Jessi Velazquez

11-17-2020

Foundations of Programming: Python

Assignment 05

**Task Manager Program**

**Introduction**

This document summarizes the task of creating a Python script to manage a user’s constantly changing To-Do list. The program allows the user to view their current list, add items to the list, remove items from the list, and save the list at any time to a master text file to be read In to the program next time it is opened.

**Functionality of the Script**

The first thing that happens when the program is opened is that the text file “ToDoList.txt” is read into the program. The file is opened in read mode, and then is run through a “for” loop, where each row in the text file is split on the comma separator, the first value is assigned to dictionary item “Task”, and the second item is assigned to dictionary item “Priority”. This gives us a list of dictionary items, which are then placed into a list, or table by appending each dictionary item to the table. Finally, the text file is closed. This is because we have read all of the data out of it into memory, so we can manipulate it within the program now without effecting the text file (until we need to, more on this later). The code is as follows:

objFile = open(objFile, "r")  
for row in objFile:  
 lstRow = row.split(",")  
 dicRow = {"Task":lstRow[0],"Priority":lstRow[1].strip()}  
 lstTable.append(dicRow)  
objFile.close()

Next we enter the “while(True)” loop where the user input/output (I/O) portions of the program exist. We display a list of options to the user, take their input, and then move into an if/elif function to perform actions based on user’s choice.

If the user choose option 1, we display the current To-Do list to them in table format. To perform this, we use a “for” loop, where each row in the table is broken out by its dictionary key – either “Task” or “Priority”, and the dictionary values associated with the key are printed, separated by a vertical bar. This forms a rudimentary table, and since the first row in the table is a dictionary item where Task = Task and Priority = Priority, these values serve as headers. The image below displays the output.

![Text

Description automatically generated]()

If the user chooses option 2, we allow them to add a To-Do item to the list. We do so by assigning two user input variables (strTask and strPriority) to a dictionary item, where strTask is assigned as a value to the key “Task” and strPriority is assigned as a value to the key “Priority”. This dictionary item is then appended as a list row to the data table. The code is as follows:

elif (strChoice.strip() == '2'):  
 strTask = input("Enter a Task: ")  
 strPriority = input("Enter its Priority (High/Med/Low): ")  
 dicRow = {"Task": strTask, "Priority": strPriority}  
 lstTable.append(dicRow)  
 continue

If option 3 is selected, that means the user intends to remove an item from the ToDo list. To accomplish this task, we assign variable strRemove to the user input response from the prompt “What task would you like to remove?” The program then uses a “for” loop to search the dictionary value associated with key “Task” in each table row to search for a value that matches the user input. Embedded within the “for” loop is an “if” statement that states if the matching value is found, remove that row in the table. Here is the code displayed:

elif (strChoice.strip() == '3'):  
 strRemove = input("What task would you like to remove? ")  
 for dicRow in lstTable:  
 if strRemove in dicRow["Task"]:  
 lstTable.remove(dicRow)  
 continue

Option 4 is saving the ToDo list to the text file. Thus far in the program, we have read in the existing data and made changes in memory, but the text file on our hard drive has not been changed. Selecting this option will save what we currently have in memory to the text file, writing over the existing data in the text file. Old data that was in the text file and was not removed in memory will have the appearance of remaining in the text file despite being overwritten because we read that data into memory, and we now write our memory back to the file.

To accomplish this, we again open the text file ToDoList.txt, but this time in write mode. Then, we use a “for” loop that says to print just the value in each row of the table separated by a comma. The loop works through each row until there are no rows left, then the text file is closed. At this point, the message “file saved” is displayed to the user, which is the only thing they see during this option. Below is the code for this:

elif (strChoice.strip() == '4'):  
 objFile = open("ToDoList.txt", "w")  
 for dicRow in lstTable:  
 objFile.write(dicRow["Task"] + "," + dicRow["Priority"] + "\n")  
 objFile.close()  
 print("File Saved!")  
 continue

Finally, option 5 is to exit the program. This is an elif statement followed by two lines of code that tell the program to display a “program closed” message to the user before “breaking” the main if/elif statement, which closes the program.

**Summary**

In this document, we have described the process that our script undergoes to perform actions on a user’s To-Do list managed via a text file of comma separated values. The user is given control over adding, removing, and saving the list.