Charles Tung

May 16, 2021

Student ID: 9721184

Foundations of Programming, Python

Assignment 06

# Assingment06\_Charles.py

## Introduction

Homework Assignment06 of the Foundations of Programming, Python class involved creating a Assignment06\_Charles.py script. Assignment06\_Charles.py script is an extension of the Assignment05.py script that reads in a text file of tasks and produces a list of dictionaries. Except in Assignment06\_Charles.py script the file demonstrates the additional capability of classes and functions. The functions are called in the main event while loop within the class Processor.

## Show Current Data

The below screen shot in figure 1 show the current data being shown (Option 1):

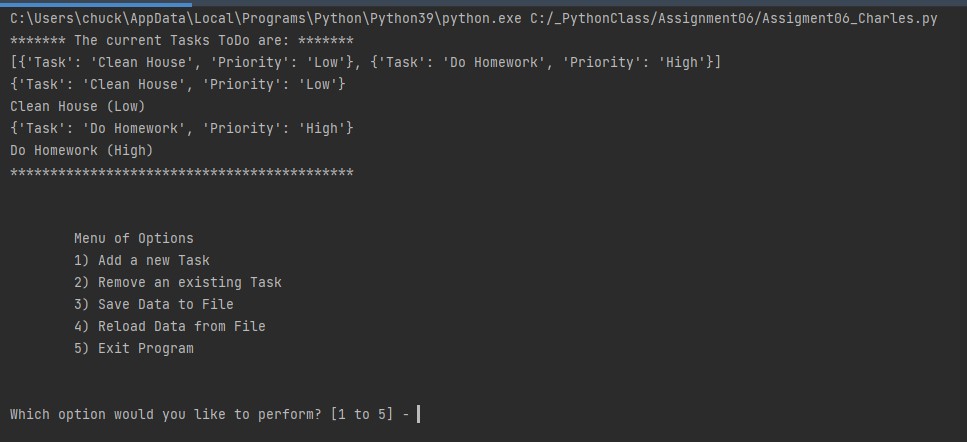


Figure 1: Screen Shot of PyCharm Showing the Current Data

## Add a New Item

The below screen shot in figure 2 shows the (Option 2) adding a new item:

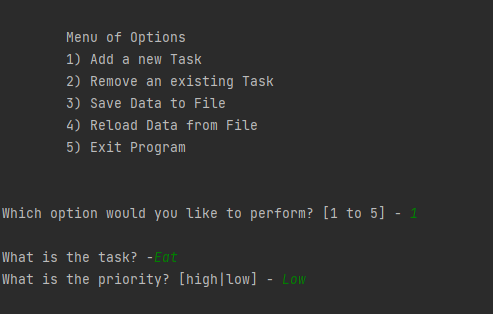


Figure 2: Screen Shot of PyCharm showing Option 1 (Add a new Item)

The below screen shot in figure 3 shows (Option 1) and the new item that was added below:

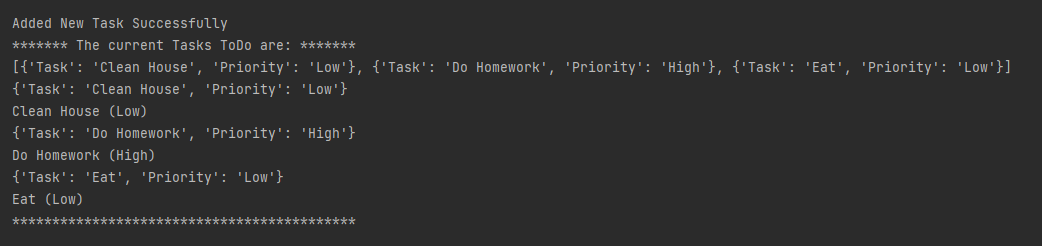


Figure 3: Screen Shot of PyCharm showing Option 1 and the additional “Eat” Task

## Save Data to File:

Option 4 was selected in figure 4 below which saved the file to a text file as shown below:

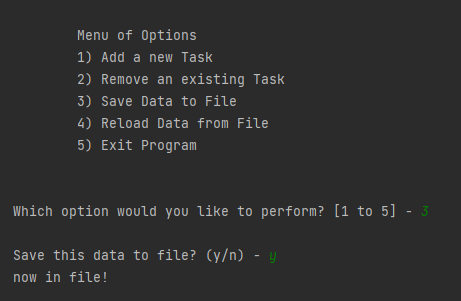


Figure 4: Screen Shot of PyCharm showing Option 4 and Saving the Data to a File

Below shows the text file ToDoList.txt output in Figure 5 below and the additional Eat task from the action shown in Figure 4 above:

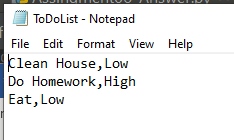


Figure 5: Screen Shot of ToDoList.txt

## Remove a task:

The script can also remove a task. Below Figure 6 shows the task Eat removed as shown below:

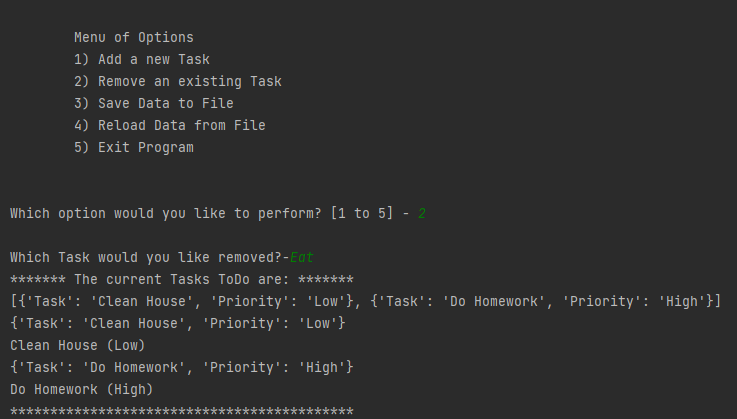


Figure 6: Screen Shot of Pycharm Showing the Task “Eat” Being Removed.

## Windows Command Screen Output

Below shows the screen shot (figure 7) from the windows command screen showing that the identical image from the PyCharm screenshot in Figure 1:

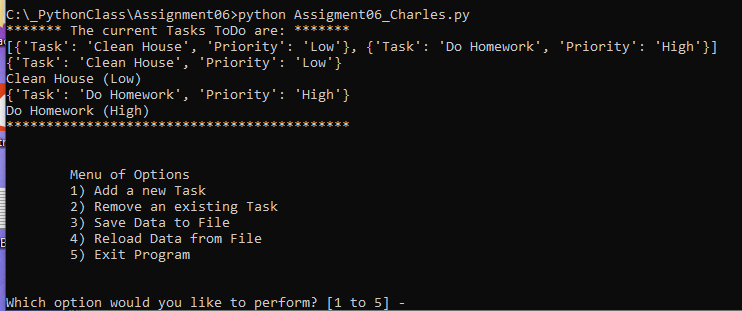


Figure 7: Windows Command Output of Assigment06\_Charles.py Script

## Source Code

The source code for Assigment06\_Charles.py was as shown in the PyCharm terminal as shown below:

# ---------------------------------------------------------------------------- #  
# 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  
# CTUNG, 5.15.2021,Modified code to complete assignment 6. Add functions  
# To Read and Write CSV Text File (ToDoList.txt) with Task and Priority,  
# Add Task to List, and Remove Task from Dictionary of Lists, Add Callouts  
# To Main Body of Script Calling out Functions  
# ---------------------------------------------------------------------------- #  
  
# Data ---------------------------------------------------------------------- #  
# Declare variables and constants  
strFileName = "ToDoList.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):  
 *""" Reads data from a file into a list of dictionary rows* ***:param*** *file\_name: (string) with name of file:* ***:param*** *list\_of\_dictionary\_rows: (list) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* list\_of\_dictionary\_rows = []  
 file = open(file\_name, "r")  
 for line in file:  
 data = line.split(",")  
 row = {"Task": data[0].strip(), "Priority": data[1].strip()}  
 list\_of\_dictionary\_rows.append(row)  
 file.close()  
 return list\_of\_dictionary\_rows, 'Success'  
  
 @staticmethod  
 def add\_data\_to\_list(list\_of\_dictionary\_rows, task, priority):  
 *"""Adds data to a list of dictionary row* ***:param*** *list\_of\_dictionary\_rows" (list) of dictionary adding data to it* ***:param*** *task: (string) with name of task* ***:param*** *priority: (string) with name of priority  
 """* row = {"Task": str(task).strip(), "Priority": str(priority).strip()}  
 list\_of\_dictionary\_rows.append(row)  
  
 @staticmethod  
 def remove\_data\_from\_list(list\_of\_dictionary\_rows, strKeyToRemove):  
 *"""Removes data from a list of dictionary rows* ***:param*** *list\_of\_dictionary\_rows: (list) of dictionary rows removing from it* ***:param*** *strKeyToRemove: Task to Remove* ***:return****: Revised list of dictionary rows  
 """* sucess\_status = False  
 row\_number = 0  
 for row in list\_of\_dictionary\_rows:  
 task, priority = row.values()  
 #why does this not work  
 if task == strKeyToRemove:  
 del lstTable[row\_number]  
 sucess\_status = True  
 row\_number += 1  
 return list\_of\_dictionary\_rows, 'Success'  
  
 @staticmethod  
 def write\_data\_to\_file(file\_name, list\_of\_rows):  
 # TODO: Add Code Here!  
 f = open(file\_name, "w")  
 for row in list\_of\_rows:  
 f.write(str(row["Task"]) + ',' + str(row["Priority"] + '\n'))  
 f.close()  
 print("now in 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(lstTable):  
 *""" 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: \*\*\*\*\*\*\*")  
 print(lstTable)  
 for row in lstTable:  
 print(row)  
 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():  
 *"""Gets data for a dictionary rows* ***:return****: (tuple) of string with tasks and priority  
 """* task = str(input("What is the task? -")).strip()  
 priority = str(input("What is the priority? [high|low] - ")).strip()  
 print() # Add an extra lin for looks  
 return task, priority  
  
 @staticmethod  
 def input\_task\_to\_remove():  
 *"""Gets a task from the user to remove* ***:return****: (string) task to remove  
 """* task = str(input("What task would you like to remove? - ")).strip()  
 return task  
  
  
# Main Body of Script ------------------------------------------------------ #  
  
# Step 1 - When the program starts, Load data from ToDoFile.txt.  
lstTable, status = Processor.read\_data\_from\_file(strFileName) # 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': # Adds a New Tasks  
 tplData = IO.input\_new\_task\_and\_priority() # Outputs Tuple with New Task & Priority  
 Processor.add\_data\_to\_list(lstTable, tplData[0], tplData[1]) # Adds Data to List  
 IO.print\_current\_Tasks\_in\_list(lstTable)  
 print("Added New Task Successfully")  
 continue # to show the menu  
  
 elif strChoice == '2': # Removes an Existing Task  
 strKeyToRemove = input("Which Task would you like removed?-")  
 blnItemRemoved = Processor.remove\_data\_from\_list(lstTable, strKeyToRemove)  
 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":  
 Processor.write\_data\_to\_file(strFileName, lstTable) # Writes 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':  
 lstTable.clear()  
 lstTable, status = Processor.read\_data\_from\_file(strFileName) #Outputs List and Sucess  
 IO.print\_current\_Tasks\_in\_list(lstTable)  
 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 8: Screenshot of Source Code to Assigment06\_Charles.py Written in PyCharm’s Terminal

The main event loop is the While Loop that calls out difference functions organized in the two classes: Processor and IO. The IO Class functions are used to help the input and outputting of the script. The Processor class functions are used to help process the data within the script.

## Summary

This paper summarizes some of the Assigment06\_Charles.py script. This script is an extension of the Assigment05.py script that reads in and writes to a list of tasks to the ToDoList.txt file. This script demonstrated the ability of Python to use both classes and functions. Classes are used to help organize the functions to help make the script easier to read.