Nick Hertlein

February 16, 2021

IT FDN 110 B: Foundations Of Programming: Python

Assignment05

GitHub URL: https://github.com/nhertlein/IntroToProg-Python.git

Lists & Dictionaries

# Introduction

This document will illustrate the knowledge gained from the fifth lecture and the media portion of the third assignment. In addition, I will cover the creation of a Python script to track a to do list by “task” and “priority” entered by the user. The program will give a number of different options to allow the user to read from a current saved to do list, see what is currently in the to do list, add a task to the list, remove a task from the list, and write the list to a text file. One difference with this assignment is that we had a “starter” file with some comments to guide us, but also ensure we followed a specific structure.

# Writing the To Do List Script

For this week’s assignment I made a project in PyCharm in the Assignment05folder of the C:\\_PythonClass directory. As we were starting with a “starter” file for this assignment the first step included updating the header change log.

The first section of the script was labeled “Data” and there was a comment to use that space to declare variables and constants. There were some variables pre-established for us to use, so I kept those for consistency. One thing I changed was creating constants for the filename and filepath and using objFile as an actual file object. As the filename and filepath were constants I made the variable names in capital letters to follow proper form. Another thing I did was populate the strMenu variable with the actual menu text to help make the code more readable and the menu more accessible if it needs to be changed in the future. See Figure 1 for variable and constant layout.

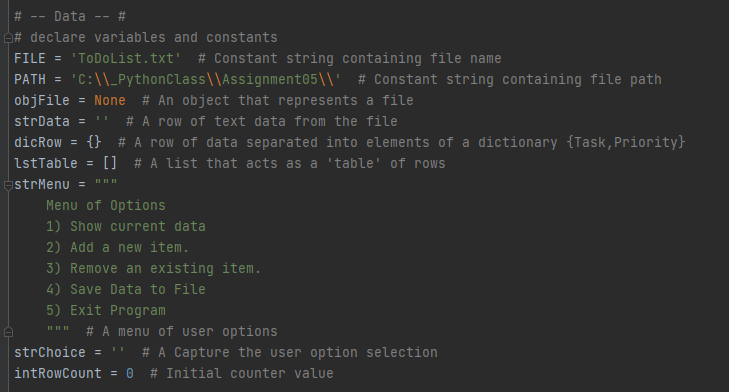


Figure 1. Declaration of variables

The next section of the script was focused on data processing. This section was focused on bringing data in from a .CSV text file and saving the pair of values into a dictionary structure. The dictionary structure was then added to the bottom of the to do list (Figure 2). The lstTable was one of the items initialized in the first section as a blank list, so it could be added to without causing an error.

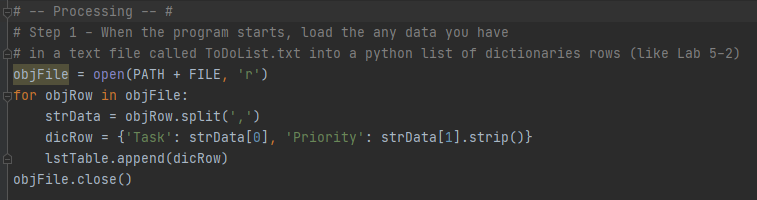


Figure 2. Reading data from a file into the list format

The last section of the script was the input/output section of the script which was the biggest section. The main feature of the section was a while loop that repeated to allow the user to make selections until they chose to exit. The first option the user could select was the ability to print the current list of tasks and priority. Printing of the tasks was handled by a for loop which broke out each row of the list and separated each component of the row by their dictionary keys (Figure 3).

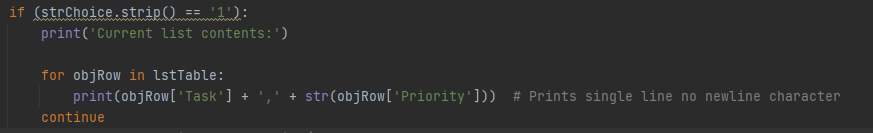


Figure 3. Printing each line of the list to the user.

The next option allowed the user to make a new entry to the list. When this option is selected the user is prompted to enter a task, and the task string is formatted so the first letter is a capital letter and the rest of the string is lowercase. The priority was required to be an integer or else the user would get an error and opportunities to enter an integer until one was entered. When the proper entries were made, the entries were saved to a dictionary structure and then appended to the current list. The user was then notified the task had been saved to the list (Figure 4).

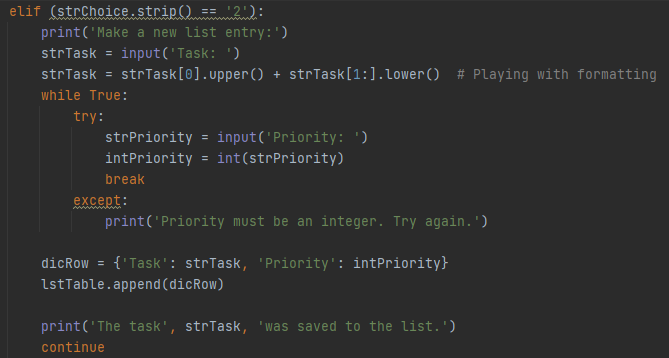


Figure 4. Saving a user entry to the list

The third option allowed the user to remove an entry from the list. The user is prompted to enter the task they would like to remove from the list and the string is converted to lowercase text to be insensitive to how the text was input. In order to determine which row to remove from the list the program went down the list one row at a time looking for a match and removed the line. The index of the row to remove was determined by a counter that incremented each row the task was not found. If the number of iterations matched the length of the list it as determined the item the user wanted to remove could not be located and they were given another try to remove an item. After the task was removed from the list the user was notified.

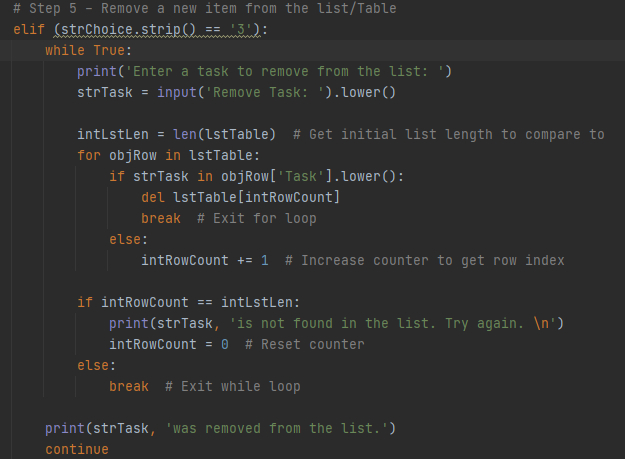


Figure 5. Removing a user entry from the list

The fourth option allowed the user to save the list to a \*.txt file. The text file was opened with the “write” option so that any previous list elements would be erased so the list loaded in at the beginning of the script would not be duplicated. The rows from the list were written to the file by use of a for loop to write each line of the list. Afterward the file is closed and the user is notified what file the information was saved in as well as the path (Figure 6).

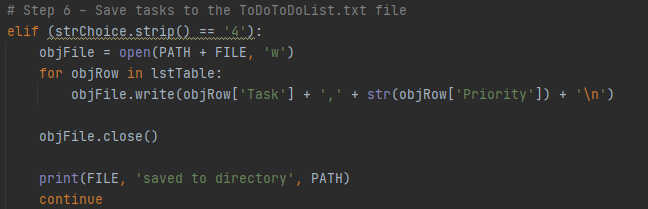


Figure 6. Writing the users list to the file

The final user option is an option to exit the program. This option features notifying the user the program will be exited and breaking out of the while loop.

I ran the program initially in the PyCharm environment to make sure the program would work how I intended to save and remove entries from my list (Figures 7 & 8). The script was a success and my entries were saved to the ToDoList.txt file in the Assignment05 folder (Figure 9).

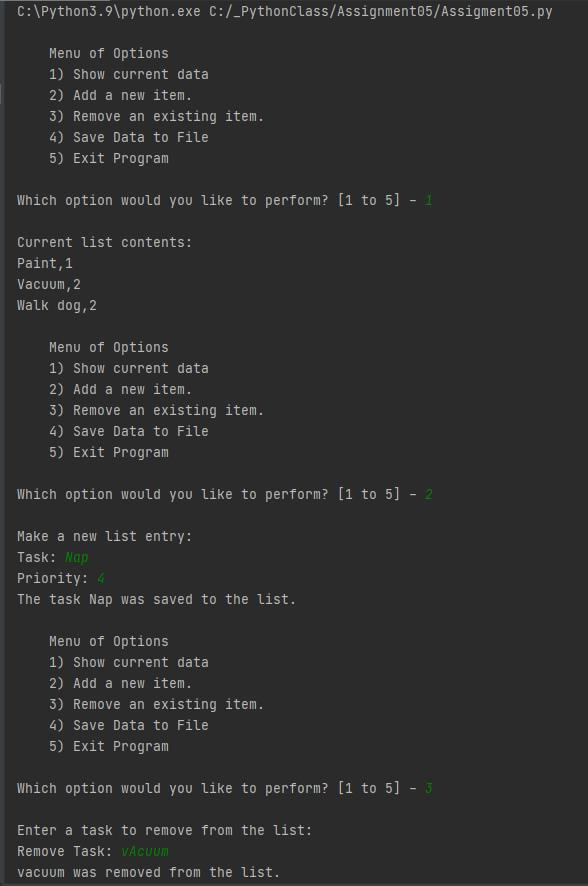


Figure 7. Running the program in Pycharm – Part 1

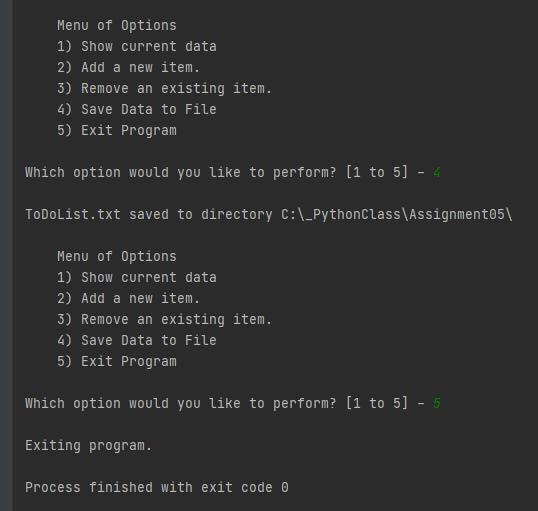


Figure 8. Running the program in Pycarm – Part 2

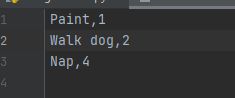


Figure 9. Proof the program actually saved the entry to the file correctly

After running the program successfully in PyCharm I wanted to make sure it would execute correctly from the system terminal. I accessed the terminal through PyCharm and made another inventory list (Figure 10).

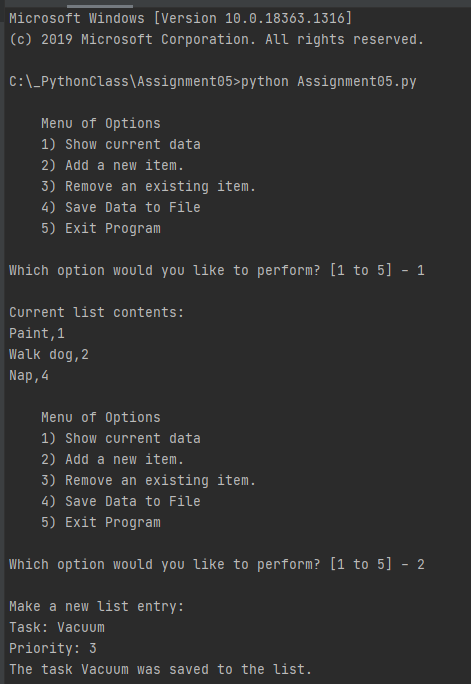


Figure 10. Running the program in the Terminal

# Summary

This module was intended to be an introduction to lists and dictionaries. The biggest difficulty for me in this assignment was figuring out how to get the correct index of the list to remove an item. It seemed this might have been easier if the whole list was a dictionary, but I can also see the advantage of having each row of the list be its own contained dictionary structure. Either way, this module was a good way to get practice accessing data through the various formats!