*David Jamieson*

*8/16/20*

*IT FDN 110 A Su 20: Foundations Of Programming: Python*

*Assignment 06*

*GitHub:* <https://github.com/djamies1/-IntroToProg-Python-Mod06>

**Assignment 06 – Documentation**

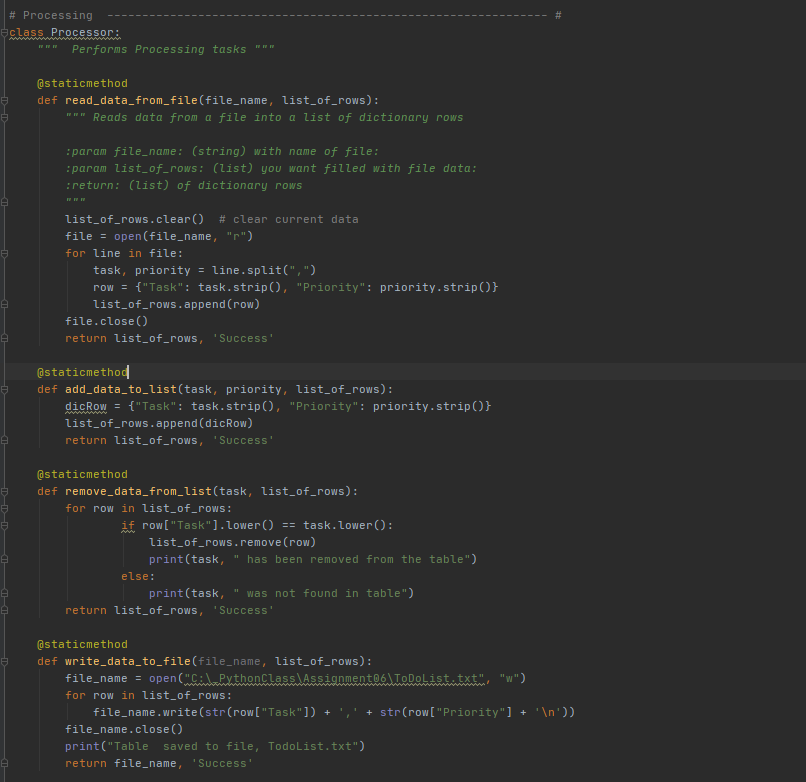
**Summary:**

The purpose of this assignment was to add functions to a preexisting script that could process large volumes of repetitive code more efficiently. This piece of python code loops through a menu of options and allows the user to make a number of edits to a simple table, which stores a task and it’s priority level in a .txt file. Each menu option calls to either a processing data function (based on the class in the python script) or an input/output script which interacts with the user. Some of the user options call from both classes, for example requesting a user’s input to adda new task requires input from the user as well as processing the data into the temporary table. User’s inputs are stored in a temporary table which they can edit while looping through the menu options and they have the ability to wipe the table clean with the text file data or save all their changes to the text file. This is a basic application that shows how python can be used for data management, in a very limited format.

**Scripting:**

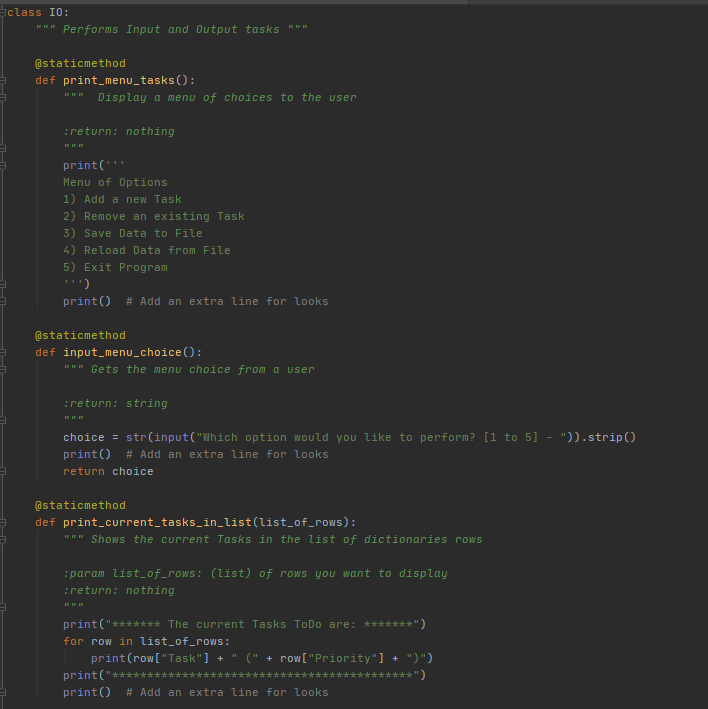
The scripting piece of this assignment was to add every type of input/output and processing data function stored inside of a function. Each function required was listed out and listed as either an input/output function (stored in the IO class) or a processing function (stored in the Processor class). Variables for the text file and various variables needed (likes strings for task, priority, user choice, etc) are defined at the start of the document and called via user inputs in the various functions.

Fig1 below shows the various processing data functions:

Fig1:

* Read\_data\_from\_file – loops through the current .txt file and stores the data into a table for other functions to call against. Also displays current data to user.
* Add\_data\_to\_list – adds use inputs for task or priority to the existing lstTable variable
* Remove\_data\_from\_list- loops through each row in the existing table and matches against the user’s input for task to remove. If it matches, it removes the row, if not it checks the next row.
* Write\_data\_to\_file – this function opens up the existing text file and saves the current tables data to the .txt file.

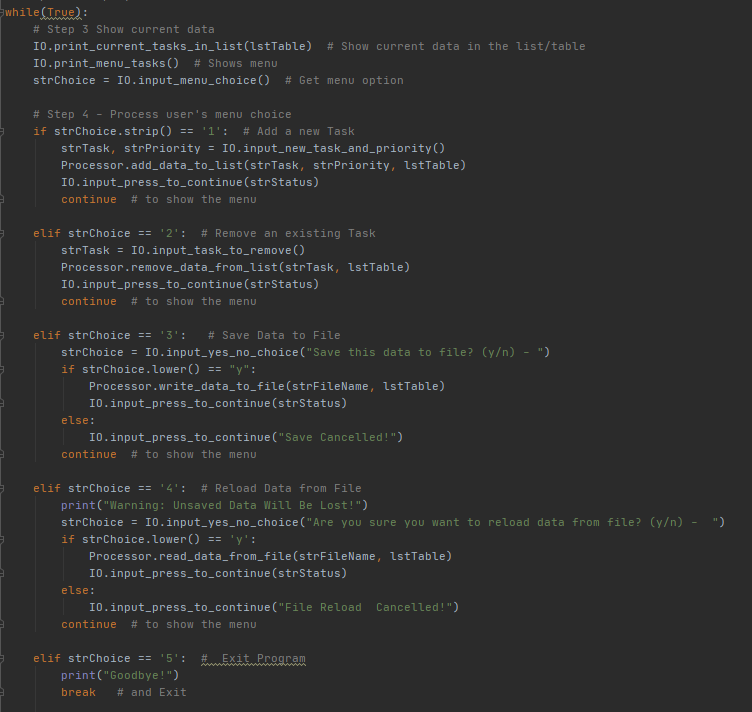
In Fig2 below, you will see the list of input/output functions which are used to request and display user input and output:

Fig2:



* Input\_yes\_no\_choice – this function asks the user if they want to continue with a y or n
* Input\_press\_to\_continue – this makes the user click enter to proceed to next step
* Input\_new\_task\_and\_priority – this prompts the user for a new task or priority to add to table
* Input\_task\_to\_remove – this prompts user for a task to remove from table

The final piece of scripting was to call all these functions to create a functioning application. The code to accomplish this is listed below (Fig 3), calling each function based on the options the user selects:

Fig 3:

**Conclusion:**

This method of scripting with functions takes a bit more effort for a smaller application like this script, however it shows how efficient using functions can be with a larger application. Once the functions have been created, they can be called easily and their code executed very efficiently. This can save a lot of time by eliminating the need to re-write the same piece of code over and over, by instead just calling the function.