Inderjot Brar

February 22, 2021

IT FDN 110 B

Assignment 06

[<GitHub\_url>](https://github.com/IBRAR21/IntroToProg-Python-Mod06)

ASSIGNMENT 6

**Introduction**

This assignment documents the steps I took to create and test a Python Script that uses classes and functions to organize the code that can read data from a text file and allows user to modify the data. The script uses dictionary and list objects to load, store and modify the data from the text files.

## **Creating the Script**

First, I created a ‘ToDoFile’ text file, containing 2 types of data – ‘Task’ and ‘Priority’. In the Assignment06\_Starter.py script, I first added the code for the functions ‘add\_data\_to\_list’, ‘remove\_data\_from\_list’ and ‘write\_data\_to\_file’ under the ‘Processor’ class. (Figure 1)

Graphical user interface, text, application, email

Description automatically generated

***Figure 1: Code added for functions in ‘Processor’ class***

The ‘**add\_data\_to\_list’** function receives *3 parameters* – task, priority and list of rows, creates a *new dictionary row* using task and priority, *appends* it to the list of rows identified by the user, and *returns* the updated list. The ‘**remove\_data\_from\_list’** function receives *2 parameters* – task and list of rows, uses ‘*if’* function within a ‘*for*’ loop to match the user input with the values stored under *Task key* in each row of the list and *remove* that row from the list. An empty variable ‘*count*’ is initialized before running the loop and its value is increased by 1 if a row is removed from the list. ‘*If’* function is then used to customize the status message *returned* to the user depending on the value of count variable. The ‘**write\_data\_to\_file**’ function receives 2 parameters – file name and list of rows, opens the file in *write* mode and uses ‘*for*’ to *write* each row of data from the list table to the file.

Next, I added the code for the functions ‘input\_new\_task\_and\_priority’, and ‘input\_task\_to\_remove’ and ‘write\_data\_to\_file’ under the ‘IO’ (input/output) class. Both the functions used ‘*input*’ function to capture user input of s*tring* type and used *strip* and *lower* methods to format the data returned to the user. (Figure 2)

**Graphical user interface, text, application, email

Description automatically generated**

***Figure 2: Code added for functions in ‘IO’ class***

Lastly, I added code in the main body of the script to call the required functions from the ‘IO’ and ‘Processor’ class for any given user choice. (Figure 3) For example, if the user chooses to add a new task and priority, ‘*input\_new\_task\_and\_priority*’ function of the ‘*IO*’ class is called which will run the code to get user input and then return the data in the form of a tuple which is assigned to the variables ‘*strTask*’ and ‘*strPriority*’. These 2 variables and ‘*lstTable*’ are then passed on the function ‘*add\_data\_to\_list*’ of the ‘*Processor*’ class to update the list of tasks. ‘*strStatus*’ variable is assigned to the 2nd string of the tuple *returned* from the function and displayed to user to inform them of the status of the task.

**Graphical user interface, text, application

Description automatically generated**

***Figure 3: Code added to call functions in the main body of the script***

## **Testing the Script in PyCharm**

After saving the script, I ran the script in PyCharm to test my code. I successfully entered a task and its priority and saved the data in the text file before exiting. (Figure 4)

Text

Description automatically generated with medium confidence

***Figure 4: Script module successfully tested in PyCharm.***

## **Running the Script in OS Command**

Next, I ran the script from the Terminal on my Mac to ensure that it works properly. I was able to successfully test removing a task from the file and reloading the data from file before exiting. (Figure 5)

Text, letter

Description automatically generated

***Figure 5: Script executed from Terminal on Mac OS.***

## **Verifying text file**

Lastly, I located the text file and verified the data was updated properly. In my code, I had used a different format to write the data than the one used in the script and I was able to verify that my formatting had worked as desired. (Figure 6)

Text, chat or text message

Description automatically generated

***Figure 6: Verification of TodoFile.txt file.***

## **Summary**

Using the Module 06 instructions, I was able to successfully create and run a script that uses class and functions to write blocks of code. Using functions is a great way to organize our code for greater readability and to achieve greater efficiency in the main body of the script.

**Link to GitHub repository:** <https://github.com/IBRAR21/IntroToProg-Python-Mod06>