Assignment-13.3

Name: Y. Poojitha

HallTicket No:2303A51499

Batch: 08

Lab 13: Code Refactoring Using AI Assistance Improving Legacy Code for Readability, Maintainability, and Performance

Task 1: Refactoring – Removing Code Duplication

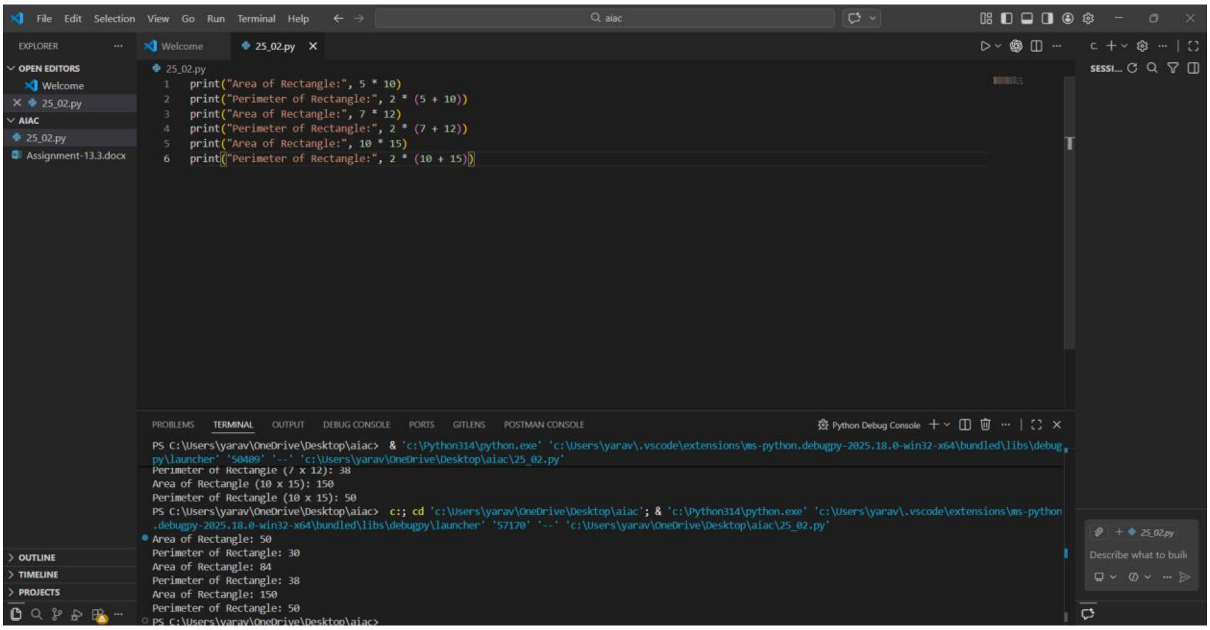
Objective

To eliminate repeated logic by extracting reusable functions.

Prompt Used

“Refactor the following Python code to remove duplication and create reusable functions with proper docstrings.”

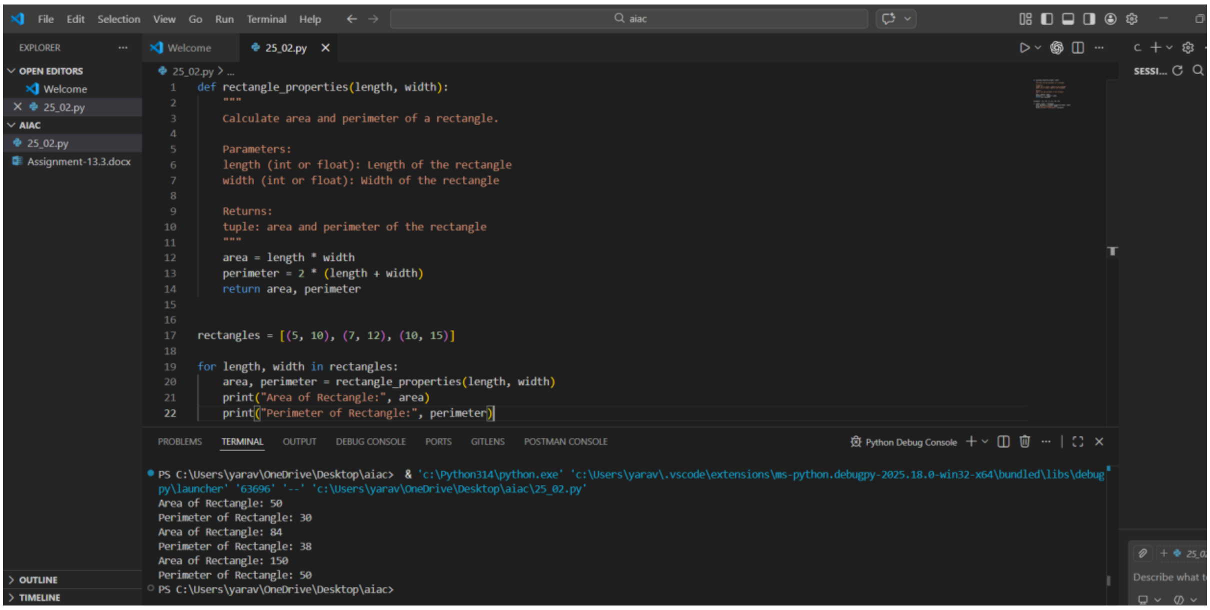
Legacy Code



```python
print("Area of Rectangle:", 5 * 10)
print("Perimeter of Rectangle:", 2 * (5 + 10))
print("Area of Rectangle:", 7 * 12)
print("Perimeter of Rectangle:", 2 * (7 + 12))
print("Area of Rectangle:", 10 * 15)
print("Perimeter of Rectangle:", 2 * (10 + 15))
```

```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debug
py\launcher' '50409' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
Perimeter of Rectangle (7 x 12): 38
Area of Rectangle (10 x 15): 150
Perimeter of Rectangle (10 x 15): 50
PS C:\Users\yarav\OneDrive\Desktop\aiac>  c:; cd 'c:\Users\yarav\OneDrive\Desktop\aiac'; & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python
.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57170' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
Area of Rectangle: 50
Perimeter of Rectangle: 30
Area of Rectangle: 84
Perimeter of Rectangle: 38
Area of Rectangle: 150
Perimeter of Rectangle: 50
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Refactored Code



```python
def rectangle_properties(length, width):
    """
    Calculate area and perimeter of a rectangle.

    Parameters:
    length (int or float): Length of the rectangle
    width (int or float): Width of the rectangle

    Returns:
    tuple: area and perimeter of the rectangle
    """
    area = length * width
    perimeter = 2 * (length + width)
    return area, perimeter


rectangles = [(5, 10), (7, 12), (10, 15)]

for length, width in rectangles:
    area, perimeter = rectangle_properties(length, width)
    print("Area of Rectangle:", area)
    print("Perimeter of Rectangle:", perimeter)
```

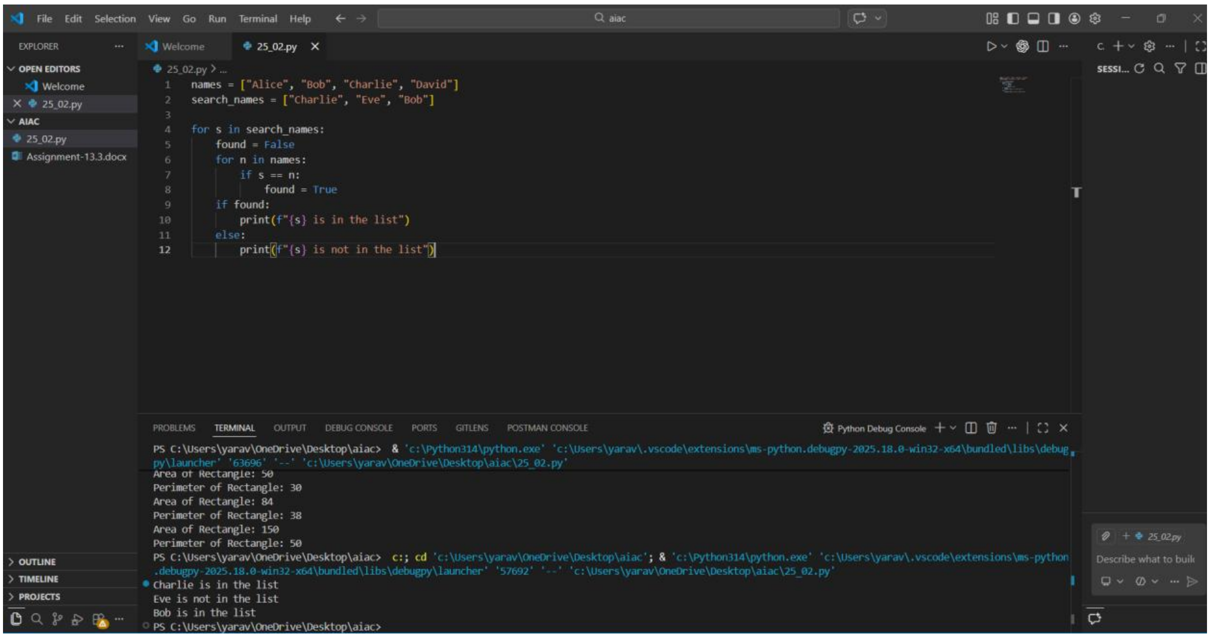
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63696' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
Area of Rectangle: 50
Perimeter of Rectangle: 30
Area of Rectangle: 84
Perimeter of Rectangle: 38
Area of Rectangle: 150
Perimeter of Rectangle: 50
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Task 2: Refactoring – Optimizing Loops and Conditionals

Objective

Improve performance by replacing nested loops.

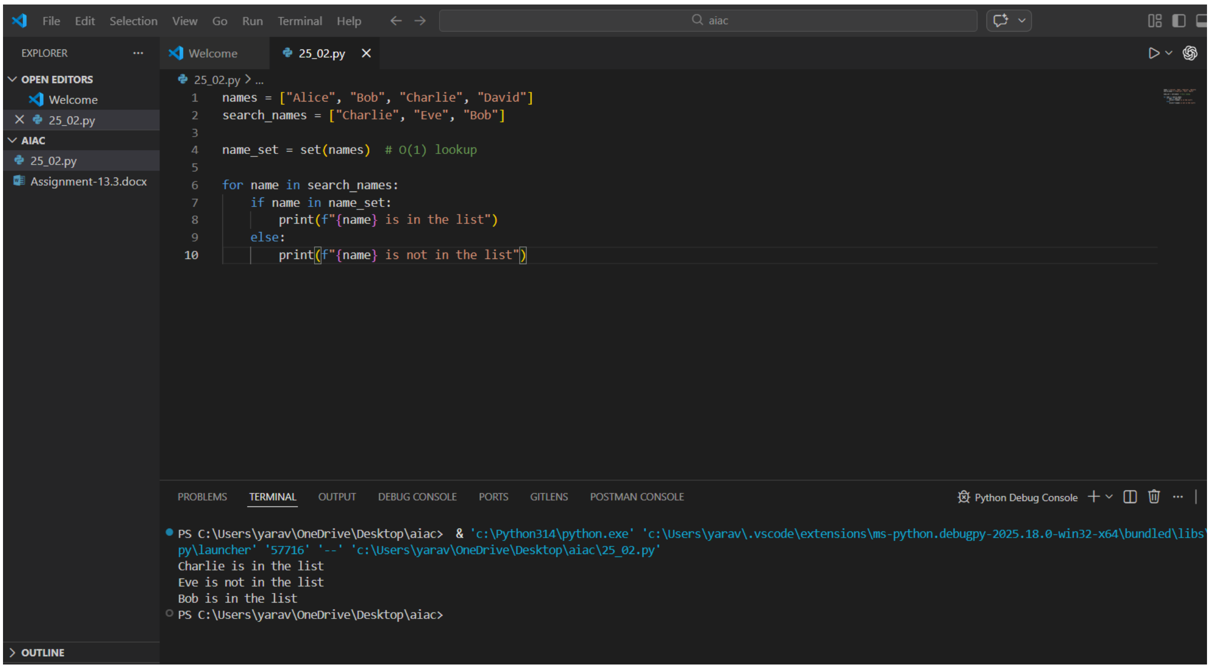
Legacy Code



```python
names = ["Alice", "Bob", "Charlie", "David"]
search_names = ["Charlie", "Eve", "Bob"]

for s in search_names:
    found = False
    for n in names:
        if s == n:
            found = True
    if found:
        print(f"{s} is in the list")
    else:
        print(f"{s} is not in the list")
```

```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63696' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
Area of Rectangle: 50
Perimeter of Rectangle: 30
Area of Rectangle: 84
Perimeter of Rectangle: 38
Area of Rectangle: 150
Perimeter of Rectangle: 50
PS C:\Users\yarav\OneDrive\Desktop\aiac>  c:; cd 'c:\Users\yarav\OneDrive\Desktop\aiac'; & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57692' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
Charlie is in the list
Eve is not in the list
Bob is in the list
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Refactored Code (Using Set Lookup)



```
names = ["Alice", "Bob", "Charlie", "David"]
search_names = ["Charlie", "Eve", "Bob"]

name_set = set(names)  # O(1) lookup

for name in search_names:
    if name in name_set:
        print(f"{name} is in the list")
    else:
        print(f"{name} is not in the list")
```

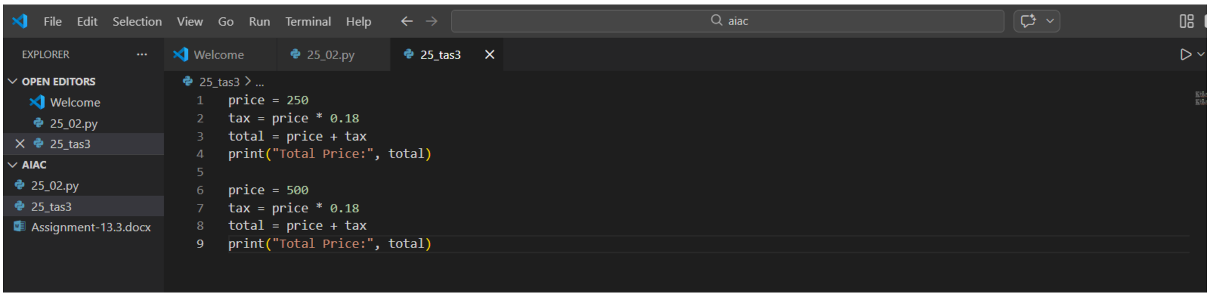
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\
py\launcher' '57716' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
Charlie is in the list
Eve is not in the list
Bob is in the list
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Task 3: Extracting Reusable Functions

Objective

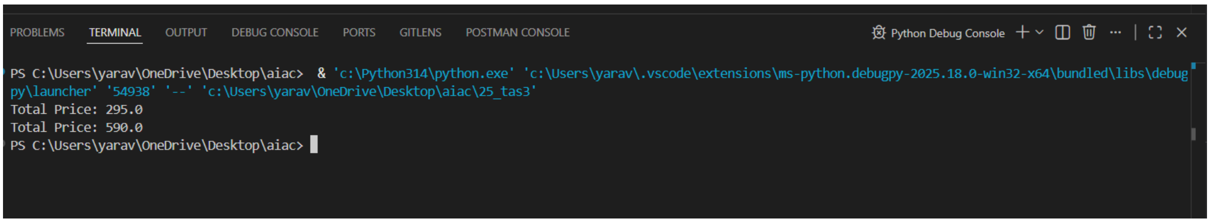
Modularize price and tax calculations.

Legacy Code



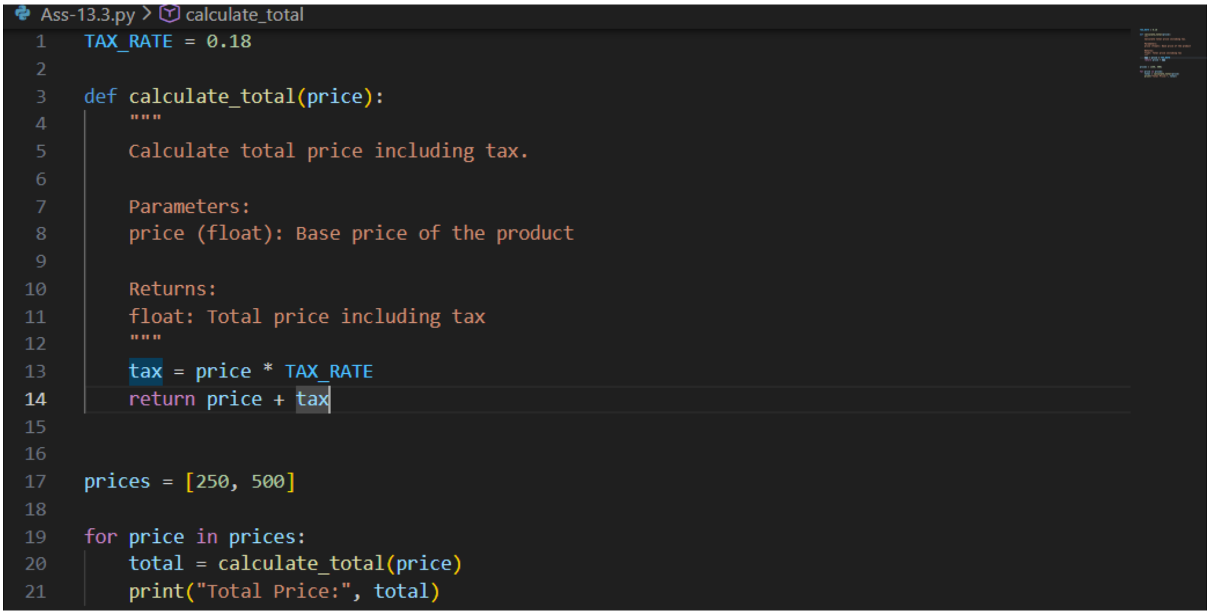
```
price = 250
tax = price * 0.18
total = price + tax
print("Total Price:", total)

price = 500
tax = price * 0.18
total = price + tax
print("Total Price:", total)
```

Output:



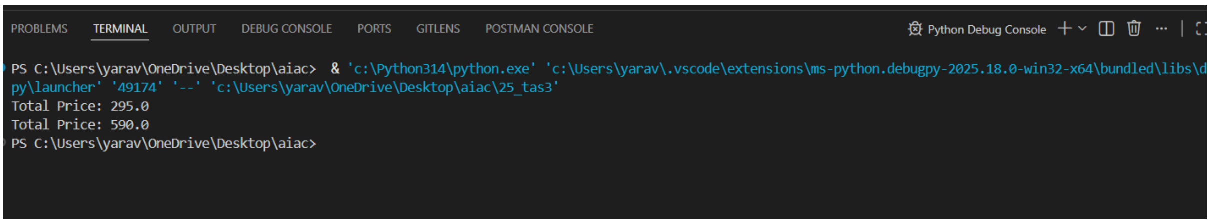
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debug
py\launcher' '54938' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_tas3'
Total Price: 295.0
Total Price: 590.0
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Refactored Code



```python
TAX_RATE = 0.18

def calculate_total(price):
    """
    Calculate total price including tax.

    Parameters:
    price (float): Base price of the product

    Returns:
    float: Total price including tax
    """
    tax = price * TAX_RATE
    return price + tax


prices = [250, 500]

for price in prices:
    total = calculate_total(price)
    print("Total Price:", total)
```

Output:



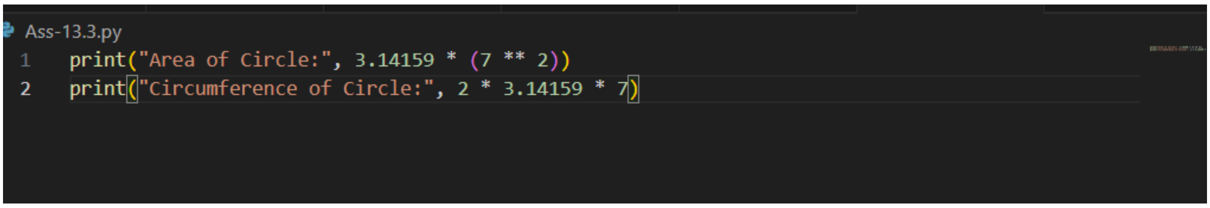
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\d
py\launcher' '49174' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_tas3'
Total Price: 295.0
Total Price: 590.0
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Task 4: Replacing Hardcoded Values with Constants

Objective

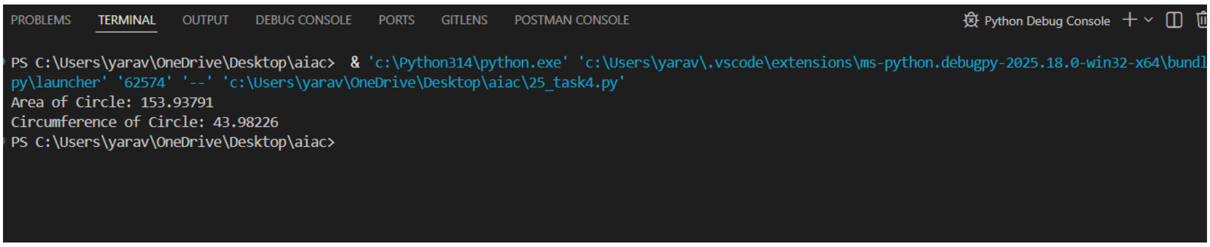
Replace magic numbers with named constants.

Legacy Code



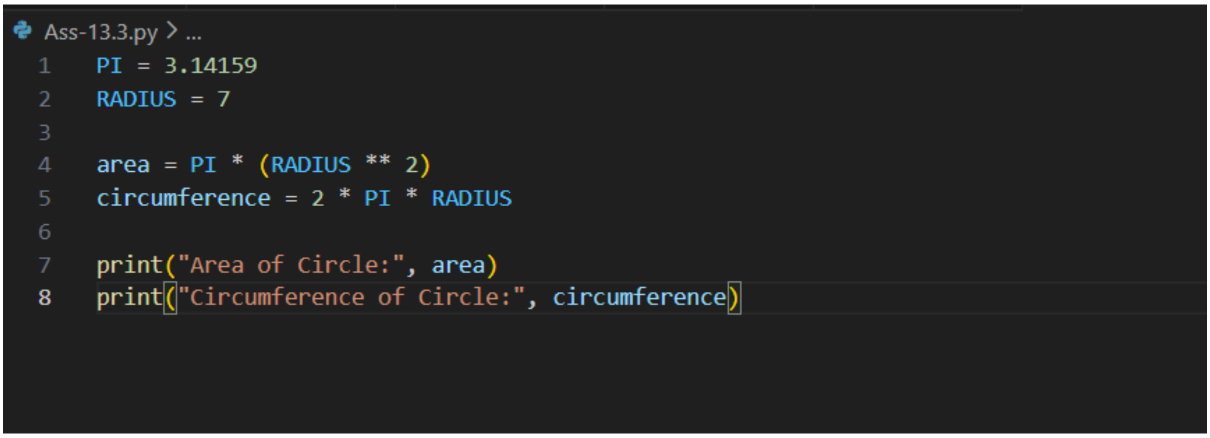
```python
print("Area of Circle:", 3.14159 * (7 ** 2))
print("Circumference of Circle:", 2 * 3.14159 * 7)
```

Output:



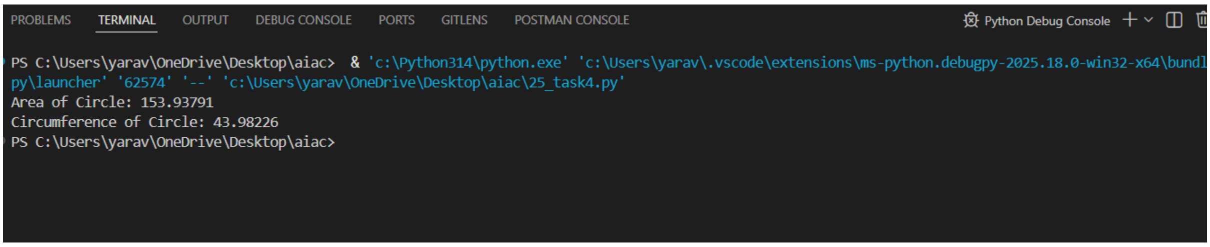
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundl
py\launcher' '62574' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_task4.py'
Area of Circle: 153.93791
Circumference of Circle: 43.98226
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Refactored Code



```python
PI = 3.14159
RADIUS = 7

area = PI * (RADIUS ** 2)
circumference = 2 * PI * RADIUS

print("Area of Circle:", area)
print("Circumference of Circle:", circumference)
```

Output:



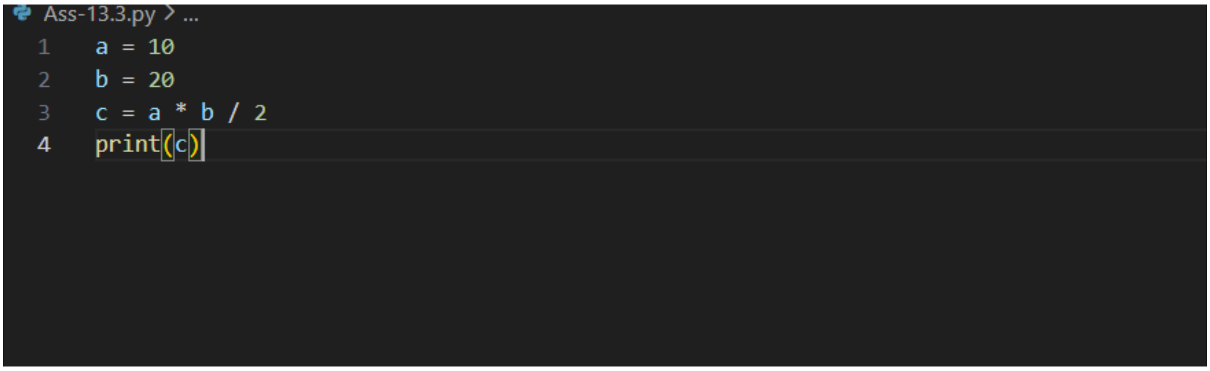
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundl
py\launcher' '62574' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_task4.py'
Area of Circle: 153.93791
Circumference of Circle: 43.98226
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Task 5: Improving Variable Naming and Readability

Objective

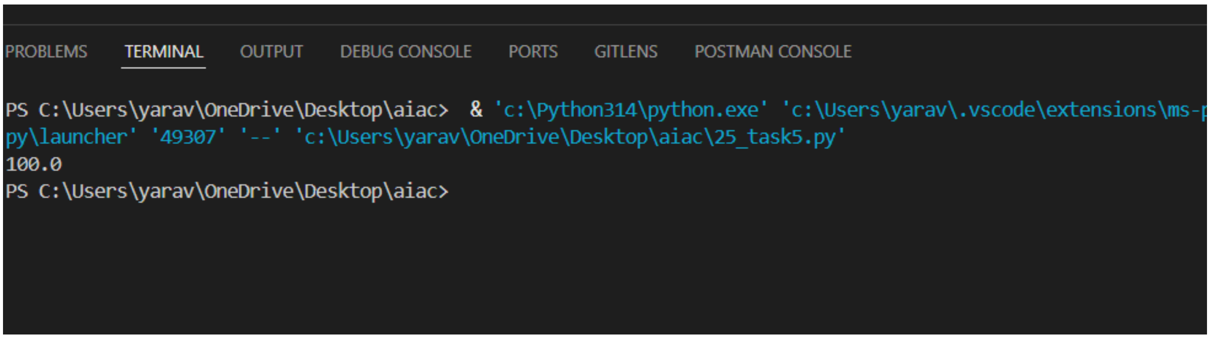
Use descriptive variable names.

Legacy Code



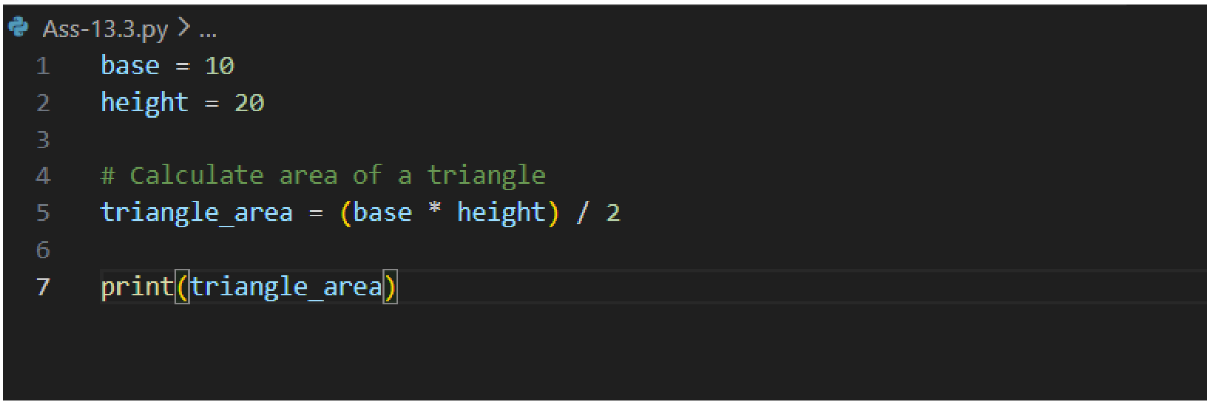
```python
a = 10
b = 20
c = a * b / 2
print(c)
```

Output:



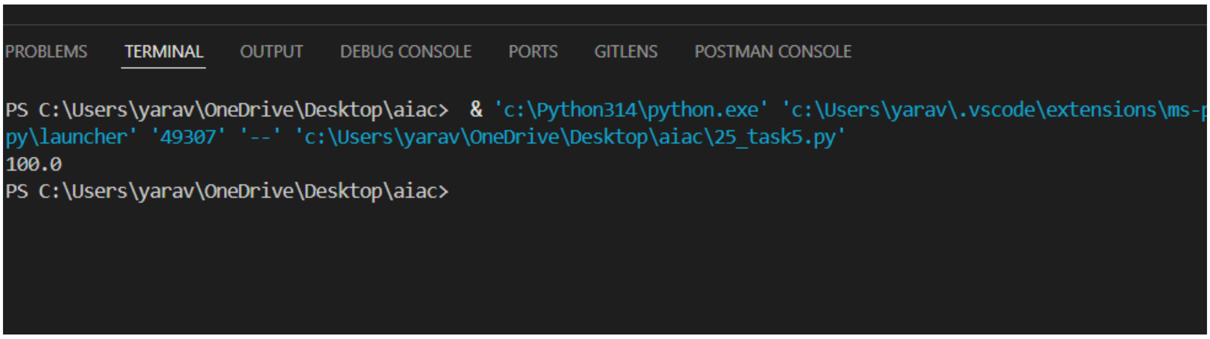
```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-p
py\launcher' '49307' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_task5.py'
100.0
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Refactored Code



```python
base = 10
height = 20

# Calculate area of a triangle
triangle_area = (base * height) / 2

print(triangle_area)
```

Output:



```
PS C:\Users\yarav\OneDrive\Desktop\aiac>  & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-p
py\launcher' '49307' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_task5.py'
100.0
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Conclusion

In this lab, AI-assisted refactoring significantly improved the quality of legacy Python code. Code duplication was removed, inefficient loops were optimized, reusable functions were created, magic numbers were replaced with constants, and meaningful variable names were introduced. The refactored programs are now more readable, maintainable, scalable, and performance-efficient while preserving original functionality.