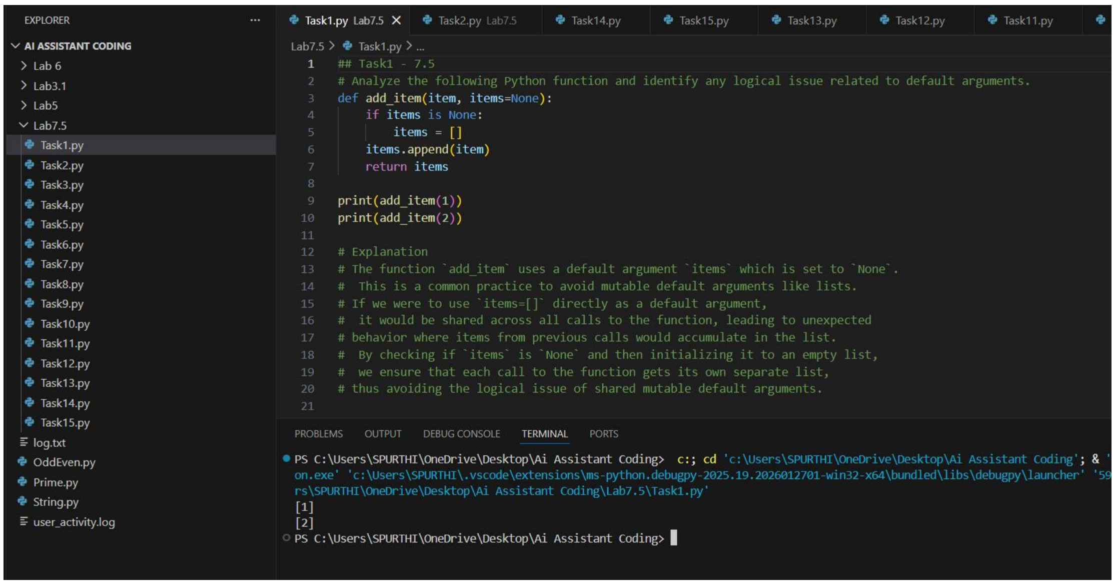
AIAC LAB

LAB 7.5

2303A51802
Billa Spurthi
Batch 29

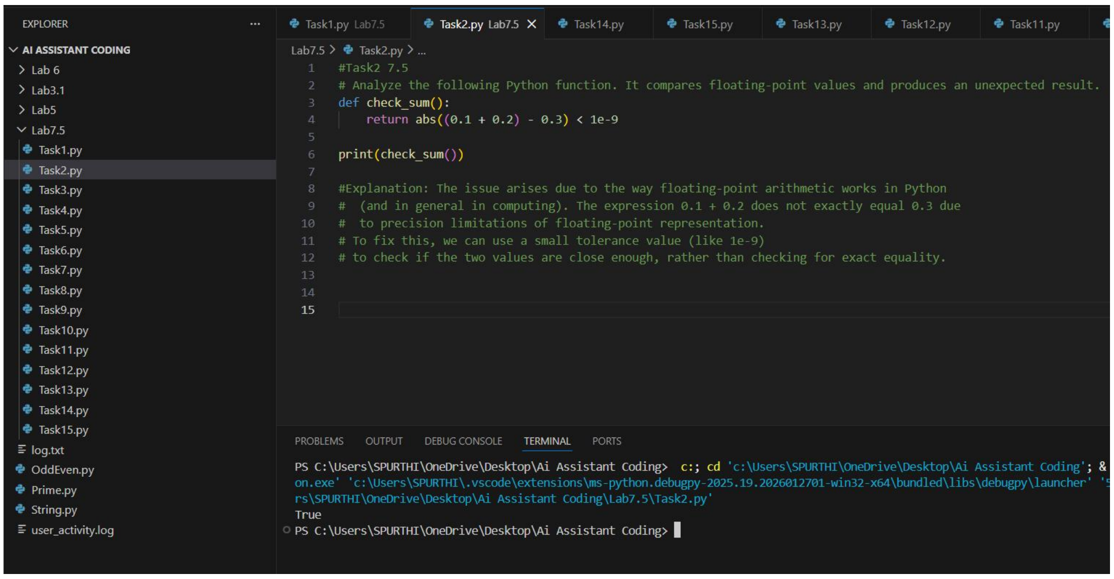
Task1:



```python
## Task1 - 7.5
# Analyze the following Python function and identify any logical issue related to default arguments.
def add_item(item, items=None):
    if items is None:
        items = []
    items.append(item)
    return items

print(add_item(1))
print(add_item(2))

# Explanation
# The function `add_item` uses a default argument `items` which is set to `None`.
#  This is a common practice to avoid mutable default arguments like lists.
# If we were to use `items=[]` directly as a default argument,
#  it would be shared across all calls to the function, leading to unexpected
# behavior where items from previous calls would accumulate in the list.
#  By checking if `items` is `None` and then initializing it to an empty list,
#  we ensure that each call to the function gets its own separate list,
# thus avoiding the logical issue of shared mutable default arguments.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & '
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '59
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task1.py'
[1]
[2]
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

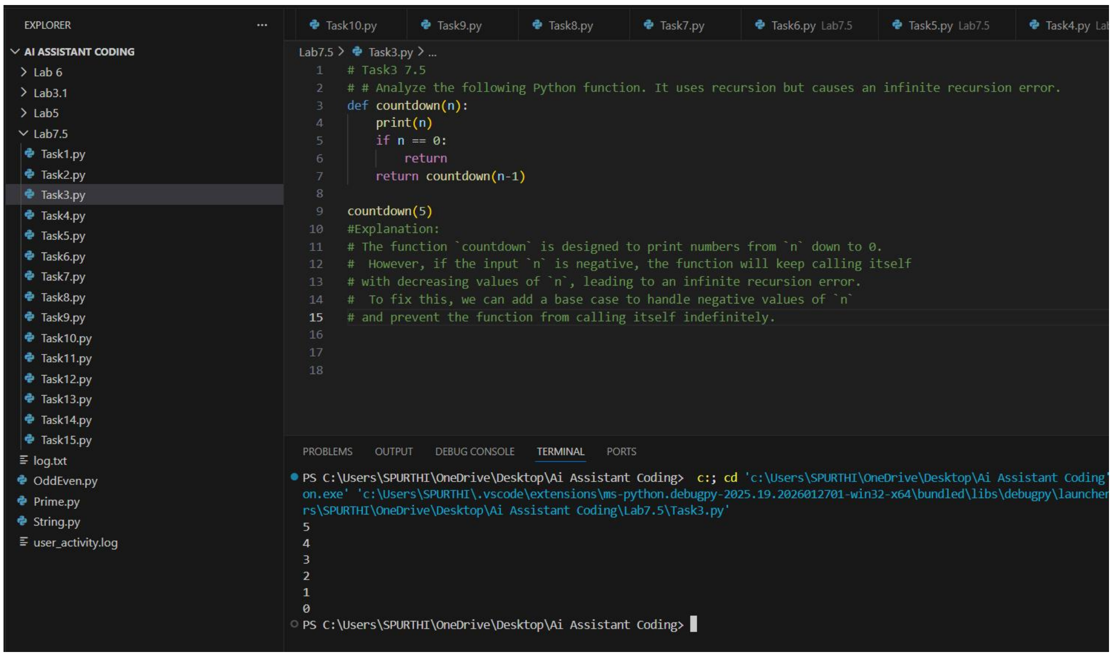
Task2:



```python
#Task2 7.5
# Analyze the following Python function. It compares floating-point values and produces an unexpected result.
def check_sum():
    return abs((0.1 + 0.2) - 0.3) < 1e-9

print(check_sum())

#Explanation: The issue arises due to the way floating-point arithmetic works in Python
#  (and in general in computing). The expression 0.1 + 0.2 does not exactly equal 0.3 due
#  to precision limitations of floating-point representation.
# To fix this, we can use a small tolerance value (like 1e-9)
# to check if the two values are close enough, rather than checking for exact equality.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; &
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task2.py'
True
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

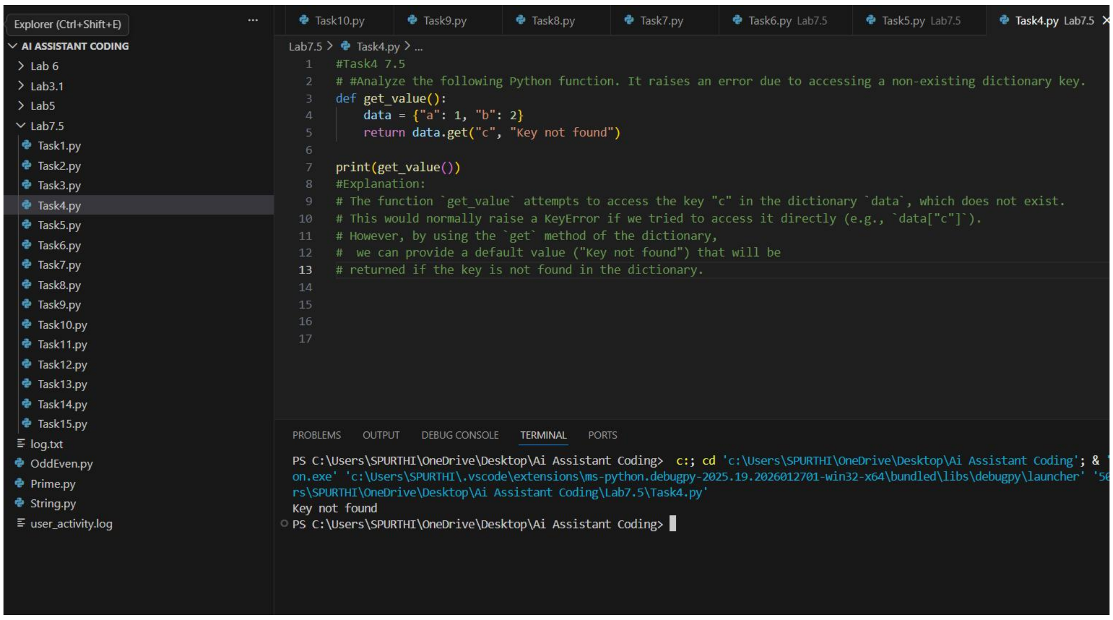
Task3:



```python
# Task3 7.5
# # Analyze the following Python function. It uses recursion but causes an infinite recursion error.
def countdown(n):
    print(n)
    if n == 0:
        return
    return countdown(n-1)

countdown(5)
#Explanation:
# The function `countdown` is designed to print numbers from `n` down to 0.
#  However, if the input `n` is negative, the function will keep calling itself
# with decreasing values of `n`, leading to an infinite recursion error.
#  To fix this, we can add a base case to handle negative values of `n`
# and prevent the function from calling itself indefinitely.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task3.py'
5
4
3
2
1
0
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

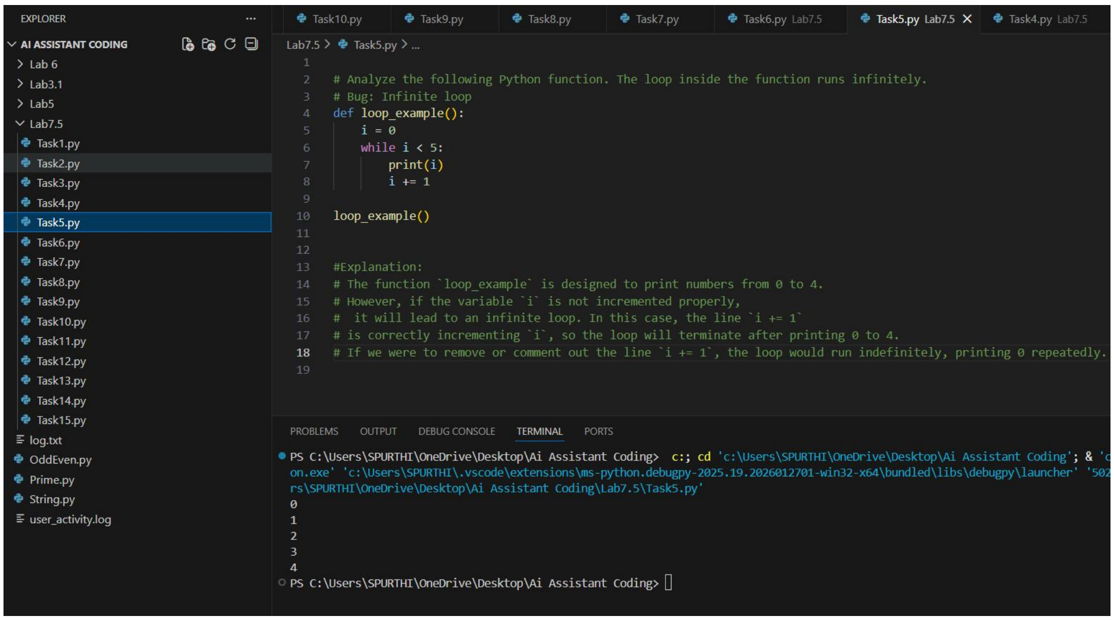
Task4:



```python
#Task4 7.5
# #Analyze the following Python function. It raises an error due to accessing a non-existing dictionary key.
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", "Key not found")

print(get_value())
#Explanation:
# The function `get_value` attempts to access the key "c" in the dictionary `data`, which does not exist.
# This would normally raise a KeyError if we tried to access it directly (e.g., `data["c"]`).
# However, by using the `get` method of the dictionary,
#  we can provide a default value ("Key not found") that will be
# returned if the key is not found in the dictionary.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & '
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '5
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task4.py'
Key not found
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

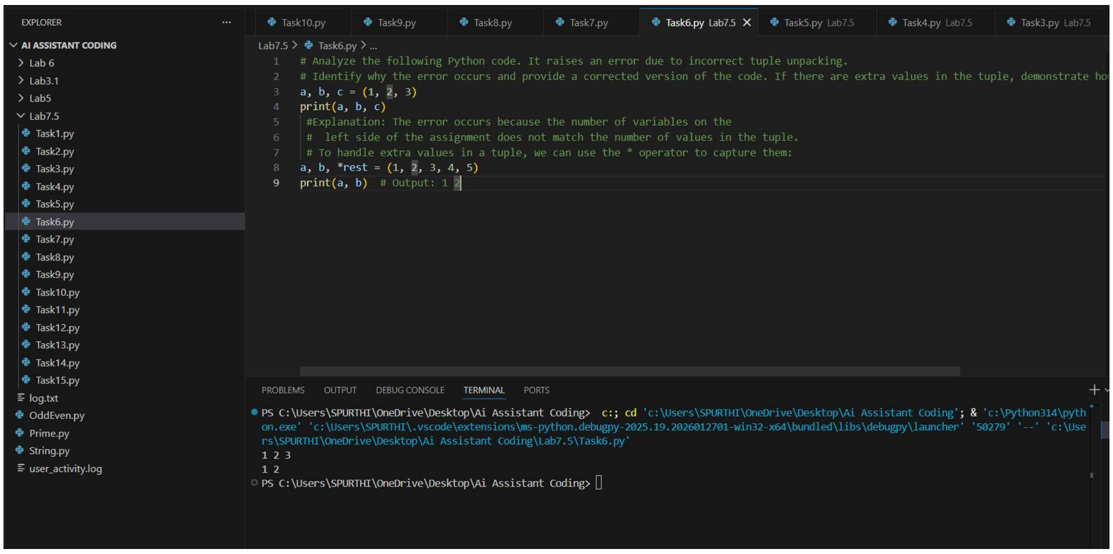
Task5



```python

# Analyze the following Python function. The loop inside the function runs infinitely.
# Bug: Infinite loop
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1

loop_example()


#Explanation:
# The function `loop_example` is designed to print numbers from 0 to 4.
# However, if the variable `i` is not incremented properly,
#  it will lead to an infinite loop. In this case, the line `i += 1`
# is correctly incrementing `i`, so the loop will terminate after printing 0 to 4.
# If we were to remove or comment out the line `i += 1`, the loop would run indefinitely, printing 0 repeatedly.

```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '502
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task5.py'
0
1
2
3
4
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

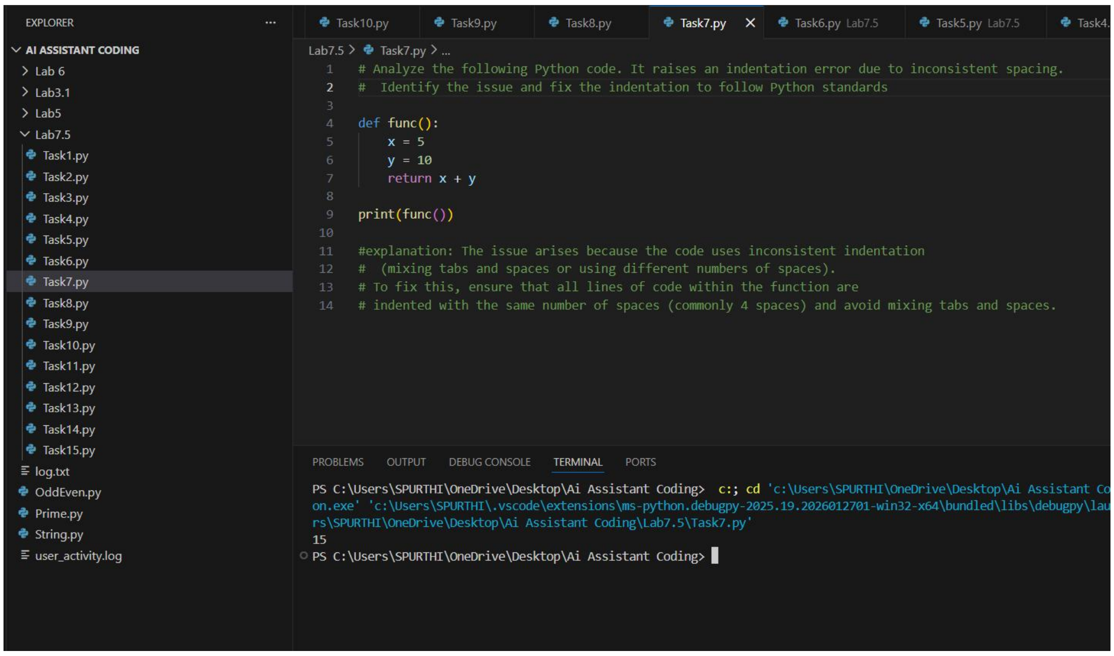
Task6:



```python
# Analyze the following Python code. It raises an error due to incorrect tuple unpacking.
# Identify why the error occurs and provide a corrected version of the code. If there are extra values in the tuple, demonstrate ho
a, b, c = (1, 2, 3)
print(a, b, c)
 #Explanation: The error occurs because the number of variables on the
 #  left side of the assignment does not match the number of values in the tuple.
 # To handle extra values in a tuple, we can use the * operator to capture them:
a, b, *rest = (1, 2, 3, 4, 5)
print(a, b)  # Output: 1 2
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c:\Python314\pyth
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '50279' '--' 'c:\Use
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task6.py'
1 2 3
1 2
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

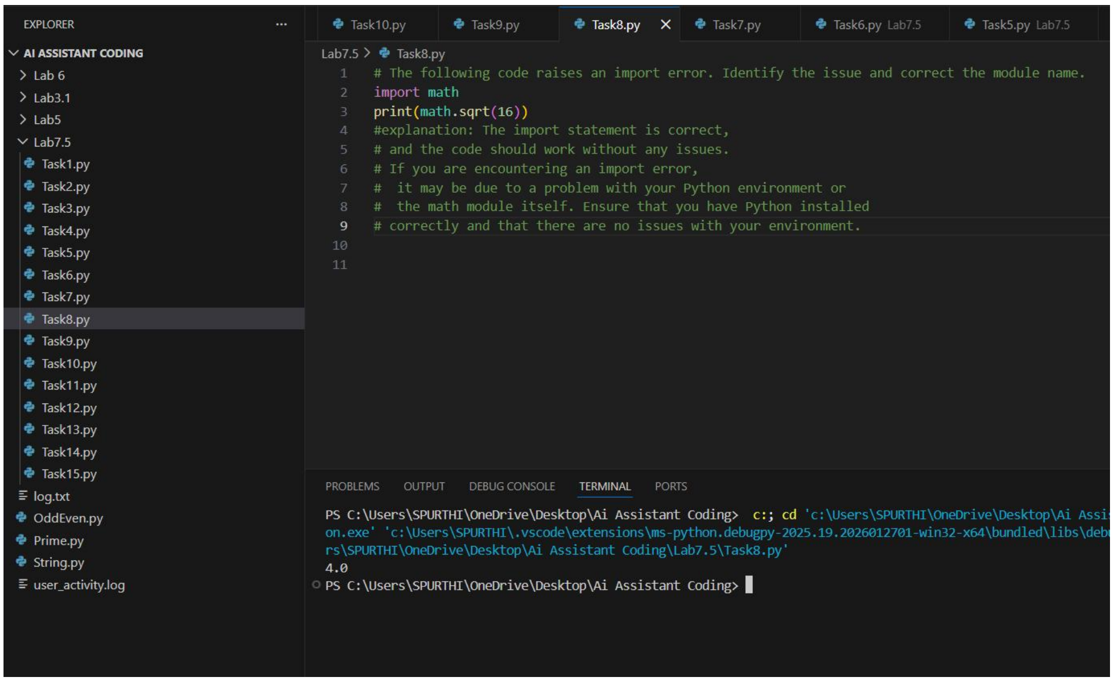
Task7:



```python
# Analyze the following Python code. It raises an indentation error due to inconsistent spacing.
#  Identify the issue and fix the indentation to follow Python standards

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

print(func())

#explanation: The issue arises because the code uses inconsistent indentation
#  (mixing tabs and spaces or using different numbers of spaces).
# To fix this, ensure that all lines of code within the function are
# indented with the same number of spaces (commonly 4 spaces) and avoid mixing tabs and spaces.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Co
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\lau
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task7.py'
15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

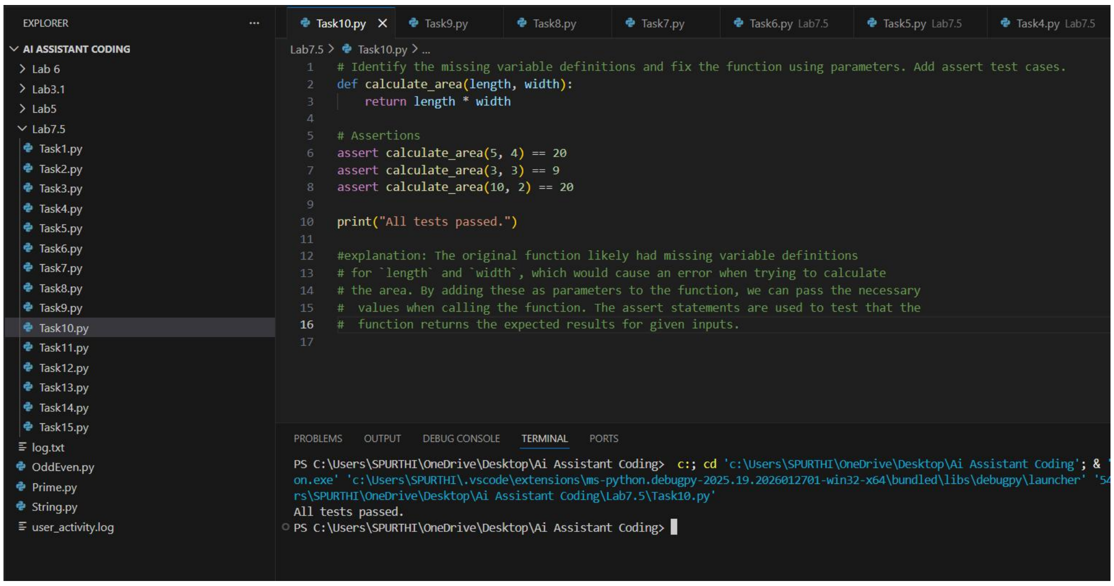
Task8:



```python
# The following code raises an import error. Identify the issue and correct the module name.
import math
print(math.sqrt(16))
#explanation: The import statement is correct,
# and the code should work without any issues.
# If you are encountering an import error,
#  it may be due to a problem with your Python environment or
#  the math module itself. Ensure that you have Python installed
# correctly and that there are no issues with your environment.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assi
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debu
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task8.py'
4.0
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

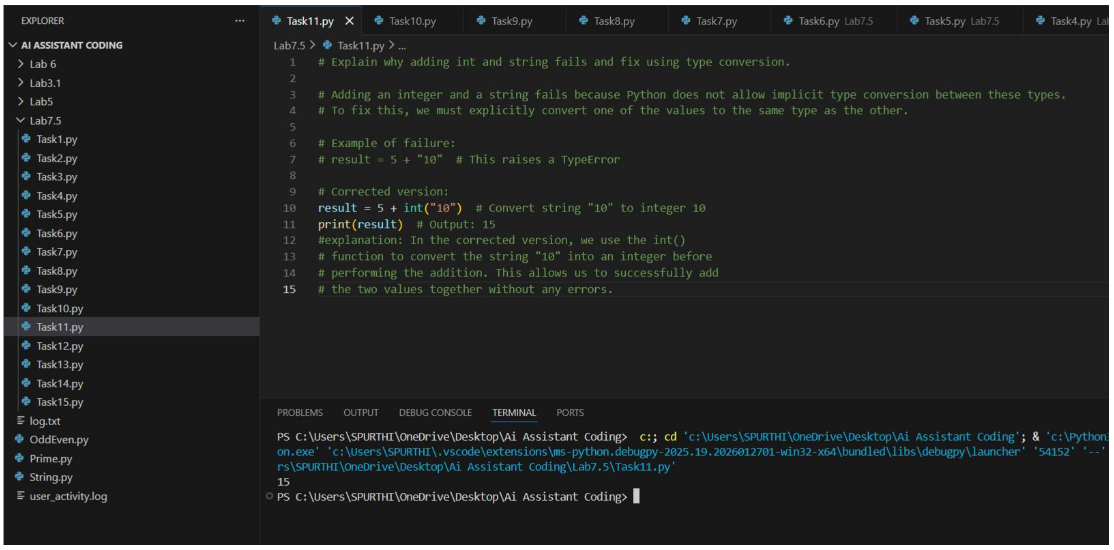
Task10:



```python
# Identify the missing variable definitions and fix the function using parameters. Add assert test cases.
def calculate_area(length, width):
    return length * width

# Assertions
assert calculate_area(5, 4) == 20
assert calculate_area(3, 3) == 9
assert calculate_area(10, 2) == 20

print("All tests passed.")

#explanation: The original function likely had missing variable definitions
# for `length` and `width`, which would cause an error when trying to calculate
# the area. By adding these as parameters to the function, we can pass the necessary
#  values when calling the function. The assert statements are used to test that the
#  function returns the expected results for given inputs.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & '
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '54
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task10.py'
All tests passed.
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

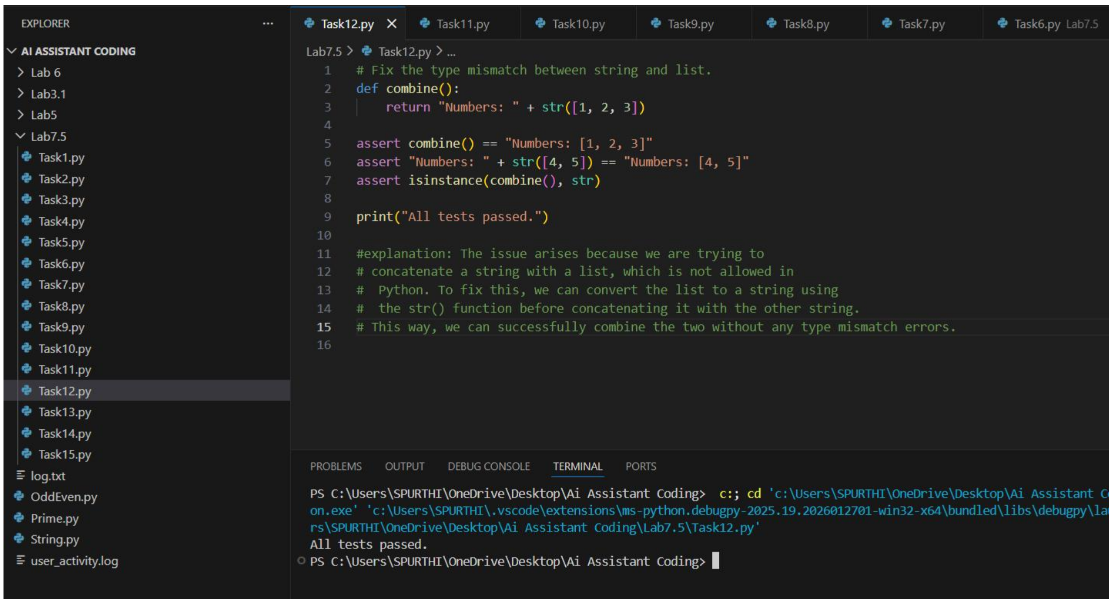
Task11



```python
# Explain why adding int and string fails and fix using type conversion.

# Adding an integer and a string fails because Python does not allow implicit type conversion between these types.
# To fix this, we must explicitly convert one of the values to the same type as the other.

# Example of failure:
# result = 5 + "10"  # This raises a TypeError

# Corrected version:
result = 5 + int("10")  # Convert string "10" to integer 10
print(result)  # Output: 15
#explanation: In the corrected version, we use the int()
# function to convert the string "10" into an integer before
# performing the addition. This allows us to successfully add
# the two values together without any errors.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c:\Python
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '54152' '--'
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task11.py'
15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

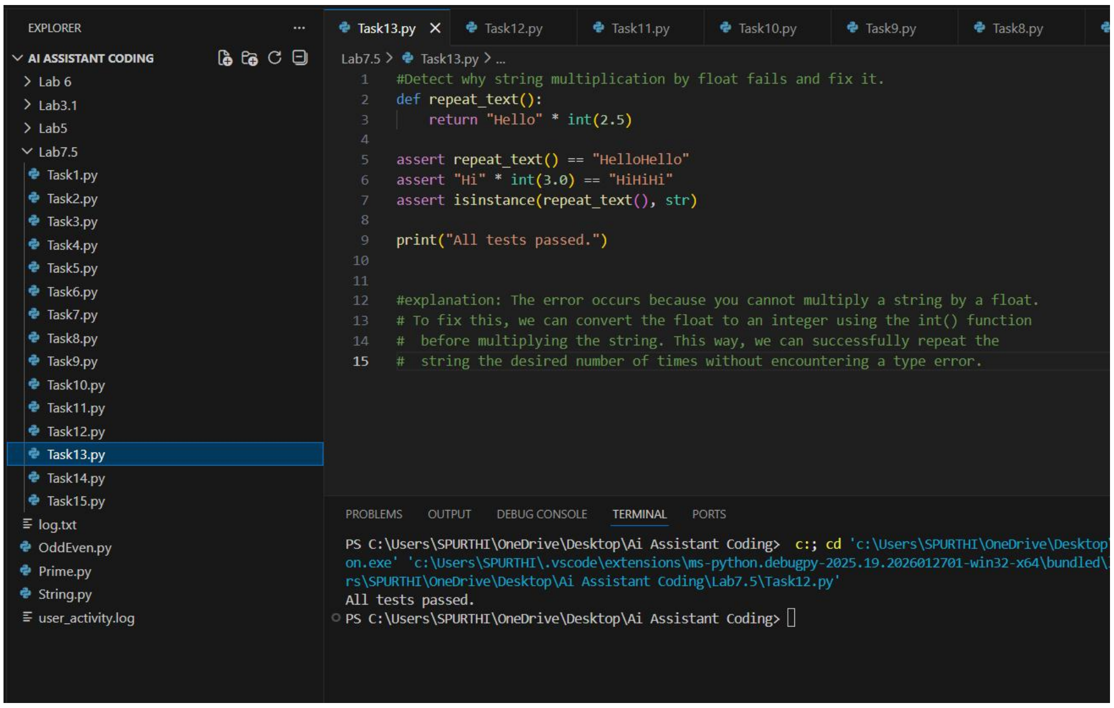
Task12:



```python
# Fix the type mismatch between string and list.
def combine():
    return "Numbers: " + str([1, 2, 3])

assert combine() == "Numbers: [1, 2, 3]"
assert "Numbers: " + str([4, 5]) == "Numbers: [4, 5]"
assert isinstance(combine(), str)

print("All tests passed.")

#explanation: The issue arises because we are trying to
# concatenate a string with a list, which is not allowed in
#  Python. To fix this, we can convert the list to a string using
#  the str() function before concatenating it with the other string.
# This way, we can successfully combine the two without any type mismatch errors.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant C
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\la
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task12.py'
All tests passed.
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

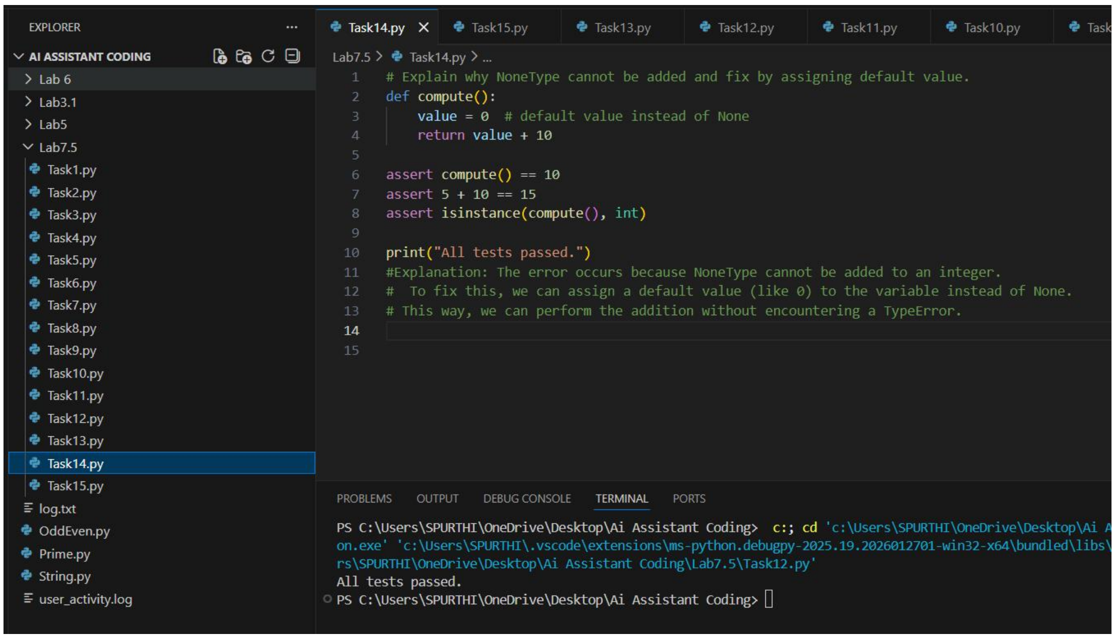
Task13



```python
#Detect why string multiplication by float fails and fix it.
def repeat_text():
    return "Hello" * int(2.5)

assert repeat_text() == "HelloHello"
assert "Hi" * int(3.0) == "HiHiHi"
assert isinstance(repeat_text(), str)

print("All tests passed.")


#explanation: The error occurs because you cannot multiply a string by a float.
# To fix this, we can convert the float to an integer using the int() function
#  before multiplying the string. This way, we can successfully repeat the
#  string the desired number of times without encountering a type error.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task12.py'
All tests passed.
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

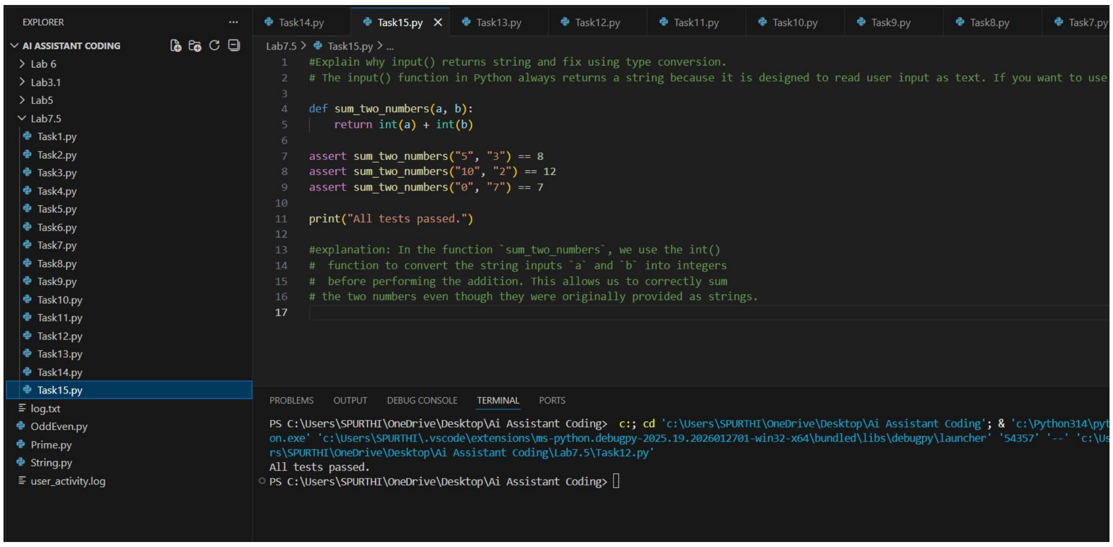
Task14:



```python
# Explain why NoneType cannot be added and fix by assigning default value.
def compute():
    value = 0  # default value instead of None
    return value + 10

assert compute() == 10
assert 5 + 10 == 15
assert isinstance(compute(), int)

print("All tests passed.")
#Explanation: The error occurs because NoneType cannot be added to an integer.
#  To fix this, we can assign a default value (like 0) to the variable instead of None.
# This way, we can perform the addition without encountering a TypeError.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai A
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task12.py'
All tests passed.
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

Task15



```python
#Explain why input() returns string and fix using type conversion.
# The input() function in Python always returns a string because it is designed to read user input as text. If you want to use

def sum_two_numbers(a, b):
    return int(a) + int(b)

assert sum_two_numbers("5", "3") == 8
assert sum_two_numbers("10", "2") == 12
assert sum_two_numbers("0", "7") == 7

print("All tests passed.")

#explanation: In the function `sum_two_numbers`, we use the int()
#  function to convert the string inputs `a` and `b` into integers
#  before performing the addition. This allows us to correctly sum
# the two numbers even though they were originally provided as strings.
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>  c:; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c:\Python314\pyt
on.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '54357' '--' 'c:\Us
rs\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\Lab7.5\Task12.py'
All tests passed.
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> 
```