```python
# ---------------------------------------------------------------------------- #
# 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
# RRoot,1.1.2030,Modified code to complete assignment 6
# ---------------------------------------------------------------------------- #

# Data ---------------------------------------------------------------------- #
# Declare variables and constants
strFileName = "ToDoFile.txt"  # The name of the data file
strChoice = ""  # Capture the user option selection
# objFile = None   # An object that represents a file
# lstTable = []  # A dictionary that acts as a 'table' of rows
# strData = ""  # A row of text data from the file
# dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
# Data ---------------------------------------------------------------------- #


# Processing  --------------------------------------------------------------- #
class DataProcessor:
    """ Processes the data in a list of dictionaries to and from a text file """

    @staticmethod
    def read_file_to_list_of_dictionaries(file_name):
        """ Reads data from a file into a list of dictionary rows

        :param file_name: (string) with name of file
        :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

    @staticmethod
    def write_file_from_list_of_dictionaries(file_name, list_of_dictionary_rows):
        """ Write data to a file from a list of dictionary rows

        :param file_name: (string) with name of file
        :param list_of_dictionary_rows: (list) of dictionary data saved to file
        :return: (bool) with status of success status
        """
        success_status = False
        file = open(file_name, "w")
        for row in lstTable:
            file.write(row["Task"] + "," + row["Priority"] + "\n")
        file.close()
        success_status = True
        return success_status

    # TODO: Create more functions that perform various Processing task as needed (Done)
    @staticmethod
    def add_data_to_list_of_dictionaries(list_of_dictionary_rows, task, priority):
        """ Adds data to a list of dictionary rows

        :param list_of_dictionary_rows: (string) with name of list your adding data to
        :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_of_dictionaries(list_of_dictionary_rows, task_to_remove):
        """ Removes a row of data from a list of dictionary rows

        :param list_of_dictionary_rows: (list) of dictionary data to remove a row from
        :param task_to_remove: (string) with name of the task in the dictionary's 'Task' key
        :return: (bool) with status of success status
        """
        success_status = False  # Create a boolean Flag for loop
        row_number = 0  # Create a counter to identify the current dictionary row in the loop
        # Search though the table or rows for a 'Task' key match
        while row_number < len(lstTable):
            # Search current row column 0
            if task_to_remove == str(list(dict(lstTable[row_number]).values())[0]):
                del lstTable[row_number]  # Delete the row if a match is found
                success_status = True  # Set the flag to indicate at least one row was removed
            row_number += 1  # Increase counter to get next row
        return success_status

# Processing  ------------------------------------------------------------- #

# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """ A class for perform Input and Output """

    @staticmethod
    def print_menu_items():
        """ Print a menu of choices to the user

        :return: nothing
        """
        print('''
        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Remove an existing item.
        4) Save Data to File
        5) Reload Data from File
        6) Exit Program
        ''')
        print()  # Add an extra line for looks

    @staticmethod
    def print_current_list_items(list_of_rows):
        """ Print the current items in the list of dictionaries rows

        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        print("******* The current items ToDo are: *******")
        for row in list_of_rows:
```

```python
            print(row["Task"] + " (" + row["Priority"] + ")")
        print("*******************************************")
        print()  # Add an extra line for looks

    @staticmethod
    def print_data_removed_status(success_status):
        """ Print the status of the task removal process

        :param success_status: (bool) status you want to display
        """
        if success_status:
            print("The task was removed.")
        else:
            print("I'm sorry, but I could not find that task.")
        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 6] - ")).strip()
        print()  # Add an extra line for looks
        return choice

    # TODO: Create more functions that perform various IO tasks as needed (Done)
    @staticmethod
    def input_task_and_priority():
        """ Gets data for a dictionary row

        :return: (tuple) of strings with task and priority
        """
        task = str(input("What is the task? - ")).strip()
        priority = str(input("What is the priority? [high|low] - ")).strip()
        print()  # Add an extra line for looks
        return task, priority

# Presentation (Input/Output)  -------------------------------------------- #

# Main Body of Script  ---------------------------------------------------- #

# Step 1 - When the program starts, Load data from ToDoFile.txt.
lstTable = DataProcessor.read_file_to_list_of_dictionaries(strFileName)  # read file data

# Step 2 - Display a menu of choices to the user
while True:
    IO.print_menu_items()  # Shows menu
    strChoice = IO.input_menu_choice()  # Get menu option

    # Step 3 - Process user's menu choice
    # Step 3.1 Show current data
    if strChoice.strip() == '1':
        IO.print_current_list_items(lstTable)
        continue  # to show the menu

    # Step 3.2 - Add a new item to the list/Table
    elif strChoice.strip() == '2':
        # Step 3.2.a - Ask user for new task and priority
        # ToDo: Place IO code in a new function (Done)
        tplData = IO.input_task_and_priority()
        # Step 3.2.b  Add item to the List/Table
        # ToDo: Place processing code in a new function (done)
        DataProcessor.add_data_to_list_of_dictionaries(lstTable, tplData[0], tplData[1])
        IO.print_current_list_items(lstTable)
        continue  # to show the menu

    # Step 3.3 - Remove a new item to the list/Table
    elif strChoice == '3':
        # Step 3.3.a - Ask user for item and prepare searching while loop
        # ToDo: Place processing code in a new function
        strKeyToRemove = input("Which TASK would you like removed? - ")  # get task user wants
deleted
        blnItemRemoved = DataProcessor.remove_data_from_list_of_dictionaries(lstTable,
strKeyToRemove)

        # Step 3.3.c - Update user on the status of the search
        # ToDo: Place IO code in a new function
        IO.print_data_removed_status(blnItemRemoved)

        # Step 3.3.d - Show the current items in the table
        IO.print_current_list_items(lstTable)  # Show current data in the list/table
        continue  # to show the menu

    # Step 3.4 - Save tasks to the ToDoFile.txt file
    elif strChoice == '4':
        # Step 3.4.a - Show the current items in the table
        IO.print_current_list_items(lstTable)  # Show current data in the list/table

        # Step 3.4.b - Ask if user if they want save that data
        # Double-check with user
        strYesOrNo = str(input("Save this data to file? (y/n) - ")).strip().lower()
        if "y" == strYesOrNo:
            # ToDo: Place processing code in a New function
            if DataProcessor.write_file_from_list_of_dictionaries(strFileName, lstTable):
                input("Data saved to file! Press the [Enter] key to return to menu.")
            else:  # Let the user know the data was not saved
                input("New data was NOT Saved, but previous data still exists! " +
                      "Press the [Enter] key to return to menu.")
            continue  # to show the menu

    # Step 3.5 - Reload data from the ToDoFile.txt file
    # (clears the current data from the list/table)
    elif strChoice == '5':
        # Warn user of data loss
        print("Warning: This will replace all unsaved changes. Data loss may occur!")
        strYesOrNo = input("Reload file data without saving? [y/n] - ")  # Double-check with
user
        if strYesOrNo.lower() == 'y':
            lstTable.clear()
            lstTable = DataProcessor.read_file_to_list_of_dictionaries(strFileName)
            IO.print_current_list_items(lstTable)
        else:
            input("File data was NOT reloaded! Press the [Enter] key to return to menu.")
            IO.print_current_list_items(lstTable)
        continue  # to show the menu

    # Step 3.6 - Exit the program
    elif strChoice == '6':
        break  # and Exit
```

```
# Main Body of Script  ---------------------------------------------------- #
```