CP1404/CP5632 2016 SP2 Assignment 1 – Shopping List 1.0

Task:

You are to plan and then code a console-based program in Python 3, as described in the following information and sample output. This assignment will help you build skills using selection, repetition, file input/output, exceptions, lists, functions and string formatting. Do not to define any of your own classes. Assignment 2 will build on this program with more advanced code constructs including classes and a Graphical User Interface (GUI). Work incrementally on this task: complete small parts of it at a time rather than trying to get it all working at once.

Program Overview:

This program is a simple shopping list that could also be used for TODO lists or similar purposes. The program maintains a list of items in a file, and each item has:

- name, price, priority (1, 2 or 3), whether it is required or completed

Users can choose to see the list of required items or completed items, including a total of the (estimated) price of all of those items. The lists will be sorted by priority.
Users can add new items and mark items as completed.
They cannot change items from completed to required.

Program Functionality Details:

Ensure that your program has the following features, as demonstrated in the sample output below.
Your program should:

- display a welcome message with your name in it
- display a menu for the user to choose from
- return to the menu after each action and loop until the user chooses to quit
- error-check user inputs as demonstrated in the sample output
- load a CSV (Comma Separated Values) file of items (just once at the very start); a sample CSV file is provided for you
- *when the user chooses **list**:* display a neatly formatted list of all the required items with their prices (lined up) and priorities
- *when the user chooses **show completed**:* display a similarly formatted list of completed items
- *when the user chooses **add**:* prompt for the item's name, price and priority, error-checking each of these, then add the item to the list in memory (not to the file)
- *when the user chooses **complete**:* display the list of required items (same as for list), then allow the user to choose one (error-checked), then change that item to completed
    - if no items are required, then a message should be displayed and the user returned to the menu
- *when the user chooses **quit**:* save the items to the CSV file, overwriting the file contents

Planning:

Write up the algorithms in pseudocode for the two functions: **load items** and **complete an item**.
Do this in a docstring (comment) directly above each of these functions.
At the very top of your code file, add a docstring containing your **name, date, brief program details** and **a link to your project on GitHub** (private repository).
Follow the guide to good pseudocode and examples presented in the subject to ensure this is done to a high standard.
You may show this part of the assignment to your tutor during practical time to get comments or suggestions.

Coding Requirements and Suggestions:

- Make use of named constants where appropriate.
- Use functions appropriately for each significant part of the program: this is the divide-and-conquer problem-solving approach. Remember that functions should "do one thing".
  Look for situations where functions can be used to reduce code duplication (e.g. displaying the completed and required lists is very similar).
- For efficiency, you should only load the items file once. Store the results appropriately in a list of lists and pass that to any functions that need access to it.
- Note that the menu choice should handle uppercase and lowercase letters.
- Use exception handling where appropriate to deal with input errors (including entering numbers and selecting items).
- Check the rubric carefully to see any other aspects of the coding that you will be assessed on.

Output Requirements:

Sample output from the program is provided below. Ensure that your program matches the sample output including spaces, spelling, and especially the formatting of the item lists. Think of this as helpful guidance as well as training you to pay attention to detail. The sample output is intended to show the full range of situations including user input error handling.

Submission:

Submit **one** Python 3 (.py) code file containing your comments and pseudocode.
Please name the file like: **FirstnameLastnameA1.py**
e.g. if your name were Miles Davis, the filename would be MilesDavisA1.py
Submit your single .py file by uploading it on LearnJCU under Assessment.

Git/GitHub:

You must use Git version control and keep your project updated in a *private* repository on GitHub.
Make sure you register for the student discount so you get free private repositories:
https://education.github.com/discount_requests/new
You must add your lecturer and tutor as collaborators so they can access your repository:
https://help.github.com/articles/inviting-collaborators-to-a-personal-repository
You are assessed on your use of version control including commits and commit messages, so commit regularly (each logical chunk or milestone) and use meaningful commit messages in the imperative voice. Your commits should show steady work completed over reasonable time, not all in a short period.
You must be easily identifiable from your GitHub username.

Due:

Submit your assignment by the date and time specified on LearnJCU.
Submissions received after this date will incur late penalties as described in the subject outline.

Integrity:

The work you submit for this assignment must be your own. You are allowed to discuss the assignment with other students and get assistance from your peers, but you may not do any part of anyone else's work for them and you may not get anyone else to do any part of your work. Note that this means you should never give a copy of your work to anyone or accept a copy of anyone else's work. Submissions that are detected to be too similar to that of another student will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.
If you require assistance with the assignment, please ask **general** questions on the discussion forum, or get **specific** assistance with your own work by talking with your lecturer or tutor.

Sample Output:

The following sample run was made using a CSV file that contained:

```
Fish fingers,12.95,2,r
Metal detector,42.5,3,r
Coffee beans,40.0,1,r
```

Bold green text below shows user input for this sample.

```
Shopping List 1.0 - by Lindsay Ward
3 items loaded from items.csv
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> c
No completed items
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> r
Required items:
0. Coffee beans         $  40.00 (1)
1. Fish fingers         $  12.95 (2)
2. Metal detector       $  42.50 (3)
Total expected price for 3 items: $95.45
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> M
0. Coffee beans         $  40.00 (1)
1. Fish fingers         $  12.95 (2)
2. Metal detector       $  42.50 (3)
Total expected price for 3 items: $95.45
Enter the number of an item to mark as completed
>>> 1
Fish fingers marked as completed
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> C
Completed items:
0. Fish fingers         $  12.95 (2)
Total expected price for 1 items: $12.95
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> a
Item name:     (blank spaces entered)
Input can not be blank
Item name: Watch
Price: $not much
Invalid input; enter a valid number
Price: $-50
Price must be >= $0
Price: $50
Priority: low
Invalid input; enter a valid number
```

```
Priority: 0
Priority must be 1, 2 or 3
Priority: 2
Watch, $50.00 (priority 2) added to shopping list
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> M
0. Coffee beans         $  40.00 (1)
1. Watch                $  50.00 (2)
2. Metal detector       $  42.50 (3)
Total expected price for 3 items: $132.50
Enter the number of an item to mark as completed
>>> 2
Metal detector marked as completed
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> m
0. Coffee beans         $  40.00 (1)
1. Watch                $  50.00 (2)
Total expected price for 2 items: $90.00
Enter the number of an item to mark as completed
>>> two
Invalid input; enter a number
>>> 2
Invalid item number
>>> 0
Coffee beans marked as completed
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> M
0. Watch                $  50.00 (2)
Total expected price for 1 items: $50.00
Enter the number of an item to mark as completed
>>> 0
Watch marked as completed
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> m
No required items
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> r
No required items
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> C
Completed items:
0. Coffee beans         $  40.00 (1)
1. Fish fingers         $  12.95 (2)
2. Watch                $  50.00 (2)
3. Metal detector       $  42.50 (3)
```

```
Total expected price for 4 items: $145.45
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> e
Invalid menu choice
Menu:
R - List required items
C - List completed items
A - Add new item
M - Mark an item as completed
Q - Quit
>>> Q
4 items saved to items.csv
Have a nice day :)
```

At the end of this run, the CSV file contained:

```
Fish fingers,12.95,2,c
Metal detector,42.5,3,c
Coffee beans,40.0,1,c
Watch,50.0,2,c
```

Marking Scheme:

Ensure that you follow the processes and guidelines taught in class in order to produce high quality work. Do not just focus on getting the program working. This assessment rubric provides you with the characteristics of exemplary, good, competent, marginal and unacceptable work in relation to task criteria.

Criteria	Exemplary (4)	Good (3)	Competent (2)	Marginal (1)	Unacceptable (0)
Planning **Pseudocode for algorithms**	Clear, well-formatted, consistent and accurate pseudocode that completely and correctly solves the problem, for two functions.	Exhibits aspects of exemplary (left) and competent (right)	Some but not many problems (e.g. an incomplete solution, inconsistent use of terms, inaccurate formatting, not for two functions).	Exhibits aspects of competent (left) and unacceptable (right)	Very many problems or pseudocode not done.
Program Execution **Correctness** Worth double (8/6/4/2/0)	Program works correctly for all functionality required.		Program mostly works correctly for most functionality, but there is/are some required aspects missing or that have problems.		Program works incorrectly for all functionality required.
Error checking	Invalid inputs are handled well using exceptions and control logic as instructed, for all user inputs.		Invalid inputs are mostly handled correctly as instructed, but there is/are some problem(s), e.g. exceptions not well used.		Error checking is not done or is very poorly attempted.
Similarity to sample output (including all formatting)	All outputs match sample output perfectly, or only one minor difference, e.g. wording, spacing.		Multiple differences (e.g. typos, spacing, formatting) in program output compared to sample output.		No reasonable attempt made to match sample output. Very many differences.
Quality of Code **Identifier naming**	All function, variable and constant names are appropriate, meaningful and consistent.		Multiple function, variable or constant names are not appropriate, meaningful or consistent.		Many function, variable or constant names are not appropriate, meaningful or consistent.
Use of code constructs	Appropriate and efficient code use, including good logical choices for data structures and loops, good use of constants, etc.		Mostly appropriate code use but with definite problems, e.g. unnecessary code, poor choice of data structures or loops, no use of constants.		Many significant problems with code use.
Use of functions	Functions and parameters are appropriately used, functions reused to avoid code duplication.		Functions used but not well, e.g. incorrect/missing parameters or calls, unnecessary duplication or main code outside main function.		No functions used or functions used very poorly.
Formatting	All formatting is appropriate, including correct indentation, horizontal spacing and consistent vertical line spacing. PyCharm shows no formatting warnings.		Multiple problems with formatting reduces readability of code. PyCharm shows formatting warnings.		Readability is poor due to formatting problems. PyCharm shows many formatting warnings.
Commenting	Helpful block/inline comments and meaningful docstrings for all functions, top docstring contains all program details (name, date, basic description, GitHub URL).		Comments contain some noise (too many/unhelpful comments) or some missing program details in top docstring or some inappropriate or missing block/inline comments.		Commenting is very poor either through having too many comments (noise) or too few comments.
Version Control (GitHub)	Git/GitHub used effectively and the repository contains a number of commits with good messages that demonstrate incremental code development.		Aspects of the use of version control are poor, e.g. not many commits, or meaningless messages that don't represent valuable incremental development.		Git/GitHub not used at all.