Assignment 2: Store Stock Calculator

Description

You are going to create a program that monitors stock for a small business. Every day the cashier will run your script, which will import a CSV file with the current stock. The CSV file looks like this:

```
"Item","Current Stock","Price per Item"
"Apple","3","1.00"
"Banana","4","2.50"
"Orange","5","1.50"
```

Your program will import this information, and present it in a manner that is easy to read:

```
#   Item    Current Stock    Price per Item
=========================================
1.  Apple         3                  $ 1.00
2.  Banana        4                  $ 2.50
3.  Orange        5                  $ 1.50
```

Your table doesn't have to be identical to this, but it should have three columns which are properly justified.

During the day, the cashier will be able to enter items that have been sold using a user prompt.

```
Select a number (1-3) to indicate a sale, or 'e' to indicate end of
day:
```

Each time the cashier enters a number, the current stock will decrement by 1. If current stock reaches 0, stop decrementing (your store can't have negative stock!) You will use another variable to monitor 'lost sales', you will increment this number each time the cashier enters that item's number *after* it has reached zero.

When the cashier enters 'e', we have reached end of day and the store closes. Your program will then generate three reports:

Total Sales

This is simply a table which indicates the total sales, including totals for each item, and a grand total:

```
Total Sales
#   Item        Sales    Price per Item     Total
=============================================
1.  Apple         3         $ 1.00         $ 3.00
```

```
2.  Banana          1          $ 2.50          $ 2.50
3.  Orange          3          $ 1.50          $ 3.00
TOTAL                                          $ 8.50
```

Lost Sales

These are sales that were lost because of missing stock. On this day the cashier entered 1 four times for apples. 3 of those counted for sales, and the last was considered a lost sale.

```
Lost Sales
#   Item         Sales    Price per Item     Total
================================================
1.  Apple          1          $ 1.00          $ 1.00
TOTAL                                          $ 1.00
```

Restock Report

This report will tell the manager how many of each item will be needed to address the current demand. The demand for apples was 4, for example. The restock should also add an additional **20%** for each item (rounded to the nearest integer) to account for variation in demand:

```
Restock
#   Item         Demand 20%  Total Demand  Current Stock From Warehouse
=================================================================================
1.  Apple          4     1        5              0                       5
2.  Banana         1     0        1              3                       0
3.  Orange         3     1        4              2                       2
```

If the current stock is greater than the total demand, your ‘from warehouse’ number should be zero, we shouldn’t be sending stock back to the warehouse.

Finally, as an optional exercise: you can export a CSV file with ‘current stock’ updated. Assume that current stock now includes the items brought in from the warehouse.

First Milestone

For your first milestone, you will need to submit the following:

Program Requirements

- To start, we are going to ignore the part of the program dealing with importing the CSV file. Assume only one item: apples.
- Use the assignment 1 to plan how the program will work, as the cashier enters 1 for a variable number of times, and then ‘e’ to end the day.
- Save this as a Word file and submit it to Blackboard.
- Once you have planned your algorithm for one item, you can extend this using items in a list once you start on the final submission.

- Name this file `requirements.docx`.

Code

- Again, for the start of your code, ignore all tasks having to do with CSV files. Also ignore formatting your table.
- Work with only one item: apples. Set variables `item_name = 'Apple'`, `stock = 3`, and `item_price = 1.0`.
- Create a script that will allow the cashier to only enter `1` or `e`.
- Print the relevant values, again don't worry about formatting these numbers into a table.
- Name this file `assign2.py`.

Your first milestone should include both of these files.

Second Milestone

Your second milestone will be a continuation of `assign2.py`. At this point you have completed lab 7 and should understand how to work with CSV files.

Your `assign2.py` should contain all the code from the first milestone, as well as the following:

- Code to check command line arguments for the CSV filename (including error handling).
- Code to import all the data from the CSV file. When we import data from CSV files, we generally get a *list of dictionaries*.
- Code to present the CSV data in a table, as seen in the first sample output above.
- The code from the first milestone should now be using values from the *first dictionary* created from the CSV data, rather than from `item_name`, `stock`, or `item_price`. However, you don't need to create any other tables, or work with items other than apples.

Your second milestone should again be named `assign2.py`, and should contain any code that you missed from the first milestone.

The Final Submission

Name your assignment `assign2.py`. This should be a working script that I can run on my computer, which will fulfill the program requirements.

- Modify your code so that we are now handling each item listed in the CSV file. Your handling of stock should be *dynamic*: you may have more than 3 items in it.
- Modify your output so that the relevant data is organised in the tables you see above.

- Add a function (or functions) to export your data into a new CSV file. This new CSV filename should be defined in a *second* command line argument.
- Make sure that your script is properly documented.

Your submission should:

- contain a docstring at the top, with your name, myseneca ID, and description of the program. For example:

```
'''
Name: Eric
Myseneca: esmith1
Description: Enter your description of the program here.
'''
```

- Your script should only import `csv`, `sys`, and `os`.
- Your script must **use the design patterns taught in class**. Code that obviously comes from external sources will be flagged for Academic Integrity violation.
- You script must contain documentation, including:
  - The top docstring
  - function-level docstrings
  - in-line comments explaining *why* you are doing what you're doing in the code.

Ideally, your script should run without errors. However if there are errors, I will award partial marks if you're on the right track.

You will submit this final version of `assign2.py` to Blackboard.

Academic Integrity

A reminder that the college has a **Zero Tolerance Policy** in regards to academic integrity. In the professional world, an employee who plagiarises is a huge liability for the company. Code falls under intellectual copyright laws: incidents involving stolen have caused litigation and millions of dollars in legal fees for companies. In the professional world, being caught stealing code **will** result in termination and possibly legal action.

On the other hand, it is expected that junior developers will make mistakes, hit roadblocks, get frustrated, get confused, introduce bugs and need help from a senior developer or mentor. Events are accepted and even celebrated as 'part of the process.' It's always better to ask for help: **knowing when you need help is just as important a skill as learning to code.**

Even if your code is not working or is incomplete, you will most likely get partial marks for what you have accomplished. Submitting code that is not your own, however, will always get a zero.

What Makes It Plagiarism?

You can absolutely communicate with classmates, and refer to outside sources. When you are discussing problems, use *pseudocode* or flow charts to explain. Sharing one or two lines of code is acceptable, sharing more than two lines of code is not. **Never share your assignment with other students, even if they say it is for reference**. You will be charged with an academic integrity violation and receive a zero.

Rubric

Concept	Points
First Milestone	2.5
Second Milestone	2.5
Basic Function	5
Error Handling	2.5
Documentation and Comments	2.5
Total	**15**