Dominic Ferenczy  
May 18, 2021  
Foundations of Programming: Python  
Assignment 06  
[uwferenczy/IntroToProg-Python-Mod06 (github.com)](https://github.com/uwferenczy/IntroToProg-Python-Mod06)

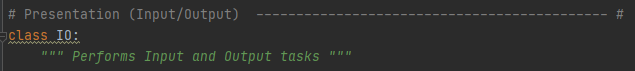
How to Use Classes and Functions in an Application

# Introduction

This document details how I applied what I have learned in module 5 of the Introduction to Programming class to modify an existing script that manages a “ToDo list.” Topics covered in this document include how to use classes to organize code, how to add functions to a class, how to utilize classes and functions in the main script of an application and how to test the application.

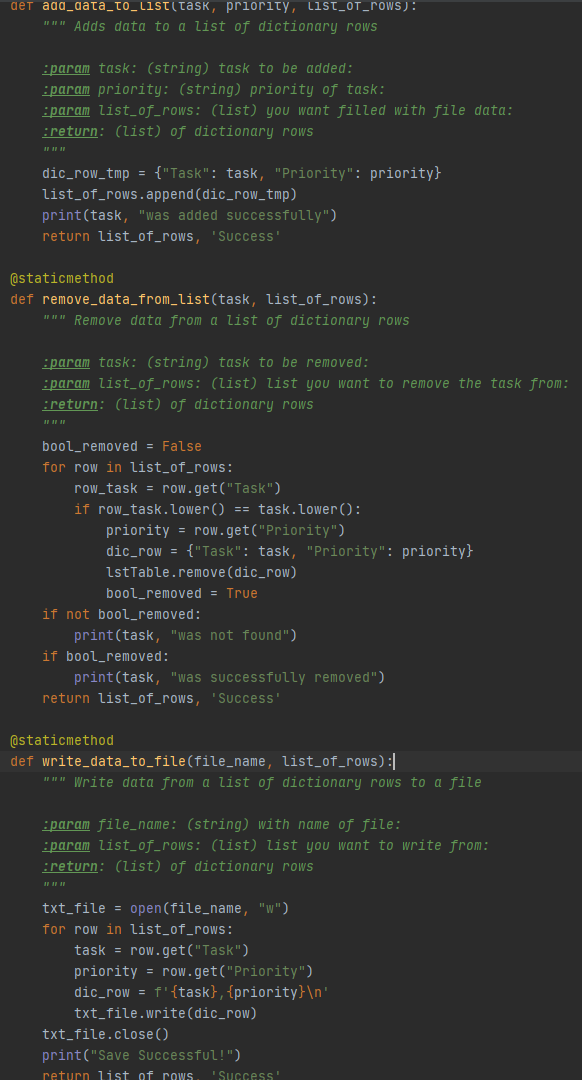
# Using Classes to Organize Code

This assignment utilized a starter file that needed to be updated to add classes and functions to assignment 5. The starter file had two classes: Processor and IO. Classes can be used as containers that group multiple functions with similar themes and can be called in the main script. Defining and using classes makes the main code easy to read and understand because it can read like pseudo-code. Classes are initialized by using class, followed by the name of the class and a colon. It is best practice to start each class off with a comment on the intended functionality of the class. The class IO can be seen in Figure 1.

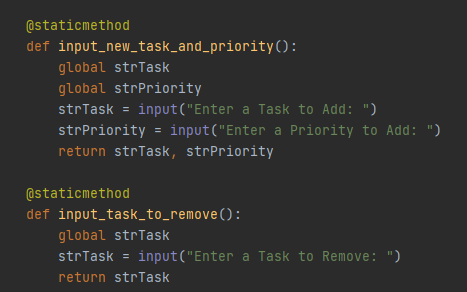
  
Figure 1: PyCharm showing the IO class definition and functionality comment

# Adding Functions to a Class

There were multiple functions inside of each class in the starter file. Each function definition is preceded by @staticmethod. The functions are defined by def, followed by the name of the function (needs to be in snake case, and parenthesis with any required fields inside of it. The starter file identified the add\_data\_to\_list, remove\_data\_from\_list and write\_data\_to\_file functions as needing to be updated in the Processor class. It is important to include comments in each function that defines the purpose of the function, the parameters and the return. The required parameters were identified by the starter code. These are the parameters that get passed from outside of the function, to the function. I utilized my code from assignment 5 for each one of these functions. After pasting my code, I made sure that I updated the variables in the function to the parameter names that get passed to the function. The resulting code can be seen in figure 2.

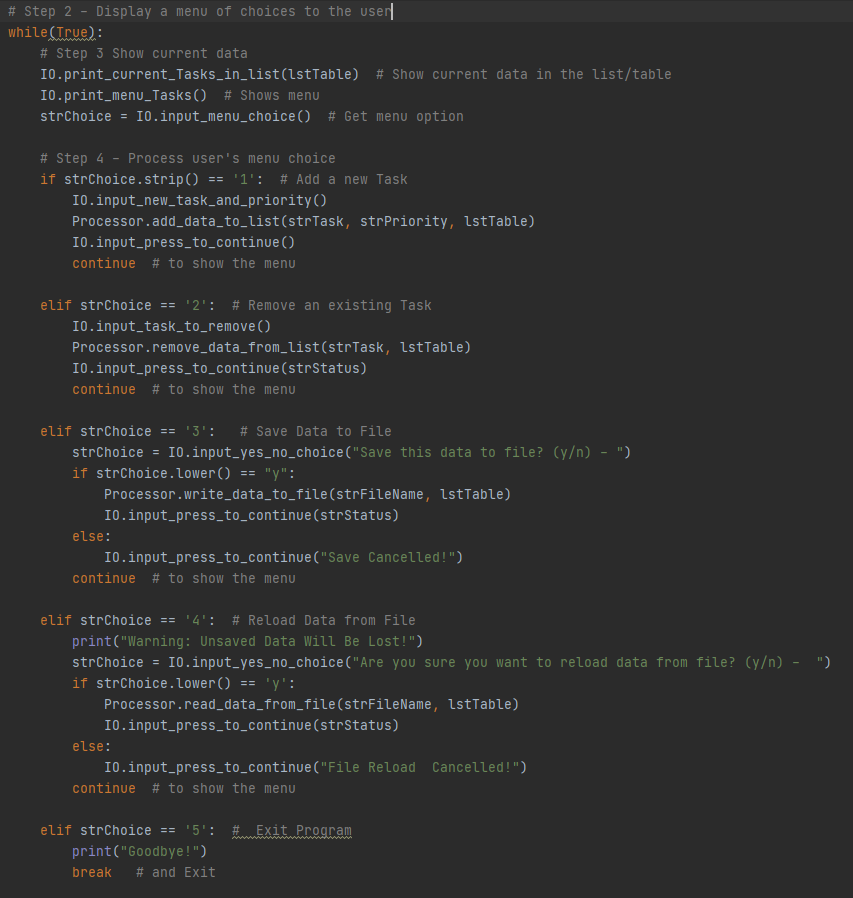
  
Figure 2: PyCharm showing code from assignment 5 refactored to separate functions

The IO class required code to be added to the input\_new\_task\_and\_priority and input\_task\_to\_remove functions. I utilized my code from assignment 5 for these functions as well. I noticed that the beginning of the starter script contained the strTask and strPriority variables and wanted to pass the data from the functions to the variables so I can pass them into other functions in the main script. Variables inside of functions are shadowed, meaning that they do not exist outside of the function by default. To update the strTask and strPriority variables, I declared both of them inside of each function as global variables and added them to the return for each function. The resulting code can be seen in figure 3.

  
Figure 3: PyCharm showing code for calling and returning global variables

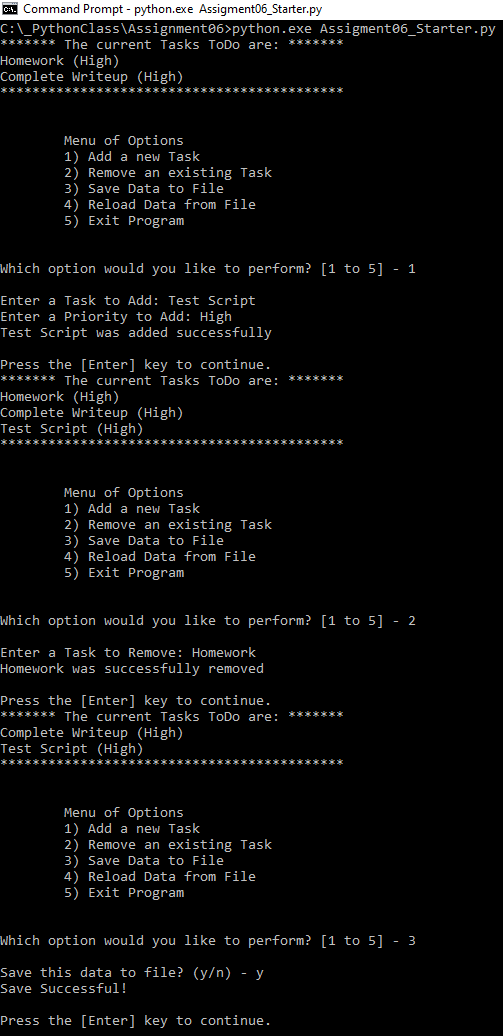
# Integrating Classes and Functions into Main Script

The starter.py code identified that code needed to be added to each of the menu choices in the main section. Functions, classes and parameters can be called by writing the class followed by a period, followed by the function, followed by parenthesis with any parameters in them. I added the class and function definitions that applied to each choice under each menu selection as shown in figure 4.

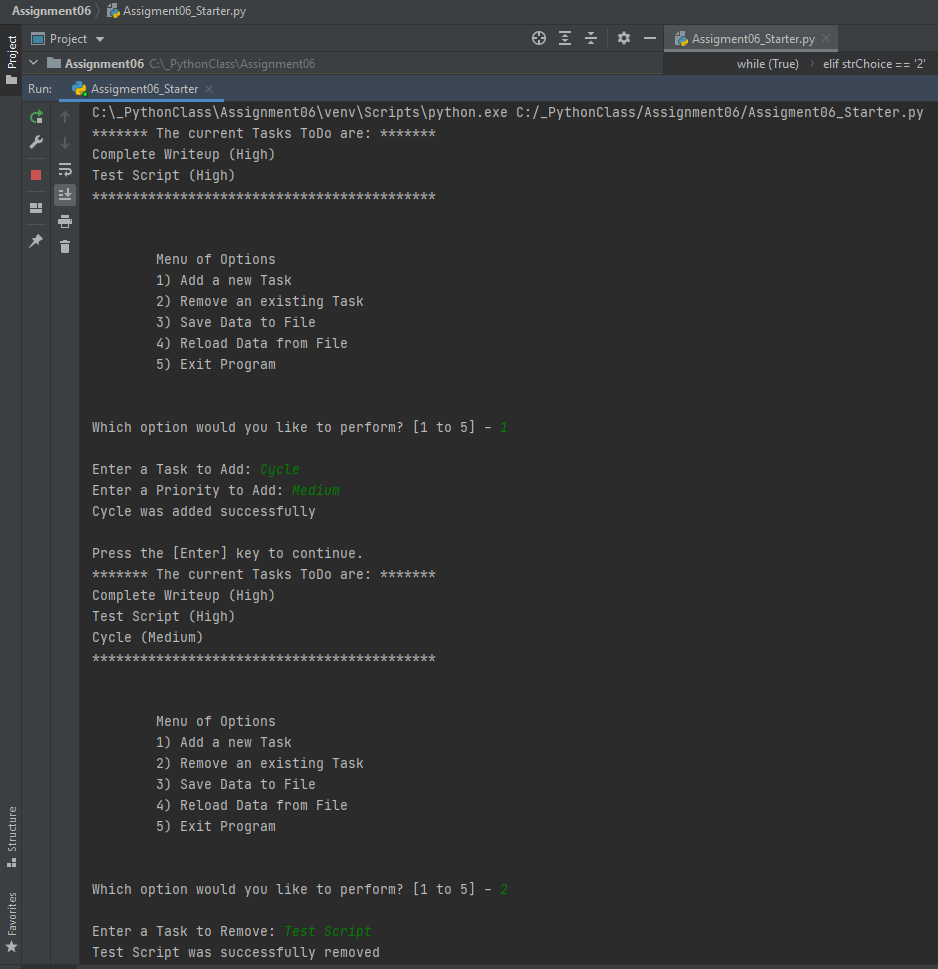
  
Figure 4: PyCharm showing IO and Processor classes and variables being used in main code

# Testing Application

I tested the application in command prompt as shown in figure 5. It ran correctly in command prompt and matched the assignment requirements.

  
Figure 5: Command Prompt showing program running correctly

I then tested the application in PyCharm as shown in figure 6. It ran correctly in PyCharm and matched the assignment requirements.

  
Figure 6: PyCharm showing program running correctly

I then opened the ToDoList.txt file to verify that the new items were added as shown in figure 7. The items were added correctly.

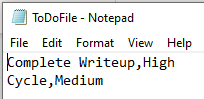


Figure 7: Notepad showing items added correctly

An important note with testing the code is that functions can be tested independently before adding to the main code. I used “scratches” in PyCharm to develop and test each function before integrating them into my code.

# Summary

This document captured what I learned in module 6 of the class. I documented the steps I took to use classes to organize code, add functions to a class, utilize classes and functions in the main script of an application and to test the application.