Overview

Write a Python program that reads item-quantity pairs from a CSV file and performs various analyses and operations on the data.

Input

- The program should read data from a CSV file with two columns: one for item names and the other for quantities.
- Download: input1.csv

Tasks

1. **Read Data from CSV**: Read the item-quantity pairs from a CSV file.

2. **Total Quantity per Item**: Calculate and display the total quantity for each item.

3. **Count of Each Item**: Display how many times each item appears in the list.

4. **Median, Max, Min, and Average Quantities per Item**: Calculate and display the median, maximum, minimum, and average quantities for each item.

    - Median Formula
        1. **If the number of observations (n) is odd:** The median is the value at the middle position. It can be calculated using the formula: **Median = Value at position((n+1)/2)** Here, **n** is the total number of observations.
        2. **If the number of observations (n) is even:** The median is the average of the two middle values. It can be calculated using the formula: **Median = (Value at position(n / 2) + Value at position((n / 2)+1)) / 2** Again, n is the total number of observations.

5. **Total of All Items**: Calculate and display the total quantity of all items combined.

6. **Top 5 and Bottom 5 Quantities**: Identify and display the top 5 and bottom 5 quantities among the items, along with their respective items.

    - Just sort based on the top 5 quantities, you don't need to care about sorting the keys
7. **Grouping Quantities into Ranges with Specific Output Formatting**:

    - Define quantity ranges (e.g., 0-10, 11-20, 21-30, etc.).
    - Group items and their quantities into these ranges.
    - Display the output with the specified indentation format.

```
Unset
0-10
 Apple
  3
  7
 Orange
  7
  9
  10
```

Requirements

- Use pure Python, without external libraries like NumPy or Pandas.
- The code should be well-structured, commented, and optimized for performance.

Evaluation Criteria

- Correctness: The program should accurately perform all required calculations and display the results as specified.
- Code Quality: Clean, well-organized, and understandable code.
- Efficiency: Optimized for handling large datasets.
- Data Presentation: Accurate representation of data in the specified format.

Submission

- Submit a Python script (.py file) capable of running in a standard Python environment.
- The script should include comments explaining the approach and any assumptions made.

Example Output

```
Total quantity per item: {'item_27': 695, 'item_11': 876, 'item_02': 782, 'item_05': 815, 'item_24': 991, 'item
Count per item: {'item_27': 30, 'item_11': 36, 'item_02': 31, 'item_05': 34, 'item_24': 35, 'item_28': 30, 'ite
Median per item: {'item_27': 20.0, 'item_11': 25.0, 'item_02': 22, 'item_05': 22.0, 'item_24': 28, 'item_28': 1
Max per item: {'item_27': 49, 'item_11': 49, 'item_02': 50, 'item_05': 50, 'item_24': 50, 'item_28': 50, 'item_
Min per item: {'item_27': 2, 'item_11': 2, 'item_02': 1, 'item_05': 3, 'item_24': 4, 'item_28': 5, 'item_16': 1
Avg per item: {'item_27': 23.166666666666668, 'item_11': 24.333333333333332, 'item_02': 25.225806451612904, 'it
Total quantity of all items: 25515
Top 5 quantities: [('item_15', 50), ('item_24', 50), ('item_28', 50), ('item_13', 50), ('item_04', 50)]
Bottom 5 quantities: [('item_16', 1), ('item_08', 1), ('item_30', 1), ('item_29', 1), ('item_12', 1)]
Range 0-10: {'item_27': [9, 5, 6, 5, 2, 8, 9, 10], 'item_05': [6, 6, 5, 6, 5, 3, 3], 'item_10': [1, 3, 7, 10, 4
Range 11-20: {'item_28': [15, 17, 12, 16, 16, 19, 11, 14, 11, 20, 11, 14, 15], 'item_16': [11, 13, 16, 15, 14,
Range 21-30: {'item_11': [29, 27, 27, 22, 23, 26, 27, 24, 21], 'item_06': [21, 22, 25, 23, 24, 28, 22, 22, 21,
Range 31-40: {'item_24': [38, 38, 34, 38, 34, 36, 38, 33], 'item_02': [33, 37, 31, 40], 'item_22': [31, 37, 39,
Range 41-50: {'item_02': [49, 44, 49, 50, 50, 45, 44], 'item_15': [50, 45, 46, 49, 48, 44, 44, 45, 50, 47, 42],
```