Chrissy Henderson

November 20, 2020

IT FDN 110 A Au 20: Foundations Of Programming: Python

Assignment 06

https://github.com/chrissyhend1/IntroToProg-Python-Mod06

# Creating a Multi-Option Task List Script with Functions in Python

## Introduction

In this module, we learned about working with functions and classes and organizing code appropriately for presentation and clarity. Functions, when called on by the program in the main body of the script, were created to process the data and to organize the input. Sectioning off these three steps can help to present that data in a readable way for others looking at the raw code.

## Script Creation

I was asked to edit a script template that loads in data from a text file and places it into a table. The program then gives the user five different menu options to select from which includes viewing the data, adding a new data item, removing an existing item, saving the data to a file, and exiting the program. However, this script includes several processing and input functions that are presented in two separate steps before running the script that selects and runs the functions as needed.

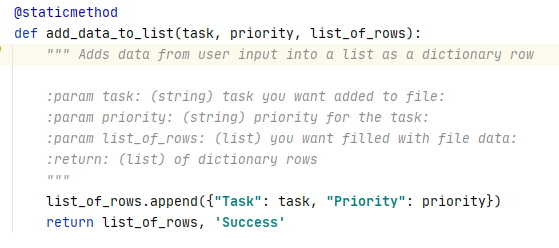
## Data step

Although already written, the first step of the program is to declare variables to be used later on. This way, the data type of the variable is already determined and future data can be easily appended and added to the variables without having to declare them later. This also gives an idea to a future reader of this code what variables are present.

## Processing step

### Function “add\_data\_to\_list”

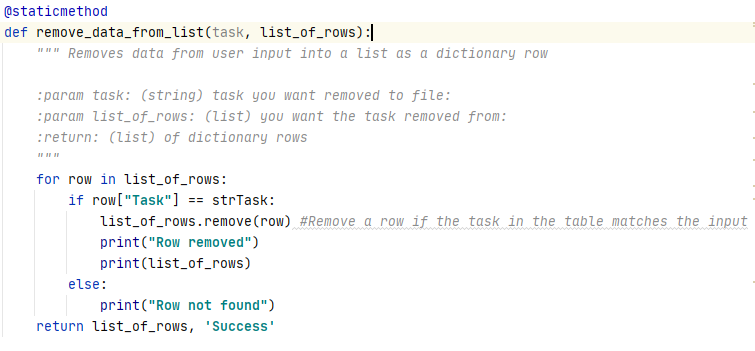
My first step after the variable declaration was to add code to the function that will write data to a list (Figure 1). When run, the function is going to look for a variable declared as “task”, “priority”, and “list\_of\_rows”. With those three variables, the program will append a dictionary row of the “task” and “priority” variables to the “list\_of\_rows” variable and return the “list\_of\_rows” to the program on a successful run. When it returns a variable, it does not print it to the user, but just keeps that variable for the program to call on in whatever the next step is. Additionally, I added some parameter descriptors at the start of the function to help the user figure out what each variable was.



***Figure 1. The function that will add a task and priority to a list variable as a dictionary row***

### Function “remove\_data\_from\_list”

I next added code to the function that would remove data from the called on list (Figure 2). When run, the function will look for the variable “task” and “list\_of\_rows”. The program takes the user input of “strTask” and then a loop will begin for each of the rows in the “1stTable”. Because this function needs user input, the program will need to ask for this before running this particular function. If the loop hits a task in the dictionary that matches the input from the user, the row will be removed and the loop will print a statement saying the row has been removed and then will print the whole table without that row. If it does not find it, the loop will print that the row has not been found. I added some parameter descriptors at the start of the function to help the user define the variables as well.

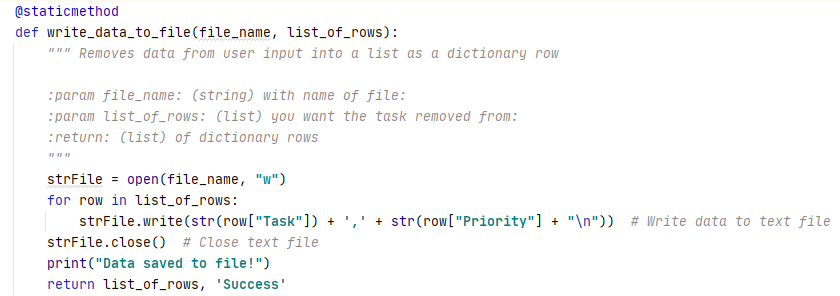


***Figure 2. The function that will remove a row from a list using user input***

### Function “write\_data\_to\_file”

The next function to add to was the one that would write data to a file when called on. The function requires a file name and the “list\_of\_rows” that the user wants written to the file (Figure 3). When run, the function will open the file that has the name of the variable “file\_name” in write mode, will format the “list\_of\_rows” data, and will then write it to the file in the file variable created earlier. If the file does not exist, the program will create it when the file is opened.

After this, the file is closed, and the program prints “Data saved to file!” to let the user know that the data was saved. I made sure to put this line after the “write” and “close” lines so that if something goes wrong with writing the file, the user does not see the printed statement before this error. The function returns the “list\_of\_rows” on a successful run.

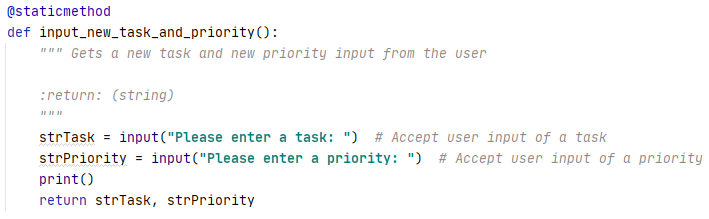


***Figure 3. The function that will write data to a file when called on***

## Presentation step

### Function “input\_new\_task\_and\_priority”

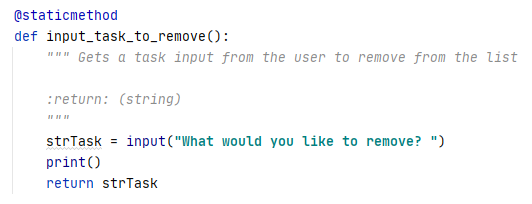
This function is located in the presentation step which includes asking for input and output. Bases on the function’s name, I deduced that the function was meant to ask the user to input both a new task and new priority. I set up two lines of code to assign the variables “strTask” and “strPriority” (which were set up in the data declaration step) to the input from the user (Figure 4). The program would display a message to the user to enter in a task and assign it to “strTask” and then to enter a priority and assign that to “str\_Priority”. The function would print an empty line for looks and then return “strTask” and “strPriority” to the program.



***Figure 4. The function that will gather user input of a task and priority***

### Function “input\_task\_to\_remove”

This function is similr to the “input\_new\_task\_and\_priority” function in that it asks for input from the user but this is for just a task to remove. This does not include the priority. The function will take the input from the user, print a blank line, and return the input to the program to be used later (Figure 5).

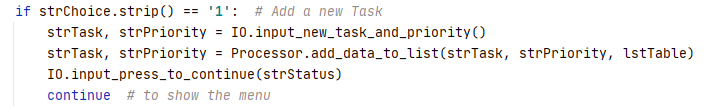


***Figure 5. The function that will remove a task from a list using input from a user***

## Main Body of Script

### Processing user menu choice 1

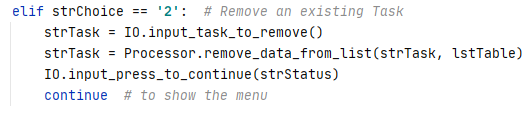
Based on the menu, when the user enters in “1”, the program should add a new task to the list using the functions I created and the ones already present in the processing and presentation steps. First, we would need input from the user so I pulled in the function “input\_new\_task\_and\_priority” in the class “IO” (Figure 6). The function returns two variables to the program (strTask and strPriority) which I assigned the names of strTask and strPriority. Next, I called on the function “add\_data\_to\_list” in the class “Processor” which needs three input variables. I told it to use the two variables we just had returned from the first function and the table name “lstTable” which will be called “list\_of\_rows” in the function. At the end, the “lstTable” will be returned from the function and data will have been added per the function’s job. The program will then continue and loop back to the menu.



***Figure 6. When a user enters in “1” as their choice, the program will run two functions to add data to a list***

### Processing user menu choice 2

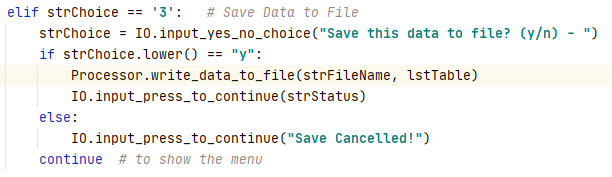
Based on the menu, when the user enters in “2”, the program should remove a task from the list using the functions I created and the ones already present in the processing and presentation steps. First, we would need input from the user so I pulled in the function “input\_ task\_to remove” in the class “IO” (Figure 7). The function returns one variables to the program (strTask) which I assigned the name of strTask. Next, I called on the function “remove\_data\_from\_list” in the class “Processor” which needs two input variables. I told it to use the one variables we just had returned from the first function and the table name “lstTable” which will be called “list\_of\_rows” in the function. At the end, the “lstTable” will be returned from the function and data will have been removed per the function’s job. The program will prompt the user to hit the carriage return and will then continue and loop back to the menu.



***Figure 7. When a user enters in “2” as their choice, the program will run two functions to remove data from a list***

### Processing user menu choice 3

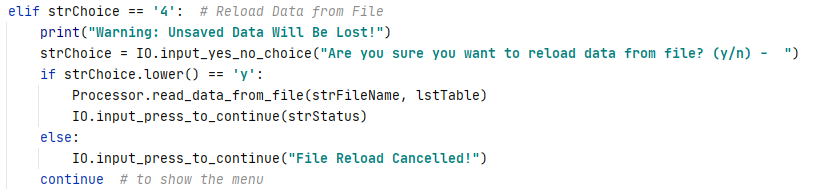
Based on the menu, when the user enters in “3”, the program will save the data using the functions I created and the ones already present in the processing and presentation steps. First, we would need input from the user of a “y” or “n” if the user wants to save their data using the “input\_yes\_or\_no\_choice” function in the class “IO” (Figure 8). This input would be assigned the name of “strChoice” and if this variable was “n” the save would be cancelled. If “strChoice” = “y”, the function will use the “strFileName” variable created in the data validation step in the “write\_data\_to\_file” function in the “processor” class using the “file\_name” and “lstTable” variables. The program will prompt the user to hit the carriage return and will then continue and loop back to the menu.



***Figure 8. Based on user input, the program will either save the file or cancel the save before looping back to the menu.***

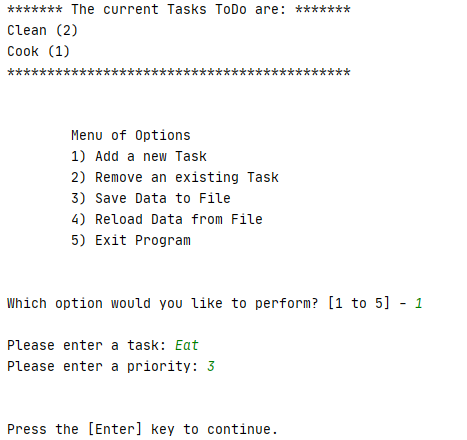
### Processing user menu choice 4

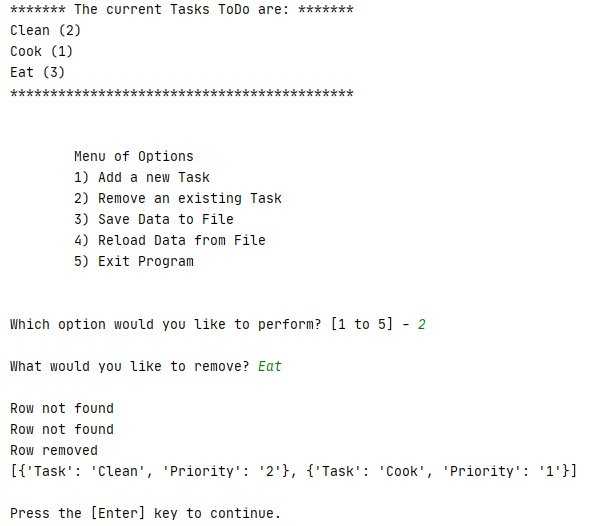
Based on the menu, when the user enters in “4”, the program will reload the data using the functions I created and the ones already present in the processing and presentation steps. First, the program will print a warning that unsaved data will be lost to warn the user. Next, the program will need input from the user of a “y” or “n” if the user wants to reload data using the “input\_yes\_or\_no\_choice” function in the class “IO” (Figure 9). This input would be assigned the name of “strChoice” and if this variable was “n” the reload would be cancelled. If “strChoice” = “y”, the function will use the “strFileName” variable created in the data validation step in the “read\_data\_from\_file” function in the “processor” class using the “file\_name” and “lstTable” variables. The program will prompt the user to hit the carriage return and will then continue and loop back to the menu.

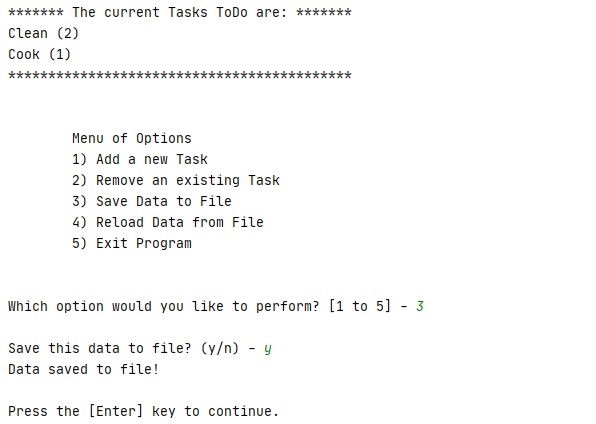


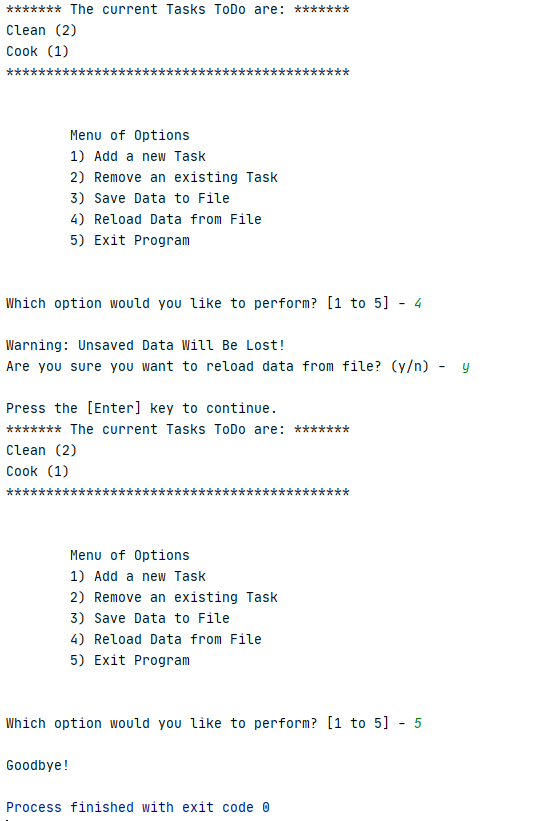
***Figure 9. The step that will reload data if the user enters in an input of “y” using two functions***

The task list program was successfully executed in PyCharm IDE and in the Windows command line, respectively (Figure 10 and 11).

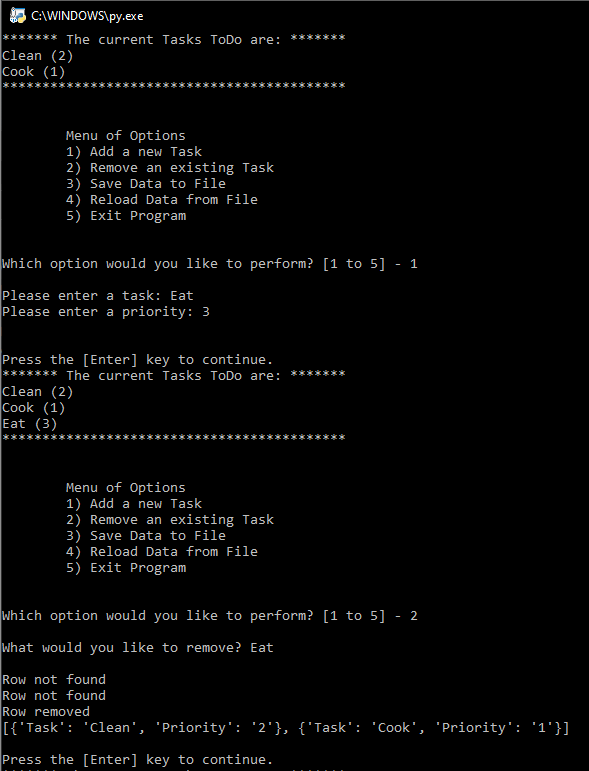


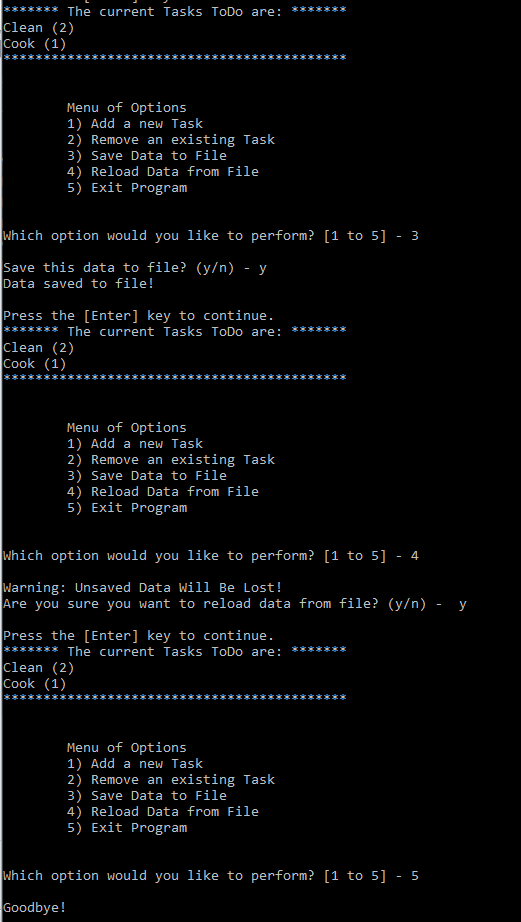






***Figure 10. The program working in the PyCharm IDE***





***Figure 11. The program working in the command window***

## Summary

In this particular assignment, I added code to a program to create a text file of tasks and priorities of these tasks, save data, and reload data. The program prints instructions with five different options. The five options allow the user to append task and priority data to a table, remove data from the table, writes the information to a text file, reload data from a file, and finally exit the program. Creating and writing to a text file is a great way to collect and store user information and can be beneficial for streamlining data entry and preventing users from accidentally changing past data. Separating functions into multiple steps is very helpful for others to come in a read the code and understand it as well.