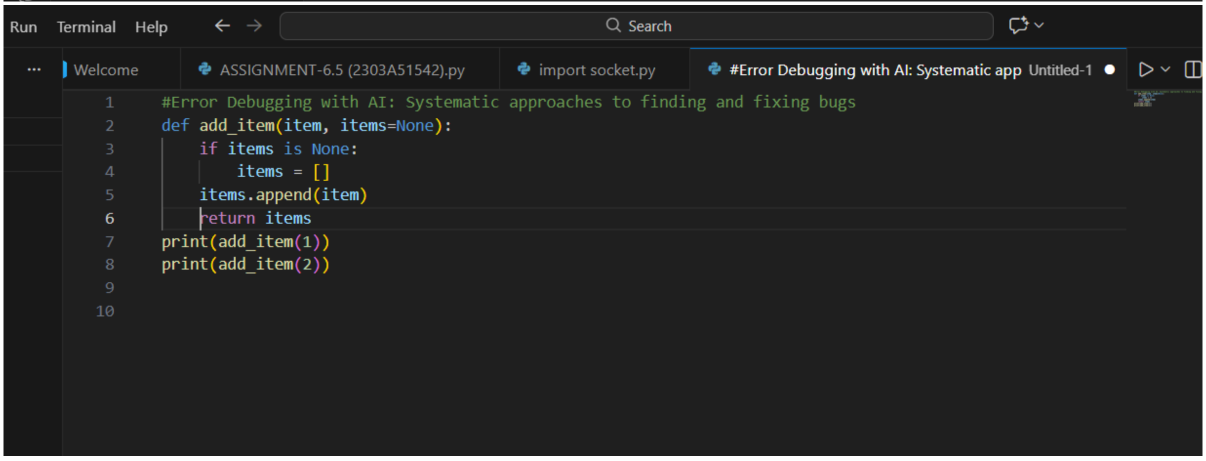
1Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it. # Bug: Mutable default argument def add_item(item, items=[]): items.append(item) return items print(add_item(1)) print(add_item(2))
Expected Output: Corrected function avoids shared list bug.

Prompt:given python program corrected function avoid shared list bug

Code:-

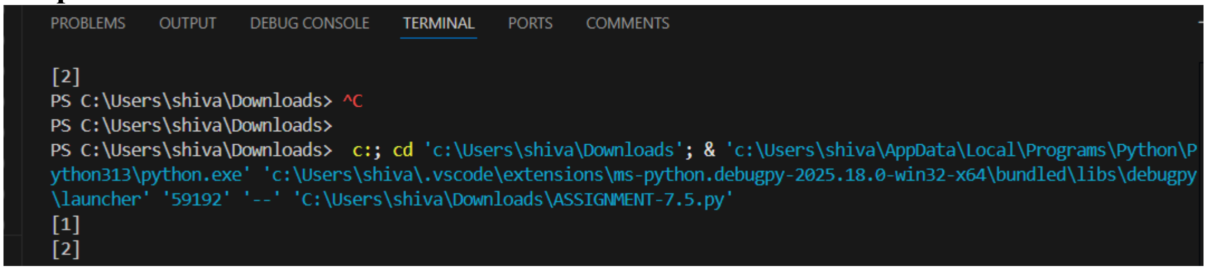


```
#Error Debugging with AI: Systematic approaches to finding and fixing bugs
def add_item(item, items=[]):    def add_item(item, items=None):
    items.append(item)                if items is None:
return items                              items = []
print(add_item(1))
print(add_item(2))
```



```
#Error Debugging with AI: Systematic approaches to finding and fixing bugs
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:-



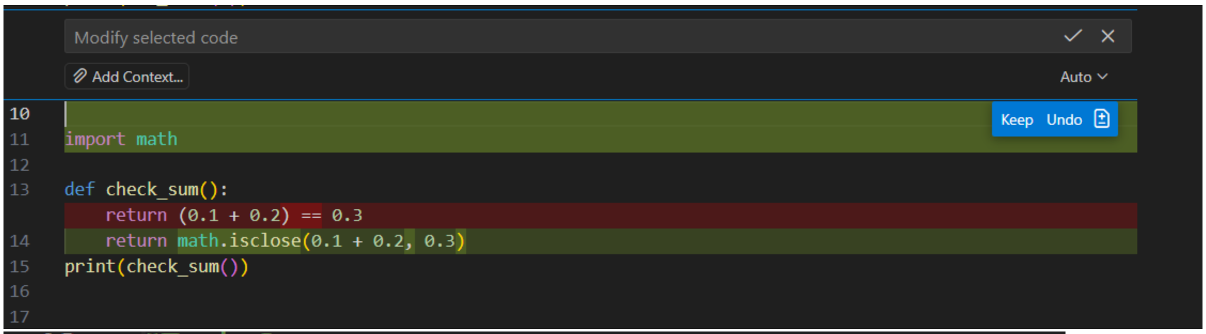
```
[2]
PS C:\Users\shiva\Downloads> ^C
PS C:\Users\shiva\Downloads>
PS C:\Users\shiva\Downloads>  c:; cd 'c:\Users\shiva\Downloads'; & 'c:\Users\shiva\AppData\Local\Programs\Python\P
ython313\python.exe' 'c:\Users\shiva\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy
\launcher' '59192' '--' 'C:\Users\shiva\Downloads\ASSIGNMENT-7.5.py'
[1]
[2]
```

Justification:
The function add_item is defined with a default parameter items that is a list. If the list is not provided, it is initialized as an empty list. The function appends the item to the list and returns the list.

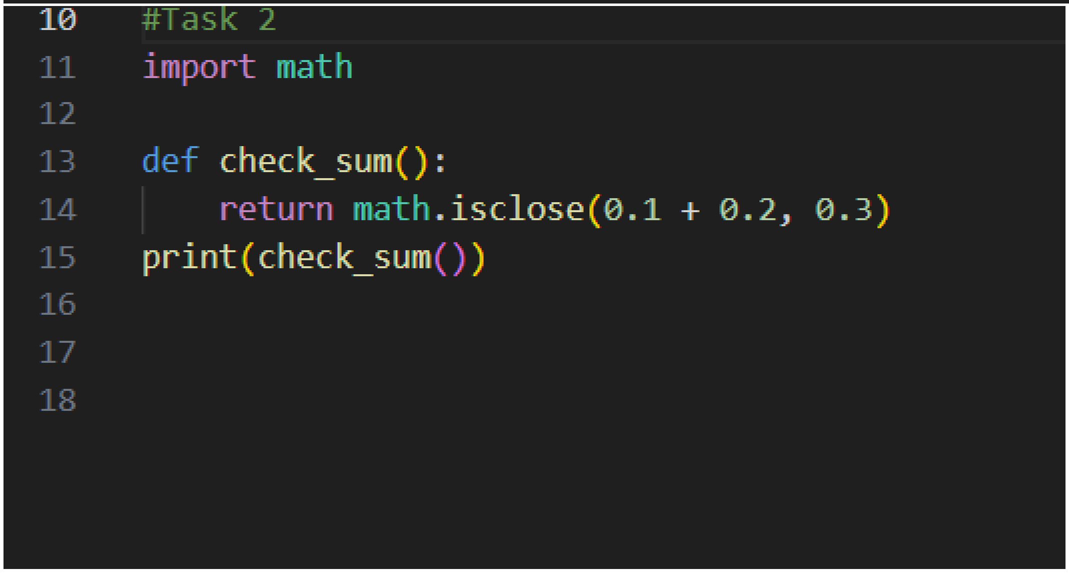
2Task: Analyze given code where floating-point comparison fails.
Use AI to correct with tolerance. #
Bug: Floating point precision issue
def check_sum(): return (0.1 + 0.2)
== 0.3 print(check_sum())
Expected Output: Corrected function

Prompt:fix the below python program using correct function with tolerance.

Code:-

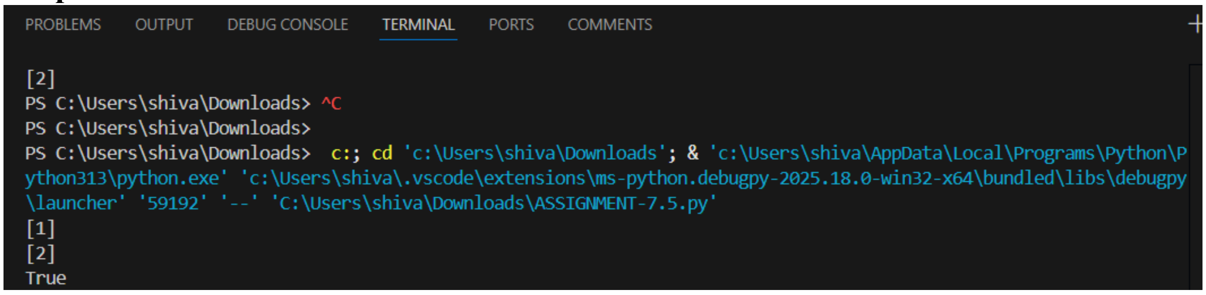


```

import math

def check_sum():
    return (0.1 + 0.2) == 0.3
    return math.isclose(0.1 + 0.2, 0.3)
print(check_sum())
```



```
#Task 2
import math

def check_sum():
    return math.isclose(0.1 + 0.2, 0.3)
print(check_sum())
```

Output:-



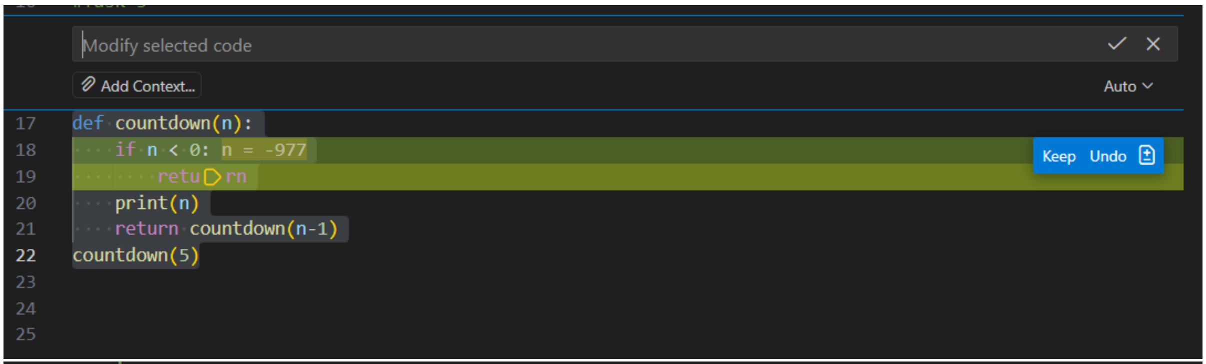
```
[2]
PS C:\Users\shiva\Downloads> ^C
PS C:\Users\shiva\Downloads>
PS C:\Users\shiva\Downloads>  c:; cd 'c:\Users\shiva\Downloads'; & 'c:\Users\shiva\AppData\Local\Programs\Python\P
ython313\python.exe' 'c:\Users\shiva\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy
\launcher' '59192' '--' 'C:\Users\shiva\Downloads\ASSIGNMENT-7.5.py'
[1]
[2]
True
```

Justification:-
The function check_sum() is used to check if the sum of 0.1 and 0.2 is equal to 0.3. The function returns True if the sum is equal to 0.3, otherwise it returns False.

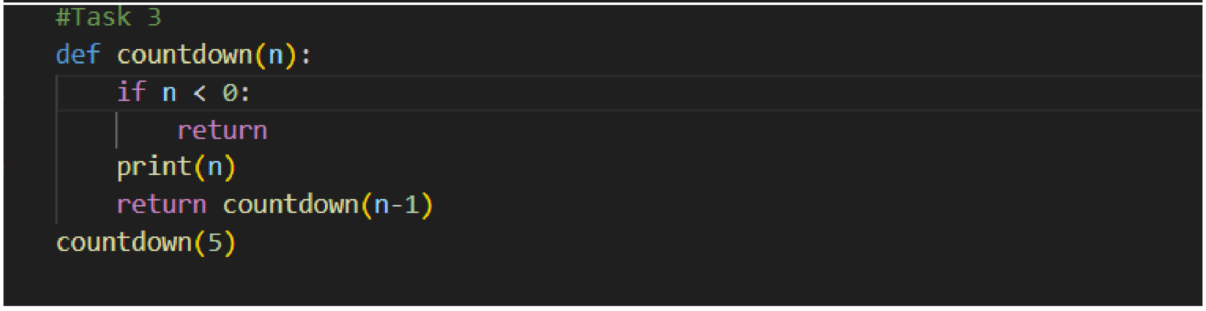
Task 3 (Recursion Error – Missing Base Case)
Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix. # Bug: No base case def countdown(n):
print(n) return
countdown(n-1)
countdown(5)
Expected Output : Correct recursion with stopping condition.

Prompt:fix the below python program using recusion with stopping condition.

Code:-

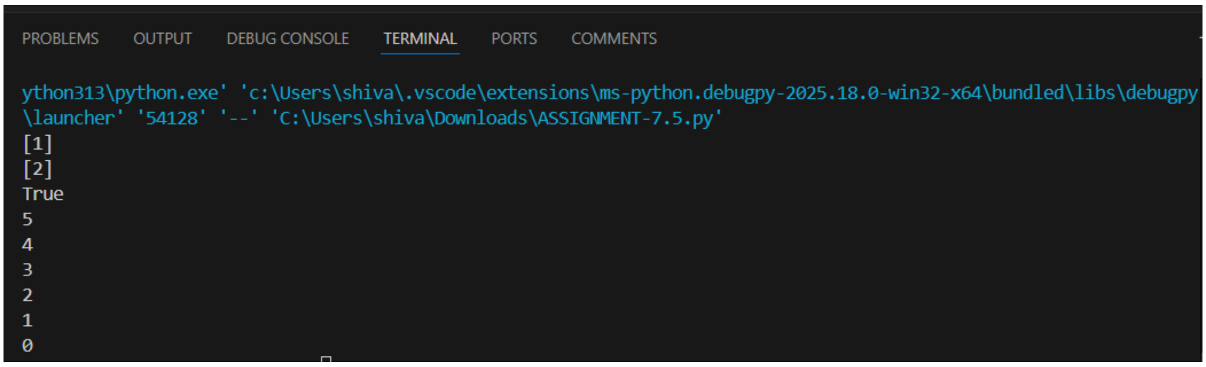


```
def countdown(n):
    if n < 0: n = -977
        retu rn
    print(n)
    return countdown(n-1)
countdown(5)
```



```
#Task 3
def countdown(n):
    if n < 0:
        return
    print(n)
    return countdown(n-1)
countdown(5)
```

Output:-



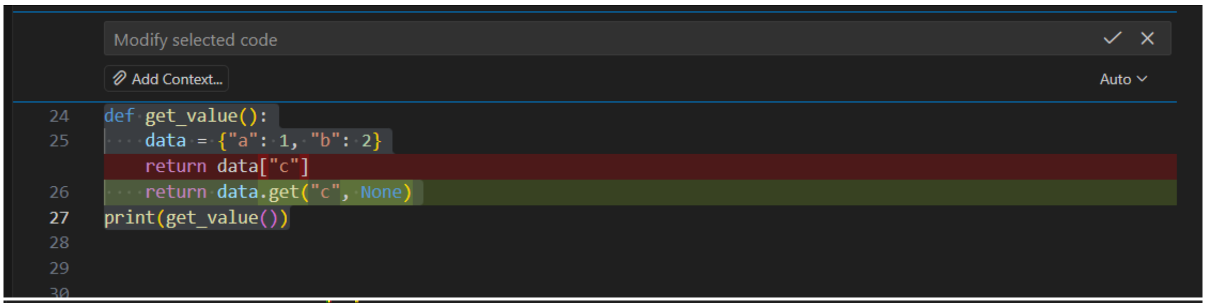
```
ython313\python.exe' 'c:\Users\shiva\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy
\launcher' '54128' '--' 'C:\Users\shiva\Downloads\ASSIGNMENT-7.5.py'
[1]
[2]
True
5
4
3
2
1
0
```

Justification:-
The function countdown is a recursive function that prints the numbers from n to 0. The stopping condition is when n is 0. The function is called with the initial value of n as 5. The function prints the value of n and then calls itself with the value of n minus 1. The function continues to call itself until the stopping condition is met.

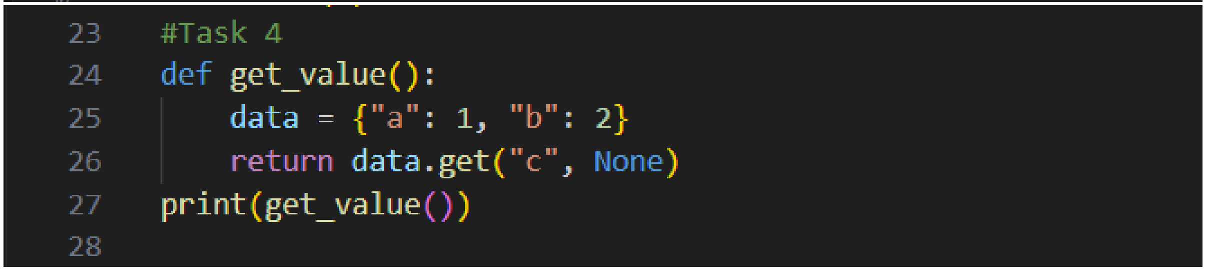
Task 4 (Dictionary Key Error)
Task: Analyze given code where a missing dictionary key causes error. Use AI to fix it.
# Bug: Accessing non-existing key
def get_value():
data = {"a": 1, "b": 2}
return data["c"]
print(get_value())
Expected Output: Corrected with .get() or error handling.

Prompt:fix the below python program using Corrected with .get() or error handling**.**

Code:-

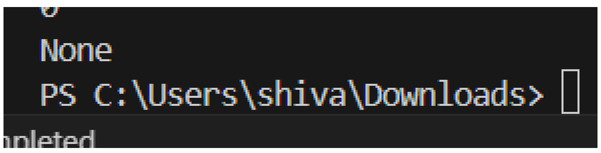


```
def get_value():
    data = {"a": 1, "b": 2}
    return data["c"]
    return data.get("c", None)
print(get_value())
```



```
#Task 4
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", None)
print(get_value())
```

Output:-



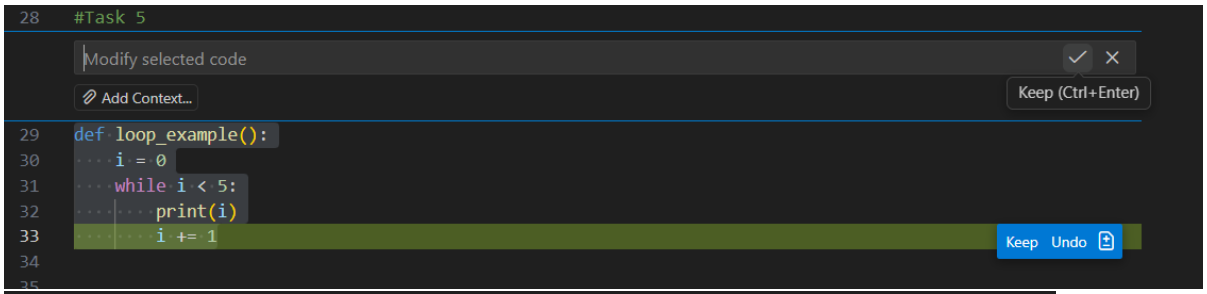
```
None
PS C:\Users\shiva\Downloads>
```

Justification:
The program is corrected by using the .get() method to get the value of the key "c" from the dictionary. If the key is not found, it returns "Key not found".

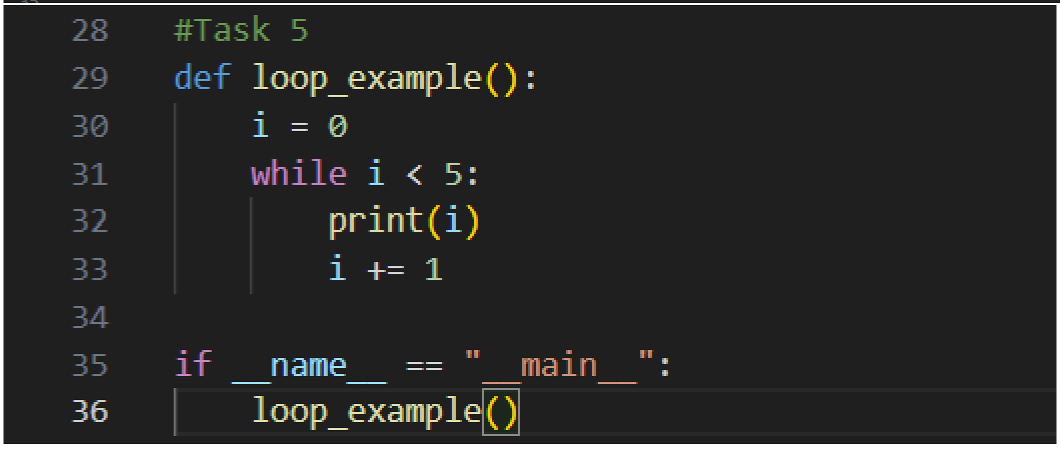
Task 5 (Infinite Loop – Wrong Condition)
Task: Analyze given code where loop never ends. Use AI to detect and fix it. # Bug: Infinite loop def loop_example(): i = 0 while i < 5:
print(i)
Expected Output: Corrected loop increments i.

Prompt:fix the below python program using Corrected loop increments i.

Code:
-

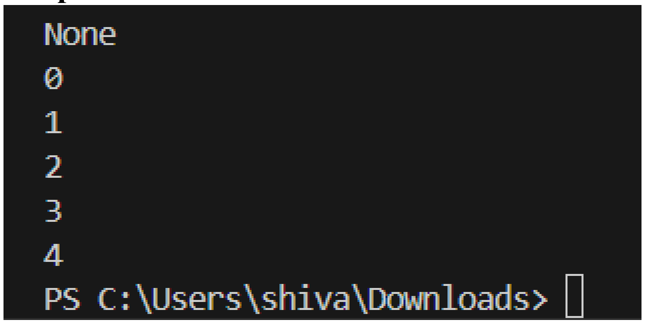


```
#Task 5
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1
```



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

if __name__ == "__main__":
    loop_example()
```

Output:-



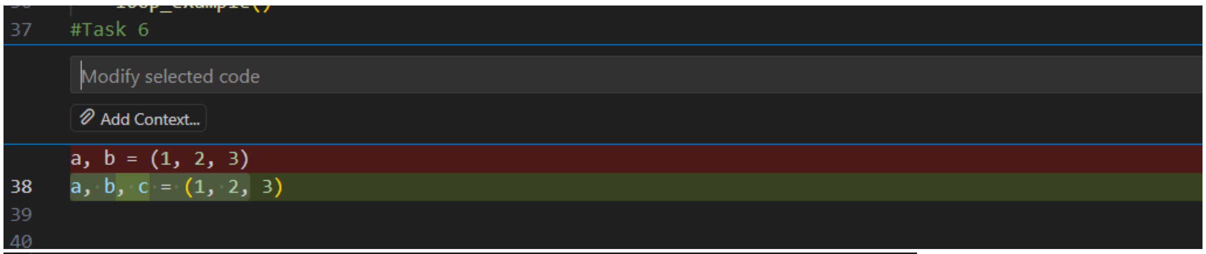
```
None
0
1
2
3
4
PS C:\Users\shiva\Downloads>
```

Justification:
The loop is incrementing i by 1 in each iteration so that the loop will terminate when i is equal to 5 and print the values of i from 0 to 4.

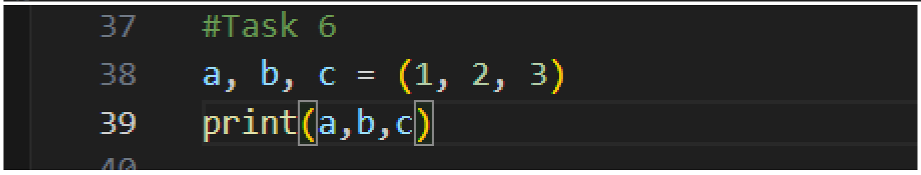
Task 6 (Unpacking Error – Wrong Variables)
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.
# Bug: Wrong unpacking
a, b = (1, 2, 3)
Expected Output: Correct unpacking or using _ for extra values.

Prompt:fix the below python program using Correct unpacking or using _ for extra values.

Code:-

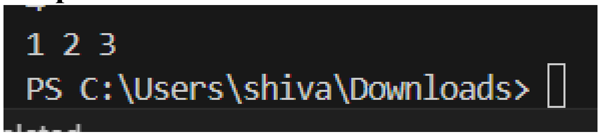


```
#Task 6
a, b = (1, 2, 3)
a, b, c = (1, 2, 3)
```



```
#Task 6
a, b, c = (1, 2, 3)
print(a,b,c)
```

Output:-



```
1 2 3
PS C:\Users\shiva\Downloads>
```

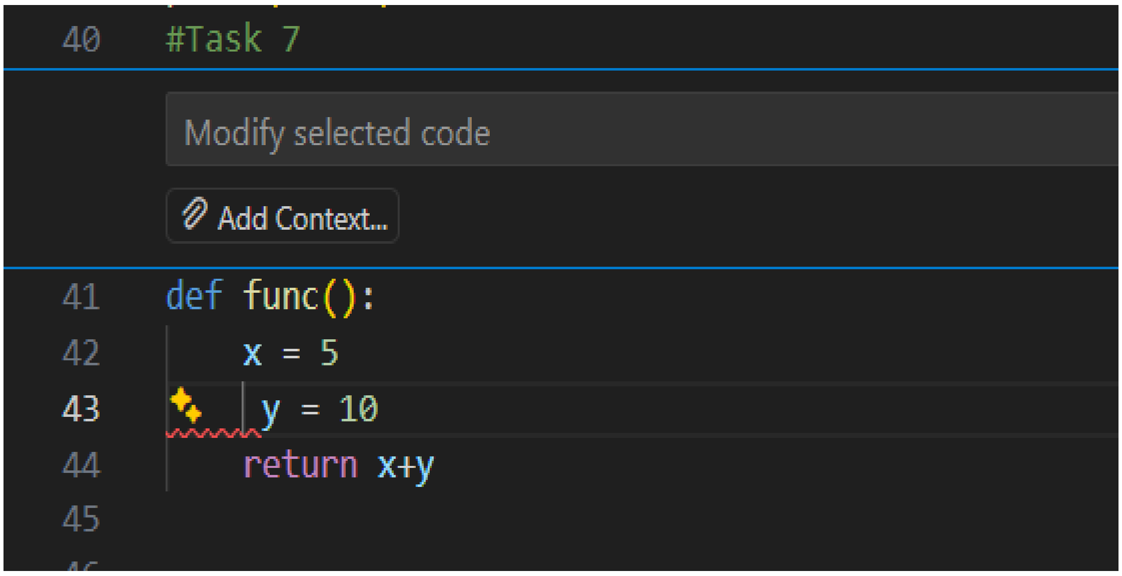
Justification:
The tuple (1, 2, 3) has exactly three values, and there are three variables (a, b, c) on the left-hand side. Since the counts match, tuple unpacking succeeds and each variable receives one value.

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.

SR UNIVERSITY

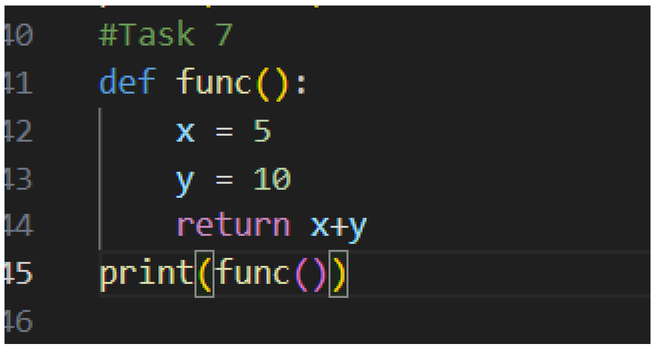
Prompt:fix the below python program using Consistent Mixed Indentation.

Code:



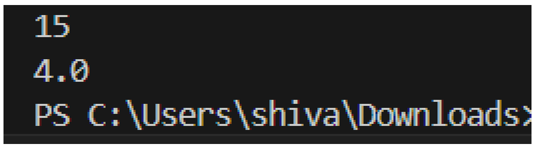
```
#Task 7
```

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



```
#Task 7
def func():
    x = 5
    y = 10
    return x+y
print(func())
```

Output:



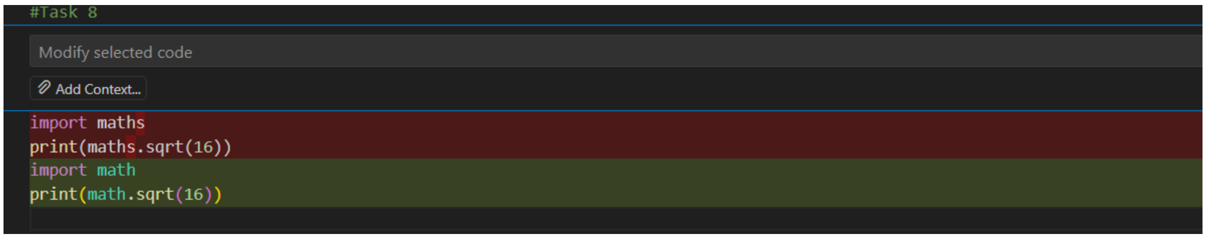
```
15
4.0
PS C:\Users\shiva\Downloads>
```

Justification:
The function is correctly indented and the return statement is correctly indented and the code is correct and the output is correct.

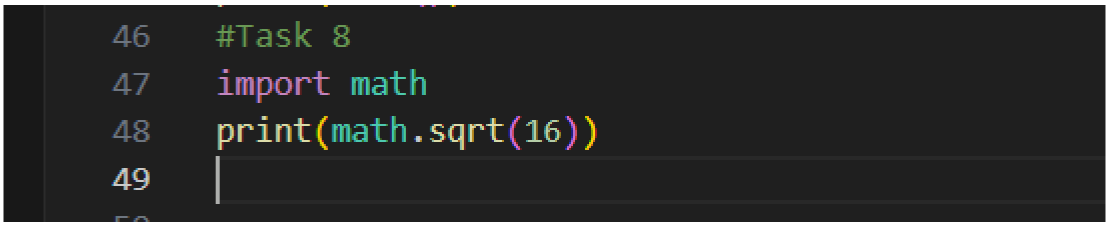
Task 8 (Import Error – Wrong Module Usage)
Task: Analyze given code with incorrect import. Use AI to fix.
# Bug: Wrong import
import maths
print(maths.sqrt(16))
Expected Output: Corrected to import math

Prompt:fix the below python program using Corrected to import math module.

Code:-

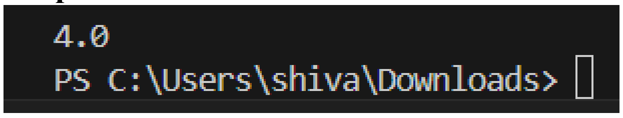


```
#Task 8
import maths
print(maths.sqrt(16))
import math
print(math.sqrt(16))
```



```
#Task 8
import math
print(math.sqrt(16))
```

Output:-



```
4.0
PS C:\Users\shiva\Downloads>
```

Justification:
The program is corrected to import the math module using the as keyword to avoid naming conflicts with the maths library and print the square root of 16.