

Tyler Tidd
2/19/21
IT FDN 110 B Wi 21
Assignment06
<https://github.com/propatronage/IntroToProg-Python-Mod06>

To Do List with Functions

Introduction

This week's assignment was similar to last week's in that we are working with a modified version of our to-do list application. This week we learned about functions and their various uses and components. For the assignment, we had to add functions to "starting code" we were given to make the to-do list work. As usual, we had to run our code worked in PyCharm and again in the Terminal. Then we had to show that everything saved appropriately in the text file.

Body

The Set Up

We were given most the code to start with. There were sections that were blank that we needed to fill in with functions. Functions are a way to group one or more statements. You start a function with "def" and indented below you have the statement that runs when the function is called. The starting code was broken into sections using Separation of Concerns. Using functions makes doing SoC much easier. You can write the processing part of the code in one section then call it from another. In the first section "Data" I saw all the variables and constants were declared. I'd need to refer to them later when I start filling out the functions.

The next section "Processing" started with a class filled with functions. A class is a way to group functions, constants, and variables. This class called "Processor" had one completed function that read data from the text file and several other's that I needed to complete. The completed one included a "docstring." A docstring is a comment describing what the function does. It's common practice to use a docstring inside the function. It's useful for other programmers who may have to modify your code in the future. You can also call a function's docstring from the terminal to describe what the function does.

Writing the code

There were 3 functions in the "Processing" class that I had to fill out. These functions would enable the user to add a task, remove a task, and save data to the text file. Since these functions operated in nearly the exact same way as last week's code I simply copied the code from last week and pasted them in their corresponding function. However, to get the code to work in the function I had to make a few changes. One change was to edit the name of the variables inside the function. Note that the variables inside the function are considered "local" meaning they can only be used inside the function. I also had to make sure that the values inside the function matched the pre-defined parameter names. Parameters allow you to pass variables into the function for processing. For these functions we used the parameters: task, priority, list_of_rows, and file_name. We could have created default parameter values by setting them = "something" but that wasn't appropriate in this case. Finally the last piece of the functions was to return the list_of_rows. The return statement ends the function and passes the results to the next part of the program.

The next segment of code was the “Presentation.” Here’s where all the inputs and outputs would take place. The predefined code started with a class called “IO.” The functions in this class would display the menu and do the various requests from the user processing around that action. The only addition I had to make to this section was to request the relevant data from the user for the adding a task and removing a task functions. The functions were predefined with the “pass” argument which tells the program to skip this section so the programmer can come back and complete it later. All I had to do was simple: just request the task and priority from the user then return their responses.

The last section was the main body of the script where all the functions that I created or were pre-populated would be called. Here I had to call the appropriate function to add or remove a task, save data to a file, or reload a file. For the first two functions, adding and removing tasks, I had to call the functions that requested inputs from the users. I stored the returned results of these functions in variables then used those variables as arguments in the next functions that would process the data. Arguments are variables passed into the parameters of functions. Then I called the processing functions. Once that was done, the script was complete

Verifying the code

The next step was to verify the code by running it in PyCharm and the Terminal.

First thing I checked was the adding a task option (Figure 1)

```
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
task4 (4)
task5 (5)
*****

Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Reload Data from File
5) Exit Program

Which option would you like to perform? [1 to 5] - 1

What is the new task name?: Task6
What is the new task priority?: 6

Press the [Enter] key to continue.
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
task4 (4)
task5 (5)
Task6 (6)
*****
```

(Figure 1: Adding a task)

Then I checked the removing a task option (Figure 2)

```
Which option would you like to perform? [1 to 5] - 2

Which item do you want to remove?: task4
task4 deleted

Press the [Enter] key to continue.
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
task5 (5)
Task6 (6)
*****
```

(Figure 2: Deleting a task)

Then I checked saving the data to a file (Figure 3)

```
Which option would you like to perform? [1 to 5] - 3

Save this data to file? (y/n) - y
```

(Figure 3: Saving data to file)

Then I checked reloading the data from file and exiting the program (Figure 4)

```
Which option would you like to perform? [1 to 5] - 4

Warning: Unsaved Data Will Be Lost!
Are you sure you want to reload data from file? (y/n) - y

Press the [Enter] key to continue.
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
task5 (5)
Task6 (6)
*****

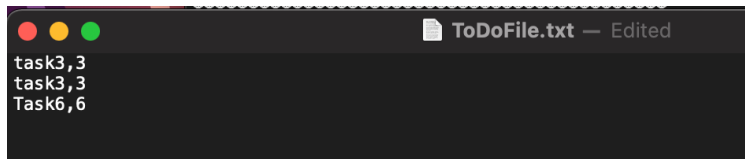
Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Reload Data from File
5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Goodbye!
```

(Figure 4: Reload and exit)

Lastly, I had to check to see if the changes were saved in the text file (Figure 5)



(Figure 5: PyCharm saved changes)

Everything worked as expected in PyCharm. Next I had to check if it works in the Terminal. First thing I did here was to open the program and add a task (Figure 6)

```
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
task5 (5)
Task6 (6)
*****

      Menu of Options
      1) Add a new Task
      2) Remove an existing Task
      3) Save Data to File
      4) Reload Data from File
      5) Exit Program

Which option would you like to perform? [1 to 5] - 1

What is the new task name?: task7
What is the new task priority?: 7

Press the [Enter] key to continue.
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
task5 (5)
Task6 (6)
task7 (7)
*****
```

(Figure 6: Opening and adding a task in the Terminal)

Next I checked the remove task and save file (Figure 7)

```
Which option would you like to perform? [1 to 5] - 2

Which item do you want to remove?: task5
task5 deleted

Press the [Enter] key to continue.
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
Task6 (6)
task7 (7)
*****

Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Reload Data from File
5) Exit Program

Which option would you like to perform? [1 to 5] - 3

Save this data to file? (y/n) - y

Press the [Enter] key to continue.
```

(Figure 7: Add task and save file in Terminal)

I reloaded the data and exited the program (Figure 8)

```
Which option would you like to perform? [1 to 5] - 4

Warning: Unsaved Data Will Be Lost!
Are you sure you want to reload data from file? (y/n) - y

Press the [Enter] key to continue.
***** The current Tasks ToDo are: *****
task3 (3)
task3 (3)
Task6 (6)
task7 (7)
*****

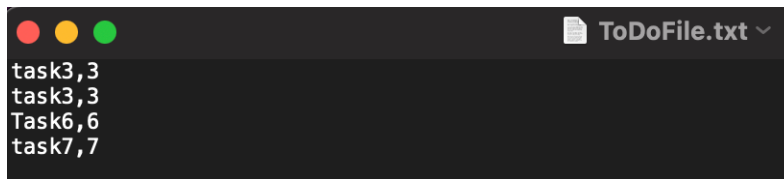
Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Reload Data from File
5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Goodbye!
```

(Figure 8: Reload data and exit the Terminal)

Finally, I checked the text file to make sure everything was saved. (Figure 9)

A terminal window with a dark background. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left and a file icon followed by the text 'ToDoFile.txt' and a dropdown arrow on the right. The terminal area contains four lines of text: 'task3,3', 'task3,3', 'Task6,6', and 'task7,7'.

```
task3,3
task3,3
Task6,6
task7,7
```

(Figure 9: Verify save in Terminal)

Conclusion

This weeks assignment was essentially to reproduce last week's assignment but by using functions. We were given a template to start with and our job was to fill out the blank functions. In this document I covered what functions are and their parts, and how I used them to build this program. I also covered how functions facilitate programming best practices. After covering how I wrote the program I showed it working by running it in PyCharm then again in the Terminal. Lastly I showed the results in the the text file.