Alison Langer

August 9, 2021

Foundations of Programming: Python

Assignment 05

Github Link: <https://github.com/alilan03/IntroToProg-Python>

Creating a Python Script for a To Do List

## **Introduction**

In this paper I will discuss the process of creating a to do list python script for assignment 05. The assignment requested a python script file that allowed a user to input a list of items to do along with each respective priority. Each pair of data is stored in a dictionary, and all pairs of data are stored in a table (list) and then written to a text file upon the user’s command. This included four main concepts of user input/output, file I/O, lists and dictionaries. This paper will begin with the initial file creation and will follow through to the final completion of the assignment with the functioning code.

### **Creating a Script File**

The first step for this assignment was to create a folder called “Assignment05” in the C: drive of the computer as a subfolder of the “\_PythonClass”. (Fig 1.1)

Graphical user interface, text, application

Description automatically generated

***Fig 1.1 Shows the “Assignment05” folder in the “\_PythonClass” folder***

The next step was to add the “Assignment05\_Starter” Python script to the “Assignment05” folder. This file contains some basic code to begin the project.

The first step I took in writing the code for this assignment was editing the header that was provided in the assignment starter to include my work to update the required sections. (Fig 1.2)

Text

Description automatically generated

***Fig 1.2 The header for the “Assignment05” file***

### **Writing the Python Code**

For the first “to-do” portion, I needed to load all previous data into the program that was already stored in the file. I accomplished this by opening the file object that was provided in read mode. From there I used a for loop to read in each task and priority. I also included an if statement to exclude any blank lines. I stored each piece of data in a dictionary and then added each dictionary to the table. (Fig 2.1)

Text

Description automatically generated

***Fig 2.1 Shows the loop for getting each task and priority from the specified file***

The next “to-do” portion was to display the current items in the table if the user chose that option from the menu. First, I printed a message to the reader to show their selection and then checked if the table was empty. If the table was empty the program would display a message saying so. If the table did have items, then the program would enter a for loop to print out each of the tasks on the to do list along with their respective priorities. (Fig 2.2)

Text

Description automatically generated

***Fig 2.2 Shows the code to display each item in the table***

The next step was to add a new item to the table. I began this section by writing the appropriate input statements to get a new task and priority from the user. From there I stored both pieces of data in a dictionary and then appended the dictionary to the table. (Fig 2.3)

Text

Description automatically generated

***Fig 2.3 The statements responsible for saving a new item to the to do list***

For the next portion, I needed to remove an item from the table. For this, I started with an input statement to get the specified item to remove from the user. Then I used a for loop to check for the matching item in the table. When the requested item was found, it was deleted from the table and a confirmation message was displayed to the user. (Fig 2.4)

Text

Description automatically generated

***Fig 2.4 The statements to remove a requested item from the table***

The next portion was to save all the tasks in the program’s table to the specified file. I began with displaying a status message. After that I reopened the file object, this time in write mode. Then I used a for loop to write each task and priority to the file before displaying a final success message. (Fig 2.5)

Text

Description automatically generated

***Fig 2.5 Shows the code for saving the program’s table to the specified “ToDoList.txt” file***

To complete the final portion of code, I wrote the appropriate print statement to exit the program upon the user’s command. This was completed with a simple input statement. (Fig 2.6)

Text

Description automatically generated

***Fig 2.6 The input statement to exit the program upon the user’s request***

### **Running the Script**

The final portion was to run the Python script in both PyCharm (Fig 3.1 - Fig 3.5) as well as a shell window (Fig 3.6 – Fig. 3.10) and record the running functionality.

Text

Description automatically generated

***Fig 3.1 Menu option 1: displaying the current data***

Text

Description automatically generated

***Fig 3.2 Menu option 2: adding a new item to the table***

Text

Description automatically generated

***Fig 3.3 Menu option 3: removing an item from the table***

Text

Description automatically generated

***Fig 3.4 Menu option 4: saving the program’s table to the specified file***

Text

Description automatically generated

***Fig 3.5 Menu option 5: exiting the program***

Text

Description automatically generated

***Fig 3.6 Menu option 1: displaying the current data***

Graphical user interface, text

Description automatically generated

***Fig 3.7 Menu option 2: adding a new item to the table***

Text

Description automatically generated

***Fig 3.8 Menu option 3: removing an item from the table***

Text

Description automatically generated

***Fig 3.9 Menu option 4: saving the program’s table to the specified file***

Graphical user interface, text

Description automatically generated

***Fig 3.10 Menu option 5: exiting the program***

The last instruction was to verify that the program did write the given information to the specified “ToDoList” text file. This was completed by opening the text file that was created by the program to verify its contents. (Fig 3.3)

Graphical user interface, text, application

Description automatically generated

***Fig 3.11 The contents of the “ToDoList” text file***

### **Summary**

In this paper discussed the process of creating the to do list script in Python for assignment 05. This script asked for the user to input a task and a priority and stored both values as a dictionary in a table (list) until the end of the program where the user could choose to store it in a text file. This program included four main programming concepts of input/output, file I/O, lists and dictionaries. Input is useful for obtaining data from the user to utilize in the program. Output is useful for displaying messages or prompts to the user. File I/O can be used to either read or write to a file. In this case we wrote the user’s input to the file. Dictionaries were new in this assignment, and they were used to create pairs of tasks and priorities. A list was then used to hold all the pairs as a table. Throughout my paper I discussed the steps and logic behind each of my decisions while coding the Python script for this assignment and concluded with the final display of the script running in a shell window.