Laboratory Exercise Number 4: Code Refactoring in Defensive Programming

Introduction

Code refactoring is an essential skill in defensive programming. It involves restructuring existing code without changing its external behavior. The goal is to improve code quality, readability, and maintainability while reducing complexity and potential bugs.

Objectives

1. Understand the importance of code refactoring in defensive programming
2. Identify code smells and areas for improvement
3. Apply refactoring techniques to improve code quality
4. Practice writing more maintainable and robust code

Activity: Refactoring a Weather Data Processor

In this lab, you will be working with a Python script that processes weather data from a CSV file. Your task is to refactor the code to improve its quality, readability, and maintainability while adding defensive programming techniques.

Starting Code

python code to refactor (file: act4.py):

```
import csv

# Original code to refactor
def process_weather_data(file_path: str) -> None:
    data = []
    with open(file_path, 'r') as f:
        reader = csv.DictReader(f)
        for row in reader:
            data.append(row)

    total_temp = 0
    total_humidity = 0
    max_temp = float('-inf')
    min_temp = float('inf')
    for item in data:
        temp = float(item['temperature'])
        humidity = float(item['humidity'])
        total_temp += temp
        total_humidity += humidity
        if temp > max_temp:
            max_temp = temp
        if temp < min_temp:
```

```
            min_temp = temp

    avg_temp = total_temp / len(data)
    avg_humidity = total_humidity / len(data)

    print(f"Average Temperature: {avg_temp:.2f}°C")
    print(f"Average Humidity: {avg_humidity:.2f}%")
    print(f"Maximum Temperature: {max_temp:.2f}°C")
    print(f"Minimum Temperature: {min_temp:.2f}°C")

# Example usage
process_weather_data('weather_data.csv')
```

The starting code can be found in the file `act4.py`. It contains a single function `process_weather_data` that reads data from a CSV file, calculates some basic statistics, and prints the results.

Weather Data

A sample weather data file `weather_data.csv` is provided. It contains three months of daily weather data with the following columns:

- date: The date of the weather record (YYYY-MM-DD format)
- temperature: The temperature in degrees Celsius
- humidity: The relative humidity as a percentage

Tasks

1. Identify at least three areas where the code in `act4.py` can be improved or refactored.
2. Refactor the code to improve its quality, readability, and maintainability.
3. Add error handling and input validation to make the code more robust.
4. Implement type hinting for better code documentation.

Bonus Tasks

- Implement additional features, such as calculating the median temperature or finding the day with the highest humidity.
- Create a command-line interface for the weather data processor.

  Run in cmd:

  ```
  python act4_sol.py weather_data.csv
  ```

Submission

Submit your refactored code in a new file named `act4_sol_LASTNAME_FIRSTNAME_YEARSECTION.py` example: `act4_sol_Garcia_John_3A.py`, along with explanations of the improvements you made and why they are beneficial in the context of defensive programming.

Evaluation Criteria

Your solution will be evaluated based on the following criteria:

1. Code quality and readability
2. Proper implementation of defensive programming techniques
3. Effective use of error handling and input validation
4. Correct implementation of type hinting
5. Implementation of bonus features (if attempted)
6. Quality of explanations for the improvements made

Resources

- Python Documentation
- PEP 8 -- Style Guide for Python Code
- Refactoring Guru
- Python Testing with pytest

Happy refactoring!