Anna Paulson

May 23, 2020

Foundations of Programming: Python

Assignment 06

<https://github.com/ampaulson/IntroToProg-Python-Mod06>

Using Classes in Python to Prioritize Tasks

# Introduction

In this paper I will discuss the steps I took to complete Assignment 06 for the class Foundations of Programming: Python. The assignment involves adding script to a start script in order to write a script that creates a list for a user to select choices from in order to add and prioritize or remove tasks in a text file. The program will be very similar to that in assignment05, but now it uses separate classes to call functions.

# Assessing the Starter Script

I opened the starter script and took a look at what kinds of steps would need to be taken in order to complete the assignment. The starter script provided specific guidance as to how to complete the script. This allowed me to determine the framework necessary to complete the assignment. The starter script outlined the menu for the user to select from in order to alter the “ToDoFile.txt” file that contained the list of tasks and priorities. When selected by the user, the options would allow them to manipulate the table of data in ToDoFile.txt with dictionary rows. The starter script has the processing and input/output tasks divided into two different sections.

# Completing the Script

I began by defining the variables. I then continued to write the processing class functions that process the actions when the menu options are selected. Since it’s a processing class section, the functions used are only to process the actions and do nothing else.

Figure 1 shows functions used that would open and close the text file, process a user’s input when they use the program, and append the proper location in the text file with their input.

A screenshot of a cell phone

Description automatically generated

***Figure 1. Processing Functions***

Figure 1 also shows how “for,” “in,” and “if” direct the program to seek out the proper location in the text file to process data.

Using classes allows for easier script navigation since classes can be collapsed and the class explains what type of functions are going to be in that particular section of the script. The functions in other sections of the script will refer to the processing class in order for the classes to work together as a whole. Figure 1 also shows how “for,” “in,” and “if” direct the program to seek out the proper location in the text file to process data.

The input and output class, or “class IO,” performs the input and output tasks by using the input() and print() functions. Figure 2 shows an example of the text in the input and output class used to give outputs to the user once they select an option from the menu.

A screenshot of a cell phone

Description automatically generated

***Figure 2. Input and Output Processing***

In figure 2 it is shown that the print() function is predominantly used to display text to the user, whereas in figure 1 the print() function isn’t used at all since the process class does not need it. This shows how the two different classes vary in purpose greatly.

Once the processing and input/output classes were complete, I moved onto the main body of the script. The main body interacts with both the processing and input/output classes to create a functioning program.

Figure 3 shows how the main body script calls the functions in the classes to run by using the defined terms “IO.” and “Processor.”

A screenshot of a cell phone

Description automatically generated

***Figure 3. Main Body of the Script***

Figure 3 shows that the other classes are called by attaching “IO” or “Processor” and then a “.” before the functions in the script. All of the functions in the script are nested under their respective “class” that is called whatever the author wants it to be called. Despite the functions being nested under different classes, they all interact with the same text file to achieve a successful task manager program.

# Running the Script in PyCharm

Figure 4 shows the script running in PyCharm and interacting with the user.

A screenshot of a cell phone

Description automatically generated

***Figure 4. Script Running in PyCharm***

Figure 4 shows the script successfully adding the user’s input and then displaying it back to them when prompted. The script is functioning with no error.

# Summary

Assignment06 achieved the same goals as Assignment05, but with improved script organization and navigation by using classes. The global variables at the top of the script remained the same, but the rest of the script changed to use more functions. As long as the functions in different classes are properly referring and calling to each other, the script’s navigation is streamlined through the use of classes. This can be particularly beneficial when writing and navigating longer scripts since classes can be minimized.