Lacey Redinger

11/6/2019

Foundations of Programming: Python

Assignment 05

Title

# Introduction

In Module 5 we learned more about Lists as well as Dictionaries. The main thing we explored in this module was working with “permanent” data, like in an external file (.txt, .csv, etc.) and temporary data within a program. Finally, we learned some tips for creating professional code, working with someone else’s code, and using GitHub.

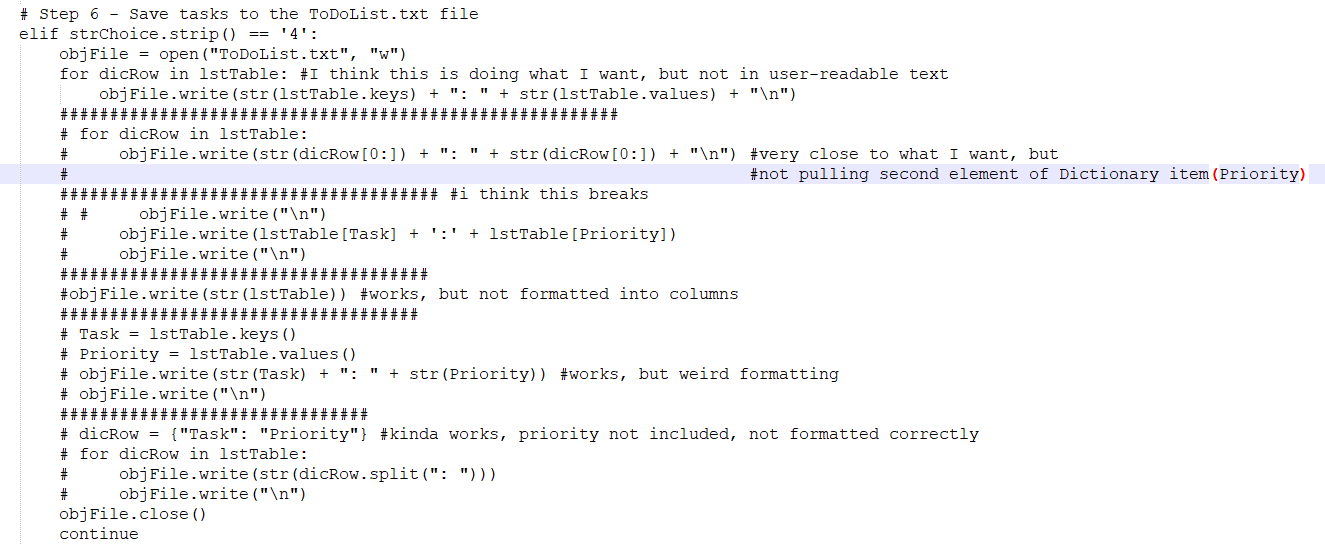
# Lists and Dictionaries

List is just a simple collection of data, a list of colors, a list of numbers, a list of teams, identified by square brackets. A dictionary is two items connected by a colon within curly brackets. A dictionary is a key and its corresponding value, which can be a list of things, or have multiple types of values. They are very similar but have different features. The main thing that is different about dictionaries, aside from syntax, from lists, is that they organize data that should be kept together. For example you could have a list of colors but you want to attach that list to a definition or group. So you could make a dictionary of colors. Example: colors = {red: primary, blue: primary, green: secondary, chartreuse: tertiary} the colors are keys, which should be unique. The type of color or definition of the color (primary, secondary, tertiary) are values. Unlike lists which can be indexed numerically, dictionaries are indexed by their keys.

# Assignment 05

I had a much harder time with this assignment, I think because pulling data from a file into “memory” or within the program was confusing to me. I’m used to working with actual files, if I wanted to write a ToDo list in real life, I would scratch it down on a post-it note, or I might write it up in a Note on my phone, or a text file. I wouldn’t run a complicated program. So in my mind, if I did pull up a program to help me write a ToDo list, I figured why not just work directly with the file. Initially I was trying to use sample code from the Module05 video and programming notes, but I was trying to use it to access or create the txt file. I got the code to mostly work, but I couldn’t figure out how to delete items from the list and I couldn’t figure out how to format the dicRows in the script or back into the txt file.

Here is some sample code (Figure 1) as I worked on trying to format my lstTable back into the txt file, wow is it ugly! Coming back and looking at this in retrospect is kind of embarrassing, but I know this is part of learning.

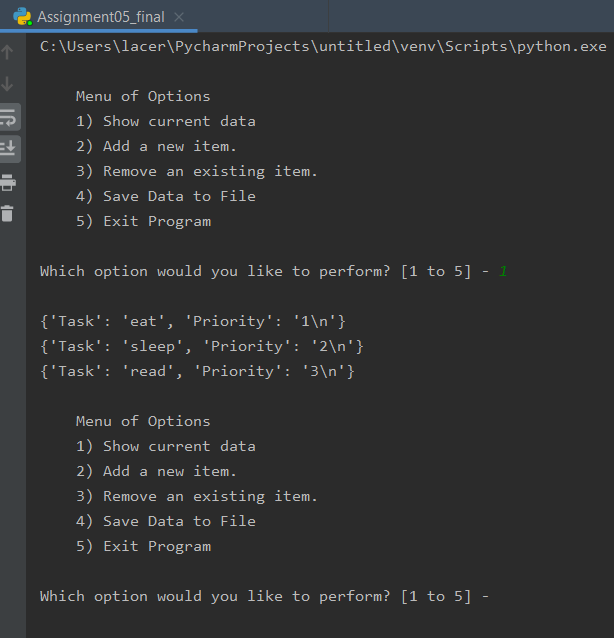


***Figure 1: Embarrassing Code, Exhibit A***

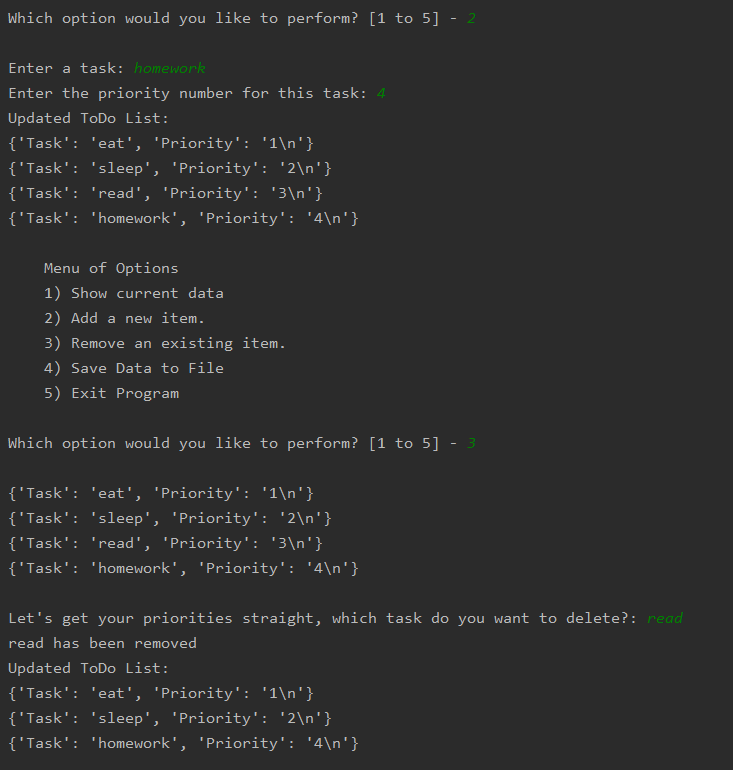
Chapter 5 of the book, presented dictionaries very differently than was explained in the course video and matched more closely with what I was reading online, so then I started second guessing myself and getting really confused. Then I tried mixing a bunch of code from the “geek” example in the book, with the code from Mod5 programming notes and things got really wacky. After making a pretty big mess I decided to try and get clarification on how the ToDo list program should start (with a pre-populated txt file, or by creating one as part of the script). Meanwhile I accidentally deleted all the code I had been struggling through for the past 2 days.

I had to start basically from scratch, but this time I was confused about what I was learning about dictionaries from the book and from the internet and the way dictionaries and dictionary rows and list tables were being explained in the course video. It was pretty tough to wrap my head around. Luckily a co-worker took a look at my code and tried to explain things I was obviously confused about (mainly dictionary rows and List tables) His explanations were more in line with the book and the internet so I re-wrote my code creating an internal dictionary within the script and then moving through all the menu option blocks. I got everything to work, very similar to the “geek” dictionary example in Chapter 5 of the book. However, not only could I not figure out writing back to the file, I also got clarification on the assignment which caused me to have to rewrite the whole thing again.

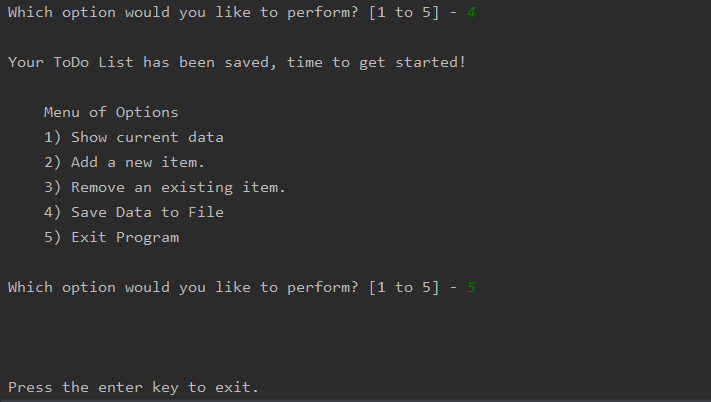
Another coworker looked at my new code, and looked at the lab examples in Mod5 Programming notes and was able to explain to me that the variable dicRow was only being used to populate the lstTable (temporary memory) with data from the txt file (permanent memory) and put it into the script for modification. All modification was happening on this temporary data pulled from the txt file, then once the user is done adding or deleting various tasks, that temporary data can overwrite the ToDoList.txt or simply exit the program and keep the txt file the way it was originally. So I made a new variable called objRow to differentiate between the dicRow pulling data from the file, and objRow accessing rows in the lstTable (while modifying the data). I think this whole assignment would have made more sense if we were only using lists, not trying to use dictionaries. Or if the txt file was pulled into a single dictionary rather than a List Table. Here are screenshots of the program running in PyCharm:



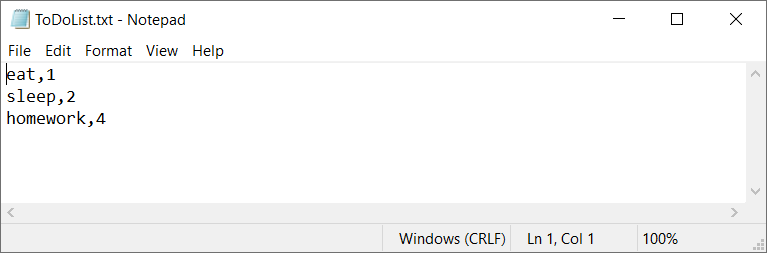
***Figure 2: ToDoList – Option 1 (PyCharm)***



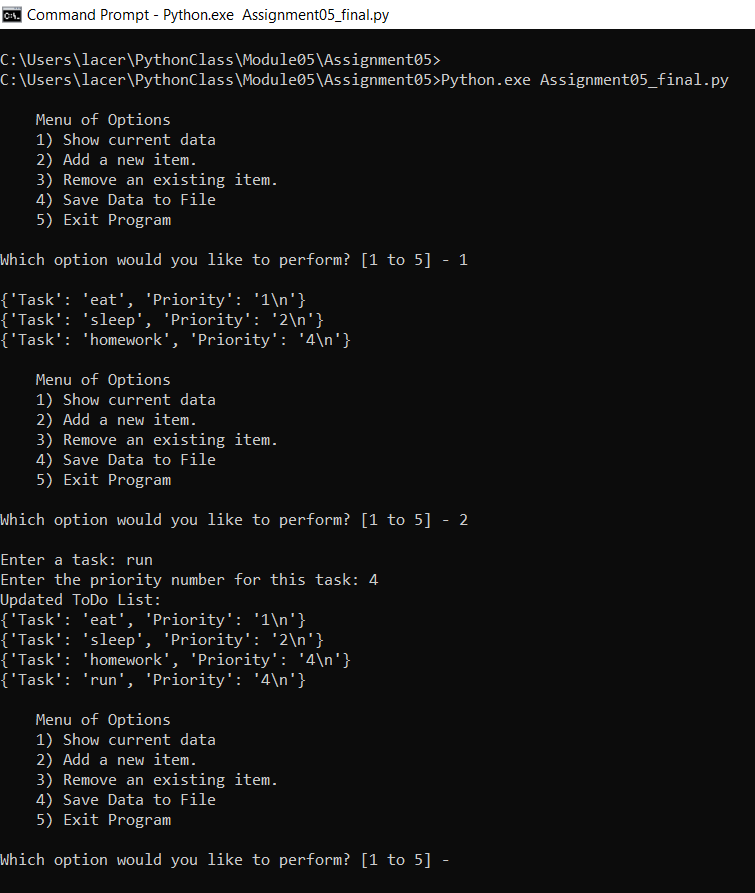
***Figure 3: ToDoList – Options 2 & 3 (PyCharm)***



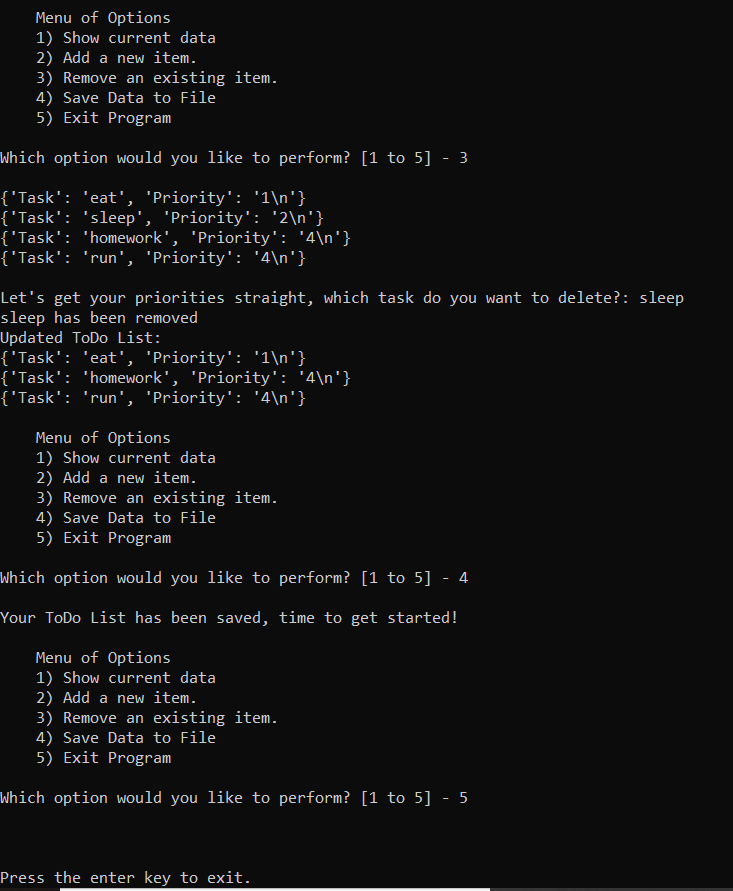
***Figure 4: ToDoList – Options 4 & 5 (PyCharm)***

***Figure 5: Updated ToDoList.txt***

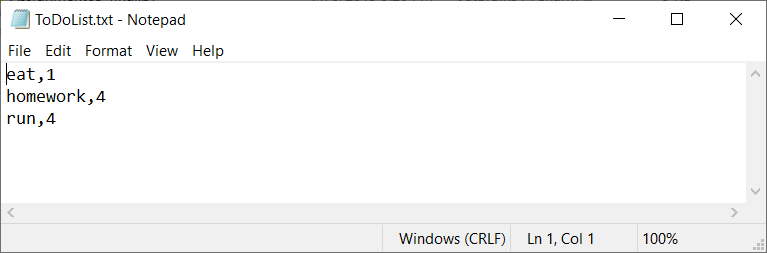
Here are screenshots of the program running in Windows Command Prompt:



***Figure 6: ToDoList Options – 1 & 2 (Windows)***

******

***Figure 7: ToDoList Options 3-5 (Windows)\***

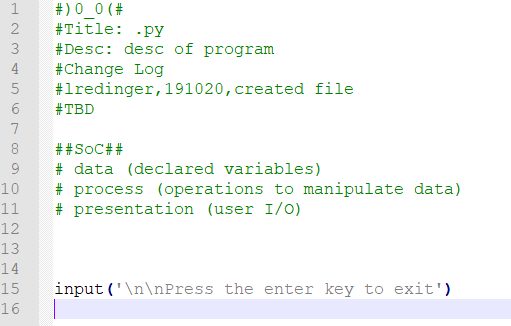


***Figure 8: Updated ToDoList.txt***

# Be Professional

Part of creating professional code includes clarity, consistency, and organization. Some ways to achieve this are to have a script template which includes a header identifying the program, the programmer, and a change-log. One also might include a Separation of Concerns which is Data (declaring variables, constants, comment descriptions), Processing (the functions and operations of the code, manipulating the data) and Presentation (this is user input and output, how the data is displayed, and parameters to keep the code from breaking. For example, using a try/except.) Including the SoC in the template can help a programmer stay organized and make sure everything is included.

There might also be helpful or common functions, preferred style or other things which could be included in a template to speed up some of the legwork. Here is an example of my script template:



***Figure 9: Script Template***

Another tool that professionals utilize is GitHub which is a place for programmers to share data, save their work, collaborate and learn. Code can be stored in repositories and changes can be committed to save progress.

# Summary

To struggle is to learn. Even though this assignment was really hard for me, I learned a lot. There are a lot of things in Python that can be used or done many different ways. This flexibility of Python makes it easy to use, but I also think it makes it harder to learn in some ways. Lists and dictionaries can be used in similar ways, and sometimes that can be confusing. I definitely have a better understanding of how temporary memory and externally saved data are used in programming, as well as how and why they are different.