Gabriela Tedeschi

8/15/2021

IT FDN 110

Assignment 06

# Functions and Classes

## Introduction

In this document, I will describe the steps I took to finish a script that creates the To List program using functions. This program uses functions and classes to separate concerns and divide the script into data, processing, input-output, and main body sections.

## Processing

To start, I created a new script, Assignment06\_GTedeschi, for completing the assignment to leave the starter unedited for reference. Then, I started working on code to complete the functions in the Processor class, which serve to process the data in different ways. I reviewed the already completed read\_data\_from\_file function so I would know how to call the function properly in the main body of the script.

For the function add\_data\_to\_list, I created a doc string that explains that the function allows you to add data to a list and outlines the parameter and what the function returns. Then, I wrote code to create a dictionary using the function’s task and priority parameters and to append the dictionary to a list.

Text

Description automatically generated

**Figure 1**: Defining add\_data\_to\_list function

For the function remove\_data\_from\_list, after creating a doc string, I created a for loop. The for loop gives the command to look at each element in a list and if the argument passed into the task parameter matches the task in that element, to remove that element from the list.

Text

Description automatically generated

**Figure 2**: Defining remove\_data\_from\_list function

For the write\_data\_to\_file function, I use the file variable to open a file and write to it. The “w” means that the existing data will be overwritten. I then use a for loop to write the task and priority from each element in the list as a new line in the file. Then, I close the file.

Text

Description automatically generated

**Figure 3**: Defining write\_data\_to\_file function

## Input/Output

The IO class contains functions that allow the program users to input data and display data back to them. After reviewing the already completed functions, I began working on the code for defining the input\_new\_task\_and\_priority function. I use two input functions to allow the user to enter a task and its priority level. I use a while loop with conditional statements to ensure that the user enters low, medium, or high for the priority level. The starter file already included a line so that the function returns the data entered.

For the input\_task\_to\_remove function, I simply use an input function so the user can input the task they want to remove. A line to return the task entered was already included.

Text

Description automatically generated

**Figure 4**: Defining input\_new\_task\_and\_priority and input\_task\_to\_remove

## Main Body

After reviewing the already complete code for steps 1 – 3, I started on option 1 in the while loop, which allows users to add a new task to their to-do list. First, I captured the output (which is the user’s input) from the IO function into a tuple made up of two variables, strTask and strPriority. Then, I passed those variables and lstTable as arguments into the Processor function. This adds the user’s input to the list. I added a line to print a confirmation message to the user.

The code for option 2 is similar to option 1. I start by capturing the output from the IO function (the user’s input) into the variable strTask. I then pass this variable and lstTable into the Processor function, which identifies the dictionary in lstTable with the matching task and removes it from the list. Again, I use a print function to let the user know what happened.

Text

Description automatically generated

**Figure 5**: Code for add new task and remove task from list options

In the option 3 section, I pass the variable that holds the file name and lstTable as arguments into the Processor function. This writes the contents of the list to the file. Again, I added a message for the user.

For option 4, I used the same process, passing the file and list variables into the read from file Processor function. This reloads the current contents of the file into the list, overwriting whatever is currently in the list. Then, I use a print function to print a message to the user again.

After option 5, I added an else clause so that if the user enters anything besides 1, 2, 3, 4, or 5, they receive a message telling them to choose a valid option. The continue statement brings them back the beginning of the loop.

Text

Description automatically generated

**Figure 6**: Code for save data to file, reload data from file, and exit options

## Running The Program

Please see the images below of the program running in the command shell and PyCharm.

**Command Shell**

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

Text

Description automatically generated

Text

Description automatically generated

**PyCharm**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

## Summary

In this document, I described the steps I took to work with a partially complete script and create a to-do list program that allows users to add data, remove data, save data to a file, and read data from a file. By using functions, I was able to separate the processing and presentation tasks into classes and simply fit these functions together like puzzle pieces in the main body of the script. This made writing the main body simpler and prevented me from having to rewrite or copy and paste longer blocks of code in multiple places.