**Patrick Woodard**

UW Foundations in Python Summer 2021

Assignment 06

<https://github.com/paw345/IntroToProg-Python-Mod06>

# Introduction

The program Assignment06.py reads the file ToDoFile.txt, then interacts with the user to work with the “To Do” list.

# The Script

For this assignment I started with the partially built script Assignment06\_Starter.py. The code in this “starter” script had the code broken out into four major sections: **Data**, **Processing**, **Presentation**, and **Main** (i.e. employing the *separation of concerns* design principle).

The **Data** section initializes the major global variables and constants. In-line comments are included to make clear the intended use of the variables and constants. I did not change anything in this section. See Figure 1 for a screenshot of the **Data** section.

Text

Description automatically generated

Figure . **Data** Section of Code

The **Processing** section of code is all contained within a **Class** named *Processor*. Four core functions are defined and run through this Processor. The first, *read\_data\_from\_file*, was part of the starter script and reads data from ToDoFile.txt. The other three add data, remove data, and write data – and all were coded as part of this assignment. These functions perform the same functions as in Assignment 05, but they are different in that they receive their arguments from the **Presentation** section. See Figure 2 for a screenshot of the first two functions in the **Processing** section.

Graphical user interface, text

Description automatically generated

Figure 2. **Processing** Section of Code (first two functions only)

The third section of code, **Presentation (Input/Output)**, contains all the code that interacts directly with the user and includes seven (7) functions. The first five (5) functions are unchanged from the starter file, the last two (2) functions were coded for this assignment and collect new tasks/priorities to add and existing tasks/priorities to delete. Local variables are used within these functions (e.g. *iTask*). See Figure 3 for a screenshot of the **Presentation** section.

Graphical user interface, text, application, email

Description automatically generated

Figure 3. **Presentation** Section of Code (unchanged functions collapsed)

Most of the **Main** (*Main Body of Script*) section of code was structured in the “starter” file as an **if/elif** statement nested within a **while(True)** statement (the one exception is the initial call of the *read\_data\_from\_file* function). All of the major **elif** statements were written as part of this assignment and they call the functions defined in the **Processing** and **Presentations** section to perform many of the desired actions. See Figure 4 for a snippet of the **Main** section.

Graphical user interface, text, application

Description automatically generated

Figure 4. **Main** Section of Code (partial)

# Running the Code and the Output File

The script operates as intended in both PyCharm and console mode. Screenshots of the code running in console mode, running in Pycharm, and input/output file ToDoFile.txt are presented in Figure 5, Figure 6, and Figure 7.

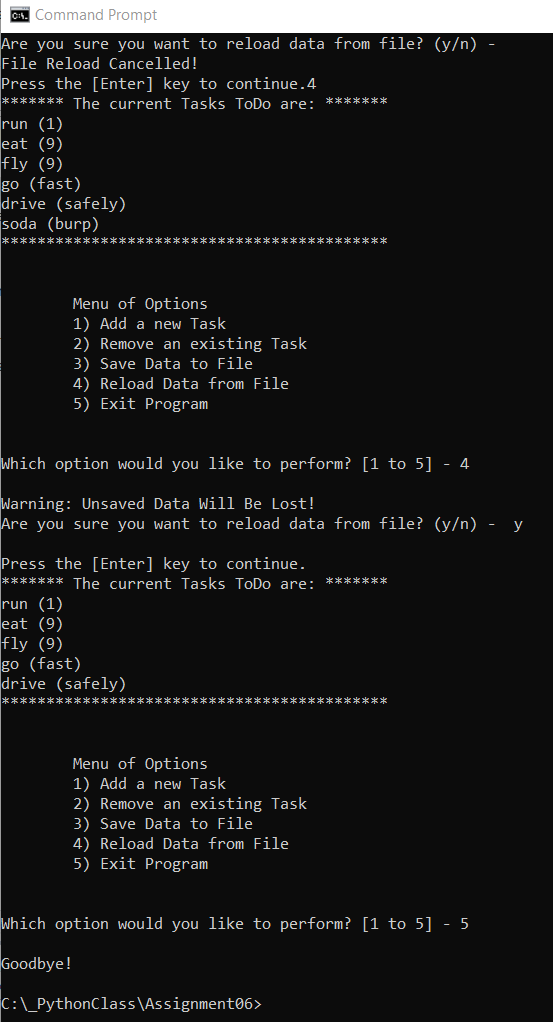


Figure 5. Screenshot of the Script Running in Console Mode

Graphical user interface, text, application, email

Description automatically generated

Figure . Screenshot of the Script Running in PyCharm

Graphical user interface, text, application

Description automatically generated

Figure . Screenshot of script output file and data contained therein

# Summary

In this assignment it becomes clear how employing the *Separation of Concerns* design principle and embedding appropriately grouped functions within the *Processing* and *Presentation* sections makes the code more readable. That said, this assignment and new structure took a bit of effort to get comfortable with. Part of the “getting comfortable” process led me to use the Pycharm debugger more actively. I set breakpoints and stepped through parts of the code while troubleshooting. This was good and I’m started to see how the debugger tool will be really valuable as programs get longer and more complicated.