Keith Burdette

February 28th, 2020

Foundations of Programming: Python

Assignment 06

<https://github.com/twkeith/IntroToProg-Python-Mod06>

**Using Python Classes and Functions**

**INTRODUCTION**

This assignment was an extension of last week’s To Do List. Instead of coding all the processing and presentation steps within the if/else statement, we used functions to that accomplish these tasks. A class for processing tasks was set up with the shell for the functions we need to add and remove items from a list. A second class for interacting with the user was created and we need to flesh out functions that asks the user about the tasks they need to add and remove. Setting it up this way allows us to follow the Separation of Concerns.

**TODO LIST PROGRAM – Part 2**

The main structure of the program was already set up. Separate classes for processing and input/output had been created. Each class had shells of functions that we needed to finish. The first function was adding data to our list. The function definition had three parameters: a task, a priority and the list of all tasks. Task and priority were strings so the first line of code I added I converted them into a dictionary with a Task field and a Priority field. In the second line, I added this row into my list of rows with an append statement. This whole function is shown in Figure 1.

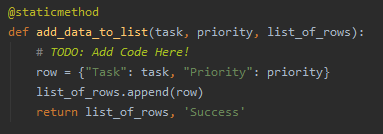


Figure 1 – Add To List Function

The second function to add to was the remove function. Task and list parameters were given for this function. I set up a for loop to roll through the items in my list. For each item in the list, I compare the task argument with the task element of the dictionary item from my list. If I find the task, I remove the item from the list and jump out of the function with a return statement passing back my modified list and a string indicating success. If there are multiple instances of the same task, this will only remove the first one found in the list. If the for loop reaches the end without finding the task, I return my list along with a fail string. This whole function is shown in Figure 2.

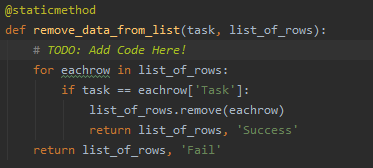


Figure 2 – Remove Items Function

The next function I needed to complete was writing to a file. This function had two parameters: a list and a filename. I created a writeobjfile that opened my filename in write mode. This will overwrite any data previously in the file. I loop through each row in my list and write it one by one to the file. I used a concatenated string of the task element and priority element separated by a comma with a carriage return at the end. After the for loop finishes, I close my file object and return my list and a success string out of the function. Since I’m not modifying the list, I probably don’t need to return the list out of the function but it was already created this way. This function code is shown in figure 3.

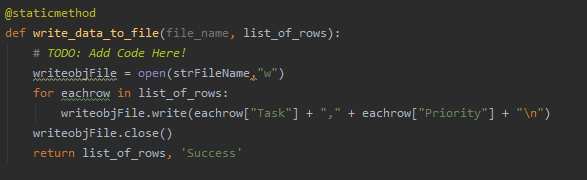


Figure 3 – Write To File Function

That finished up the processor functions we need to write. Moving onto the input/output class, many functions were already completed. The two left to finish were inputting a new task and removing a task. These could be accomplished with simple input statements and a return statement with those inputs. This is shown in Figure 4.

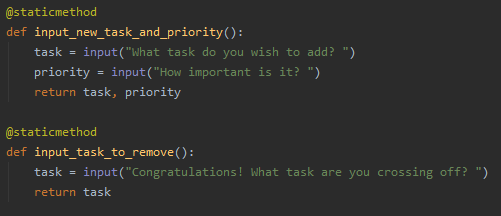


Figure 4 – User Input Functions

With our two classes finished, we now move onto the main section of the program that will execute our functions. Most of the menu navigation was already built. We just needed to add our modified functions in the appropriate places. For menu choices 1 and 2, we needed to add an input/output class function and a processor function. First, we call the input and set the appropriate return variables. We then use those variables as arguments to the processing function and return the list and status string variables. The status string is then passed into the continue function to let the user know if the function succeeded. Menu choices 1 and 2 are shown in Figure 5.

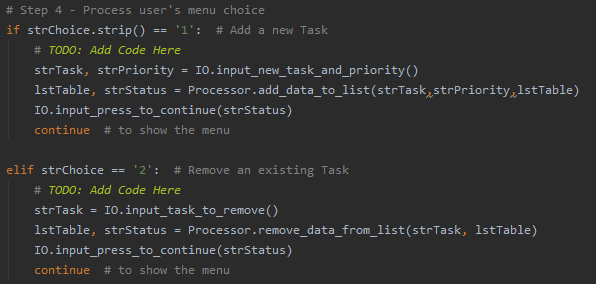


Figure 5 – Menu choices to add and remove items

The last step was to complete the menu options to write and reload data. Both of these required only one line, calling the processor function with the filename and list. The return for both of these functions were the list and the status string. These two menu choices are shown in Figure 6.

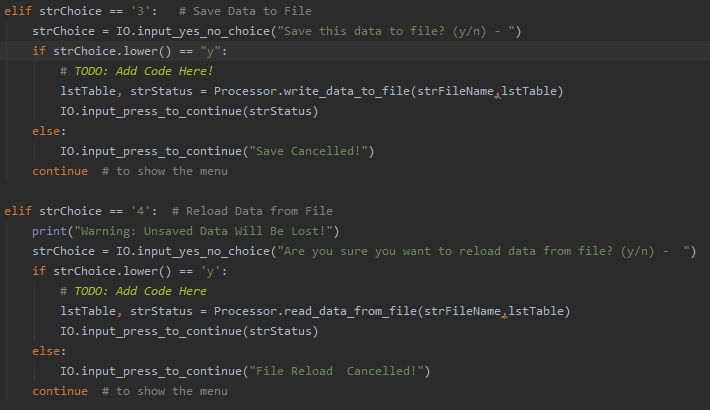


Figure 6 – Menu Choice to Read and Write Files

**TESTING**

When I first ran the program, I ran into an error because there was no file by that name. I added a task/priority pair to the text file and tried again. Figure 7 shows the initial load of the file and then an add item using menu choice 1.

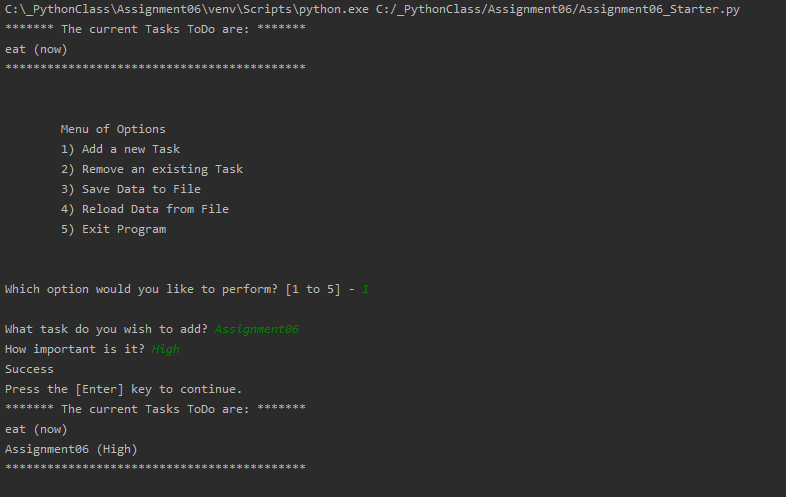


Figure 7 – Load File and Add Item

I ate a sandwich so I can cross eating off the list. I tried to remove “ate” from the list but it couldn’t find it in the list so I got a message saying it failed. I tried removing “eat” and that succeeded. This is shown in Figure 8.

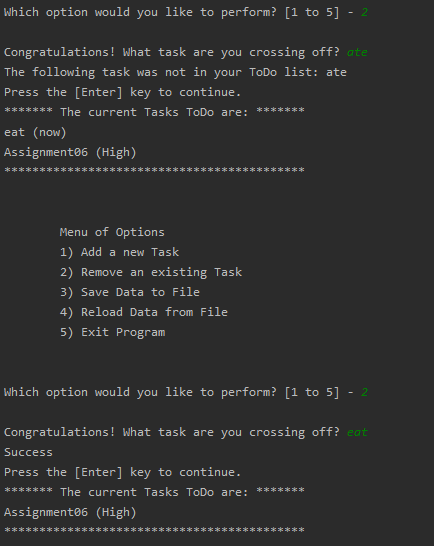


Figure 8 – Removing an item

I added a couple more items and saved that to the file. I then opened the command line and ran my code. This is shown in Figure 9.

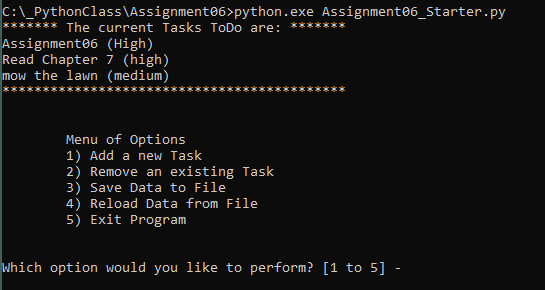


Figure 9 – Command Line File Load

I test a few more lines just to test that all the functions work. Figure 10 below shows that I can modify the lists all I want. But if I choose option 4, I can reload my currently saved list and lose any changes I have made.

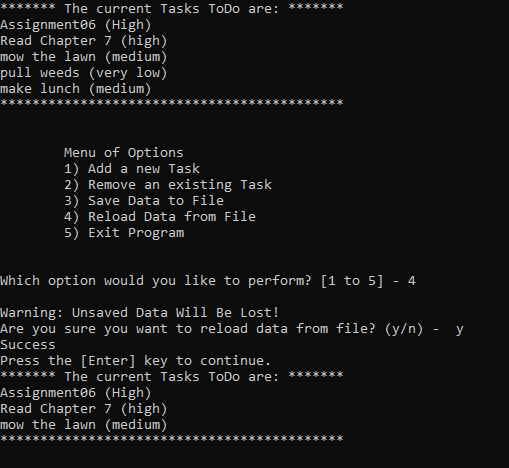


Figure 10 – Loading From File

**SUMMARY**

This assignment shows the power of functions. Your code becomes more readable if you break out the specific tasks outside the main part of your program. Functions also allow your code to be repeatable. Instead of hunting through your for repeated instances of the tasks when making changes, you can use a function and modify a single series of statements in your code. For our ToDo list, we built functions that allowed us to follow the Separation of Concerns. One class of functions allowed the user to interact with the menu and input tasks. A second class of functions built and edited the core list of tasks the user wished to track. The result is a nice text-based program that anyone can easily follow and use for their day-to-day life.