

Assignment-13.3

Name: B. Shravya

Hall Ticket No: 2303A51492

Batch: 08

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

Task 1: Refactoring – Removing Code Duplication

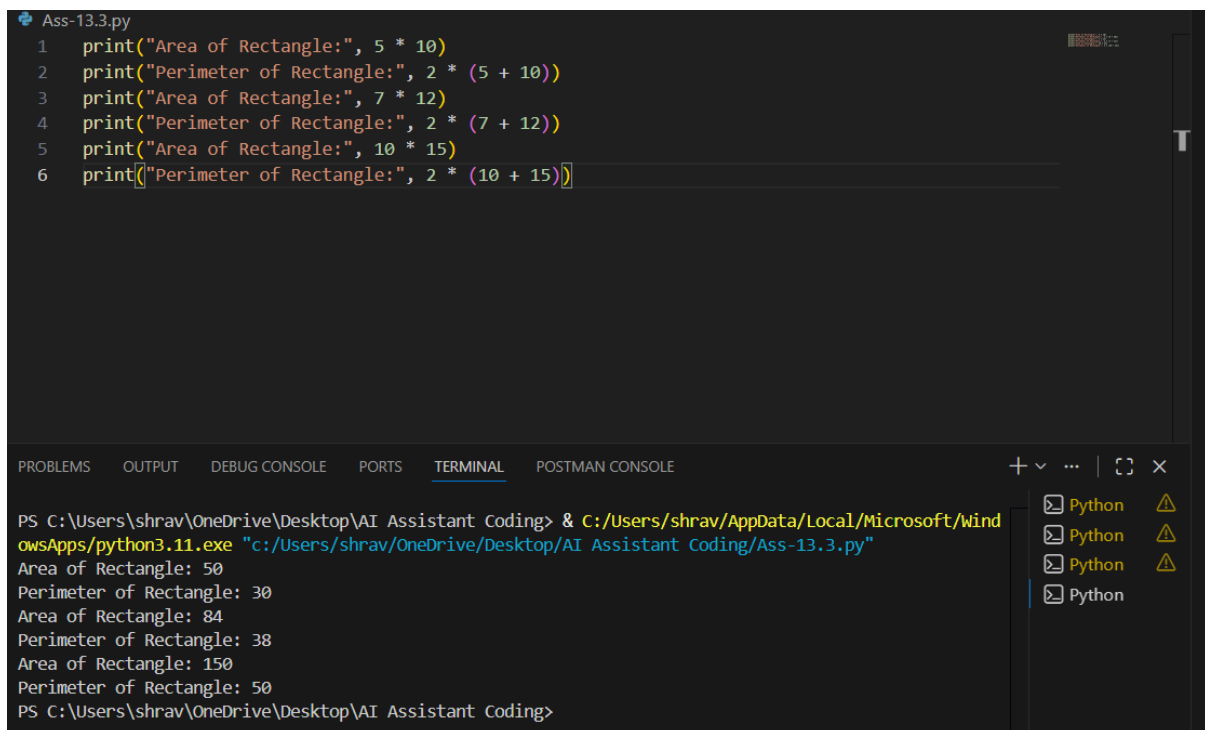
Objective

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 a code editor with a file named 'Ass-13.3.py'. The code contains six lines of print statements that calculate the area and perimeter of rectangles with different dimensions. The output window at the bottom shows the execution results of this 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))
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL POSTMAN CONSOLE

```
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding> & C:/Users/shrav/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI Assistant Coding/Ass-13.3.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\shrav\OneDrive\Desktop\AI Assistant Coding>
```

Refactored Code

```
Ass-13.3.py > ...
1 def rectangle_properties(length, width):
2     """
3     Calculate area and perimeter of a rectangle.
4
5     Parameters:
6     length (int or float): Length of the rectangle
7     width (int or float): Width of the rectangle
8
9     Returns:
10    tuple: area and perimeter of the rectangle
11    """
12    area = length * width
13    perimeter = 2 * (length + width)
14    return area, perimeter
15
16
17 rectangles = [(5, 10), (7, 12), (10, 15)]
18
19 for length, width in rectangles:
20     area, perimeter = rectangle_properties(length, width)
21     print("Area of Rectangle:", area)
22     print("Perimeter of Rectangle:", perimeter)]
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL POSTMAN CONSOLE

```
Perimeter of Rectangle: 50
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding> & C:/Users/shrav/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI Assistant Coding/Ass-13.3.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\shrav\OneDrive\Desktop\AI Assistant Coding>
```

Task 2: Refactoring – Optimizing Loops and Conditionals

Objective

Improve performance by replacing nested loops.

Legacy Code

```

Ass-13.3.py > ...
1  names = ["Alice", "Bob", "Charlie", "David"]
2  search_names = ["Charlie", "Eve", "Bob"]
3
4  for s in search_names:
5      found = False
6      for n in names:
7          if s == n:
8              found = True
9      if found:
10         print(f"{s} is in the list")
11     else:
12         print(f"{s} is not in the list")

```

Output:

```

owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI Assistant Coding/Ass-13.3.py"
Charlie is in the list
Eve is not in the list
Bob is in the list
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>

```

Refactored Code (Using Set Lookup)

```

Ass-13.3.py > ...
1  names = ["Alice", "Bob", "Charlie", "David"]
2  search_names = ["Charlie", "Eve", "Bob"]
3
4  name_set = set(names) # O(1) lookup
5
6  for name in search_names:
7      if name in name_set:
8          print(f"{name} is in the list")
9      else:
10         print(f"{name} is not in the list")

```

Output:

```

owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI Assistant Coding/Ass-13.3.py"
Charlie is in the list
Eve is not in the list
Bob is in the list
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>

```

Task 3: Extracting Reusable Functions

Objective

Modularize price and tax calculations.

Legacy Code

```

Ass-13.3.py > ...
1  price = 250
2  tax = price * 0.18
3  total = price + tax
4  print("Total Price:", total)
5
6  price = 500
7  tax = price * 0.18
8  total = price + tax
9  print("Total Price:", total)

```

Output:

```

owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI
Total Price: 295.0
Total Price: 590.0
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>

```

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     tax = price * TAX_RATE
14     return price + tax
15
16
17  prices = [250, 500]
18
19  for price in prices:
20      total = calculate_total(price)
21      print("Total Price:", total)

```

Output:

```

owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI
Total Price: 295.0
Total Price: 590.0
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>

```

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:

```
owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/
Area of Circle: 153.93791
Circumference of Circle: 43.98226
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding
```

Refactored Code

```
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:

```
owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI A
Area of Circle: 153.93791
Circumference of Circle: 43.98226
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>
```

Task 5: Improving Variable Naming and Readability

Objective

Use descriptive variable names.

Legacy Code

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

Output:

```
owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/
100.0
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>
```

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:

```
owsApps/python3.11.exe "c:/Users/shrav/OneDrive/Desktop/AI Assistant Co
100.0
PS C:\Users\shrav\OneDrive\Desktop\AI Assistant Coding>
```

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.