Lauren Parker

November 13, 2020

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

Assignment 06

Adding Functions to a Script

# Introduction

This paper will discuss the steps it took to adjust a Python script that was provided by our professor, to create an ongoing to-do list. The script provided a menu of interactive options, such as adding a new task to the to-do list, removing an existing task from the to-do list, saving the data to a file, reloading the data from the file, and exiting the program. Our job was to create and organize the functions in the script to keep the separation of concerns clean.

# Step 1: Importing Starter Script & Reviewing Code

Our professor provided us with a starter code for this assignment. The first thing I did was create a new file in PyCharm and import Professor Root’s starter code. I made sure to update the header with my name and date. Next, I reviewed Professor Root’s code to get an idea of what the code did and what needed to be added.

# Step 2: Creating Functions for Processing

After reviewing the code, it was evident that there were 3 functions that needed additional code in the Processing section. The functions were add\_data\_to\_list, remove\_data\_from\_list, and write\_data\_to\_file. You can see an example of the empty functions in Figure 1.

Text

Description automatically generated

***Figure 1: Empty functions***

The first function, add\_data\_to\_list, would need a doc string and code to add a task and priority level to a dictionary and the dictionary to a list. I added the function header by using three double quotes and adding a short description of what the function does, the definition of each parameter, and what the function returns. I closed the block of text with an additional three quotation marks. The code I used to add a task and priority to the list of dictionaries was from our previous assignment05. You can see the code I used for all three functions in figures 2 and 3.

Text

Description automatically generated

***Figure 2: Showing the function headers and code for add\_data\_to\_list and remove\_data\_from\_list***

***Text

Description automatically generated***

***Figure 3: Showing the function headers and code for write\_data\_to\_file***

# Step 3: Creating Functions for Presentation

Most of the code in the presentation section was already filled in, with the exception to 2 function, input\_new\_task\_and\_priority and input\_task\_to\_remove. You can see the empty functions in figure 4. I added doc string headers for each function and pulled the code from my previous assignment05 since I knew it worked. Once I had copied it to the file, I would run the code in PyCharm to make sure it worked and didn’t pose an error message. You can see an example of my function headers and code in figure 5.

Graphical user interface, text, application

Description automatically generated

***Figure 4: Empty functions in the presentation section of the program***

***Text

Description automatically generated***

***Figure 5: Function headers and code added to input\_new\_task\_and\_priority and input\_task\_to\_remove in the presentation section of the program***

# Step 4: Calling Functions in Main Body of Script

The next portion of code would be used to call the previously defined functions in order to make the menu run correctly. The starter code provided by Professor Root, had some blanks that needed to be filled in. He denoted these with the “#TODO: Add Code” Here comment. You can see an example of this on lines 145, 150, 157 and 167 in figure 6.

Text

Description automatically generated

***Figure 6: Starter code with comments denoting where code needed to be added***

I called the previously defined functions for each TODO section. In order to do this, I had to list the class name, followed by a period, and the name of the function. If there were parameters within the function, I added the appropriate arguments. You can see an example of this in figure 7.

***Text

Description automatically generated***

***Figure 7: Writing the data to a .txt file and ending the program.***

After adding all of my functions I made sure to run my code in PyCharm and a Shell window to make sure that it ran correctly.

# Summary

In conclusion, we have discussed how to use functions to keep code organized. We have seen how important it is to break your code into sections following the separation of concerns. We have also discussed how to call our functions in the main body of our script.