**Samir Abad**

**May 23, 2022**

**IT FDN 110A Sp. 22**

**Assignment 06**

**Github Link-** <https://github.com/mailsamir/IntroToProg-Python-Mod06>

**Task and Priority Script**

**Introduction:** In this paper I am going to talk about how I created a new python script that asks the user to add or remove a task and the corresponding priority from a list of tasks and priorities. The "ToDoFile" file will contain two columns of data, "Task" and "Priority." Load the columns into a Python Dictionary object. Each dictionary object represents one row of data, and these rows must be added to a Python *List* object to create a table of data. A starting template was provided and the missing code was added inorder to execute the option selected by the user. This script emphasizes the concept around separation of concerns by using classes and functions. Separate classes and underlying functions were created to address the processing and presentation parts.

The following information shows the detailed steps demonstrating how I went about writing the script.

**Processing functions**

**Function 1:** Define a function to read data from a file

There are 2 parameters that form the function read\_data\_from\_file. Parameter 1 is the name of the file that has the tasks and priorities data. Parameter 2 is the list that will store the task and priority data into a list. A for loop was called to read every row of the file, split the comma separated data on each row and then append the row data into a list. The function will return the list of rows.

The code for function 1 is shown below-

@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

**Function 2:** Define a function to add data to a list

This function uses 3 parameters. Parameter 1 and 2 is the task and priority to be added. Parameter 3 is the list to which the new data (task, priority) is added. The new row data is a dictionary with Task and Priority added as new data elements (key/value pair). The new row is subsequently appended and the list is then returned.

@staticmethod  
def add\_data\_to\_list(task, priority, list\_of\_rows):  
 *""" Adds data to a list of dictionary rows* ***:param*** *task: (string) with name of task:* ***:param*** *priority: (string) with name of priority:* ***:param*** *list\_of\_rows: (list) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* row = {"Task": str(task).strip(), "Priority": str(priority).strip()}  
 # *TODO: Add Code Here!* list\_of\_rows.append(row)  
 return list\_of\_rows

**Function 3:** Remove data from a list

In order to remove a row there are 2 parameters defined -task to be removed and the list from which the row is to be removed. A for loop going through each row and checking if the row contains the task string to be removed is used to find the row. Once the row is identified then that row is removed and the list of rows with remaining data is returned.

@staticmethod  
def remove\_data\_from\_list(task, list\_of\_rows):  
 *""" Removes data from a list of dictionary rows* ***:param*** *task: (string) with name of task:* ***:param*** *list\_of\_rows: (list) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* # *TODO: Add Code Here!* for row in list\_of\_rows:  
 if row["Task"].lower() == task.lower():  
 list\_of\_rows.remove(row)  
 return list\_of\_rows

**Function 4:** Write data to a file

The file name and the list within the file are parameters defined in this function. The file is opened with a write option and the for loop goes through every row in the list and writes the task and priority data in the file. The task and priority are separated by comma.

@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) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* # *TODO: Add Code Here!* file = open(file\_name, "w")  
 for row in list\_of\_rows:  
 file.write(row["Task"] + ',' + row["Priority"] + '\n')  
 file.close()  
 return list\_of\_rows

**Presentation functions:**

Function 1: Print menu of options

A print menu within the function is used to print the 4 options to the user. No parameters are required.

@staticmethod  
def output\_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) Exit Program  
 ''')  
 print() # Add an extra line for looks

Function 2: Gets the menu choice from the user

An input function is used to ask the choice from the user. And the choice is returned.

@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 4] - ")).strip()  
 print() # Add an extra line for looks  
 return choice

Function 3: Displays current tasks in the list

The list is used as a parameter that is to be displayed. A for loop goes through every row in the list and prints each row (comma separated task and priority data).

@staticmethod  
def output\_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

Function 4: Gets the task and priority to be added

Input functions are used to ask for the task and priority data to be added to the list. Both the task and priority data that is provided by the user is returned.

@staticmethod  
def input\_new\_task\_and\_priority():  
 *""" Gets task and priority values to be added to the list* ***:return****: (string, string) with task and priority  
 """* task = str(input("Enter a new task:"))  
 priority = str(input("Enter the priority:"))  
 return task, priority

Function 5: Gets the task name to be removed from the list

This function asks the user to input the task that is to be removed from the list.

@staticmethod  
def input\_task\_to\_remove():  
 *""" Gets the task name to be removed from the list* ***:return****: (string) with task  
 """* task = str(input("Enter the task to be removed:"))  
 return task

**Main body**

This code was provided and hence not edited.

Step1- The data from the file ToDoFile.txt is loaded. The file name- ToDoFile.txt and the table list is provided as arguments to the read\_data\_from\_file function.

Step2- To show the current data, the table list is provided as an argument to the output\_current\_tasks\_in\_list function. Once the data is displayed the menu options is displayed using the output\_menu\_tasks() function. The menu choice provided by the user is saved in a choice\_str variable.

If the user selects option 1 (to add data to file) then the task and priority returned by the input\_new\_task\_and\_priority() function are saved in the table list and appended to existing row of data using the add\_data\_to\_list function.

If the user selects option2 (to remove a task), then the task provided by the user as input is searched through the rows the data and if found the row is removed using the remove\_data\_from\_list() function.

If the user selects option 3 (to save the file), then the name of the file and the table list is passed on as arguments. The function write\_data\_to\_file is used to write every row of data to the file.

Option 4 exits the program.

# Step 1 - When the program starts, Load data from ToDoFile.txt.  
Processor.read\_data\_from\_file( file\_name=file\_name\_str, list\_of\_rows=table\_lst) # read file data  
  
# Step 2 - Display a menu of choices to the user  
while (True):  
 # Step 3 Show current data  
 IO.output\_current\_tasks\_in\_list(list\_of\_rows=table\_lst) # Show current data in the list/table  
 IO.output\_menu\_tasks() # Shows menu  
 choice\_str = IO.input\_menu\_choice() # Get menu option  
  
 # Step 4 - Process user's menu choice  
 if choice\_str.strip() == '1': # Add a new Task  
 task, priority = IO.input\_new\_task\_and\_priority()  
 table\_lst = Processor.add\_data\_to\_list(task=task, priority=priority, list\_of\_rows=table\_lst)  
 continue # to show the menu  
  
 elif choice\_str == '2': # Remove an existing Task  
 task = IO.input\_task\_to\_remove()  
 table\_lst = Processor.remove\_data\_from\_list(task=task, list\_of\_rows=table\_lst)  
 continue # to show the menu  
  
 elif choice\_str == '3': # Save Data to File  
 table\_lst = Processor .write\_data\_to\_file(file\_name=file\_name\_str, list\_of\_rows=table\_lst)  
 print("Data Saved!")  
 continue # to show the menu  
  
 elif choice\_str == '4': # Exit Program  
 print("Goodbye!")  
 break # by exiting loop

Text

Description automatically generated

Figure1- Output of step1

Text

Description automatically generated

Figure2- Output of step2

Text

Description automatically generated

Figure3- Output of step3

Text

Description automatically generated

Figure4- Output of step4

**Summary:**

Assignment6 expanded upon the previous assignment by introducing concepts around separation of concerns by creating classes and functions. The skill to add to an existing code is an important skill and this assignment provided exposure on how to go about editing existing code. This program also allowed me to explore PyCharm by writing and testing the program in the PyCharm development environment. Overall, this assignment was very helpful in introducing basic python scripting to a novice programmer.