Assignment_06

Mary Williams

08/19/2019

IT FDN 100 B, Foundations Of Programming: Python

https://github.com/maryaleen/IntroToProg-Python

Introduction

Assignment_05 tasked each student to create a python script that allows a user to modify and save a to do list of items and their corresponding priorities. The items should be saved in a table (a list that contains dictionary rows with "Tasks" and Priorities"). The user should be able to view the list, add an item, remove and item, save the list, as well as save and exit the program.

Assignment_06 tasked each student to refactor their code from assignment_05 to use functions and a class. I chose to create a class and functions within that class (called methods).

This document describes the steps taken to create in PyCharm and run the script in both PyCharm and the Mac OS Terminal.

Script/Code

Open a new project in PyCharm with the following directory:

marywilliams/Documents/_PythonClass/Assignment06

Copy Python file from assignment_05 and keep the name *ToDoList.py*. I wrote the script to flow in the following manner:

A. Create a class called Tasks to manage my the user's to do list.

B. Define functions within the class, or methods, to perform the different processing/actions related to the user's to do list.

C. The class must be instantiated and the first method called to read from the ToDo.txt file to create to do list, which is a table of Task and Priorities

D. The remaining methods will be called later in the menu for the user to do the following (within the while loop just like in assignment_05):

    1. View List
    2. Add Item
    3. Remove Item
    4. Save
    5. Save/Exit

E. No changes were made to the code functionality relative to assignment_05.

ToDoList.py script contents are contained at the end of this document for reference.

Running the Script

To run the script in PyCharm, use the Run menu or the button in the tool bar. See Figure 1.

To run the script in Mac OS, I opened a Terminal and navigated to the directory my script was located in: marywilliams/Documents/_PythonClass/Assignment06

Then, I used the python3 command to run my script. See Figure 2.

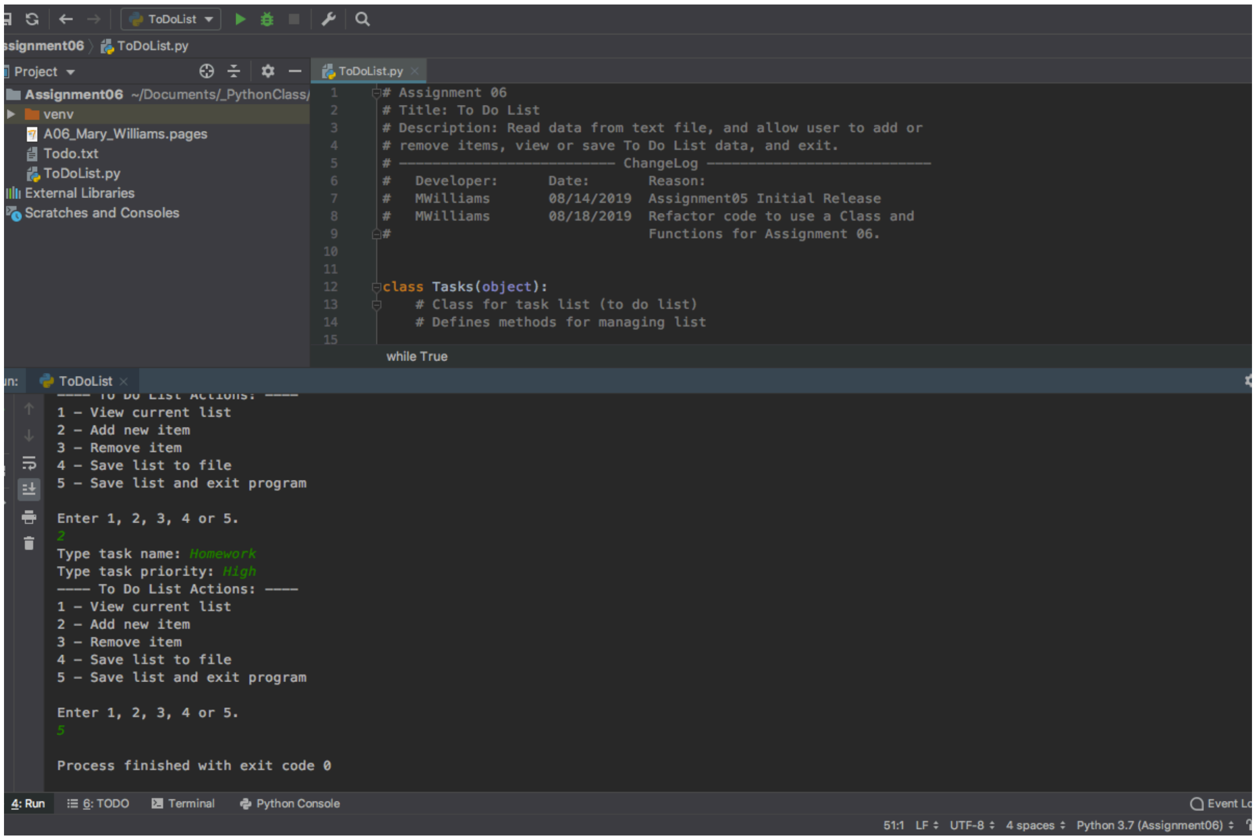
I started with the ToDo.txt file I ended assignment_05 with, which contained the following tasks/priorities:

Clean House, Low

Walk Dog, Medium

Exercise, High

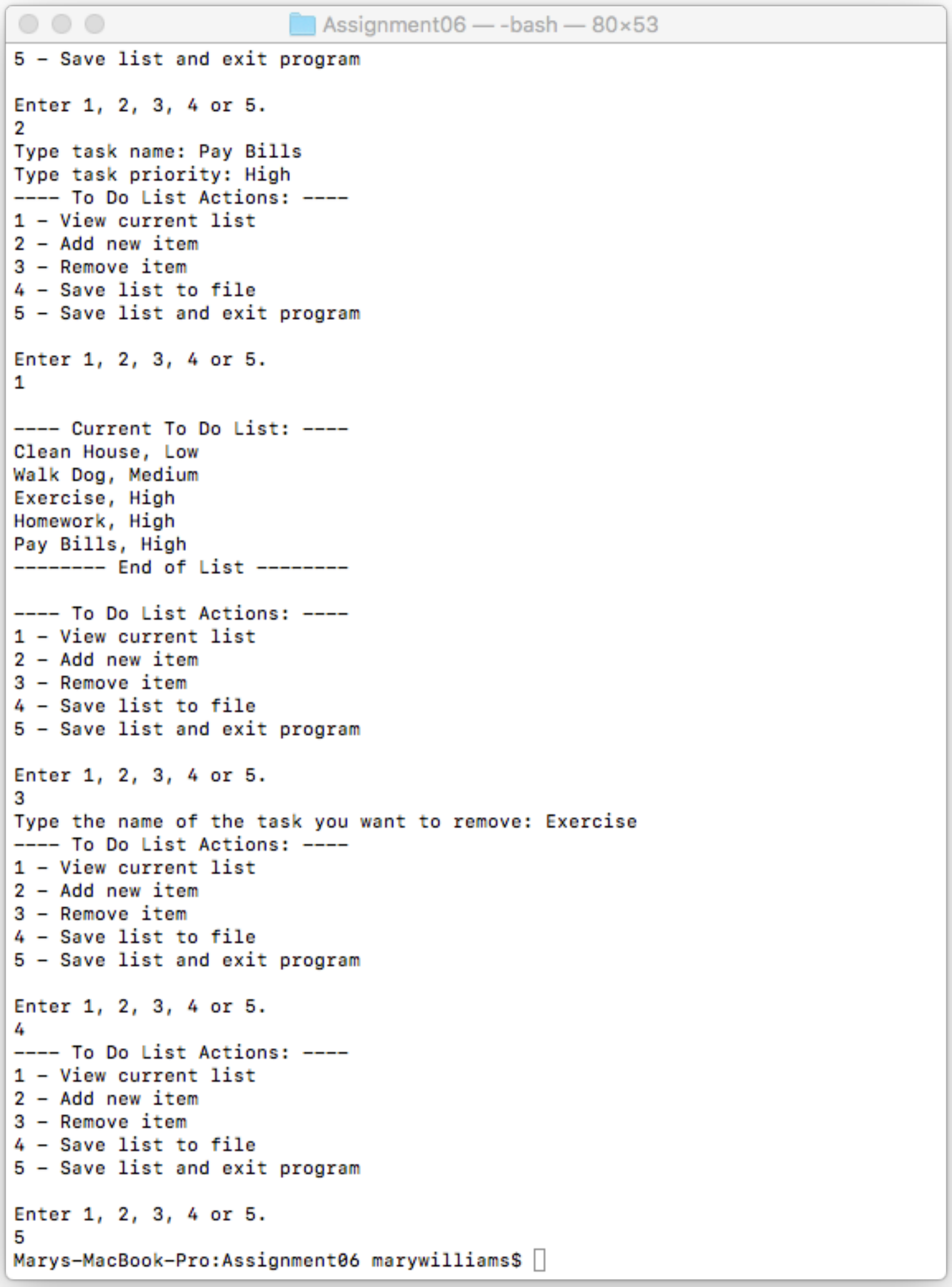
After running my assignment_06 script, the text file now contains four to do items and their priorities. Two were entered by running the script in PyCharm; one more was added and two were removed running the script in the Terminal. The text file output is shown in Figure 3.



```
# Assignment 06
# Title: To Do List
# Description: Read data from text file, and allow user to add or
# remove items, view or save To Do List data, and exit.
# ------------------------- ChangeLog -------------------------
#   Developer:      Date:        Reason:
#   MWilliams       08/14/2019  Assignment05 Initial Release
#   MWilliams       08/18/2019  Refactor code to use a Class and
#                               Functions for Assignment 06.
#


class Tasks(object):
    # Class for task list (to do list)
    # Defines methods for managing list
```

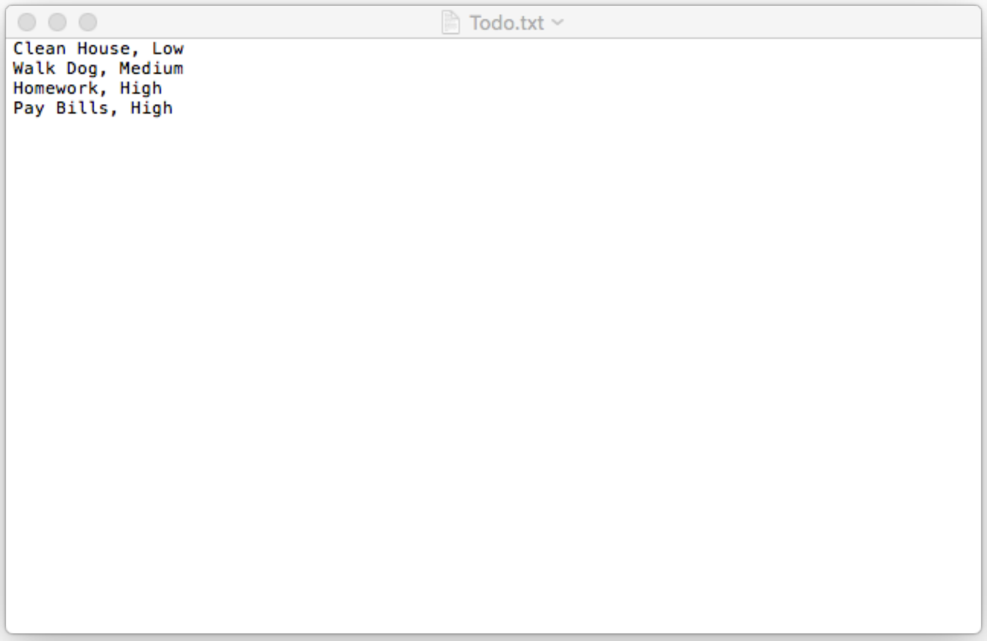
```
---- To Do List Actions: ----
1 - View current list
2 - Add new item
3 - Remove item
4 - Save list to file
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
2
Type task name: Homework
Type task priority: High
---- To Do List Actions: ----
1 - View current list
2 - Add new item
3 - Remove item
4 - Save list to file
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
5

Process finished with exit code 0
```

FIGURE 1: PYCHARM CAPTURE



```
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
2
Type task name: Pay Bills
Type task priority: High
---- To Do List Actions: ----
1 - View current list
2 - Add new item
3 - Remove item
4 - Save list to file
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
1

---- Current To Do List: ----
Clean House, Low
Walk Dog, Medium
Exercise, High
Homework, High
Pay Bills, High
-------- End of List --------

---- To Do List Actions: ----
1 - View current list
2 - Add new item
3 - Remove item
4 - Save list to file
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
3
Type the name of the task you want to remove: Exercise
---- To Do List Actions: ----
1 - View current list
2 - Add new item
3 - Remove item
4 - Save list to file
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
4
---- To Do List Actions: ----
1 - View current list
2 - Add new item
3 - Remove item
4 - Save list to file
5 - Save list and exit program

Enter 1, 2, 3, 4 or 5.
5
Marys-MacBook-Pro:Assignment06 marywilliams$
```

FIGURE 2: TERMINAL CAPTURE



```
Clean House, Low
Walk Dog, Medium
Homework, High
Pay Bills, High
```

FIGURE 3: TODO.TXT CAPTURE

ToDoList.py

```
# Assignment 06
# Title: To Do List
# Description: Read data from text file, and allow user to add or
# remove items, view or save To Do List data, and exit.
# -------------------------- ChangeLog ---------------------------
#   Developer:      Date:       Reason:
#   MWilliams       08/14/2019  Assignment05 Initial Release
#   MWilliams       08/18/2019  Refactor code to use a Class and
#                               Functions for Assignment 06.


class Tasks(object):
    # Class for task list (to do list)
    # Defines methods for managing list

    # 0 - Read from file to create list/dictionary table
    @staticmethod
    def read_todo():
        obj_file = open('ToDo.txt', 'r')  # renamed objFile to follow PEP 8
        lines = obj_file.readlines()
        todo_list = []  # renamed todoList to follow PEP 8
        for item in lines:
            row = (item.split(','))
            dictrow = {'Tasks': row[0].strip(), 'Priority': row[1].strip()}
            todo_list.append(dictrow)
        obj_file.close()
        return todo_list

    # 1 - Show current list
    @staticmethod
    def view_todo(a):
        todo_list = a
        print('\n---- Current To Do List: ----')
        for place in todo_list:
            print(place.get('Tasks') + ', ' + place.get('Priority'))
        print('-------- End of List --------\n')

    # 2 - Add new item to To Do List
    @staticmethod
    def add_task(b):
        todo_list = b
        task = input('Type task name: ')
        priority = input('Type task priority: ')
        dictrow = {'Tasks': task, 'Priority': priority}
        todo_list.append(dictrow)
        return todo_list

    # 3 - Remove item from To Do List
    @staticmethod
    def rm_task(c):
        todo_list = c
        rmtask = input('Type the name of the task you want to remove: ')
        for row in todo_list:
            if row['Tasks'] == rmtask:
                todo_list.remove(row)
        return todo_list

    # 4 - Save To Do List to file
    @staticmethod
    def save_todo(d):
        todo_list = d
        obj_file = open('ToDo.txt', 'w')
        for place in todo_list:
            obj_file.write(place.get('Tasks') + ', ' + place.get('Priority') + '\n')
        obj_file.close()


# Instantiate Tasks class as userList
userList = Tasks()
```

```
# Create initial to do list with the read_todo() method
master_list = userList.read_todo()


# Allow user to interact with to do list or exit program
while True:
    print('---- To Do List Actions: ----')
    choice = int(input('1 - View current list\n'
                       '2 - Add new item\n'
                       '3 - Remove item\n'
                       '4 - Save list to file\n'
                       '5 - Save list and exit program\n\n'
                       'Enter 1, 2, 3, 4 or 5.\n'))

    # 1 - Show current list
    if choice == 1:
        userList.view_todo(master_list)

    # 2 - Add new item to To Do List
    elif choice == 2:
        master_list = userList.add_task(master_list)

    # 3 - Remove item from To Do List
    elif choice == 3:
        master_list = userList.rm_task(master_list)

    # 4 - Save To Do List to file
    elif choice == 4:
        userList.save_todo(master_list)

    # 5 - Save To Do List and Exit Program
    elif choice == 5:
        userList.save_todo(master_list)
        exit()

    # Handle bad user input
```

```
else:
    print('Incorrect input. Try again.\n')
```