Lab Demonstration and In-Class Exercise

Consider the operations of a small business. The business keeps records of all of its transactions with its customers. The file `LabDemo.csv` contains a small example of these records; this file contains a header, labeling the four column of this file as follows:

`custName`:	A code corresponding to the customer
`SKU`:	The product being sold in this transaction, listed as a stock keeping unit (SKU)
`orderQty`:	The quantity of the product being ordered
`unitPrice`:	The price per unit (in USD) for this product in this transaction

As part of this exercise, you will write some Python code to analyze these data. However, your code should be able to work on any file as long as it follows the template given above. For example, the customer names and SKUs could be different than those in `LabDemo.csv`, and the number of different customer names and SKUs could also differ (e.g., see the file `LabDemo2.csv`).

TASK: Write code that computes the average unit price for each product sold to each customer and exports this information to a file called `CustomerSummary.txt`. Customers should be listed in alphabetical order, and products should be listed in alphabetical order under each customer. All average prices should be rounded to the nearest cent. If a customer has never purchased a particular product, the file should list `No purchase history` for this product. To simplify the report for reading, please place a blank line after listing the summary for each customer. For example, `CustomerSummary.txt` should contain the following when run on the `LabDemo.csv` file.

```
Customer: LV
Average price for A: $4.38
Average price for T: $2.51

Customer: RY
Average price for A: $2.36
Average price for T: No purchase history

Customer: WX
Average price for A: $3.36
Average price for T: $2.54
```

USEFUL TOOLS:

1.) The `csv` package for reading in your data (also the `open`, `close`, `reader` and `next` functions)
2.) Lists and Dictionaries, for storing and manipulating data (also the `append`, `sorted`, and `set` functions)
3.) Control flow tools such as `for` loops and `if` statements
4.) The `int`, `float`, and `str` functions to cast data as integers, floats, and strings (also the `format` function)

HINTS AND RULES OF THUMB:

1.) Before starting to write Python code, write some pseudocode that describes how you will carry out all tasks within your code. This will help you mentally organize your tasks and determine what functions (and other tools) you will need to complete your task.
2.) Near the top of your file, assign the names of your input and output files to string variables, and refer to these variables later when interacting with your files. This will allow you to change the names of the input and output files in one place, rather than throughout your code.
3.) Never refer to a specific customer or SKU by name in your code (i.e., do not “hardcode” them). Since they can vary from file to file, you should always gather the list of customers and SKUs from the file itself.