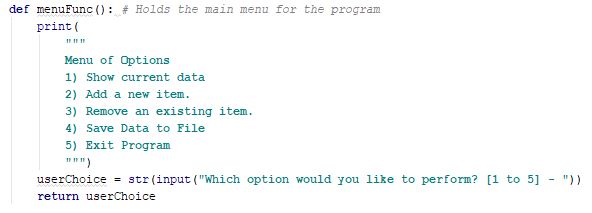
Introduction

Assignment 6 builds off assignment 5. Taking the script from last week we modified it to be in classes, functions, and to split the script into legible sections of Functions, Variables, and Presentation. This code was challenging because it took our one dimensional, top-to-bottom read script and turned it into a dymanic script that is better suited for legibility, expansion, and sharing.

# Functions

Functions are defined sections of code that are encapsulated by a term. They are recognized but the open/close brackets, (), at the end. In some cases there are things between the brackets and other times there is nothing. Custom functions are created by using the tag *def* in front of the to-be defined function. In Figure 1 you see I have created *menuFunc()* which will hold the main menu for the assignment script. Everything below and indented is included. There is nothing between the brackets which means it doesn’t take information from the main script. However, it does *return* something, it returns *userChoice* as a variable data that is used outside the function later on.



**Figure 1 show a customer function I defined. This function holds the main menu for the assignment script.**

# Classes and Methods

Classes are a way of grouping function, variable, and constants. A class makes methods out of function contained within. They’re called by typing the class, then ‘.’, then function as seen in Figure 2.



**Figure 2. Shows menuClass.showTasksFunc() class and method pair being called in the menu of the assignment script.**

We’ve seen methods before, such as the *.strip()* seen in figure 1. *.strip()* is a built in method however and isn’t one that I defined.

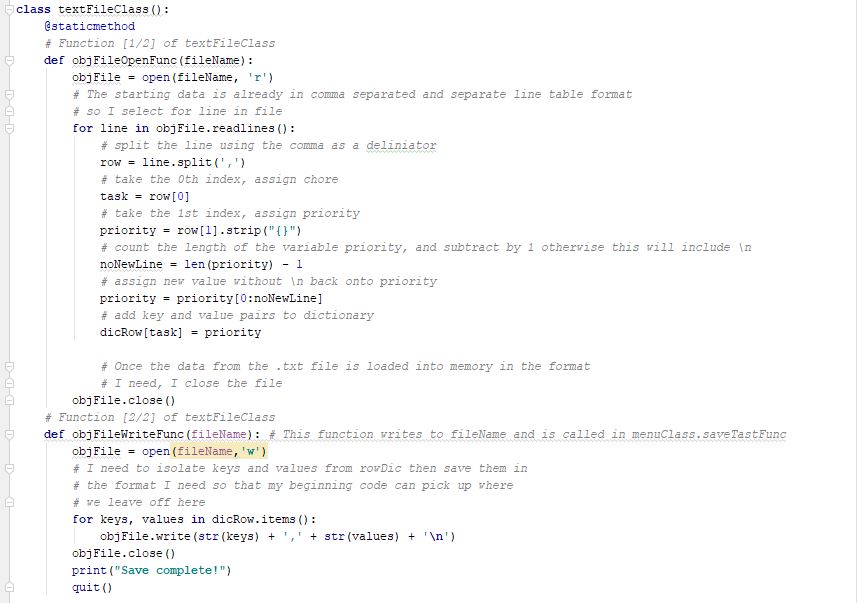
Todo.pyt, Revisisted

Armed with these new tools I revised last week’s script called Todo.py to be divided into functions.

I created two classes. One to hold the functions that manage the menu called *menuClass()* (Figure 3) and one to manage the opening, saving, and closing of the .txt file called *textFileClass()*(Figure 4).

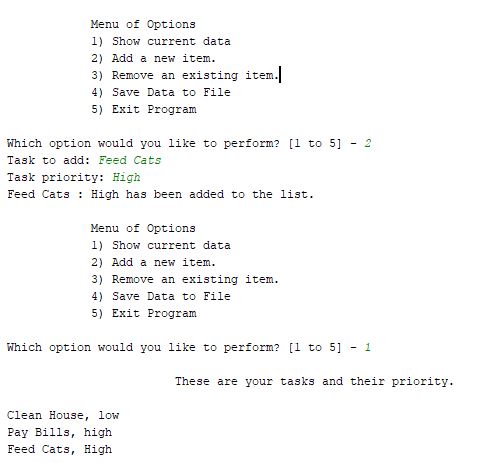


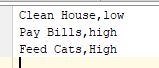
**Figure 3: Shows the entire menuClass and the 5 function contained within.**



**Figure 4: textFileClass() has two functions contained within it.**

Figures 5 show the program running PyCharm and the user selecting to add a task then repeating the task list back. Figure 6 shows that the .txt file is being updated as desired.

  
**Figure 5: The user first select *2* prompting her to input a new task and priority. In this case “Feed Cats” and “High”. The user then selects *1* and the program repeats back the current list.**

  
**Figure 6: The Todo.txt file does indeed update with the new data.**

Conclusion

Functions, classes, and methods have allowed me to take last week script and make it modular. Each function is easy to understand and can easily be used in other applications. Although this is a small example, functions are critical in larger programs not just to hold more complex code, but also for legibility and a reader’s comprehension.