

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

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows files: Welcome, 25_02.py, and Assignment-13.3.docx.
- Code Editor:** Displays the following Python code:

```
1 print("Area of Rectangle:", 5 * 10)
2 print("Perimeter of Rectangle:", 2 * (5 + 10))
3 print("Area of Rectangle:", 7 * 12)
4 print("Perimeter of Rectangle:", 2 * (7 + 12))
5 print("Area of Rectangle:", 10 * 15)
6 print("Perimeter of Rectangle:", 2 * (10 + 15))
```
- Terminal:** Shows the command run in the terminal:

```
PS C:\Users\yarav\OneDrive\Desktop\aiac> & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.python\2025.18.0-win32-x64\debug\launcher' '50409' -> 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
```

Output:

```
Area of Rectangle (5 x 10): 50
Perimeter of Rectangle (5 x 10): 30
Area of Rectangle (7 x 12): 84
Perimeter of Rectangle (7 x 12): 38
Area of Rectangle (10 x 15): 150
Perimeter of Rectangle (10 x 15): 50
```
- Python Debug Console:** Shows the command run in the terminal:

```
PS C:\Users\yarav\OneDrive\Desktop\aiac> & cd 'c:\Users\yarav\OneDrive\Desktop\aiac' & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.python\2025.18.0-win32-x64\debug\launcher' '57178' -> 'c:\Users\yarav\OneDrive\Desktop\aiac\25_02.py'
```

Output:

```
Area of Rectangle: 50
Perimeter of Rectangle: 30
Area of Rectangle: 84
Perimeter of Rectangle: 38
Area of Rectangle: 150
Perimeter of Rectangle: 50
```
- Bottom Status Bar:** Shows the file path: C:\Users\yarav\OneDrive\Desktop\aiac>

Refactored Code

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows open files: `25_02.py`, `Welcome`, and `Assignment-13.3.docx`.
- Terminal:** Displays command-line output for running the script.
- Code Editor:** Displays the `25_02.py` file content, which defines a function `rectangle_properties` to calculate area and perimeter of a rectangle.
- Output:** Shows the execution results of the script, including the area and perimeter for three rectangles: (5, 10), (7, 12), and (10, 15).

```
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.python.debug-2025.18.0-win32-x64\bundled\libs\debug\py\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: 58
PS C:\Users\yarav\OneDrive\Desktop\aiac
```

Task 2: Refactoring – Optimizing Loops and Conditionals

Objective

Improve performance by replacing nested loops.

Legacy Code

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar includes standard window controls and the VS Code logo.

Explorer View: Shows the file tree with the following structure:

- OPEN EDITORS:
 - Welcome
 - 25_02.py
- AIAC
 - 25_02.py
 - Assignment-13.3.docx

Terminal View: Displays the output of running the Python script 25_02.py. The terminal tab is active at the bottom of the interface.

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

Status Bar: Shows the current file path as 'C:\Users\yarav\OneDrive\Desktop\aiac' and the status 'aiac'.

Refactored Code (Using Set Lookup)

The screenshot shows the VS Code interface with the following details:

- Editor:** The main editor window displays the file `25_02.py` containing Python code that uses a set to look up names.
- Terminal:** The terminal window shows the output of running the script, which prints the names Charlie, Bob, and Alice, indicating they are in the list.
- File Explorer:** The sidebar shows files like `Welcome`, `25_02.py`, and `Assignment-13.3.docx`.
- Bottom Bar:** The bottom navigation bar includes tabs for PROBLEMS, TERMINAL, OUTPUT, DEBUG CONSOLE, PORTS, GITLENS, and POSTMAN CONSOLE.

Task 3: Extracting Reusable Functions

Objective

Modularize price and tax calculations.

Legacy Code

The screenshot shows the VS Code interface with the following details:

- Editor:** The main editor window displays the file `25_tas3.py` containing modularized code for price and tax calculations.
- File Explorer:** The sidebar shows files like `Welcome`, `25_02.py`, `25_tas3`, and `Assignment-13.3.docx`.
- Bottom Bar:** The bottom navigation bar includes tabs for PROBLEMS, TERMINAL, OUTPUT, DEBUG CONSOLE, PORTS, GITLENS, and POSTMAN CONSOLE.

Output:

The screenshot shows the VS Code interface with the following details:

- Terminal:** The terminal window shows the output of running the script, which prints two total prices: 295.0 and 590.0.
- Bottom Bar:** The bottom navigation bar includes tabs for PROBLEMS, TERMINAL, OUTPUT, DEBUG CONSOLE, PORTS, GITLENS, and POSTMAN CONSOLE.

Refactored Code

```
Ass-13.3.py > calculate_total
1  TAX_RATE = 0.18
2
3  def calculate_total(price):
4      """
5          Calculate total price including tax.
6
7          Parameters:
8          price (float): Base price of the product
9
10         Returns:
11         float: Total price including tax
12         """
13
14         tax = price * TAX_RATE
15         return price + tax
16
17 prices = [250, 500]
18
19 for price in prices:
20     total = calculate_total(price)
21     print("Total Price:", total)
```

Output:

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS POSTMAN CONSOLE

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' '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

```
Ass-13.3.py
1 print("Area of Circle:", 3.14159 * (7 ** 2))
2 print("Circumference of Circle:", 2 * 3.14159 * 7)
```

Output:

The screenshot shows a terminal window with the following content:

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS POSTMAN CONSOLE
PS C:\Users\yarav\OneDrive\Desktop\aiac> & 'c:\Python314\python.exe' 'c:\Users\yarav\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\pylauncher' '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

The screenshot shows a terminal window with the following content:

```
ASSISTANT Ass-13.3.py > ...
1 PI = 3.14159
2 RADIUS = 7
3
4 area = PI * (RADIUS ** 2)
5 circumference = 2 * PI * RADIUS
6
7 print("Area of Circle:", area)
8 print("Circumference of Circle:", circumference)
```

Output:

The screenshot shows a terminal window with the following content:

```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS POSTMAN CONSOLE
PS C:\Users\yarav\OneDrive\Desktop\aiac> & 'c:\Python314\python.exe' 'c:\Users\yarav\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\pylauncher' '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

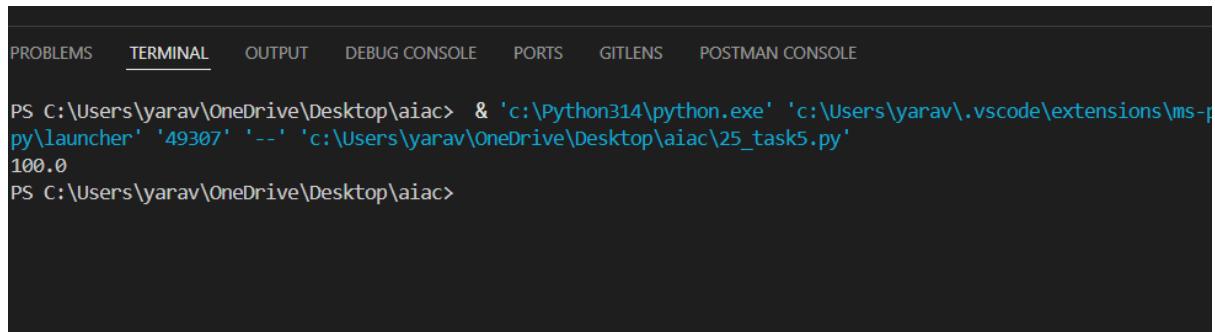
Use descriptive variable names.

Legacy Code

The screenshot shows a terminal window with the following content:

```
ASSISTANT Ass-13.3.py > ...
1 a = 10
2 b = 20
3 c = a * b / 2
4 print(c)
```

Output:



```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS POSTMAN CONSOLE

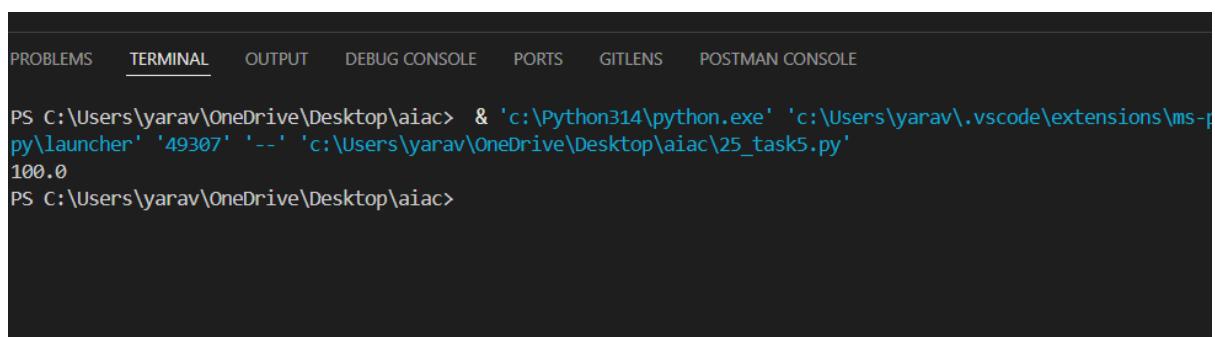
PS C:\Users\yarav\OneDrive\Desktop\aiac> & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python\launcher' '49307' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\25_task5.py'
100.0
PS C:\Users\yarav\OneDrive\Desktop\aiac>
```

Refactored Code



```
-Ass-13.3.py > ...
1  base = 10
2  height = 20
3
4  # Calculate area of a triangle
5  triangle_area = (base * height) / 2
6
7  print(triangle_area)
```

Output:



```
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS POSTMAN CONSOLE

PS C:\Users\yarav\OneDrive\Desktop\aiac> & 'c:\Python314\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python\launcher' '49307' '--' 'c:\Users\yarav\OneDrive\Desktop\aiac\Ass-13.3.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.