Naga Anusha.Perali

February 21, 2022

Python Script Assignment 6

Working with Classes and Methods

Introduction

In this assignment, I'm going to explain the steps that I have used to create a Python Program with classes and methods, which loads the data available in the input text file into list of dictionaries, where each row in the input file is stored as a python Dictionary.

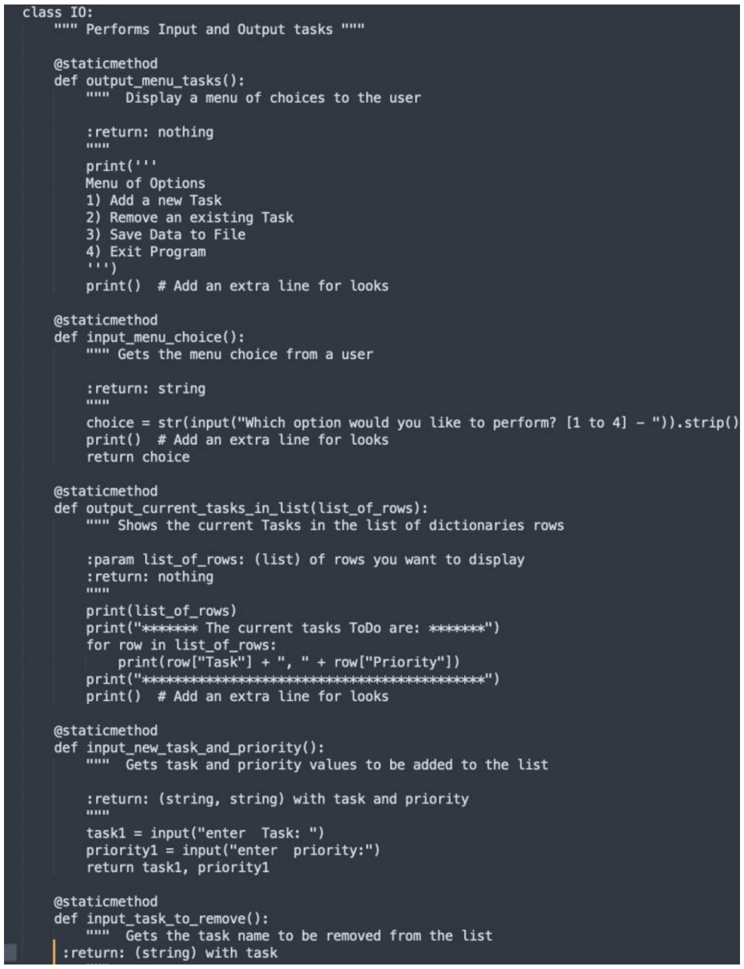
What are Classes & Methods

A function is a block of organized reusable code that is used to perform a single, related action. Python has many built in function like print () etc., you must define the function before using in the code to call the function.

A Class is used to define a particular type of object because python object can have both function and data elements. Python classes can define what methods can be used to change the state of an object, they also indicate what attributes the object can have.

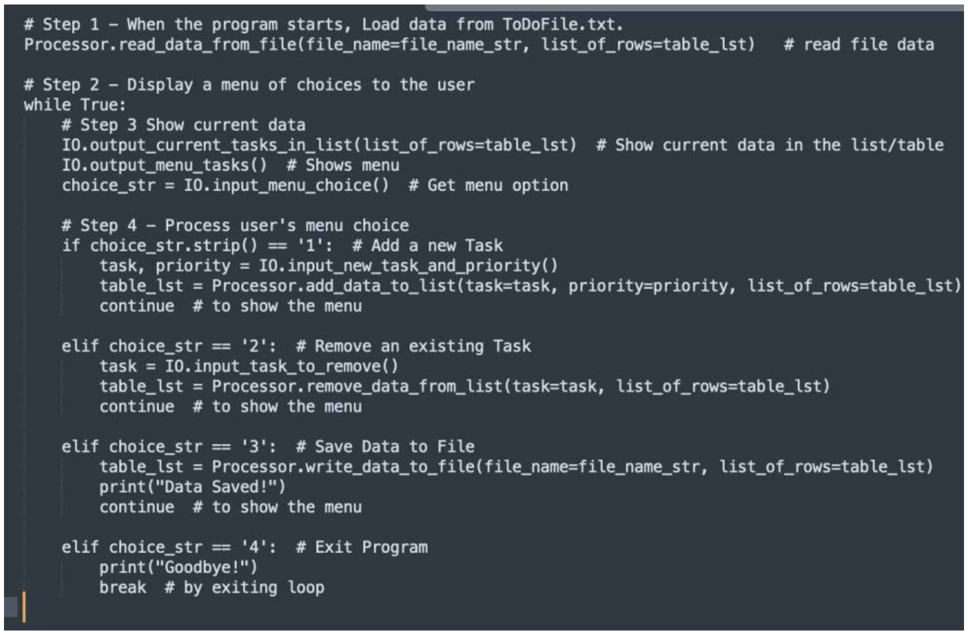
Program Structure

In the last assignment (Assignment5), I implemented the same program without using classes & methods, instead wrote all the code logic in a single python file. In this assignment, I split the program into two classes and kept the IO (Input Output) related code in one class and Processing related operations in another class and each class has multiple methods to perform different operations. All these methods are declared as "Static", so that we don't have to create an object to call these methods. Following images shows the classes & methods implemented:



```
class IO:
    """ Performs Input and Output tasks """

    @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

    @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

    @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(list_of_rows)
        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_new_task_and_priority():
        """  Gets task and priority values to be added to the list

        :return: (string, string) with task and priority
        """
        task1 = input("enter  Task: ")
        priority1 = input("enter  priority:")
        return task1, priority1

    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list
     :return: (string) with task
```

Picture 9: Class Io with multiple methods.

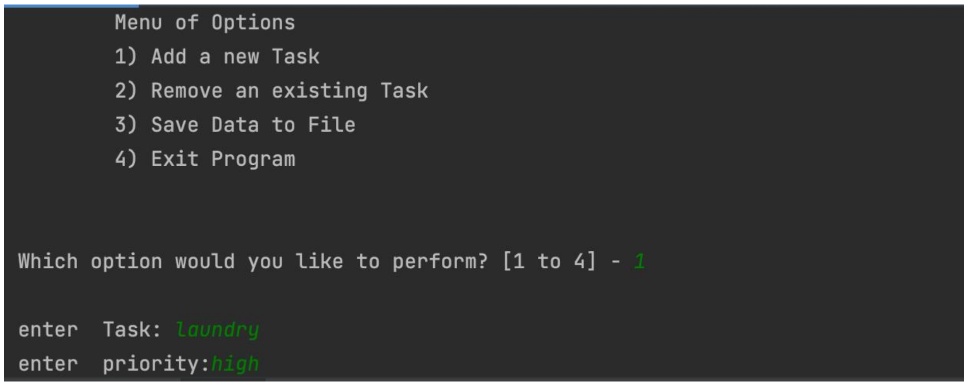


```
# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)   # read file data

# 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
```

Picture 10: Class Processor.

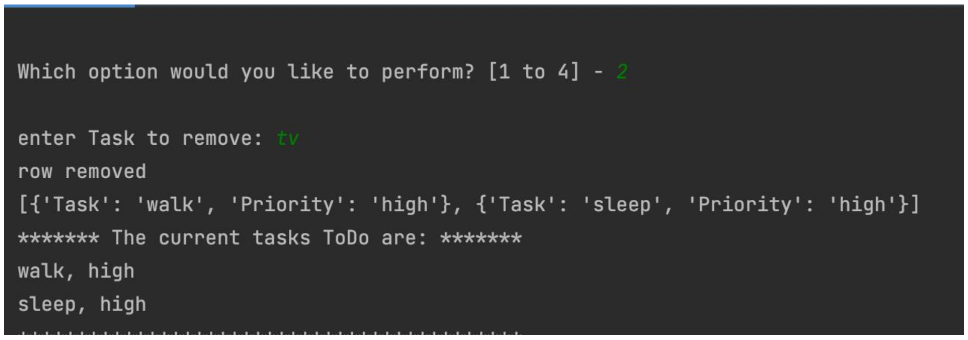
Program Steps

I displayed the menu of options to the user such as Adding the new task, Remove and save the data to file and exit program to the text file when user makes that choice. User options are shown the following images.



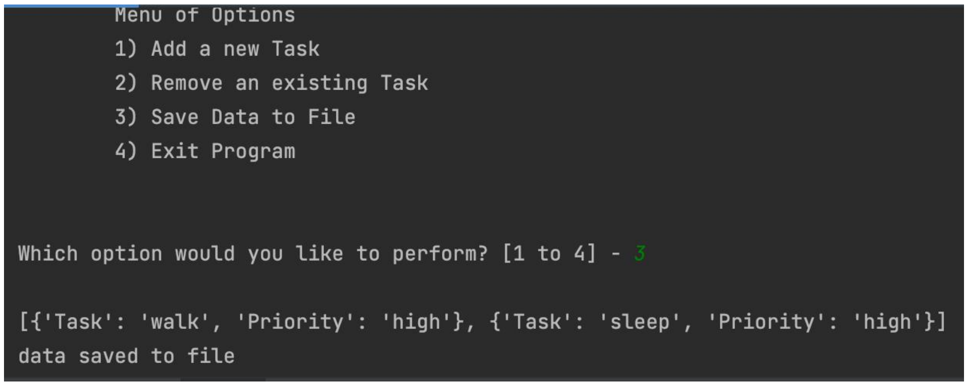
```
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

enter  Task: laundry
enter  priority:high
```

Picture 1: Displaying the options to choose.



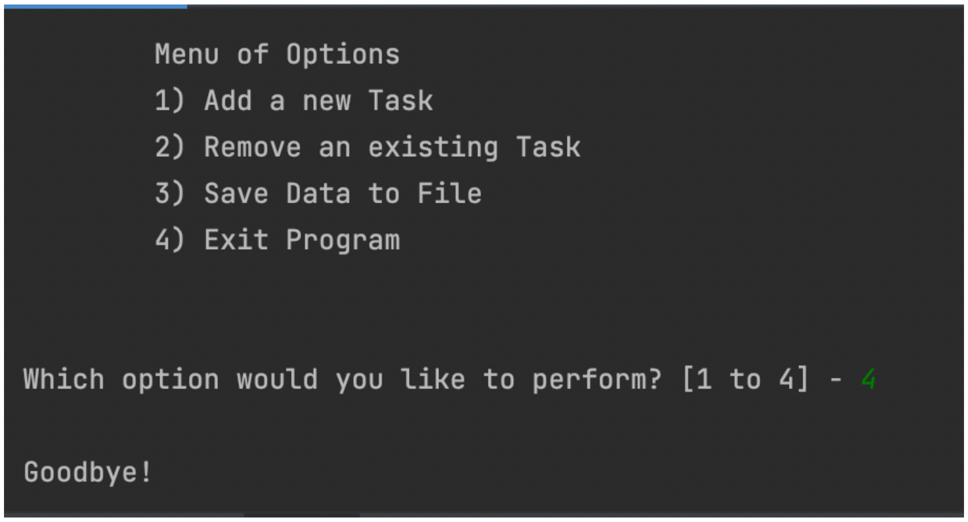
```
Which option would you like to perform? [1 to 4] - 2

enter Task to remove: tv
row removed
[{'Task': 'walk', 'Priority': 'high'}, {'Task': 'sleep', 'Priority': 'high'}]
******* The current tasks ToDo are: *******
walk, high
sleep, high
```

Picture 2: Removing an existing task.



```
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 3

[{'Task': 'walk', 'Priority': 'high'}, {'Task': 'sleep', 'Priority': 'high'}]
data saved to file
```

Picture 3: saving the data to text file.

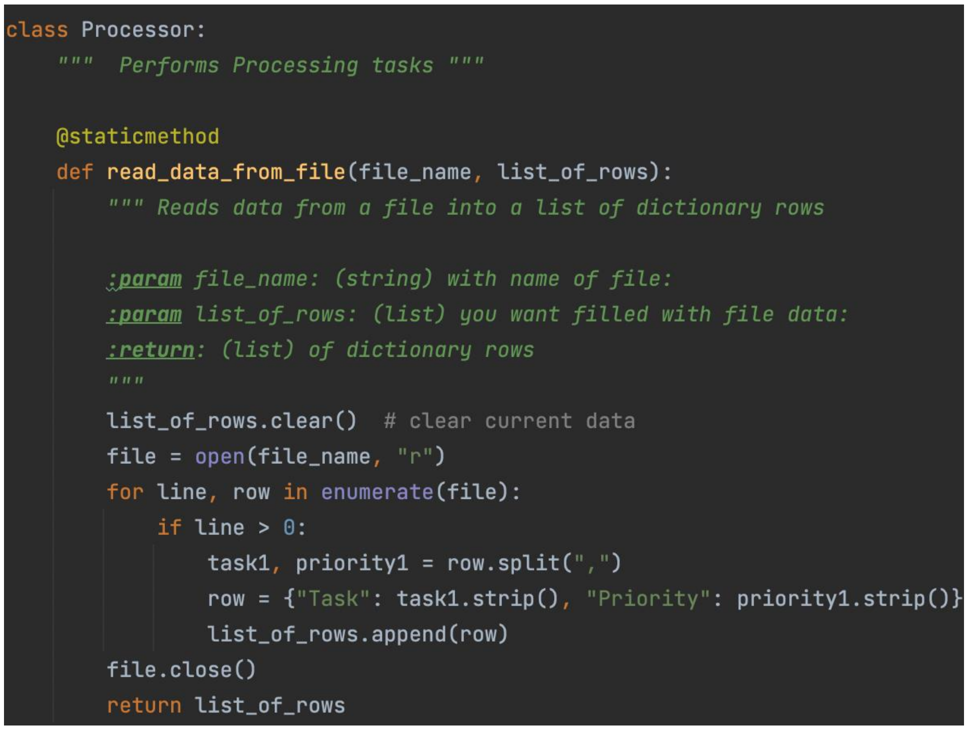


```
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Goodbye!
```

Picture 4: Exit the program.

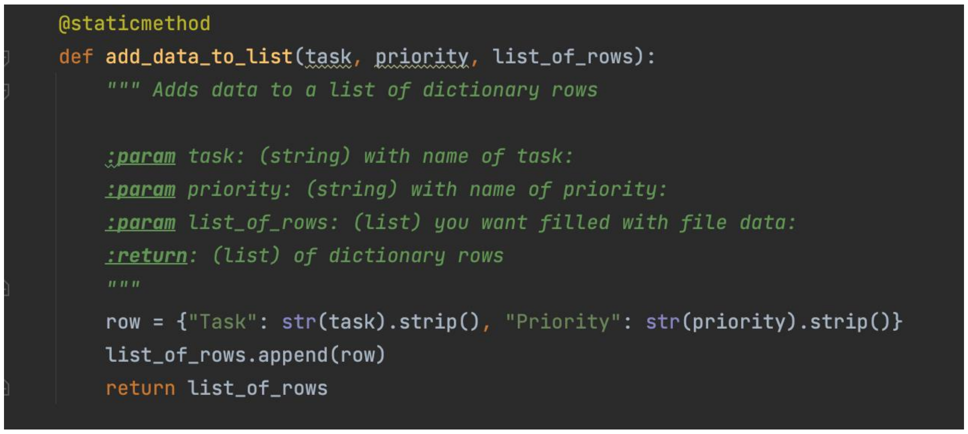
Code Implementation

I read the input text file, iterated through each row in the input text file and stored each row of data into a dictionary and all the data available in the input text file is stored in list of dictionaries. Each time when user selects the option1, new input is taken from the user and added to the list and when the user selects option2, input is taken from the user to remove an existing task, when the user selects option3, entered data is saved to file and option4 to exit the program. Pictures shown below



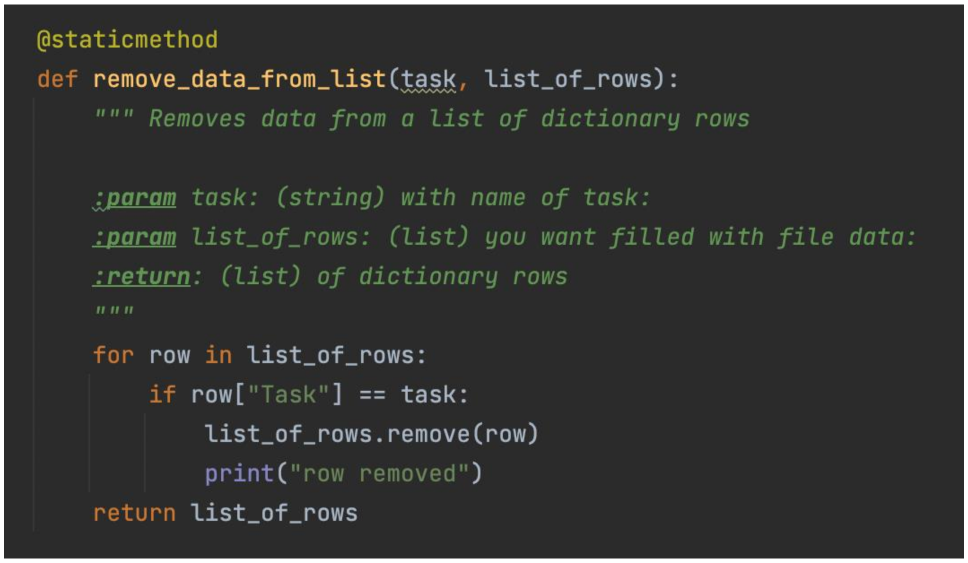
```python
class Processor:
    """  Performs Processing tasks """

    @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, row in enumerate(file):
            if line > 0:
                task1, priority1 = row.split(",")
                row = {"Task": task1.strip(), "Priority": priority1.strip()}
                list_of_rows.append(row)
        file.close()
        return list_of_rows
```

Picture 5: Reading the data from the file into list of dictionary rows.



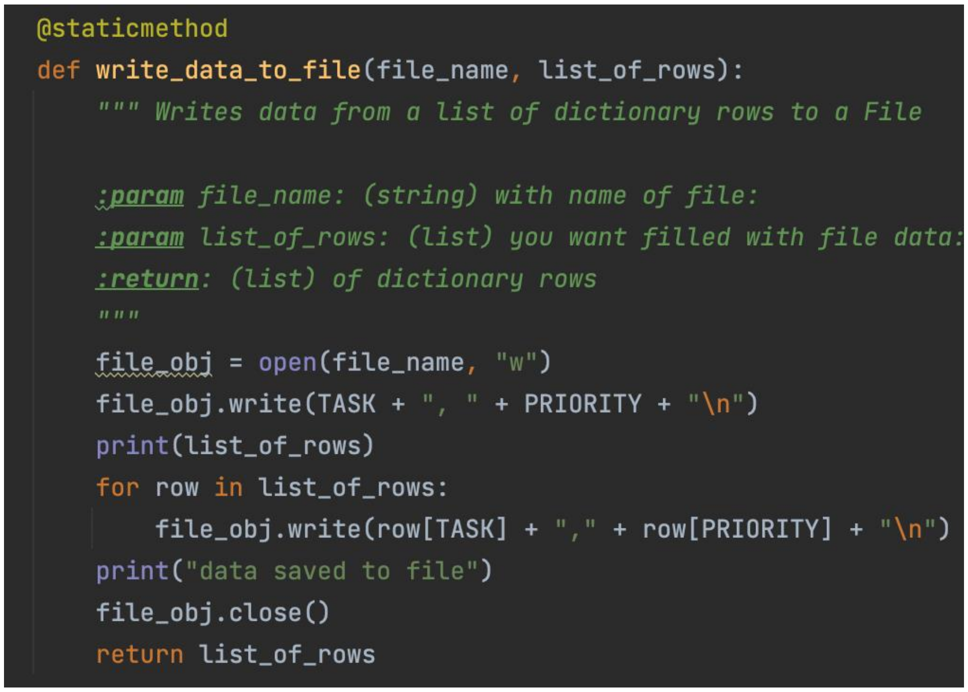
```python
    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        """ Adds data to a list of dictionary rows

        :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()}
        list_of_rows.append(row)
        return list_of_rows
```

Picture 6: Adding new task to the list.



```python
@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
    """
    for row in list_of_rows:
        if row["Task"] == task:
            list_of_rows.remove(row)
            print("row removed")
    return list_of_rows
```

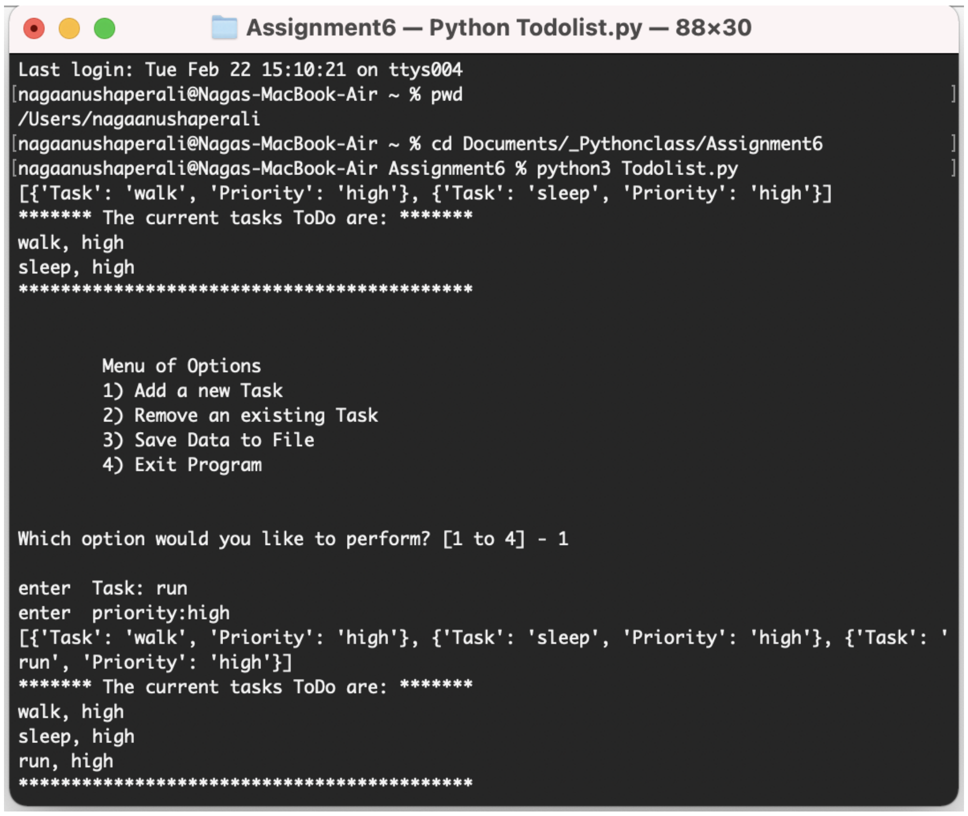
Picture 7: Removing an existing from the list.



```python
@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
    """
    file_obj = open(file_name, "w")
    file_obj.write(TASK + ", " + PRIORITY + "\n")
    print(list_of_rows)
    for row in list_of_rows:
        file_obj.write(row[TASK] + "," + row[PRIORITY] + "\n")
    print("data saved to file")
    file_obj.close()
    return list_of_rows
```

Picture 8: writing the data from the list of dictionaries rows to file.

I also ran this program in the terminal window for results refer image below.



```
Last login: Tue Feb 22 15:10:21 on ttys004
nagaanushaperali@Nagas-MacBook-Air ~ % pwd
/Users/nagaanushaperali
nagaanushaperali@Nagas-MacBook-Air ~ % cd Documents/_Pythonclass/Assignment6
nagaanushaperali@Nagas-MacBook-Air Assignment6 % python3 Todolist.py
[{'Task': 'walk', 'Priority': 'high'}, {'Task': 'sleep', 'Priority': 'high'}]
******* The current tasks ToDo are: *******
walk, high
sleep, high
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

enter  Task: run
enter  priority:high
[{'Task': 'walk', 'Priority': 'high'}, {'Task': 'sleep', 'Priority': 'high'}, {'Task': '
run', 'Priority': 'high'}]
******* The current tasks ToDo are: *******
walk, high
sleep, high
run, high
*******************************************
```

GitHub Webpage

Creating a simple GitHub web page that which is associated with the repository, I start off by creating the new repository as “ITFnd100-mod6”, set that as Public and initiate the Readme file and create the repository. After setting up the repository I have created a folder by clicking on the file button and add the file name as “index.md” (ITFnd100-mod6/docs/index.md) and then add some simple text to the webpage and click on the commit the new file button. To configure the website as a reused webpage Goto settings and select GitHub Pages and choose the docs folder. Link for the webpage I created https://nagaausha.github.io/ITFnd100-mod6/

I also uploaded my documentation, python script to the GitHub repository   here is the link https://github.com/NagaAusha/ITFnd100-mod6

Summary

In this Assignment by referring to module 6 video and by referring the information in the text book I learned to work with Classes and Methods and perform some operations, and I learn how to Create folder, file and create a webpage using the GitHub.