Patricia Coonrad

May 25, 2020

IT FDN 100 A Sp 20: Foundations Of Programming: Python

Assignment 06

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

Add Functions to ToDo List Script

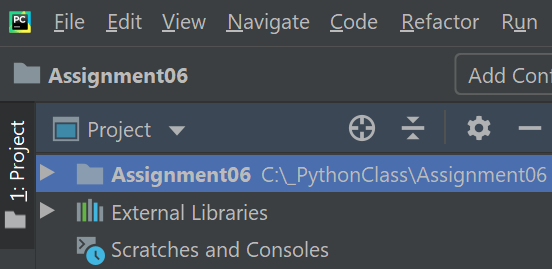
# Introduction

In this assignment I will describe the steps I took to add functions to the Python program I created for Assignment 05 called ToDo List. I will demonstrate some of the concepts I learned during the course, such as use of a script template, functions, parameters, arguments, class, ‘Separations of Concerns’, Doc Strings, to name a few.

# Add Functions to the ToDo List Script

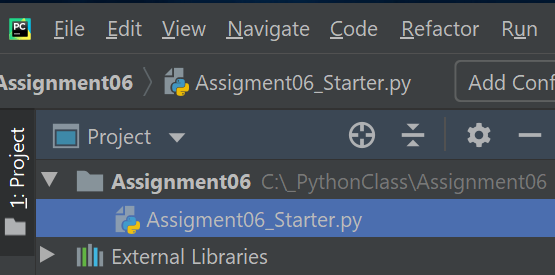
In this exercise, I will describe the steps I took to add functions the ToDo List program. I will then run the script and capture the image of the script execution using both PyCharm and the Command Window.

1. As per assignment directions, I created a new Project in PyCharm that uses the \_PythonClass\Assignment06 folder as its location (Figure 1).



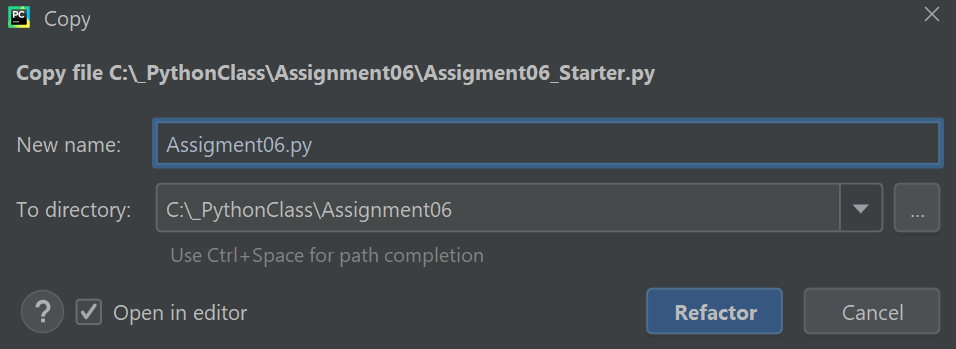
***Figure 1. Assignment06 PyCharm Project***

1. I added the starter file ‘Assignment06\_Starter.py’ to the project. This file was provided by instructor Randy Root and was a starting template I had to modify and use for my program (Figure 2).

****

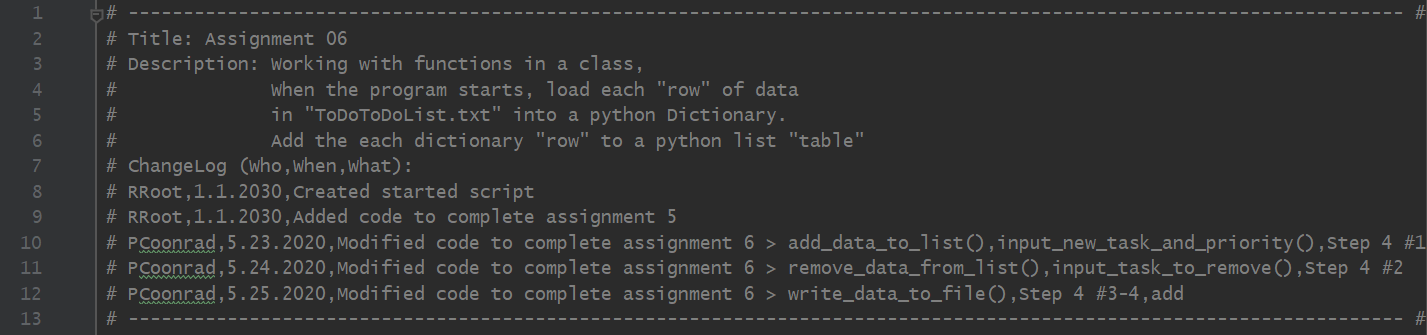
***Figure 2. Adding the ‘Assignment06\_Starter.py’ file to the project***

1. I made a copy of the template file and named it ‘Assignment06.py’ (Figure 3).



***Figure 3. Copying the starter file and naming ‘Assignment06.py’***

1. Once the ‘Assignment06.py’ file was created, I started modifying the script. I updated the script header change log to reflect the changes I made to the code. The script header is a comment at the beginning of the script. It includes information about the script such as title, brief description of the program, and change log. (Figure 4).



***Figure 4. Assignment06 Script header***

1. The objective of the assignment was to convert processing code and input/output code into functions whenever possible so that the code is more flexible and more manageable.

A function is a set of programming statements. The programming statements can be executed by calling the name of the function, and the function will then perform the tasks of the statements. Functions can be either standalone or grouped inside a class. A class is a name group of variables and functions. Whenever you have a function inside a class, it is called a method.

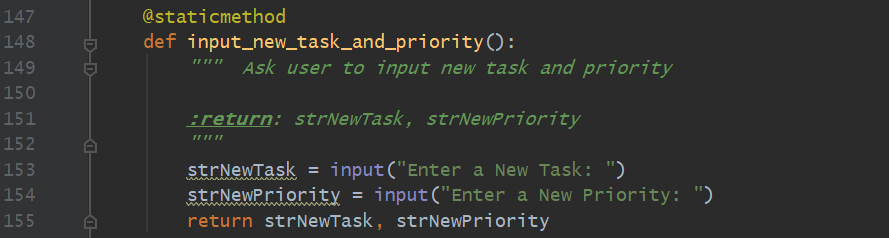
In Python, you have to define the function first at the top of your file before you use. You have to load it into memory before you execute it.

We use functions to process data and to present data to the user. When using functions you can pass values for processing using parameters. A parameter is a variable being used by the function.

Applications are usually made up of three layers: the data layer, presentation layer, and the processing layer. The data layer is where the data is stored. The presentation layer is where you get information from the user and display information to the user. The processing layer is where there is some kind of calculation that would be in the processing layer. The process of separating the application code into these three layers is called ‘Separation of Concerns’.

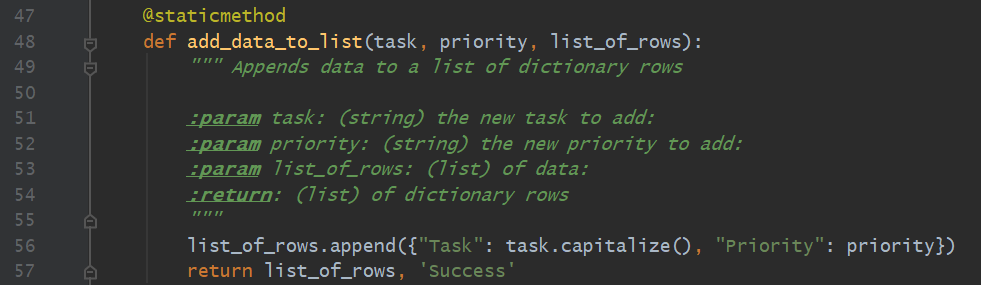
Note: I added Doc String (known as docstring in Python) as a header to beginning of the functions. This can be a helpful tool in PyCharm specifically to display developer’s notes.

1. I first worked on the “Add a New Task” option from the menu. Under the Presentation (Input/Output) layer, I converted the input\_new\_task\_and\_priority() method as per below. Since this is part of the class IO (Input/Output), I only included code related to getting input from the user (Figure 5).



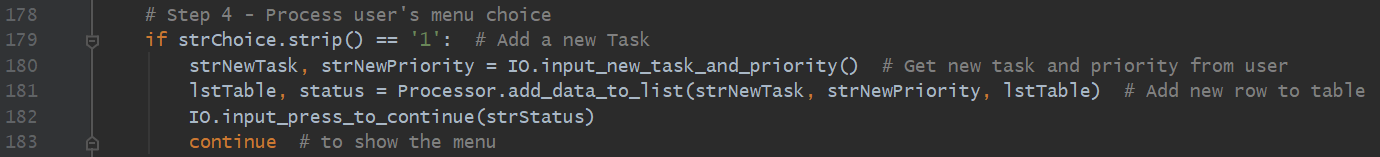
***Figure 5. Presentation code to add task and priority***

1. I then went to the Processing layer and added the processing code for add\_data\_to\_list() method Figure 6).



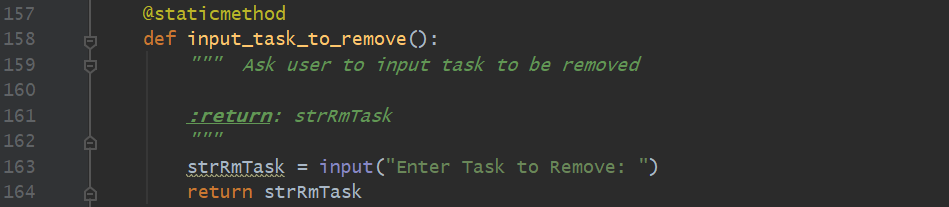
***Figure 6. Processing code to add task and priority***

1. Under the Main Body of Script, I called the presentation and processing functions to execute the tasks for adding a new task and priority to the list (Figure 7).



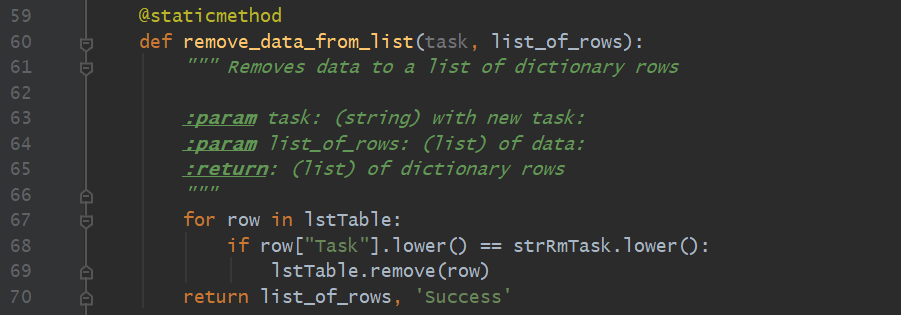
***Figure 7. Calling the functions to add new task and priority***

1. I then worked on the “Remove an existing Task” option from the menu. Under the Presentation (Input/Output) layer, I converted the input\_ask\_to\_remove() method as per below. Since this is part of the class IO (Input/Output), I only included code related to getting input from the user related to the task to remove (Figure 8).



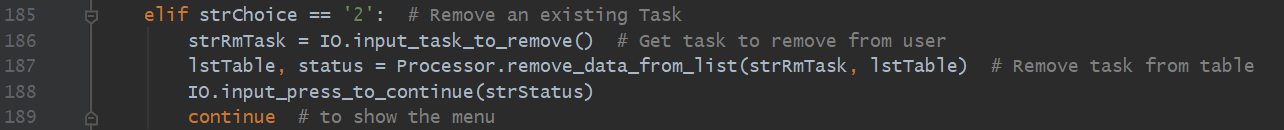
***Figure 8. Presentation code to remove task***

1. I then went to the Processing layer and added the processing code for remove\_data\_to\_list() method (Figure 9).



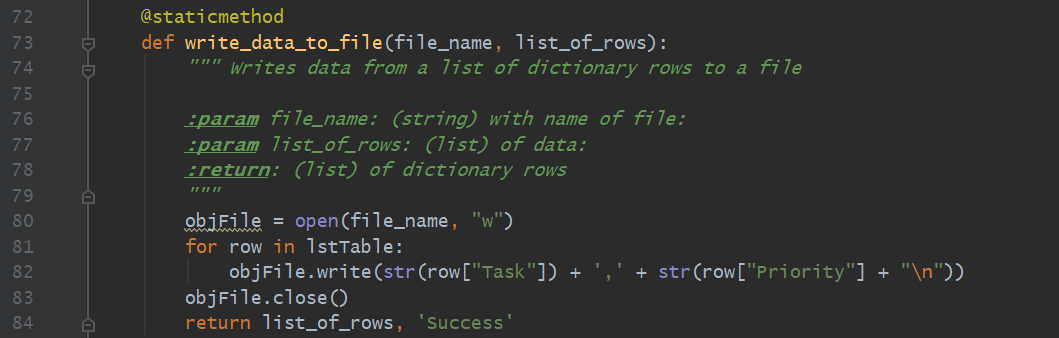
***Figure 9. Processing code to remove task***

1. Under the Main Body of Script, I called the presentation and processing functions to execute the tasks for removing the task from the list (Figure 10).



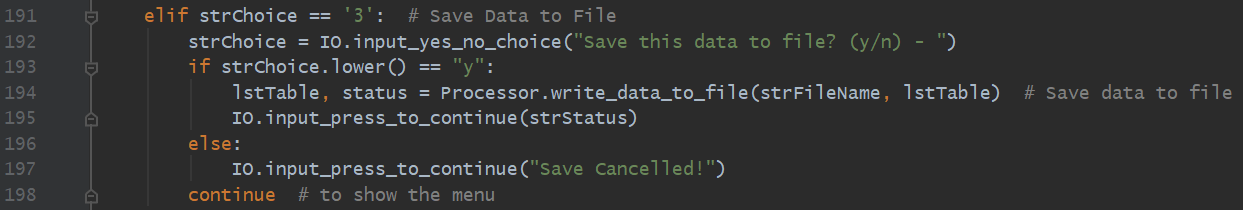
***Figure 10. Calling the functions to remove the task***

1. Next I worked on “Save Data to File” option from the menu. Under the Processing code, I declared the write\_data\_to\_file() function as per below (figure 11).



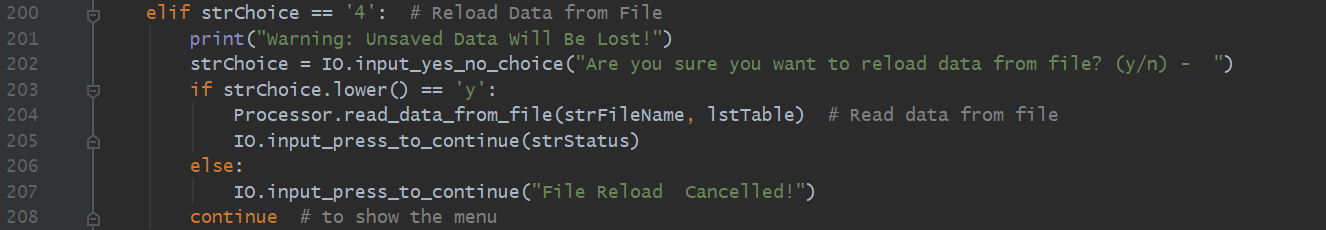
***Figure 11.***

1. Under the Main Body of Script, I called the processing and presentation functions to execute the task of writing to the file (Figure 12).



***Figure 12. Calling the functions to write data to file***

1. Lastly, under the Main Body of Script, I called the processing and presentation functions that had already been declared to execute the task of reading (reloading) data from the file (Figure 13).



***Figure 13. Calling the functions to write data to file***

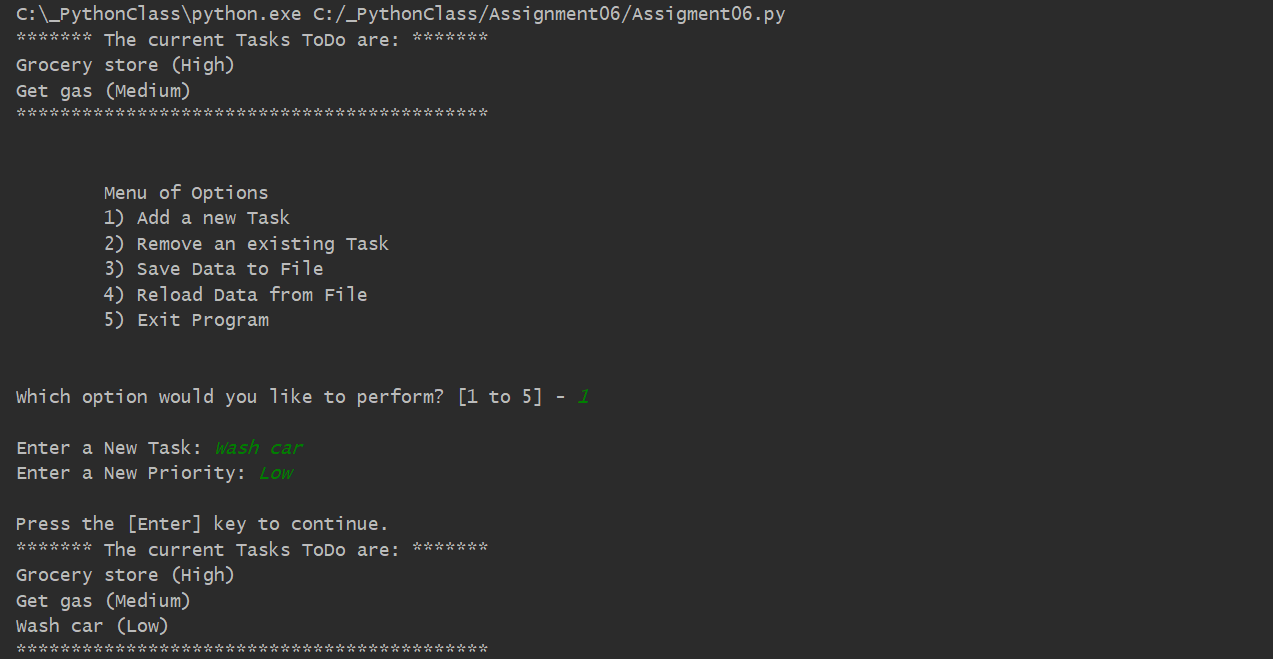
1. Below is a copy of the entire script in PyCharm (Figure 14).

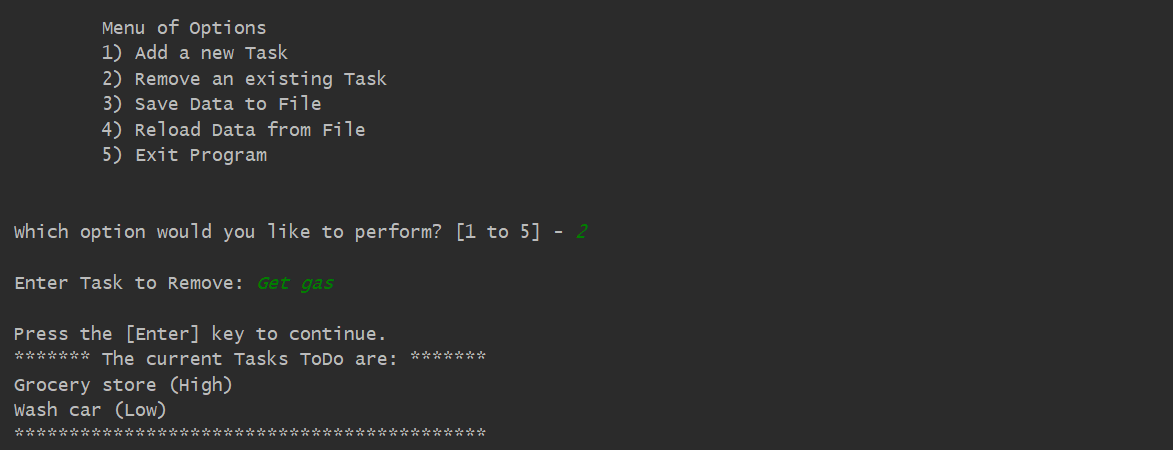
# --------------------------------------------------------------------------------------------------------------------------------------------------------------------- #  
# Title: Assignment 06  
# Description: Working with functions in a class,  
# When the program starts, load each "row" of data  
# in "ToDoToDoList.txt" into a python Dictionary.  
# Add the each dictionary "row" to a python list "table"  
# ChangeLog (Who,When,What):  
# RRoot,1.1.2030,Created started script  
# RRoot,1.1.2030,Added code to complete assignment 5  
# PCoonrad,5.23.2020,Modified code to complete assignment 6 > add\_data\_to\_list(),input\_new\_task\_and\_priority(),Step 4 #1  
# PCoonrad,5.24.2020,Modified code to complete assignment 6 > remove\_data\_from\_list(),input\_task\_to\_remove(),Step 4 #2  
# PCoonrad,5.25.2020,Modified code to complete assignment 6 > write\_data\_to\_file(),Step 4 #3-4,add  
# --------------------------------------------------------------------------------------------------------------------------------------------------------------------- #  
  
# Data ---------------------------------------------------------------------- #  
# Declare variables and constants  
strFileName = "ToDoFile.txt" # The name of the data file  
objFile = None # An object that represents a file  
dicRow = {} # A row of data separated into elements of a dictionary {Task,Priority}  
lstTable = [] # A list that acts as a 'table' of rows  
strChoice = "" # Captures the user option selection  
strTask = "" # Captures the user task data  
strPriority = "" # Captures the user priority data  
strStatus = "" # Captures the status of an processing functions  
  
# Processing --------------------------------------------------------------- #  
class Processor:  
 *""" Performs Processing tasks """* @staticmethod  
 def read\_data\_from\_file(file\_name, list\_of\_rows):  
 *""" Reads data from a file into a list of dictionary rows* ***:param*** *file\_name: (string) with name of file:* ***:param*** *list\_of\_rows: (list) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* list\_of\_rows.clear() # clear current data  
 file = open(file\_name, "r")  
 for line in file:  
 task, priority = line.split(",")  
 row = {"Task": task.strip(), "Priority": priority.strip()}  
 list\_of\_rows.append(row)  
 file.close()  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def add\_data\_to\_list(task, priority, list\_of\_rows):  
 *""" Appends data to a list of dictionary rows* ***:param*** *task: (string) the new task to add:* ***:param*** *priority: (string) the new priority to add:* ***:param*** *list\_of\_rows: (list) of data:* ***:return****: (list) of dictionary rows  
 """* list\_of\_rows.append({"Task": task.capitalize(), "Priority": priority})  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def remove\_data\_from\_list(task, list\_of\_rows):  
 *""" Removes data to a list of dictionary rows* ***:param*** *task: (string) with new task:* ***:param*** *list\_of\_rows: (list) of data:* ***:return****: (list) of dictionary rows  
 """* for row in lstTable:  
 if row["Task"].lower() == strRmTask.lower():  
 lstTable.remove(row)  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def write\_data\_to\_file(file\_name, list\_of\_rows):  
 *""" Writes data from a list of dictionary rows to a file* ***:param*** *file\_name: (string) with name of file:* ***:param*** *list\_of\_rows: (list) of data:* ***:return****: (list) of dictionary rows  
 """* objFile = open(file\_name, "w")  
 for row in lstTable:  
 objFile.write(str(row["Task"]) + ',' + str(row["Priority"] + "\n"))  
 objFile.close()  
 return list\_of\_rows, 'Success'  
  
# Presentation (Input/Output) -------------------------------------------- #  
class IO:  
 *""" Performs Input and Output tasks """* @staticmethod  
 def print\_menu\_Tasks():  
 *""" Display a menu of choices to the user* ***:return****: nothing  
 """* print('''  
 Menu of Options  
 1) Add a new Task  
 2) Remove an existing Task  
 3) Save Data to File   
 4) Reload Data from File  
 5) Exit Program  
 ''')  
 print() # Add an extra line for looks  
  
 @staticmethod  
 def input\_menu\_choice():  
 *""" Gets the menu choice from a user* ***:return****: string  
 """* choice = str(input("Which option would you like to perform? [1 to 5] - ")).strip()  
 print() # Add an extra line for looks  
 return choice  
  
 @staticmethod  
 def print\_current\_Tasks\_in\_list(list\_of\_rows):  
 *""" Shows the current Tasks in the list of dictionaries rows* ***:param*** *list\_of\_rows: (list) of rows you want to display* ***:return****: nothing  
 """* print("\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*")  
 for row in list\_of\_rows:  
 print(row["Task"] + " (" + row["Priority"] + ")")  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 print() # Add an extra line for looks  
  
 @staticmethod  
 def input\_yes\_no\_choice(message):  
 *""" Gets a yes or no choice from the user* ***:return****: string  
 """* return str(input(message)).strip().lower()  
  
 @staticmethod  
 def input\_press\_to\_continue(optional\_message=''):  
 *""" Pause program and show a message before continuing* ***:param*** *optional\_message: An optional message you want to display* ***:return****: nothing  
 """* print(optional\_message)  
 input('Press the [Enter] key to continue.')  
  
 @staticmethod  
 def input\_new\_task\_and\_priority():  
 *""" Ask user to input new task and priority* ***:return****: strNewTask, strNewPriority  
 """* strNewTask = input("Enter a New Task: ")  
 strNewPriority = input("Enter a New Priority: ")  
 return strNewTask, strNewPriority  
  
 @staticmethod  
 def input\_task\_to\_remove():  
 *""" Ask user to input task to be removed* ***:return****: strRmTask  
 """* strRmTask = input("Enter Task to Remove: ")  
 return strRmTask  
  
# Main Body of Script ------------------------------------------------------ #  
  
# Step 1 - When the program starts, Load data from ToDoFile.txt.  
Processor.read\_data\_from\_file(strFileName, lstTable) # read file data  
  
# Step 2 - Display a menu of choices to the user  
while (True):  
 # Step 3 Show current data  
 IO.print\_current\_Tasks\_in\_list(lstTable) # Show current data in the list/table  
 IO.print\_menu\_Tasks() # Shows menu  
 strChoice = IO.input\_menu\_choice() # Get menu option  
  
 # Step 4 - Process user's menu choice  
 if strChoice.strip() == '1': # Add a new Task  
 strNewTask, strNewPriority = IO.input\_new\_task\_and\_priority() # Get new task and priority from user  
 lstTable, status = Processor.add\_data\_to\_list(strNewTask, strNewPriority, lstTable) # Add new row to table  
 IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu  
  
 elif strChoice == '2': # Remove an existing Task  
 strRmTask = IO.input\_task\_to\_remove() # Get task to remove from user  
 lstTable, status = Processor.remove\_data\_from\_list(strRmTask, lstTable) # Remove task from table  
 IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu  
  
 elif strChoice == '3': # Save Data to File  
 strChoice = IO.input\_yes\_no\_choice("Save this data to file? (y/n) - ")  
 if strChoice.lower() == "y":  
 lstTable, status = Processor.write\_data\_to\_file(strFileName, lstTable) # Save data to file  
 IO.input\_press\_to\_continue(strStatus)  
 else:  
 IO.input\_press\_to\_continue("Save Cancelled!")  
 continue # to show the menu  
  
 elif strChoice == '4': # Reload Data from File  
 print("Warning: Unsaved Data Will Be Lost!")  
 strChoice = IO.input\_yes\_no\_choice("Are you sure you want to reload data from file? (y/n) - ")  
 if strChoice.lower() == 'y':  
 Processor.read\_data\_from\_file(strFileName, lstTable) # Read data from file  
 IO.input\_press\_to\_continue(strStatus)  
 else:  
 IO.input\_press\_to\_continue("File Reload Cancelled!")  
 continue # to show the menu  
  
 elif strChoice == '5': # Exit Program  
 print("Goodbye!")  
 break # and Exit

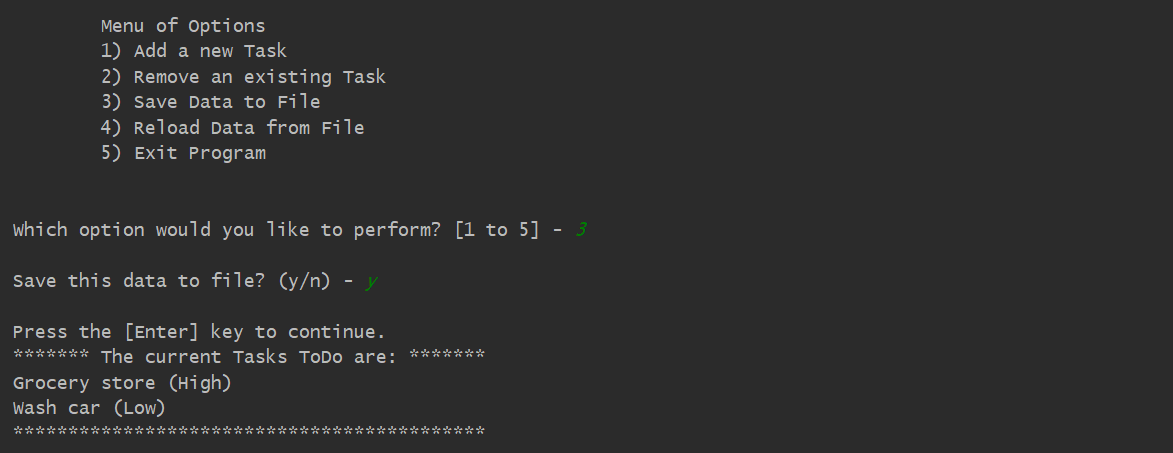
***Figure 14. Copy of the script in PyCharm***

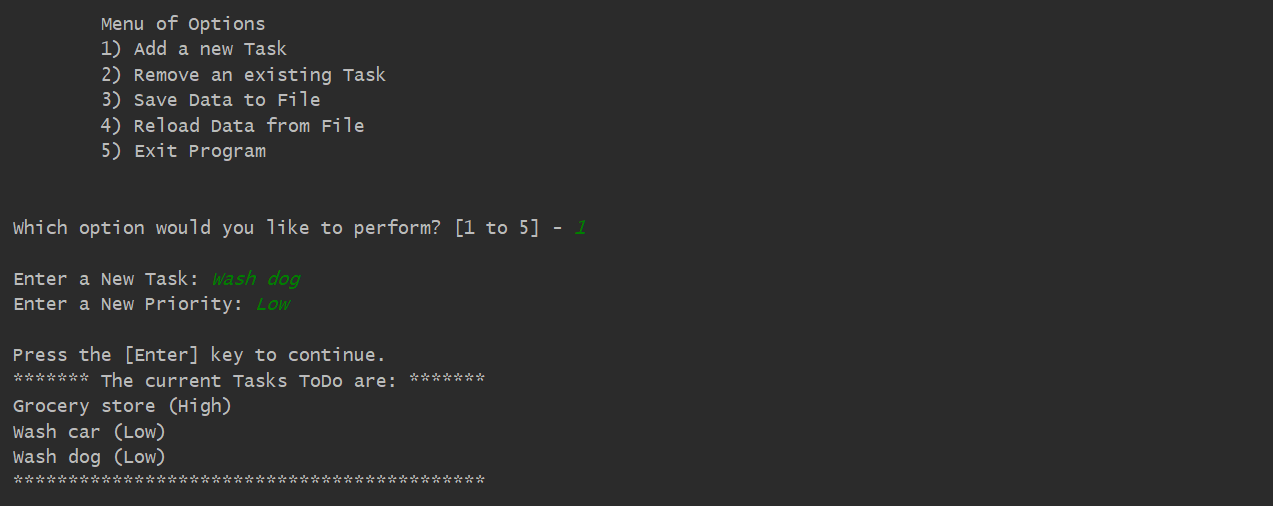
# Running the ToDo List Program Using PyCharm

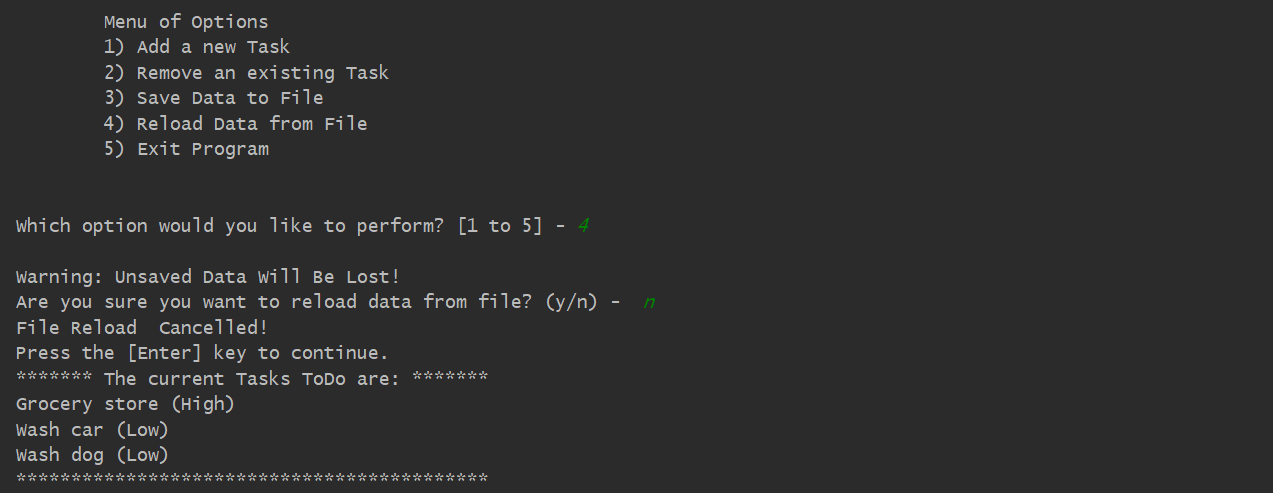
After the code was complete, I successfully ran the program using PyCharm (Figure 15).

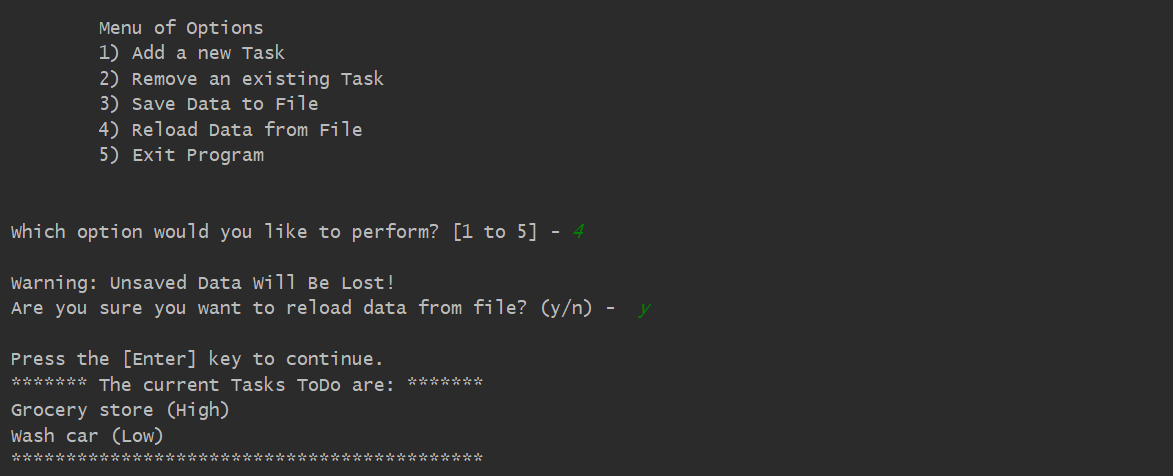


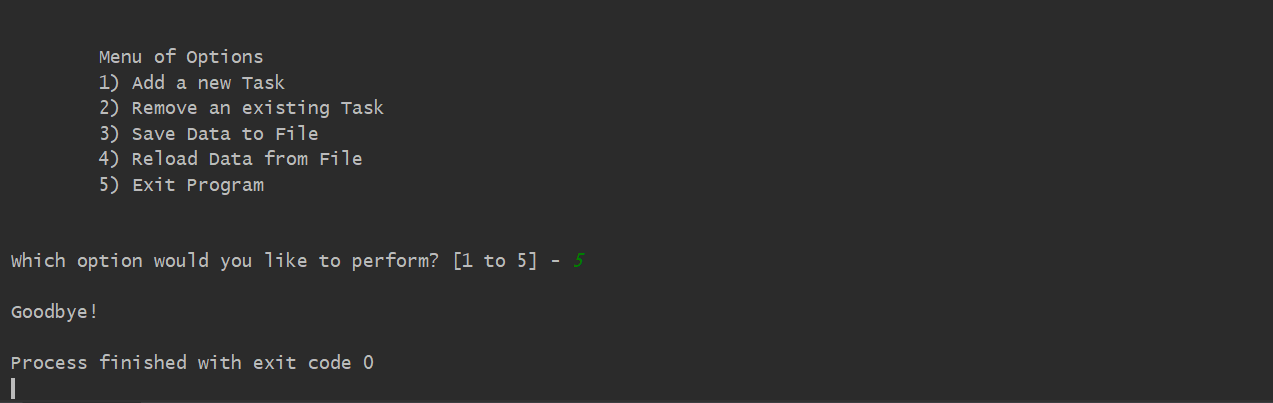








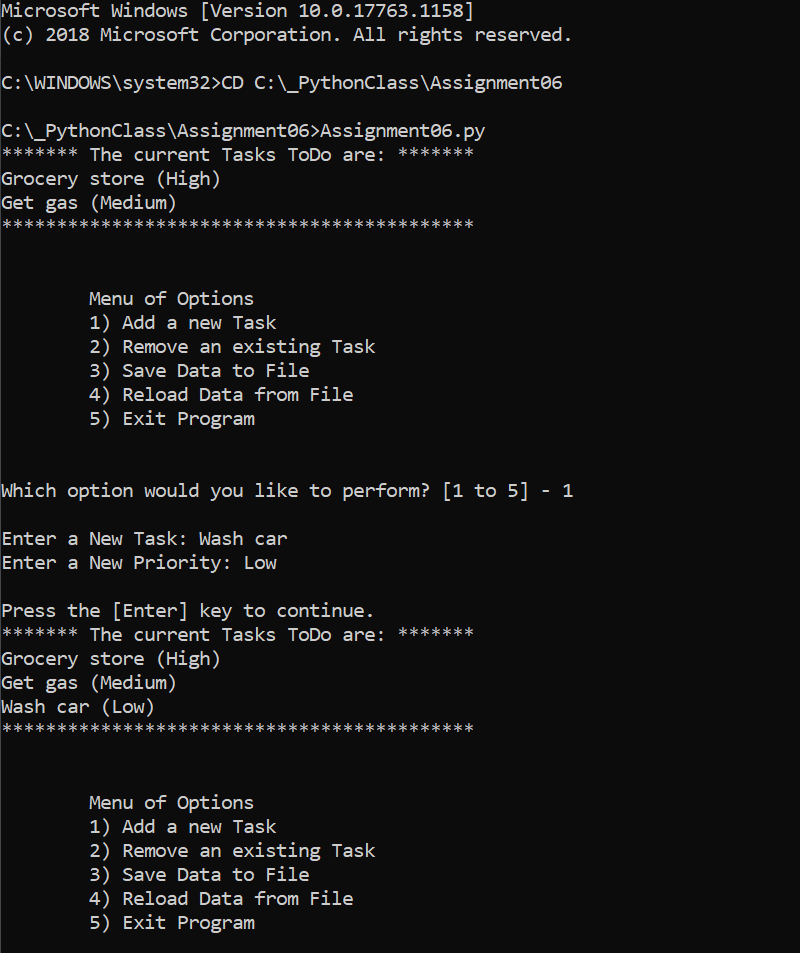


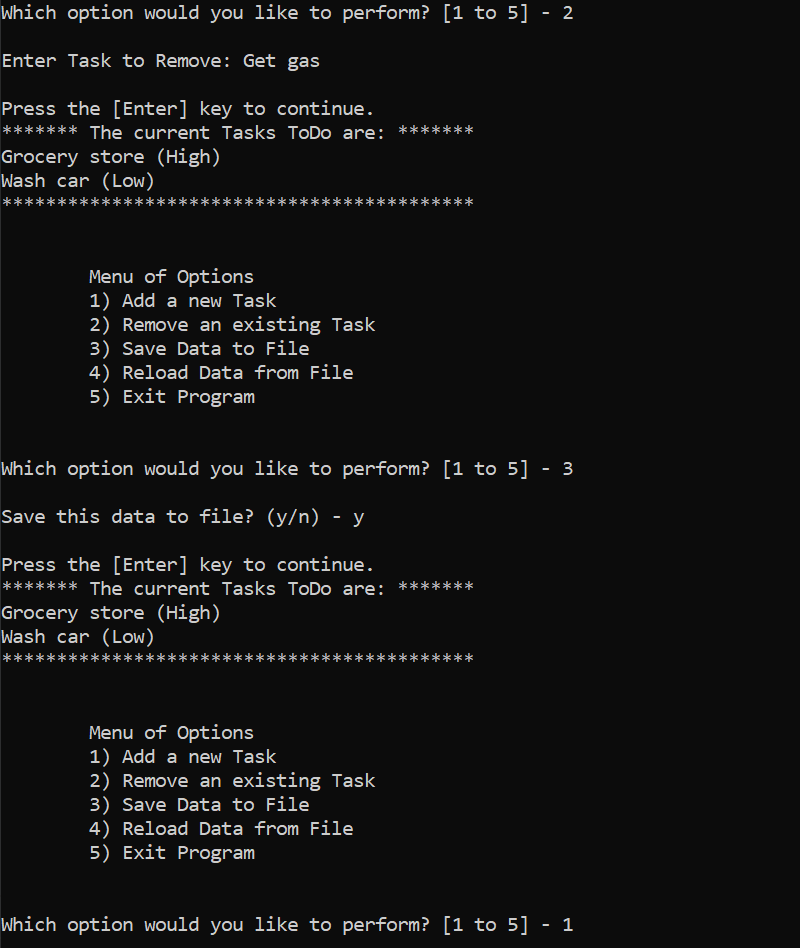


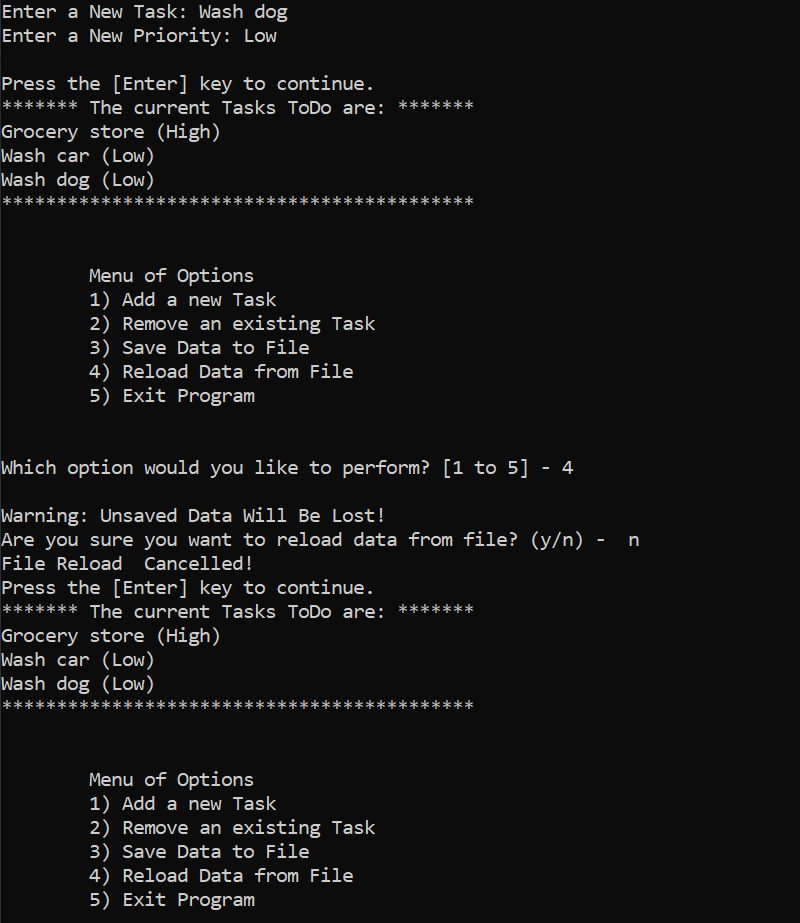
***Figure 15. Screenshot of the script running in PyCharm***

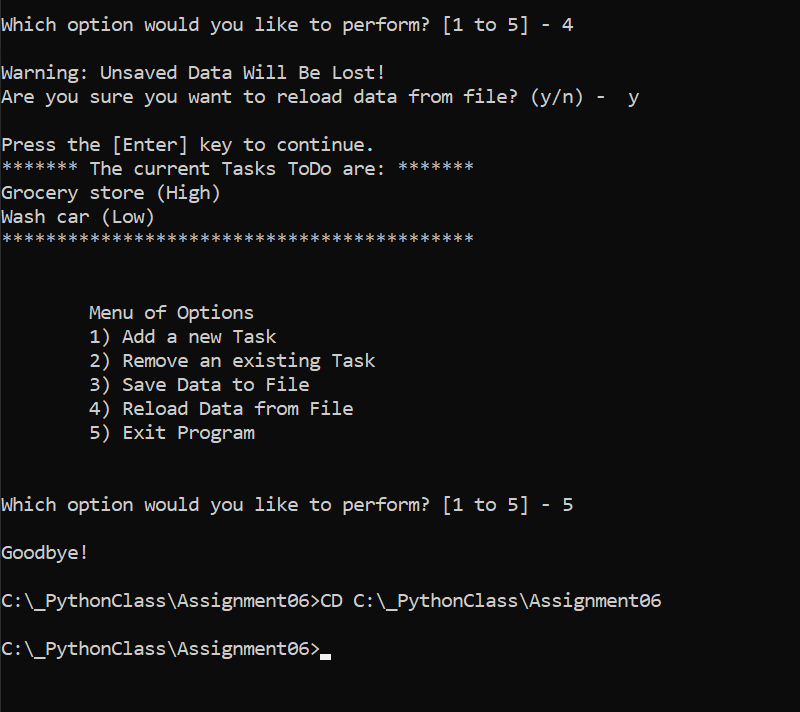
# Running the ToDo List Program Using the Command Window

And I also successfully ran the program using the Command Window (Figure 16).





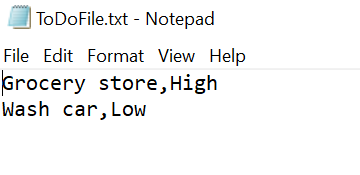




***Figure 16. Screenshot of the script running in a Command Window***

# Verify that it Worked

After I ran the program, I located the text file and opened it in a text editor to verify the file had data and it was correct (Figure 17).



***Figure 17. Verifying that the file has the correct data***

Summary

In the Module 06, I have modified a script template for the ToDo List program and have demonstrated the use of different concepts I learned during the course such as as use of a script template, functions, parameters, arguments, class, ‘Separations of Concerns’, Doc Strings, to name a few. By watching the instructional videos, reading the textbook and consulting additional documentation, I was able to successfully modify the ToDo List program by adding functions, and successfully ran the program using PyCharm and Command Window.