CMSC 201 Spring 2018

Project 1 – Pantry Planner

Read this entire document before you begin!

Assignment: Project 1 – Pantry Planner
Due Date:
Design Document: Friday, March 30th, 2018 by 8:59:59 PM
Project: Friday, April 6th, 2018 by 8:59:59 PM
Value: 80 points

Collaboration: For Project 1, **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
```

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 using functions to break a large task down into smaller parts. You will be required to turn in a design document before you turn in your actual code.

This is the first assignment where you've had to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberately constructing your program and how it will work, rather than coding as you go along, or starting without a plan.

A list of functions required for this project are found later in this document. You are ***required*** *to follow the provided design exactly*! You may add additional functions, but you must implement all of the specified functions as described in the design overview.

At the end, your Project 1 file must run without any errors.
It must also be called proj1.py (case sensitive).

Additional Instructions – Creating the proj1 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 1 files. We recommend calling it `proj1`, 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 1 is designed to give you lots (and lots) of practice with functions, as well as help you become more familiar and comfortable with 2D lists. You'll need to use **`while`** loops, control statements like **`if/else`**, passing in parameters, returning from functions, shallow copies of lists, and algorithmic thinking.

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

```
scl enable python33 bash
```

Task

You'll be implementing a relatively simple pantry planner that allows the user to enter the food they have available, to view the entered food, and to determine how many "complete" meals they could make from the food they have.

See the sample output available on Blackboard for detailed examples of how everything works, and continue reading for more information.

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

For now, you should pay special attention to the sections about:

- Comments
  - ***Function header comments***
  - Please note that the "Input" and "Output" in the function header comment do NOT mean what is shown on the screen with `print()`, or what is gotten from the user with `input()`. They refer to the **parameters** taken in, and the **return value**. (Both "Input" and/or "Output" may be None if appropriate.)
- Constants
  - You must use constants instead of magic numbers or strings!!!
- Make sure to read the last page of the Coding Standards document, which prohibits the use of certain tools and Python keywords.

Additional Specifications

For this assignment, **you must follow the design overview** in this document.

For this assignment, you do need to worry about "input validation." You may assume that the user will enter an integer, but it may be negative or outside of the allowable range.
If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

Project

The project is worth a total of 80 points. Of those points 10 will be based on your design document, 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 make sure that you begin thinking about your project in a serious way early on in the process. 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 design1.txt.**

For Project 1, the design overview is included in this document, and you are **required to follow it**. For future projects, you will be creating the design entirely on your own, and may choose to design it however you like.

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
3. Function headers
    a. A complete function header comment for each function
    b. You do not need to include the function's code in your design
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)

Although you will be presented with a design overview, you must still create the function headers and the pseudocode for `main()` on your own.

A start for your design is provided on Blackboard under "Assignments". Follow the layout and format of that document. You can also copy it using:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/design1.txt .
```

NOTE: The sample design provided is not complete, and is missing many constants and function header comments. You must add them!

Your **`design1.txt`** file will be compared to the **`proj1.py`** file that you submit. Minor changes to the design are acceptable. 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 decision is ultimately up to you.)

To submit your design document, use

```
linux1[4]% submit cs201 PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]%
```

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard, and is called "sample1.txt". Look at the sample output before reading the notes below.
(Yours does not have to match the sample output exactly, but it should be similar.)

Other Hints

- Writing up and testing each of your functions individually will make your life much, much easier.
- Commenting your code as you write it or even before you write it (think pseudocode) will make the process much simpler, and will also allow the TAs to help you more effectively during the process.

- This project lends itself very well to a "top down" implementation, where functions are first made as dummy functions, and you ensure that everything works together. (Dr. Gibson's first draft of the program just said "printing the pantry, beep boop" and similar things.)

Design Information

You are required to implement and use at least the following eight functions for Project 1, in addition to **`main()`**. You may implement more functions if you think they are necessary, but the eight below must be implemented **and used**, or you will lose significant points on your project grade. The information for each function is below.

Printing Functions

As the name suggests, these functions print information to the user. **None** of them return values – they only print information, which does NOT count as output. Some of them take in parameters, while others do not.

- **`def printMenu(menuList)`**
  - Prints out any of the menus in the program, including adding the "Quit" choice and option at the end of the menu

```
E - Enter food
V - View pantry
P - Possible complete meal?
Q - Quit
```

  - Takes in a list of the menu options
  - Returns nothing, since it is a print function

- **`def viewPantry(foodList)`**
  - Prints out the entire contents of the pantry, sorted by food type

```
Vegetables:
     carrot          25 calories     61 grams
     kale chips      50 calories     99 grams
     bell pepper     24 calories     119 grams
```

  - Takes in the full food list
  - Returns nothing, since it is a print function

- **`def printFoods(foodList, foodType)`**
  - Prints out the details of a single food type
  - Takes in the full food list, and the string of the food type to print
  - Returns nothing, since it is a print function
    - **NOTE:** This function is designed to mainly be a helper function, and is meant to be used by another function to accomplish its task.

Helper Functions

As the name suggests, these functions "help" other functions, by performing small tasks that will be needed by many pieces of the program. They are often called from functions other than `main()`.

- **`def validIntInput(minimum, maximum, message)`**
  - Gets a valid integer from the user that falls within a certain range (between **`minimum`** and **`maximum`**, inclusive), using the provided string of **`message`** when asking the user for input.
  - Takes in two integers and a prompt
    - The prompt is used by the function when asking for input
    - The two integers are the min and max values (inclusive)
  - Returns an integer
  - **HINT**: Look at Lecture 13 for an example of a similar function, **`getValidInt()`**, which does not contain a prompt parameter

  - **IMPORTANT:** This function is also set up to allow the user to enter numbers that have no maximum value. It does this by checking to see if the value of the **`maximum`** parameter is "NO MAX". See the sample output for an example of this behavior when entering a food's calories,

- **`def validMenuChoice(menuList, message)`**
  - Gets a valid choice from the user that exists in the menu options
  - Takes in a list of menu options and a message to use for input
  - Returns a single-character string representing the user's choice

UPDATED!!!

- **`def createMenuChoices(menuList)`**
  - Creates the single-character menu choices derived from the menu options provided
  - Takes in a list of menu options
  - Returns a list of single-character menu choices
    - For example, given **`["Pet dog","Throw ball", "Rub belly"]`**, the function should return **`["P", "T", "R"]`**

General Functions

These are the "heavy lifters" of the program, and do the majority of the work, and are almost always called from `main()`. They often call other functions, often more than one.

- **`def addFood(foodList)`**
  - Allow the user to add a new food to their pantry
    - User is prompted for food type, name, calories, and weight
    - **NOTE:** The user should be allowed to quit when choosing the food type, but not at any other time.
  - Takes in the full food list
  - Returns nothing; the food list is a shallow copy and should be edited in place, so any changes made will "stick" regardless

- **`def countComplete(foodList)`**
  - Count the number of "complete" meals that are possible from the pantry's contents
    - A "complete" meal is one consisting of a single food from each of the five food types
  - Takes in the full food list
  - Returns an integer; the number of "complete" meals possible
  - **WARNING:** This function is probably the most complex one in the whole project, so take some time to think carefully about what it needs to do and plan how it will accomplish that before you start writing any code for it.

Additional Details

- Although "Quit" is not included in any of the actual menu option lists, it must be included at the end of the printed menu, and a user must be allowed to choose "Q" to quit from that menu.
- Calories entered by the user have no maximum value.
- Weight entered by the user has a maximum value of 11340 grams (this is 25 pounds, which is many refrigerator shelves' weight capacity).
- We've provided a sample "starting" pantry that you can copy into your program to use for testing. Before submitting your project, you **MUST REMOVE** this sample pantry from your code.
  - (See the sample output file for how to download the file.)

Helpful Checklist

We understand that the first big project can be overwhelming, which is why we've provided you with a detailed description of all the functions, along with sample output, a starter design, and more. Below, you'll find a summary of the key actions and where important documents can be found.

Design:

- ☐ Download the starter design from Dr. Gibson's public directory:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/design1.txt .
```

- ☐ Read through the function descriptions provided in this document, and create function header comments for each function in your design
- ☐ Look at the sample output on Blackboard, and begin sketching out an idea in pseudocdoe for how your **`main()`** will work
  - ☐ Look at the function headers you wrote to gain more insight

Coding:

- ☐ [Optional step] Make a copy of your design (or your design so far), as a framework for starting the project
- ☐ Going off of your design, start coding up your project piece by piece
  - ☐ Use top-down or bottom-up implementation – your choice!
  - ☐ Don't code up everything at once, and test as you go
- ☐ If you make changes to your design while coding, and it's before the design due date, update your design and resubmit it!
- ☐ Come to office hours when you need help or get stuck

General:

- ☐ Submit the design to PROJ1_DESIGN, **not** PROJ1
- ☐ Start working on the project itself before March 30th if possible, since actually coding it up will improve your design as you go
- ☐ Test your code with the user choices made in the sample output
  - ☐ [Optional step] Download the file of the pre-populated pantry, and copy it into your own project for testing (we'll shown how to do this in class)

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

    - If you do this, don't forget to remove it before submitting!

Submitting

Once your **`proj1.py`** or **`design1.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the project in multiple times, as you reach new milestones. To do so, run **`submit`** as normal.)

To submit your design file (which is due Friday, March 30th, 2018 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]%
```

To submit your project file (which is due Friday, April 6th, 2018 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ1 proj1.py
Submitting proj1.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 assignment 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.**