Assignment 05: To Do List

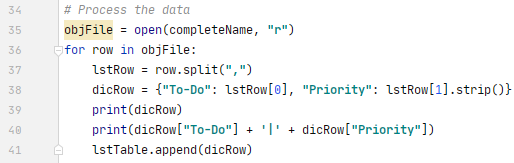
# Introduction

This week, students were introduced to lists and dictionaries and used these to develop a user-inputted To Do list. The input was read from a .txt file and a menu was displayed to the user to view the table, add an item, remove an item, or save and exit the program.

# Solution

# Process the Data

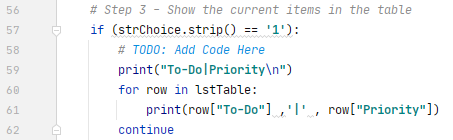
The .txt file was read via an open function in “read” mode, as show below in Figure 1, line #35. For each row within the .txt file, it is split by using the comma character as the delimiter. This split is needed to create the list. The columns for each list (index) are attributed to a key (line 38). At this point, we have created a dictionary from our .txt file and the columns can be called upon by name “To-Do” or “Priority” and we no longer need to reference the column numbers.

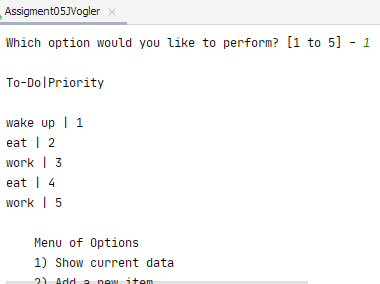


***Figure 1: Processing the data and resulting output***

# Menu Selection #1: View Table

The starter program included the menu code, so the next action was to build what happened when the user selections option #1 – read the table. For each row in the table, we display the dictionary keys and values (Line #60-61). Continue function takes us back to our menu options once the for loop is complete.

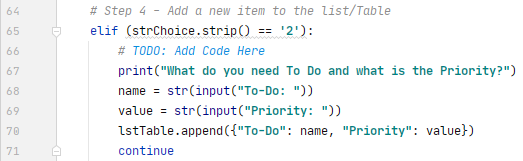




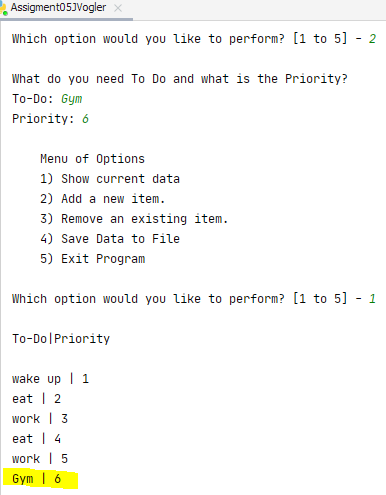
***Figure 2: Show the current table code and resulting output***

# Menu Selection #2: Add Item to Table

Data was added to the existing table by prompting the user for their new “To-Do” item and its “Priority” and storing their input as variables “name” and “value” as shown in lines #67-69. The append function was used on the table and the desired key/value pair (“name” and “value” variables) were added to the end of the list. The braces were used to add the key/value pair to the dictionary, line #70.



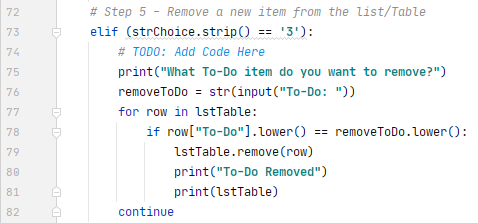
***Figure 3: Adding an item code***



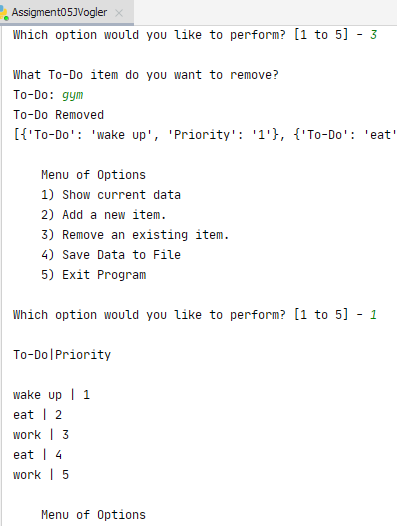
***Figure 3: Added “Gym” with Priority 6 to table***

# Menu Selection #3 – Remove Data

Removal of a table row was accomplished by requesting the user to type in what To-Do item they want removed. This input was stored as “removeToDo” variable and a for loop was performed on the “To-Do” items to find the row which matched “removeToDo”. The lower() function in line 78 of Figure 4 was used to convert all spelling to lower-case, to account for the various capitalization a user man enter (i.e. “GYM”, “gym”, “Gym”). In the example above, “Gym” was entered and we removed it successfully even though “gym” was entered as the desired item to be removed (Figure 5).



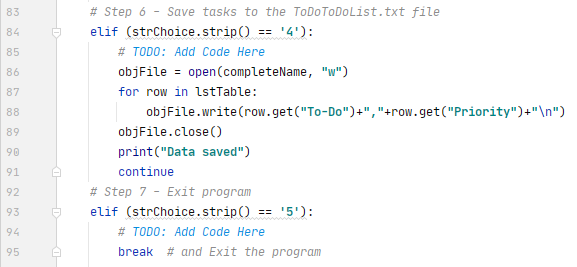
***Figure 4: Remove item code***



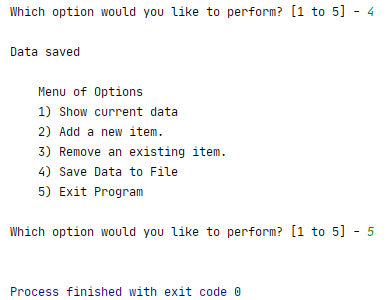
***Figure 5: Removal of To-Do item “Gym”***

# Menu Selection #4 & 5: Save and Exit

Saving the updated table to the .txt file is accomplished by being in write mode, as shown in line #86. For each row of the table, the “To-Do” keys and the “Priority” values are written to the file (separated by a comma and ends with a new line special character), and once each row of the table has been iterated through, the file closes and the write process is complete (lines #86-89). A “Data saved” message is displayed to the user. To exit the project, menu selection 5 calls the ‘break’ statement.



***Figure 6: Save and Exit code***



***Figure 7: Save and Exit output***

# Command Processor and .txt

