**Name:** Gail Provancha

**Date:** August 7, 2023

**Course:** IT FDN 110 A Foundations of Programming: Python

**Assignment:** Assignment05

**GitHub:** <https://github.com/provgl1/IntroToProg-Python>

Working with Dictionaries and Files

# Introduction

This document will go over the steps needed to write a Python Script that provides a menu for the user to make selections that can add, delete, and save to a “to do list”. This script will be utilizing lists, dictionaries, while and for loops. Below I will go into more detail regarding how the script is to be written.

# Creating a Header, Declare Variables and Pseudo-Code

To start, you will be adding to a preexisting script and naming it ”ToDoList.py”. You will want to update the header for the script, which will include a title, description and a change log. The header is to provide information regarding the script for individuals, including yourself, on the purpose of the script and when and who made modifications.

The next section is to declare your variables, even though not required since Python creates variables when values are assigned, it still is useful information for future code updates and readability.

And finally, pseudo-code is used to create an outline of the steps needed to write this script. This helps break the script into sections and explains what the programmer is trying to accomplish in easily understandable terms. (Figure 1).



Figure 1: Header, Declare variables, and Pseudo-Code

# Step 1 – Load the Data into a List

You will use the open() function, which requires the name of the file “objFile” and the mode “r” to read the file. The file will already need to exist for the “r” to work.

The “for loop” will be used to go through all the rows of data, splitting the data into a list based on “,”. The rows of data will then be saved to a dictonary “dicRow”and in turn appended to a list “lstTable”. Once the last row of information in the collection is read, the “for loop” will automatically end, which makes it different from a “while loop” which requires a condition to be met.

Since the text file “objFile-ToDoList.txt” needs to already exist, error handling was added using the “Try-Except.” If an error occurs processing the block of code under “try:”, then the process will jump to the “except:” which will print a message to the user (Figure 2).

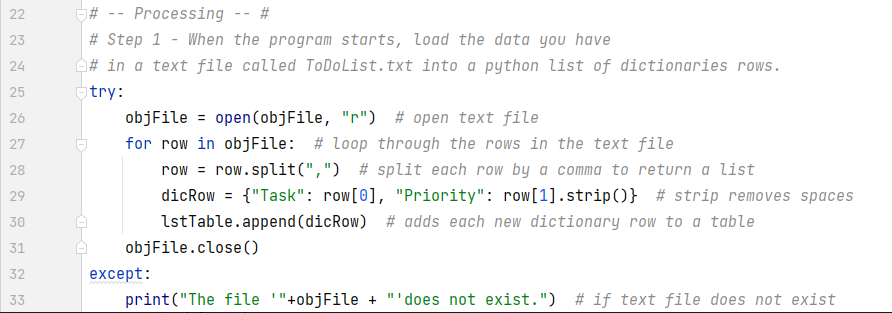


Figure 2: Reading File into List

# Step 2 – Display a Menu of Options

A “while loop” will be used to execute a block of code until the condition becomes “False”. To start, a menu will be printed to show the different choices (Figure 3).

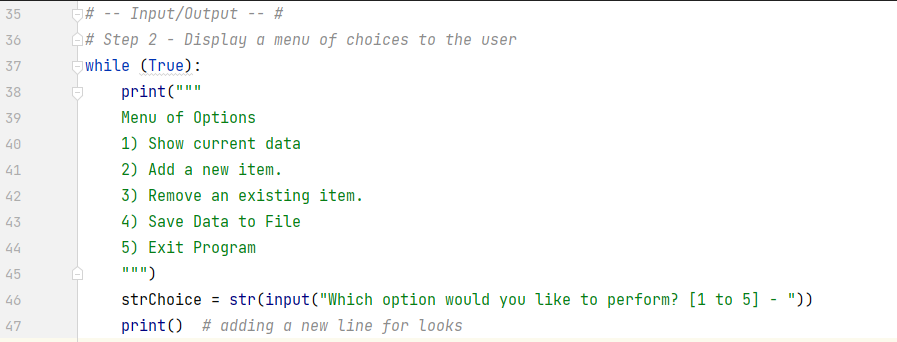


Figure 3: Menu Options

The “if”, “elif”, and “else” commands are sequences that are similar to an “If/Then” statement. Once a condition in the sequence is meet, then that block of code associated with the condition is performed and the rest of the conditions are ignored. The script will continue to loop, providing the menu options, until the user selects option 5, at which point the “while” condition has been meet and the loop stops (Figure 4).



Figure 4: if, elif, sequence

# Step 3 - Menu: Option 1, Show current data

When menu option 1 is selected, the current items in the “lstTable will be displayed. The “for loop” will be used to go through all the rows, printing each Task and Priority in the table (Figure 5) and (Figure 6).

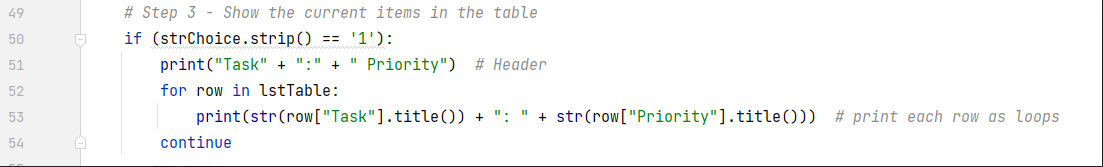


Figure 5: code: display items in table

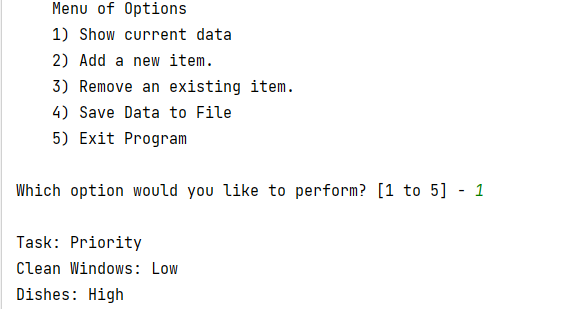


Figure 6: running script in Pycharm

# Step 4 - Menu: Option 2, Add a new item

When menu option 2 is selected, the user will be requested to provide a Task and that Task’s Priority, which will be appended to the Table “lstTable” (Figure 7) and (Figure 8).

# 

Figure 7: code: add new item

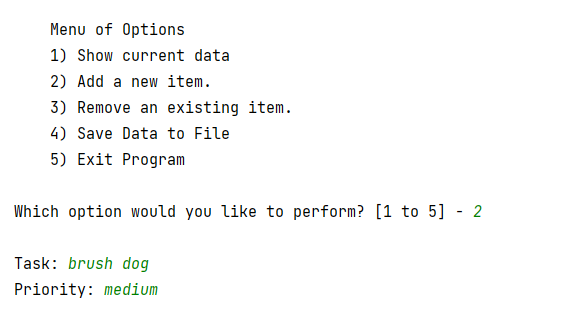


Figure 8: running script in Pycharm

# Step 5 - Menu: Option 3, Remove an existing item

When menu option 3 is selected, the user will be requested to provide a Task, which will be removed from the list. If the task is removed, the user will be notified, otherwise the menu options will be displayed (Figure 9) and (Figure 10).

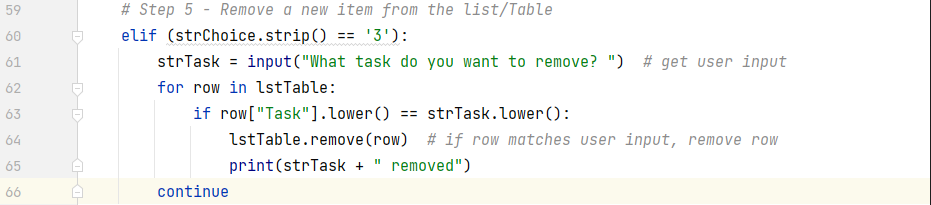


Figure 9: code: delete item

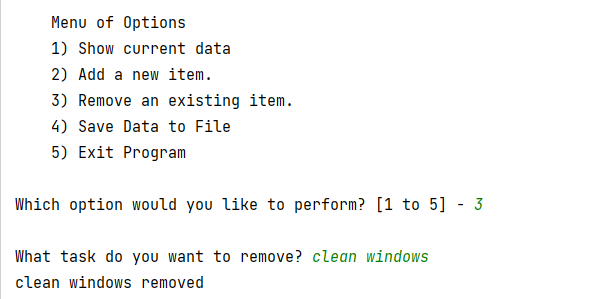


Figure 10: running script in Pycharm

# Step 6 - Menu: Option 4, Save data to File

When the menu option 4 is selected, the items in the “lstTable” will be saved back to the “ToDoList.txt” file. This will involve opening the text file, writing information to that text file, and closing the file. The information will not be saved until the file is closed (Figure 11) and (Figure 12)

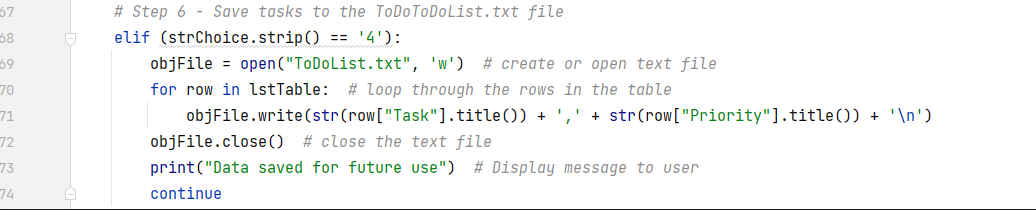


Figure 11: code: save to text file

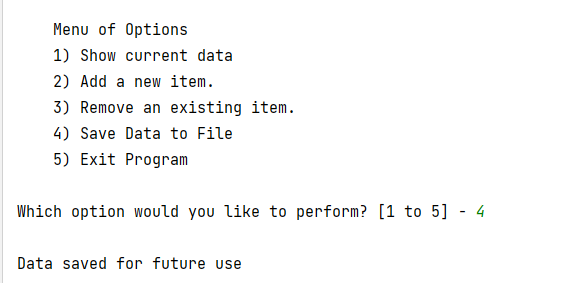


Figure 12: running script in Pycharm

# Step 7 - Menu: Option 5, Exit Program

When the menu option 5 is selected, the script will end. Notice the “break” is used, which causes the “while” loop at the beginning of the script to become “False” and the loop to end (Figure 13).

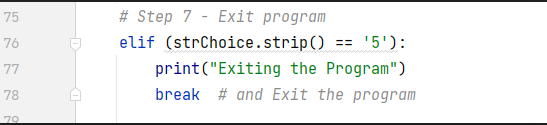


Figure 13: code: Exit the program

# Testing the Script in CMD line

Now the script will be run from Command Prompt to make sure it also runs as expected. Open a command prompt window “CMD”. Type in “cd” and then the pathway to the directory where the python script is saved “C:\\_PythonClass\Module05\Assignment05”. Click Enter. Next type “python” followed by the name of the python script “ToDoList.py”. Click “Enter” again to see the script run. Provide the necessary inputs as if you are the user and verify the result is what you expected (Figure 14).

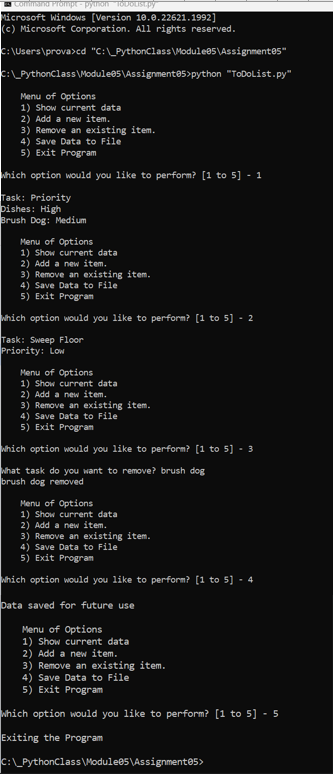


Figure 14: running script in Command Line

Open up the “ToDoList.txt” file to verify the results have changed based on the new inputs (Figure 15).

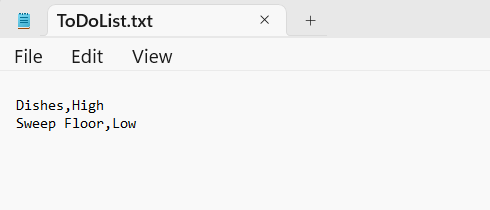


Figure 15: Testing in Command Line, results

# Summary

In summary, this document goes over the steps needed to write a Python script that provides a menu where a user can make different selections to update a to-do-list, including viewing their current list, adding tasks, removing tasks and saving the latest version of their list. This script introducted dictonaries and reinforced learnings around loops, lists, and reading & writing to text files.