Katrina Taylor

August 17, 2020

Foundations of Programming: Python

Assignment06

<https://github.com/katrinataylor/ITFnd110-Mod06>

**Classes and Functions**

# **Introduction**

The goal of this assignment is to modify a script that manages a To Do list. This script already loads data from a file into a Python List of Dictionary objects and I’ve added more functions to organize the code.

# **Text File**

First, I had to create the text file my program would load titled ToDoList.txt (Figure 1).

A screenshot of a cell phone

Description automatically generated

***Figure 1: ToDoList.txt text file***

# **Python Script**

Next, I started working on the additional Python Script that would carry out the program through functions.

## **Starter Script**

This particular program then required use of a starter script (Figure 3).

# ---------------------------------------------------------------------------- #  
# 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  
# <Your Name>,<Date>,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")  
 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!* return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def remove\_data\_from\_list(task, list\_of\_rows):  
 # *TODO: Add Code Here!* return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def write\_data\_to\_file(file\_name, list\_of\_rows):  
 # *TODO: Add Code Here!* 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():  
 pass # *TODO: Add Code Here!* # return task, priority  
  
 @staticmethod  
 def input\_task\_to\_remove():  
 pass # *TODO: Add Code Here!* # 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* IO.input\_press\_to\_continue(strStatus)  
 continue # to show the menu  
  
 elif strChoice == '2': # Remove an existing Task  
 # *TODO: Add Code Here* 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":  
 # *TODO: Add Code Here!* 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':  
 # *TODO: Add Code Here!* 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 3: The starter script***

## **Adding a New Task to the ToDo List**

The end user is presented with 5 options on how to proceed while running the program. I started with writing the script for option 1: adding a new task to the ToDo list (Figure 4).

if strChoice.strip() == '1': # Add a new Task  
 strTask, strPriority = IO.input\_new\_task\_and\_priority() # Get the task name and priority  
 lstTable, strStatus = Processor.add\_data\_to\_list(strTask, strPriority, lstTable) # Add to list  
 IO.input\_press\_to\_continue(strStatus) # Get status  
 continue # to show the menu

***Figure 4: The script for adding a new task to the ToDo list***

In order to capture what the end user would like to add, the script here calls for class “IO” and specifically function “input\_new\_task\_and\_priority”. This function uses input() to capture both the new task and it’s priority and then returns those two strings in a tuple (Figure 5). This tuple is then saved to variables in the main body of the script which in turn is passed through the next function (Figure 6).

@staticmethod  
def input\_new\_task\_and\_priority():  
 *""" Gets the new task and priority the user wants to add* ***:return****: task (string) and priority (string)  
 """* task = input('Enter a task to add: ')  
 priority = input('Enter the task\'s priority: ')  
 return task, priority

***Figure 5: The script for collecting the new task and priority from the end user***

Next, the program needed to perform the actual step of adding the task and priority to the list of dictionaries. To do this, the script calls on class “Processor” and function “add\_data\_to\_list”. This function adds the user’s input into a dictionary and then uses append() to add it to the current list. It returns the new list and the ‘Success’ status in a tuple (Figure 6).

@staticmethod  
def add\_data\_to\_list(task, priority, list\_of\_rows):  
 *""" Adds task and priority from user input into a dictionary then adds dictionary to list* ***:param*** *task: (string) with name of task:* ***:param*** *priority: (string) with priority of task:* ***:param*** *list\_of\_rows: (list) you want appended with user input data:* ***:return****: (list) of dictionary rows  
 """* row = {"Task": task.strip(),"Priority": priority.strip()}  
 list\_of\_rows.append(row)  
 return list\_of\_rows, 'Success'

***Figure 6: The script for adding the task and priority into the list dictionary***

## **Removing a Task from the ToDo List**

I then worked on the script for option 2: removing an existing task from the ToDo list (Figure 7).

elif strChoice == '2': # Remove an existing Task  
 strTask = IO.input\_task\_to\_remove() # Get the task name  
 lstTable, strStatus = Processor.remove\_data\_from\_list(strTask, lstTable) # Remove from list  
 IO.input\_press\_to\_continue(strStatus) # Get status  
 continue # to show the menu

***Figure 7: The script removing an existing task***

In order to capture what the end user would like to remove, the script here calls for class “IO” and specifically function “input\_ task\_to\_remove”. This function uses input() to capture said the task returns the string (Figure 8). This string is then saved to variables in the main body of the script which in turn is passed through the next function (Figure 9).

@staticmethod  
def input\_task\_to\_remove():  
 *""" Gets the task the user wants to remove* ***:return****: task (string)  
 """* task = input('Enter a task to remove: ')  
 return task

***Figure 8: The script for collecting the task to be removed from the end user***

Next, the program needed to perform the actual step of removing the task dictionary from the list of dictionaries. To do this, the script calls on class “Processor” and function “remove\_data\_from\_list”. This function uses a for loop to run through each dictionary in the list to check if it the dictionary to remove, and if it is then the remove() function is used to actually remove that dictionary. It returns the new list returns the new list and the ‘Success’ status in a tuple (Figure 9).

@staticmethod  
def remove\_data\_from\_list(task, list\_of\_rows):  
 *""" Uses user input to remove a task's dictionary from list* ***:param*** *task: (string) with name of task:* ***:param*** *list\_of\_rows: (list) you want the task's dictionary removed from:* ***:return****: (list) of dictionary rows  
 """* for row in list\_of\_rows:  
 if row.get("Task") == task:  
 list\_of\_rows.remove(row)  
 return list\_of\_rows, 'Success'

***Figure 9: The script for removing a task dictionary from the overall list dictionary***

## **Saving the current ToDo List to a File**

I then worked on the script for option 3: saving the current ToDo list to a file (Figure 10).

elif strChoice == '3': # Save Data to File  
 strChoice = IO.input\_yes\_no\_choice("Save this data to file? (y/n) - ") # Get user response to continue  
 if strChoice.lower() == "y": # Continue with saving data to file  
 lstTable, strStatus = Processor.write\_data\_to\_file(strFileName, lstTable) # Save the data to the file  
 IO.input\_press\_to\_continue(strStatus) # Get status  
 else:  
 IO.input\_press\_to\_continue("Save Cancelled!") # Cancel current operation to save data to file  
 continue # to show the menu

***Figure 10: The script in the main body of the script saving the current ToDo List to a file***

In order to perform the save, the script here calls for class “Processor” and specifically function “write\_data\_to\_file”. The function starts by opening our file in “w” writing mode. Then a for loop is used to iterate through our current list of dictionaries and adds the dictionary information to our file one-by-one using the write() function. Finally, the filed is closed and the new list of dictionaries from the newly saved files is returned with the ‘Success’ status in a tuple (Figure 11).

@staticmethod  
def write\_data\_to\_file(file\_name, list\_of\_rows):  
 *""" Saves current list to file* ***:param*** *file\_name: (string) with name of file:* ***:param*** *list\_of\_rows: (list) you want to save to file:* ***:return****: (list) of dictionary rows  
 """* file = open(file\_name, "w")  
 for row in list\_of\_rows:  
 task = row.get('Task')  
 priority = row.get('Priority')  
 file.write(task + ',' + priority + '\n')  
 file.close()  
 return list\_of\_rows, 'Success'

***Figure 11: The script in the write\_data\_to\_file function saving the current ToDo List to a file***

## **Reload Data that’s Currently Stored in File**

I then worked on the script for option 4: reloading the data that’s currently stored in the file. To accomplish this, I used a function that was already provided in the starter file, “read\_data\_from\_file” (from the “Processor” class) which returned the refreshed list of dictionaries created from the file and the ‘Success’ status in a tuple (Figure 12).

elif strChoice == '4': # Reload Data from File  
 print("Warning: Unsaved Data Will Be Lost!") # Warn user actions of adding/removing data will be wiped clean  
 strChoice = IO.input\_yes\_no\_choice("Are you sure you want to reload data from file? (y/n) - ") # get user response to continue  
 if strChoice.lower() == 'y':  
 lstTable, strStatus = Processor.read\_data\_from\_file(strFileName, lstTable) # Reload file data  
 IO.input\_press\_to\_continue(strStatus) # Get status  
 else:  
 IO.input\_press\_to\_continue("File Reload Cancelled!") # Cancel current operation to reload data  
 continue # to show the menu

***Figure 12: The script to reload data from the file***

## **Exiting the Program**

The last option, option 5, to exit the program was already written (Figure 3) and therefore the script was now complete.

# **Script Output**

When the script is ran, the end user will not see the header comments or any other comments. The end user will only see the statements printed from the print() and input() functions, in addition to whatever they type when they interact with the program. The script was successful at accomplishing the goal of this assignment (Figure 13 and Figure 14).

/Users/katrinataylor/Documents/\_PythonClass/Assignment06/venv/bin/python /Users/katrinataylor/Documents/\_PythonClass/Assignment06/Assignment06\_Starter.py

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

Wash dishes (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 1

Enter a task to add: Walk

Enter the task's priority: High

Success

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

Wash dishes (Medium)

Walk (High)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 3

Save this data to file? (y/n) - y

Success

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

Wash dishes (Medium)

Walk (High)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 2

Enter a task to remove: Walk

Success

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

Wash dishes (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 4

Warning: Unsaved Data Will Be Lost!

Are you sure you want to reload data from file? (y/n) - y

Success

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks ToDo are: \*\*\*\*\*\*\*

Wash dishes (Medium)

Walk (High)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 5

Goodbye!

Process finished with exit code 0

***Figure 13: The output in PyCharm after running and interacting with my program***

***A screenshot of text

Description automatically generated***

***A screenshot of a cell phone

Description automatically generated***

***Figure 14: The output in Terminal after running and interacting with my program***

The saved file now contains the input the user provided (Figure 15).

***A screenshot of a cell phone

Description automatically generated***

***Figure 15: The text file shows the saved data***

# **Summary**

In this assignment I modified a script that manages a To Do list. This script already loaded data from a file into a Python List of Dictionary objects and I added more functions to organize the code.