CMSC 201 Spring 2017
Project 2 – Vending Machine

Assignment: Project 2 – Vending Machine
Due Date:
Design Document: **Saturday**, April 15th, 2017 by 8:59:59 PM
Project: Friday, April 21st, 2017 by 8:59:59 PM
Value: 80 points

Collaboration: For Project 2, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

For Project 2 you will have to turn in a “design document” in addition to the actual code. The design document is intended to help you practice deliberate construction of your program and how it will work, rather than coding as you go along, or starting without a plan.

Instructions

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on manipulating lists, calling functions, and reading from and writing to a file.

The design for Project 2 is entirely up to you – suggestions are provided within the project description, but you are not required to use them.

At the end, your Project 2 file must run without any errors.
It must also be called proj2.py (case sensitive).

Additional Instructions – Creating the proj2 Directory

During the semester, you’ll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 2 files. We recommend calling it `proj2`, and creating it inside a newly-created directory called `Projects` inside the `201` directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

Project 2 is designed to give you practice with file I/O, three-dimensional lists, and creating and calling functions. You'll need to use practically everything you've learned so far, and will need to do some serious thinking about how all of the pieces you need to create should fit together.

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Task

UMBC recently switched from being a Pepsi campus to carrying Coke products instead. They've also decided that they want to update the snack vending machines' technology for the 21st century, and have hired you to implement a prototype using the Python programming language. The new vending machines will use the balance on the user's red UMBC card to make purchases, and will display the contents of the machine digitally.

Your program will need to allow the user to:

- Read in the contents of a vending machine from a file
- Set their initial card balance (must be a positive amount)
- Display the vending machine (in a lined-up grid)
    - "Empty" slots should not display item, price, or vending "code"
- Make their selection via the vending "codes" for each item
- Check their card balance
- Add money to their card balance (must be a positive amount)
- Exit the program
    - Exiting the program will update the vending machine contents by writing the updated data to the same filename

Specification

Prior to this assignment, **you should be familiar with the entirety of the Coding Standards**, available on Blackboard under “Assignments” and linked on the course website at the top of the “Assignments” page.

You should be commenting your code, and using constants in your code (not magic numbers or strings).
Any numbers other than 0 or 1 are magic numbers!

You will **lose major points** if you do not follow the 201 coding standards.

If you have questions about commenting, whitespace, or any other coding standards, please come to office hours.

Additional Specifications

For this assignment, **you must use a three-dimensional list to store the vending machine within your program**. You also must create and call **at least five individual functions**. All other design decisions are up to you.

For this assignment, you do need to worry about “input validation.” You may assume that the user will enter the correct type of input (for example, an integer if one is asked for, a float if one is asked for), but the input may be negative, outside of the allowable range, or “bogus” (in the case of strings).

If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

It is also acceptable if your program crashes when a filename is entered that either does not exist, or has the wrong formatting for the vending machine program.

Details

The program starts by asking the user for the filename that contains the vending machine info, and the amount of money currently on their card (the amount must be positive). It should then present them with a menu (the five choices seen below).

```
    1 - Display Vending Machine
    2 - Make Selection
    3 - Display Card Balance
    4 - Add Money to Card
    5 - Quit
```

Choice 1 should display the vending machine in a lined-up grid, as seen in the sample output file. If there are zero of an item left, neither the name, cost, or code should be display – in its place in the grid it should say "empty". See the sample output for an example of this.

Choice 2 allows the user to make a selection, using the "code" associated with each item. Once selected, it should prompt the user until a valid code is chosen, re-prompting if necessary. Once a valid code is chosen, it needs to check if the user has enough money. **(Items of which there are zero left are not a valid code, and the user should be prompted to make a different choice.)** If the user does not have enough money, it should display an error message and return to the menu. If the user does have enough money, the amount of that item should decrease by 1, and the user's card balance should be charged that item's cost. Once purchased, a message should be printed with the item's name and the new card balance.

Choice 3 simply displays the user's card balance. (If it displays something like `$5.0` instead of `$5.00` that's fine.)

Choice 4 asks the user how much money they want to add to their card, and must be a positive amount.

Choice 5 quits the program, but first automatically saves the contents of the updated vending machine to the same filename as it was read in from. Foods that have a quantity of 0 should also be written back into the file. (The contents do not need to be lined up like they are in the original sample files.)

Input Files:

The input files will all have the same format, in which each line contains four things (in this order), separated by one or more spaces:

- Item name (six characters or less)
- Item price (as a float)
- Item amount (as an integer – may be zero!)
- Item code (a string – may contain letters, numbers, or both)

For simplicity's sake, every vending machine will have a **multiple of 3 items**, and your vending machine should display the contents in rows of 3 as well.

You can download two examples from Dr. Gibson's **pub** folder. Remember that your program will be over-writing these files, so you may need to re-copy them when you are testing your program. You are also highly encouraged to make your own test files – you can create and share these with your classmates if you want as well.

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/fuud.txt .
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/candi.txt .
```

Hints and Tips:

- Remember that lists are mutable as long as they are changed in place. You should not need to create a deep copy of the vending machine, as there is only a single machine.
- We suggest first writing the function to read the vending machine in from the file, then the function to print it, and finally the function to write the updated machine back to the file. If you want to see the 3D list (with brackets and commas), you can use `print(vendingMachine)`.
- You can use `strip()` and `split()` when reading, and `join()` when writing. Don't forget to cast from string to int/float, and back.
- The snack name, price, quantity, and code will always be in the same place – it might be a good idea to create constants for the indexes.
- Do NOT try to code up the entire project all at once! It won't go well!!!

Points

The project is worth a total of 80 points. Of those points, 10 will be based on your design document (creation of it and following it), 10 will be based on following the coding standards, and the other 60 will be based on the functionality and completeness of your project.

Design Document

The design document will ensure that you begin seriously thinking about your project way early on. This will not only give you important experience doing design work, but will help you gauge the number of hours you'll need to set aside to be able to complete the project. **Your design document must be called design2.txt.**

For Project 2, you are creating the design entirely on your own.
You **may NOT work with another student** to “brainstorm” a solution or discuss any general approaches or requirements. If you need assistance with the design document, come to office hours.

Your design document must have four separate parts:

1. A file header, similar to those for your assignments
2. Constants
    a. A list of all the constants your program will need, including a short comment describing what each “group” of constants is for (*e.g.,* menu options, meaning of indexes, etc.)
3. Function headers
    a. A complete function header comment for each function your plan to create, including the description, inputs, and outputs
4. Pseudocode for main()
    a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under the relevant comment (if applicable)

Your design can follow the same general format as the design for Project 1.

Your **`design2.txt`** file will be compared to the **`proj2.py`** file that you submit. Minor changes to the design are allowed. A minor change might be the addition of another function, or a small change to **`main()`**.

Major changes between the design and your project will lose you points. This would indicate that you didn't give sufficient thought to your design.
(If your submitted design doesn't work, it is generally better to lose the points on the design, and to have a functional program, rather than turning in a broken program that follows the design. The ultimate decision is up to you.)

To submit your design document, use

```
linux1[4]% submit cs201 PROJ2_DESIGN design2.txt
Submitting design2.txt...OK
linux1[5]% 
```

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard, and is called "sample2.txt". The sample output uses the "cat" command to "concatenate" a file to the screen so that you can see its contents (without having to open it up in emacs).
(Yours does not have to match the sample output exactly, but it should be similar, and the columns in the vending machine should all line up nicely.)

IMPORTANT:
You may notice that during the run of the program, the user's card balance displays as $3.3000000000000003 (instead of $3.30). This is a result of the way that the computer stores numbers, and is not an error with the program itself. You might also notice that at the end of the sample run, the user has $4.440892098500626e-16 left on their card (instead of $0.00). Again, this is not an error with the program. If your program does this as well, do not try to fix the behavior, as it may cause other problems.

Submitting

Once your **`proj2.py`** or **`design2.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the design or project in multiple times, as you reach new milestones or complete each piece. To do so, run **`submit`** as normal.)

To submit your design file (which is due Friday, April 14th, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ2_DESIGN design2.txt
Submitting design2.txt...OK
linux1[5]%
```

To submit your project file (which is due Friday, April 21st, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ2 proj2.py
Submitting proj2.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**