Moiz Vahora

08/17/2021

IT FDN 110 A

Assignment 6

https://github.com/mvahoraUW/IntroToProg-Python-Mod06

**Working with Dictionaries, Files, and Functions**

**Introduction**

The script made for this assignment reads a file called “TodDoList.txt” with a list of tasks and priorities and processes, displays, edits the dataset in the file using a menu of options provided to the user by the program. The file is then updated with the user modifications if they choose to save their edits to file. This assignment is similar to Assignment 6, where the user is modifying a template to complete the tasks outlined, but the main difference is we are now using functions to complete the tasks.

**Steps Taken**

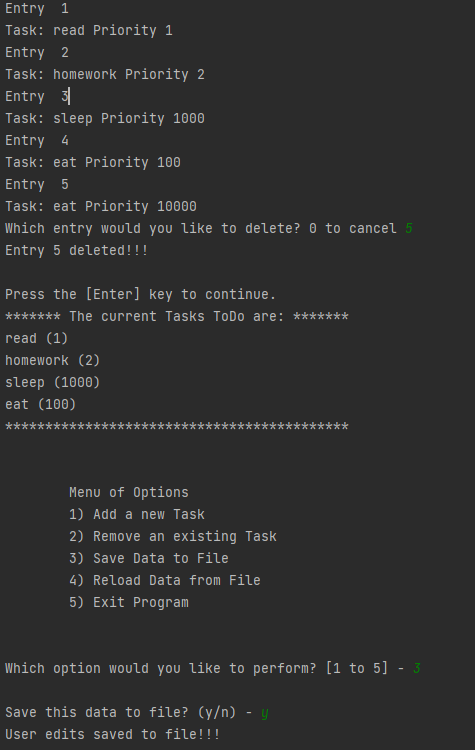
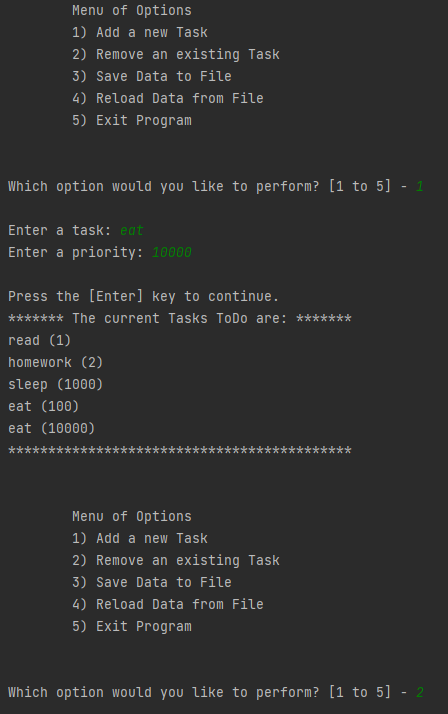
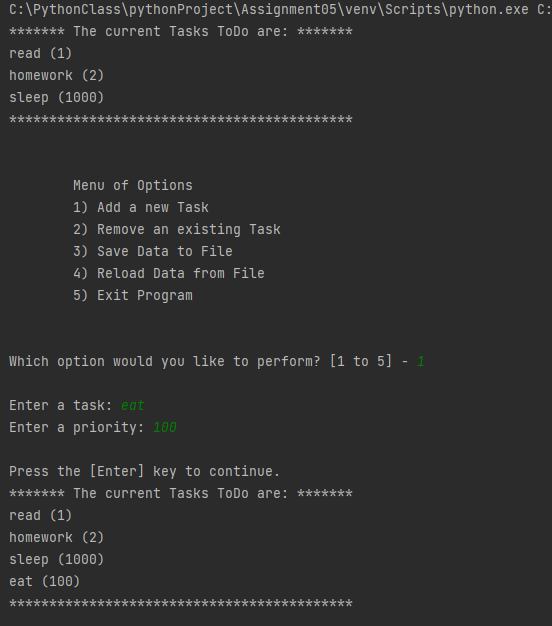
These are the step-by-step tasks taken by the author of the code to create the program used to load, edit, and save the ToDoList.txt file. This is broken down into 4 sections consisting of Data section to declare all the variables used in the program, a processing function section to process data loaded into the program, a IO function to process the inputs and outputs tasks such loading data and taking user input, and a main body section uses the functions to perform the following tasks:

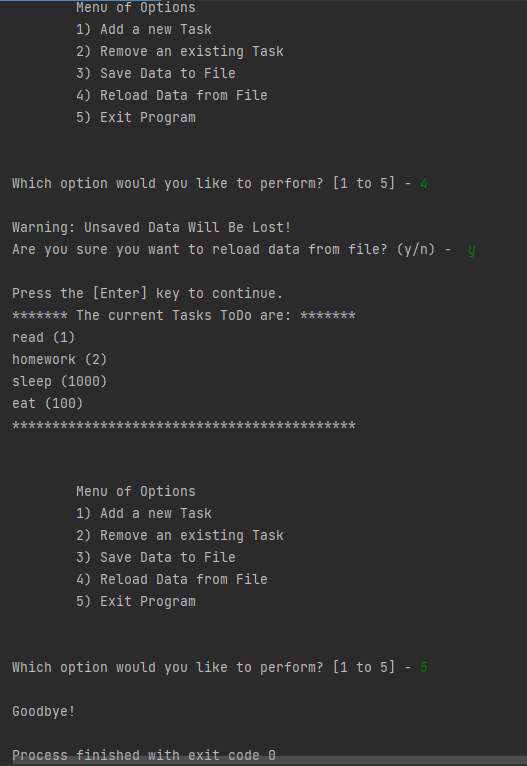
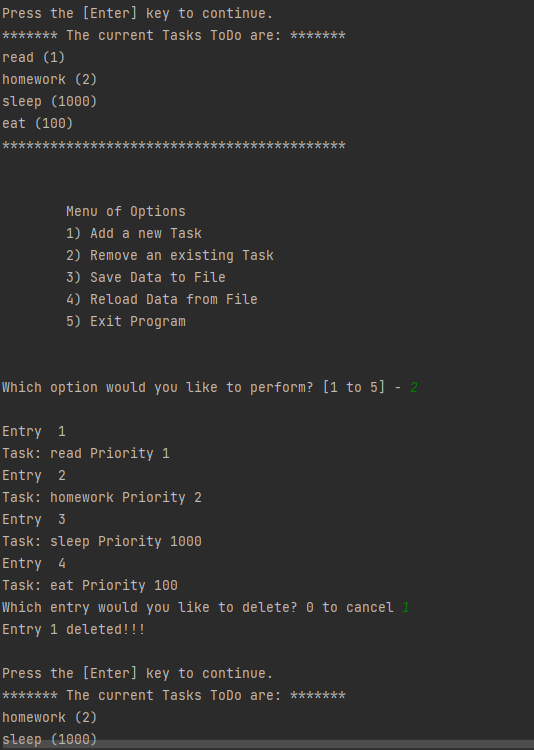
* 1. Load data from ToDoList.txt file using a function in the processor section to load data into a list for the user to process. After loading the following sections are repeated in a while loop until the user breaks it by entering option ‘5’.
  2. Displaying Menu of options to user and current data
     1. This section displays the options for the user for the list used in the data set.
     2. Print the tasks and priorities loaded into list variable using functions
     3. Print the menu tasks using using function.
     4. Obtain the menu choice using a function.
  3. Remove item from list/table
     1. This is option 2 and it displays the current data set and its index using a loop. The user is then prompted to selected which entry they would like to delete by inputting the entry number (0 if they would like to cancel) using a function in the IO section and a function in the processor section is called to remove data from the list using the user input. A message then displays the entry number deleted by the user.
  4. Save Data to File
     1. This is option 3 and the program uses a function in the IO section to ask the user if they would like to save. If the user chooses yes, a function in the processing section is called to open the ToDoList.txt file and overwrites all the information in the file using the “w” command in the open function. A for loop is used to write the each row entry from list into the ToDoList.txt file, where each row entry is a dictionary entry, which each string is separated using a comma along with a new row at the end. The file is then closed a message is printed to inform the user their edits were saved.
  5. Reload Data from file
     1. This section runs a function in the IO section to ask the user if they would like to overwrite their changes. If ‘the user chooses yes, the same function used to load the initial list is used to reload the data.
  6. Exit program

**Python Code**

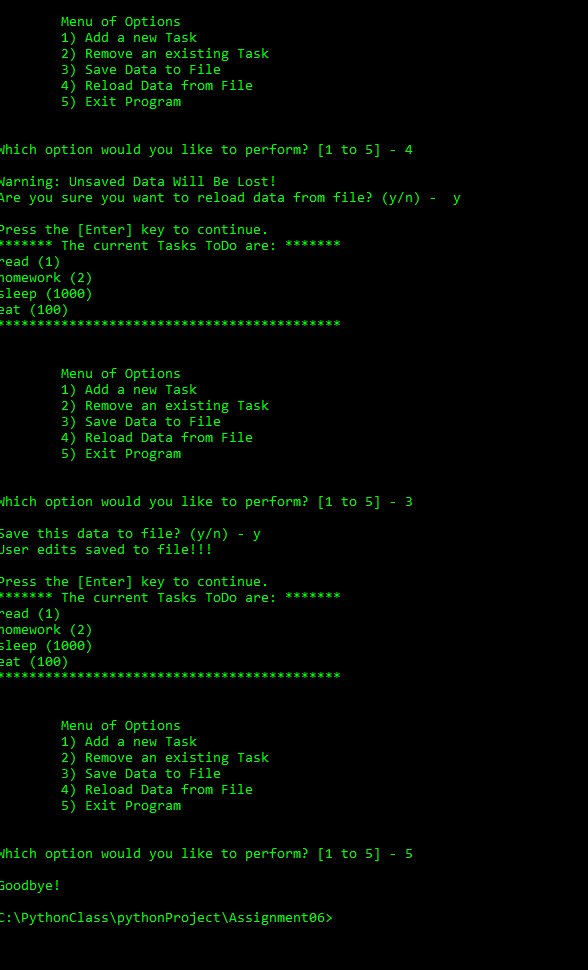
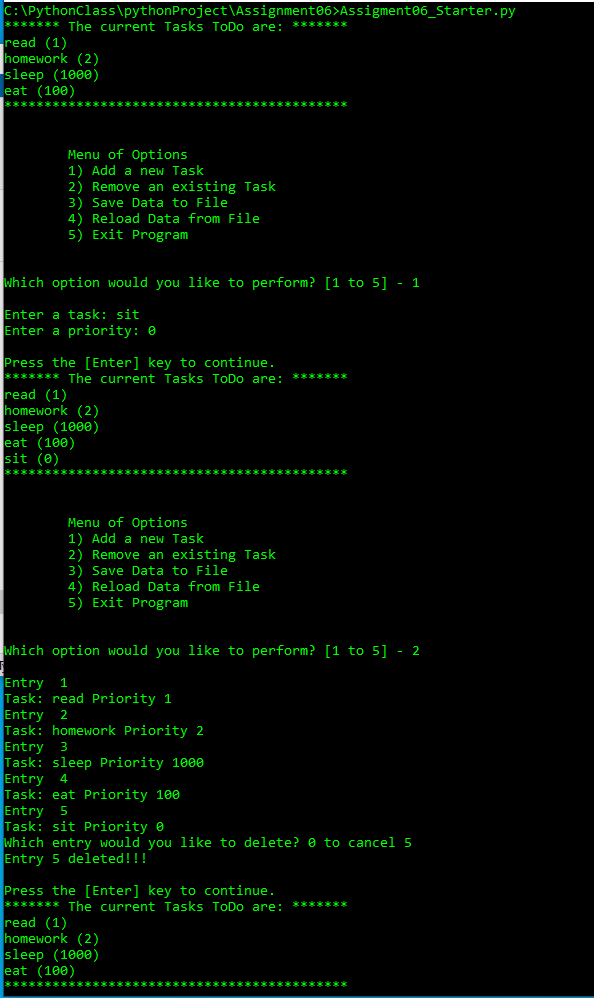
# ---------------------------------------------------------------------------- #  
# Title: Assignment 06  
# Description: Working with functions in a class,  
# When the program starts, load each "row" of data  
# in "ToDoToDoList.txt" into a python Dictionary.  
# Add the each dictionary "row" to a python list "table"  
# ChangeLog (Moiz Vahora,8/16/2021,modified starter.py file):  
# RRoot,1.1.2030,Created started script  
# RRoot,1.1.2030,Added code to complete assignment 5  
# <Moiz Vahora>,<08.16.2021>,Modified code to complete assignment 6  
# ---------------------------------------------------------------------------- #  
  
# 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")  
 # loop used to populate list\_of\_rows list  
 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):  
 # TODO: Add Code Here!  
  
 # appending list variable with user input using a dictionary  
 dicRow = {"Task": task, "Priority": priority }  
 list\_of\_rows.append(dicRow)  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def remove\_data\_from\_list(task, list\_of\_rows):  
 # TODO: Add Code Here!  
  
 # conditional statement used to remove from list, if user enters 0 nothing is deleted.  
 if task == 0:  
 # cancel deletion and return to menu  
 print("No user input deleted")  
 else:  
 # Removing selected entry (subtract 1 since count starts at 0) and displaying entry number deleted  
 list\_of\_rows.remove(list\_of\_rows[task - 1])  
 print("Entry " + str(task) + " deleted!!!")  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def write\_data\_to\_file(file\_name, list\_of\_rows):  
 # TODO: Add Code Here!  
 # Opening file and overwriting my previous input, in case we deleted entries  
 objfile\_io = open(file\_name, "w")  
  
 for objrow in list\_of\_rows:  
 # Writing entries in list to the file from each dictionary entry  
 objfile\_io.write(objrow["Task"] + ',' + objrow["Priority"] + '\n')  
 objfile\_io.close() # close file  
 print("User edits saved to file!!!")  
 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():  
 *""" asks the user to input a new task and priority*  
  
***:return****:nothing*  
 *"""*  
task = input("Enter a task: ")  
 priority = input("Enter a priority: ")  
 return task, priority  
  
 @staticmethod  
 def input\_task\_to\_remove():  
 #pass # TODO: Add Code Here!  
 *""" Asks the user which list item they would like deleted using a numerical list*  
  
***:return****:*  
 *"""*  
task = int(input("Which entry would you like to delete? 0 to cancel "))  
 return task  
  
# 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  
 # TODO: Add Code Here  
 I\_task, I\_priority = IO.input\_new\_task\_and\_priority()  
 Processor.add\_data\_to\_list(I\_task,I\_priority,lstTable)  
  
 IO.input\_press\_to\_continue(strStatus)  
  
 continue # to show the menu  
  
 elif strChoice == '2': # Remove an existing Task  
 # TODO: Add Code Here  
 for i\_entry, objRow in enumerate(lstTable):  
 # displaying current data for user reference (copied from option 1)  
 print("Entry ", i\_entry + 1)  
 print('Task: ' + objRow["Task"] + ' ' + 'Priority ' + objRow["Priority"])  
 # Calling function to ask for user input for data deletion  
 Delete = IO.input\_task\_to\_remove()  
  
 # calling function to remove data from list  
 Processor.remove\_data\_from\_list(Delete, lstTable)  
 # Calling funtion to ask the user to press enter to continue  
 IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu  
  
 elif strChoice == '3': # Save Data to File  
  
 #calling function get user input if they would like to save  
 strChoice = IO.input\_yes\_no\_choice("Save this data to file? (y/n) - ")  
 #conditional statement y to save, n to continue  
 if strChoice.lower() == "y":  
 # TODO: Add Code Here!  
 # Opening file and overwriting my previous input, in case we deleted entries.  
  
 Processor.write\_data\_to\_file(strFileName,lstTable)  
  
 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!")  
 # Ask user if they would like reload  
 strChoice = IO.input\_yes\_no\_choice("Are you sure you want to reload data from file? (y/n) - ")  
  
 #Conditional statement to detemrine, y to reload data, n to cancel and continue  
 if strChoice.lower() == 'y':  
 # TODO: Add Code Here!  
 # Call function to re-read data, using the same function to load the data when the script was started  
 Processor.read\_data\_from\_file(strFileName, lstTable)  
 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

**PyCharm Display**





**Command Prompt Display**



**Conclusion**

The script written for this assignment is to take data from the ToDoList.txt file and display/manipulate the data using dictionaries and lists. This differs from the previous assignment as a series of functions are used to process the data. There were some challenges understanding how the functions were used how to use each of them as intended, but overall the program can perform all the required tasks.