

ASSIGNMENT-13.3

Name: J.Vyshnavi

HT. No: 2303A51895

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.

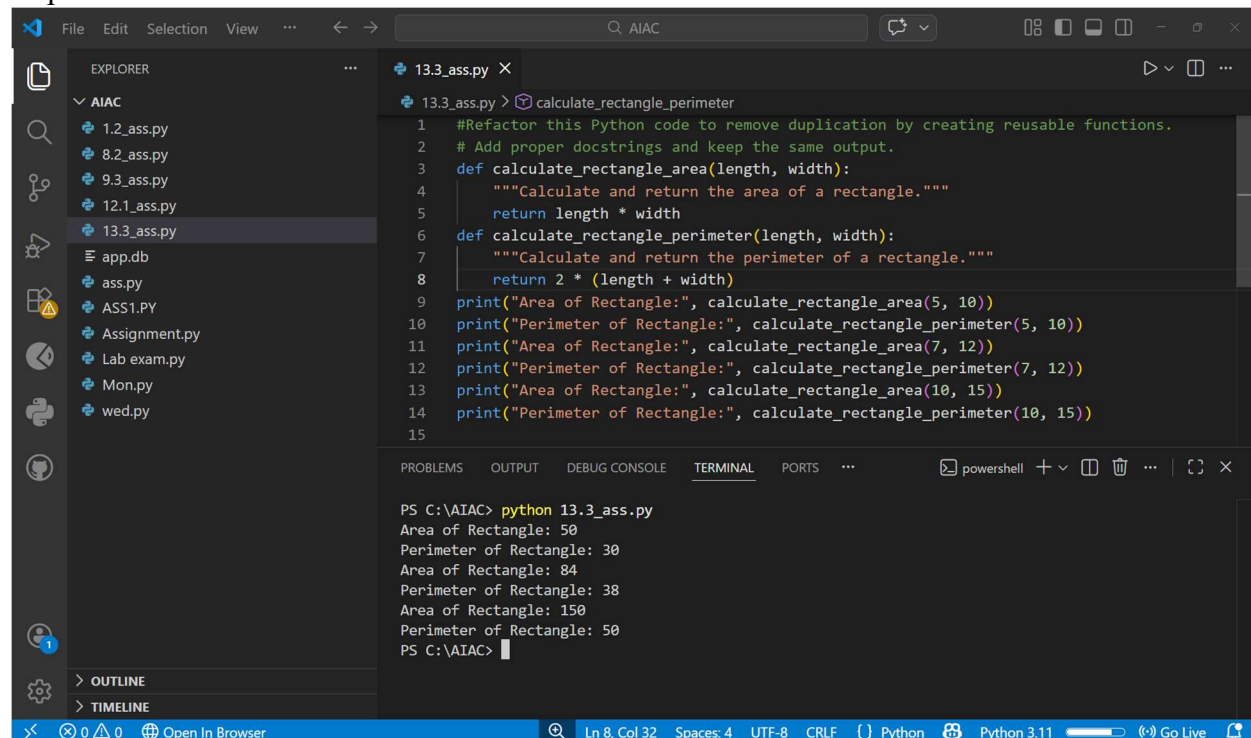
Task Description

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))
```

Expected Outcome



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'AIAC' with several Python files. The code editor displays the file '13.3_ass.py' with the following code:

```
13.3_ass.py > calculate_rectangle_perimeter
1 #Refactor this Python code to remove duplication by creating reusable functions.
2 # Add proper docstrings and keep the same output.
3 def calculate_rectangle_area(length, width):
4     """Calculate and return the area of a rectangle."""
5     return length * width
6 def calculate_rectangle_perimeter(length, width):
7     """Calculate and return the perimeter of a rectangle."""
8     return 2 * (length + width)
9 print("Area of Rectangle:", calculate_rectangle_area(5, 10))
10 print("Perimeter of Rectangle:", calculate_rectangle_perimeter(5, 10))
11 print("Area of Rectangle:", calculate_rectangle_area(7, 12))
12 print("Perimeter of Rectangle:", calculate_rectangle_perimeter(7, 12))
13 print("Area of Rectangle:", calculate_rectangle_area(10, 15))
14 print("Perimeter of Rectangle:", calculate_rectangle_perimeter(10, 15))
15
```

The terminal at the bottom shows the output of running the script:

```
PS C:\AIAC> python 13.3_ass.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:\AIAC>
```

Task 2: Refactoring – Optimizing Loops and Conditionals

Objective

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

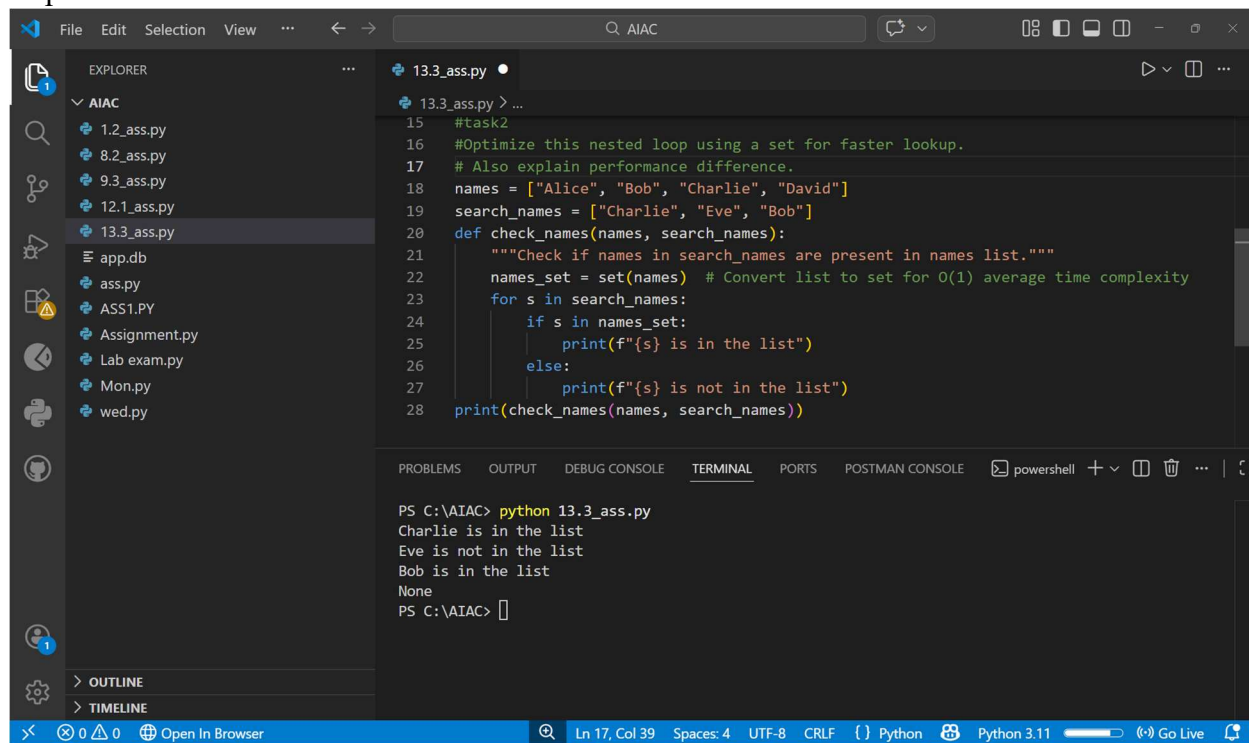
Task Description

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')
```

Expected Outcome

The screenshot shows a Visual Studio Code editor window. The Explorer sidebar on the left shows a file named '13.3_ass.py' selected. The main editor area displays the code for '13.3_ass.py', which includes a function 'check_names' that refactors the nested loop logic using a set for O(1) lookup. The code is as follows:

```
15 #task2
16 #Optimize this nested loop using a set for faster lookup.
17 # Also explain performance difference.
18 names = ["Alice", "Bob", "Charlie", "David"]
19 search_names = ["Charlie", "Eve", "Bob"]
20 def check_names(names, search_names):
21     """Check if names in search_names are present in names list."""
22     names_set = set(names) # Convert list to set for O(1) average time complexity
23     for s in search_names:
24         if s in names_set:
25             print(f'{s} is in the list')
26         else:
27             print(f'{s} is not in the list')
28     print(check_names(names, search_names))
```

The bottom panel shows the 'TERMINAL' output, which matches the expected outcome from the previous block:

```
PS C:\AIAC> python 13.3_ass.py
Charlie is in the list
Eve is not in the list
Bob is in the list
None
PS C:\AIAC>
```

Task 3: Refactoring – Extracting Reusable Functions

Objective

To modularize code by extracting calculations into reusable functions.

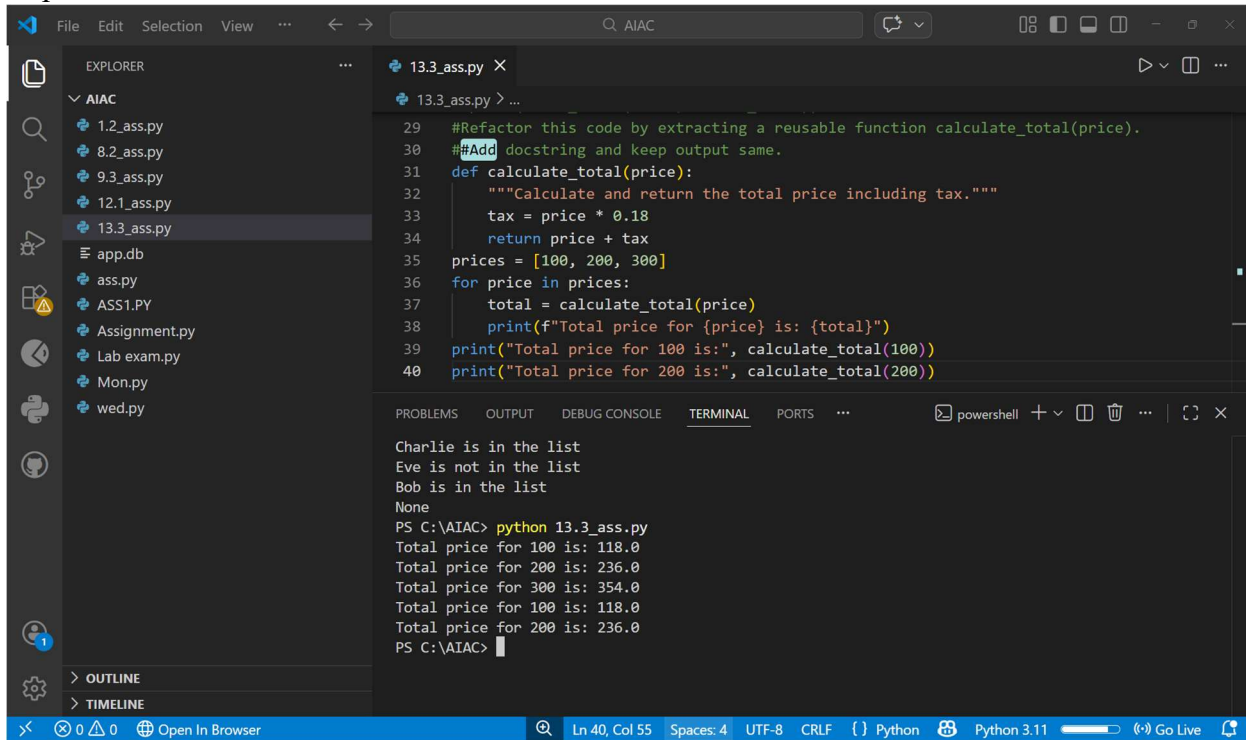
Task Description

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)
Expected Outcome
```



The screenshot shows a Visual Studio Code editor window. The Explorer sidebar on the left shows a project named 'AIAC' with several Python files. The file '13.3_ass.py' is selected and open in the editor. The editor displays the following Python code:

```
29 #Refactor this code by extracting a reusable function calculate_total(price).
30 ##Add docstring and keep output same.
31 def calculate_total(price):
32     """Calculate and return the total price including tax."""
33     tax = price * 0.18
34     return price + tax
35 prices = [100, 200, 300]
36 for price in prices:
37     total = calculate_total(price)
38     print(f"Total price for {price} is: {total}")
39 print("Total price for 100 is:", calculate_total(100))
40 print("Total price for 200 is:", calculate_total(200))
```

Below the editor, the TERMINAL panel shows the output of running the script:

```
Charlie is in the list
Eve is not in the list
Bob is in the list
None
PS C:\AIAC> python 13.3_ass.py
Total price for 100 is: 118.0
Total price for 200 is: 236.0
Total price for 300 is: 354.0
Total price for 100 is: 118.0
Total price for 200 is: 236.0
PS C:\AIAC>
```

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)
```

Expected Outcome

```
File Edit Selection View ... AIAC
EXPLORER
AIAC
1.2_ass.py
8.2_ass.py
9.3_ass.py
12.1_ass.py
13.3_ass.py
app.db
ass.py
ASS1.PY
Assignment.py
Lab exam.py
Mon.py
wed.py
OUTLINE
TIMELINE
13.3_ass.py
40 #Refactor this code by replacing magic numbers with constants.
41 PI = 3.14159
42 radius = 7
43 print("Area of Circle:", PI * (radius ** 2))
44 print("Circumference of Circle:", 2 * PI * radius)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... powershell
PS C:\AIAC> python 13.3_ass.py
Area of Circle: 153.93791
Circumference of Circle: 43.98226
PS C:\AIAC>
```

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)
```

Expected Outcome

