Alison Langer

August 16, 2021

Foundations of Programming: Python

Assignment 06

Github Link: https://github.com/alilan03/IntroToProg-Python-Mod06

Creating a Python Script for a To Do List: Functions

## **Introduction**

In this paper I will discuss the process of creating a to do list python script for assignment 06. 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 five main concepts of user input/output, file I/O, lists, dictionaries, and functions. 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 “Assignment06” in the C: drive of the computer as a subfolder of the “\_PythonClass”. (Fig 1.1)

Table

Description automatically generated

***Fig 1.1 Shows the “Assignment06” folder in the “\_PythonClass” folder***

The next step was to add the “Assignment06\_Starter” Python script to a new PyCharm project in the “Assignment06” 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 “Assignment06” file***

### **Writing the Python Code**

For the first “to-do” portion, I needed to write a function to add new data to the list based on a given task and priority from the user. I accomplished this by creating a new row (dictionary) in the table with the given data. From there I appended the new row to the table (list) and then sorted the table based on priority. Finally I returned the list and a status update message. (Fig 2.1)

Text

Description automatically generated

***Fig 2.1 Shows the code for adding a new task and priority to the to-do list***

The second “to-do” portion was to remove an item from the to-do list as specified by the user. First, I used a for loop to iterate through the table. Next, I used a condition to check if the current iteration matched the specified task. If it did match, then the item was removed from the list and the list was returned with a success message. If the item was not found, an error message was returned instead. (Fig 2.2)

Text

Description automatically generated

***Fig 2.2 Shows the code to remove a specified item from the table***

The next step was to write the data currently stored in the program to the specified file. I began by opening the file object to write to the file. From there I used a for loop to iterate through each item in the table. Using the loop, I could write each task and corresponding priority to the designated file. Finally, I closed the file object and returned the list with a success message. (Fig 2.3)

Text

Description automatically generated

***Fig 2.3 The statements responsible for writing the table to the specified file***

For the next portion, I needed to get a new task and priority from the user. For this, I started with an input statement to get the specified task from the user. Then I used another input statement to get the priority for the specified task. Finally, I returned both the task and priority given by the user. (Fig 2.4)

Text

Description automatically generated

***Fig 2.4 The statements to get a new task and priority from the user***

The next portion was to get a specified item from the user to remove from the table. For this I used an input statement to get the task from the user before returning the input. (Fig 2.5)

Text

Description automatically generated

***Fig 2.5 Shows the code for getting a task from the user to remove from the list***

To complete the next portion of code, I needed to write the code for menu option 1: Add a New Task. For this I began by calling the IO “input\_new\_task\_and\_priority” function to get a new task and priority from the user. Then I called the Processor function “add\_data\_to\_list” to add the new data pair to the list. (Fig 2.6)

Text

Description automatically generated

***Fig 2.6 The code to add a new task upon the user’s menu request***

To complete the next portion of code, I needed to write the code for menu option 2: Remove an Existing Task. To begin, I called the IO function “input\_task\_to\_remove” to get the specified task to remove from the user. Then I called the Processor function “remove\_data\_from\_list” to remove the specified task from the list. (Fig 2.7)

Text

Description automatically generated

***Fig 2.7 The code to remove an existing task upon the user’s menu request***

To complete the next portion of code, I needed to write the code for menu option 3: Save Data to File. First, the IO function “input\_yes\_no\_choice” was called to see if the user wanted to save the data to the file. If the user entered a “y” for yes, then the Processor function “write\_data\_to\_file” was called to write the list’s data to the file. (Fig 2.8)

Text

Description automatically generated

***Fig 2.8 The code to save the current data to the file upon the user’s menu request***

To complete the final portion of code, I needed to write the code for menu option 4: Reload Data from File. First, a warning message is printed before the IO function “input\_yes\_or\_no\_choice” is called to check the user’s choice about saving the data. If the user enters “y” for yes, then the Processor function “read\_data\_from\_file” is called to load the data from the file into the program. (Fig 2.9)

Text

Description automatically generated

***Fig 2.9 The code to reload the data from the file upon the user’s menu request***

### **Running the Script**

The final portion of the assignment 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. 3.9) and record the running functionality.

Text

Description automatically generated

***Fig 3.1 Menu option 1: adding a new task***

Text

Description automatically generated

***Fig 3.2 Menu option 2: removing an existing task***

Text

Description automatically generated with medium confidence

***Fig 3.3 Menu option 3: saving the data to the file***

Text

Description automatically generated

***Fig 3.4 Menu option 4: reloading the data from the file***

Text

Description automatically generated

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

Text

Description automatically generated

***Fig 3.6 Menu option 1: adding a new task***

Text

Description automatically generated

***Fig 3.7 Menu option 2: removing an existing task***

Text

Description automatically generated

***Fig 3.8 Menu option 3: saving the data to the file***

Text

Description automatically generated

***Fig 3.9 Menu option 4: reloading the data from the file***

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 06. 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 five main programming concepts of input/output, file I/O, lists, dictionaries, and functions. 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 used to create pairs of tasks and priorities. A list was then used to hold all the pairs as a table. The new concept in this project was functions that were used to organize the code into IO and Processor functions as well as reduce redundancy. 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.