

# **AI Assisted Coding**

## **Assignment – 13.3**

**K.VAMSHIDHAR || 2303A510H7 ||**

**Batch:- 8**

### Task 1: Refactoring – Removing Code Duplication

#### Objective

To eliminate repeated logic by extracting reusable functions.

#### Task Description

Use AI assistance to refactor a legacy Python script that contains repeated blocks of code calculating the area and perimeter of rectangles.

#### Starter (Legacy) Code

```
# Legacy script with repeated logic
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))
```

```
lab 13.py x
Ask Copilot or type / for commands
#2303A510H7

# Copilot Prompt:
# Refactor this script to remove duplication by creating a reusable function
# that calculates the area and perimeter of a rectangle.

# Requirements:
# - Use a function
# - Add docstrings
# - Preserve output format

# 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))

def calculate_rectangle_properties(length, width):
    """
    Calculate and print the area and perimeter of a rectangle.

    Parameters:
    length (float): The length of the rectangle
    width (float): The width of the rectangle
    """

    area = length * width
    perimeter = 2 * (length + width)

    print(f"Area of Rectangle: {area}")
    print(f"Perimeter of Rectangle: {perimeter}")

# Example usage
calculate_rectangle_properties(5, 10)
calculate_rectangle_properties(7, 12)
calculate_rectangle_properties(10, 15)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
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\komma\Desktop\3rd YEAR\AI-AC>
```

## Task 2: Refactoring – Optimizing Loops and Conditionals

### Objective

To improve performance by replacing inefficient nested loops with optimized structures.

### Task Description

Use AI to analyze and refactor a script that checks the presence of elements using nested loops.

### Starter (Legacy) Code

```
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")

```

The screenshot shows a terminal window with the following content:

```

lab 13.3.py > ...
37 # Optimize this code by replacing nested loops with a faster solution.
38 # Requirements:
39 # - Use a set for lookup
40 # - Keep identical output
41 # - Add comments explaining optimization
42
43 # names = ["Alice", "Bob", "Charlie", "David"]
44 # search_names = ["Charlie", "Eve", "Bob"]
45
46 # for s in search_names:
47 #     found = False
48 #     for n in names:
49 #         if s == n:
50 #             found = True
51 #     if found:
52 #         print(f"{s} is in the list")
53 #     else:
54 #         print(f"{s} is not in the list")
55 names = ["Alice", "Bob", "Charlie", "David"]
56 search_names = ["Charlie", "Eve", "Bob"]
57 # Convert the list of names to a set for O(1) average time complexity lookups
58 names_set = set(names)
59 # Iterate through the search names and check for their presence in the set
60 for s in search_names:
61     if s in names_set:
62         print(f"{s} is in the list")
63     else:
64         print(f"{s} is not in the list")
65
66

```

Below the code, there is a terminal output window showing the execution of the script:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Perimeter of Rectangle: 50
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 13.3.py"
Charlie is in the list
Eve is not in the list
Bob is in the list
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>

```

### Task 3: Refactoring – Extracting Reusable Functions

#### Objective

To modularize code by extracting calculations into reusable functions.

#### Task Description

Refactor a legacy script where price and tax calculations are written inline.

## Starter (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)
```

The screenshot shows a code editor window with a dark theme. The file is named 'lab 13.3.py'. The code is as follows:

```
lab 13.3.py X
lab 13.3.py > ...
69 # Refactor this code by creating a reusable function calculate_total(price).
70 # Requirements:
71 # - Include docstring
72 # - Remove duplication
73 # - Preserve output
74
75 # price = 250
76 # tax = price * 0.18
77 # total = price + tax
78 # print("Total Price:", total)
79
80 # price = 500
81 # tax = price * 0.18
82 # total = price + tax
83 # print("Total Price:", total)
84 def calculate_total(price):
85     """
86     Calculate and print the total price including tax.
87
88     Parameters:
89     price (float): The original price of the item
90     """
91     tax = price * 0.18
92     total = price + tax
93     print(f"Total Price: {total}")
94 # Example usage
95 calculate_total(250)
96 calculate_total(500)
97
```

Below the code editor, there is a terminal window showing the output of running the script:

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd Y
Total Price: 295.0
Total Price: 590.0
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

## Task 4: Refactoring – Replacing Hardcoded Values with Constants

### Objective

To improve maintainability by replacing magic numbers with named constants.

## Task Description

Use AI to identify hardcoded values and replace them with constants.

### Starter (Legacy) Code

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

```
100  # Replace hardcoded values with named constants (PI, RADIUS).
101  # Improve readability and maintain identical output.
102
103  # print("Area of Circle:", 3.14159 * (7 ** 2))
104  # print("Circumference of Circle:", 2 * 3.14159 * 7)
105  PI = 3.14159
106  RADIUS = 7
107  area = PI * (RADIUS ** 2)
108  circumference = 2 * PI * RADIUS
109  print(f"Area of Circle: {area}")
110  print(f"Circumference of Circle: {circumference}")
111  +
112 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3rd YEAR/AI-AC/lab 13.3.py"
Area of Circle: 153.93791
Circumference of Circle: 43.98226
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```

## Task 5: Refactoring – Improving Variable Naming and Readability

### Objective

To enhance readability using descriptive variable names and comments.

### Task Description

Refactor a script with unclear variable names.

### Starter (Legacy) Code

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

```
.lab 13.3.py X
lab 13.3.py > ...
114 # Improve variable naming and readability.
115 # Requirements:
116 # - Use descriptive variable names
117 # - Add inline comments
118 # - Preserve output
119
120 # a = 10
121 # b = 20
122 # c = a * b / 2
123 # print(c)
124
125 # Calculate the area of a triangle using base and height
126 base_length = 10 # Length of the triangle's base
127 height_length = 20 # Height of the triangle
128 # Area of a triangle is (base * height) / 2
129 triangle_area = (base_length * height_length) / 2
130 print(triangle_area)
131 ✘

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC> & "C:/Program Files/Python312/python.exe" "c:/Users/komma/Desktop/3
100.0
PS C:\Users\komma\Desktop\3rd YEAR\AI-AC>
```