**Name : **Leonardo Espinoza

**Date : **August 16, 2023

**Course : **Foundations of Programming, Python

Assignment 06

Github : https://github.com/lesping/IntroToProg-Python-Mod06

The To Do List Program using Functions

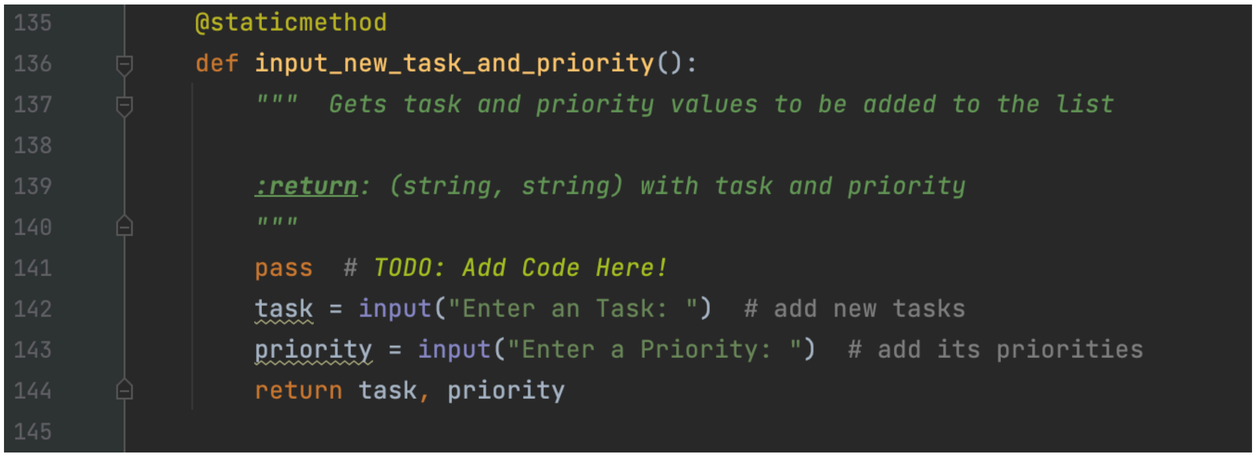
Introduction

During module 6, we learned how to work on creating scripts using functions and simple classes. This allows us to transform large scripts and with certain levels of complexity, into smaller and more manageable chunks of code. We also learned how to execute the code using the PyCharm debugger and how to create Github web pages. This week the challenge was to complete and modify a provided script that manages a To Do List using Functions and Classes. Following section describe the steps carried out to complete the script from assignment 6.

I. Completing the To Do List script using Functions

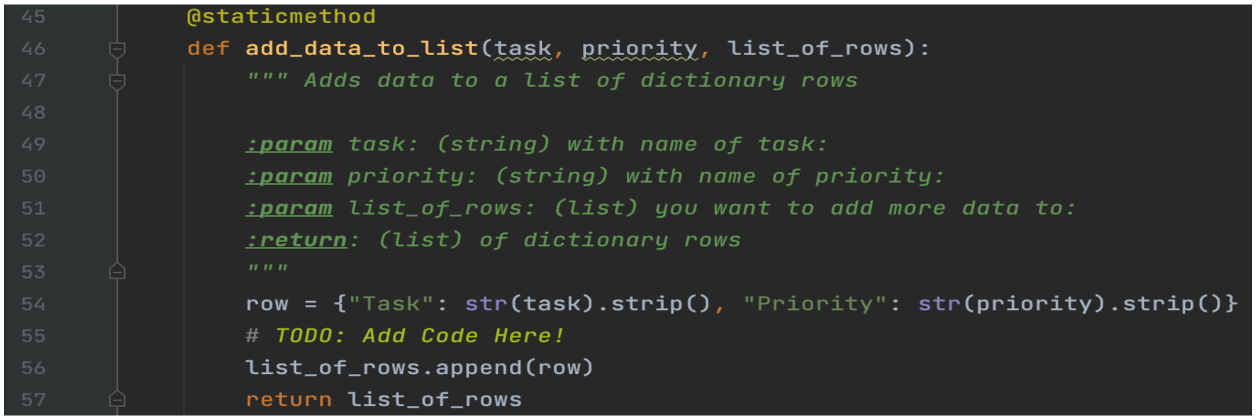
I.1. Completing the program code

The initial template had several functions ready to be executed in the program, therefore the first step I took was to write code using the input() statement to request the user to enter new tasks and their respective priority, for which I used the variables local "task" and "priority", the values were captured with the "Return" function (Figure 1). Then, in the "add_data_to_list" function, I used the captured values to add them in the form of dictionary to the table list, using the Append() statement, and also capture the "list_of_rows" value with the "Return" function (Figure 2).



```
    @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
        """
        pass  # TODO: Add Code Here!
        task = input("Enter an Task: ")  # add new tasks
        priority = input("Enter a Priority: ")  # add its priorities
        return task, priority

```

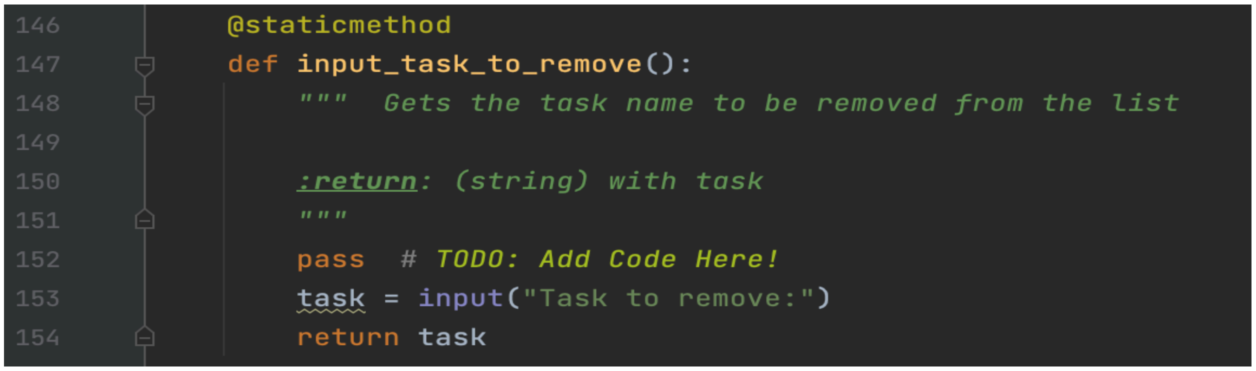
Figure 1: Code to enter new data into the To Do List



```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 to add more data to:
        :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
```

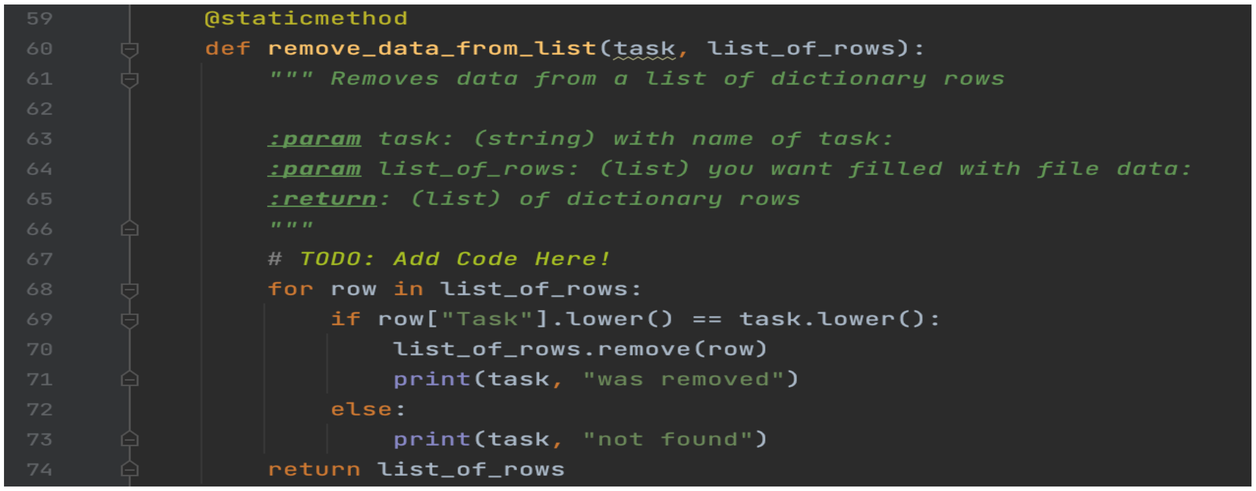
Figure 2: Code to add data in table list as a dictionary row

The next step was to complete the functions that allow the user to remove a task from the list and its respective priority. To achieve that, I used the input() statement to ask the user to enter the item to remove and captured its value in the local variable “task” using the “Return” function (Figure 3). Then, in the "remove_data_from_list" function, I used the “task” parameter and For loop to search for the item within the table list and with the if() conditional, if the value is found in the list it is removed, otherwise the program only informs the user that the value is not found in the table (Figure 4).



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

        :return: (string) with task
        """
        pass   # TODO: Add Code Here!
        task = input("Task to remove:")
        return task
```

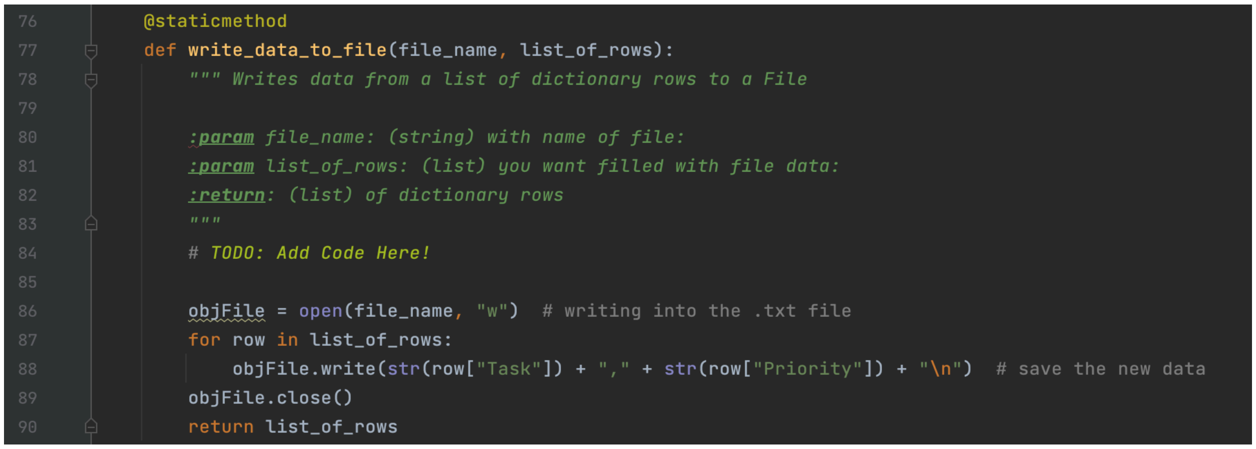
Figure 3: Code to ask for item to delete



```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
        """
        # TODO: Add Code Here!
        for row in list_of_rows:
            if row["Task"].lower() == task.lower():
                list_of_rows.remove(row)
                print(task, "was removed")
            else:
                print(task, "not found")
        return list_of_rows
```

Figure 4: Code to remove data from the table list

The next step was to complete the code for the "write_data_to_file" function. I used the *open()* function with the option *"w"* to add the new data in the plain text file where the information will be stored (I created a folder called 'Assignment06' where I also saved my script from PyCharm), then added the *For Loops* to be able to collect the data in the form of two-dimensional list (dictionary row) and leave it saved in the form of table in the .txt file, prior to close the program.

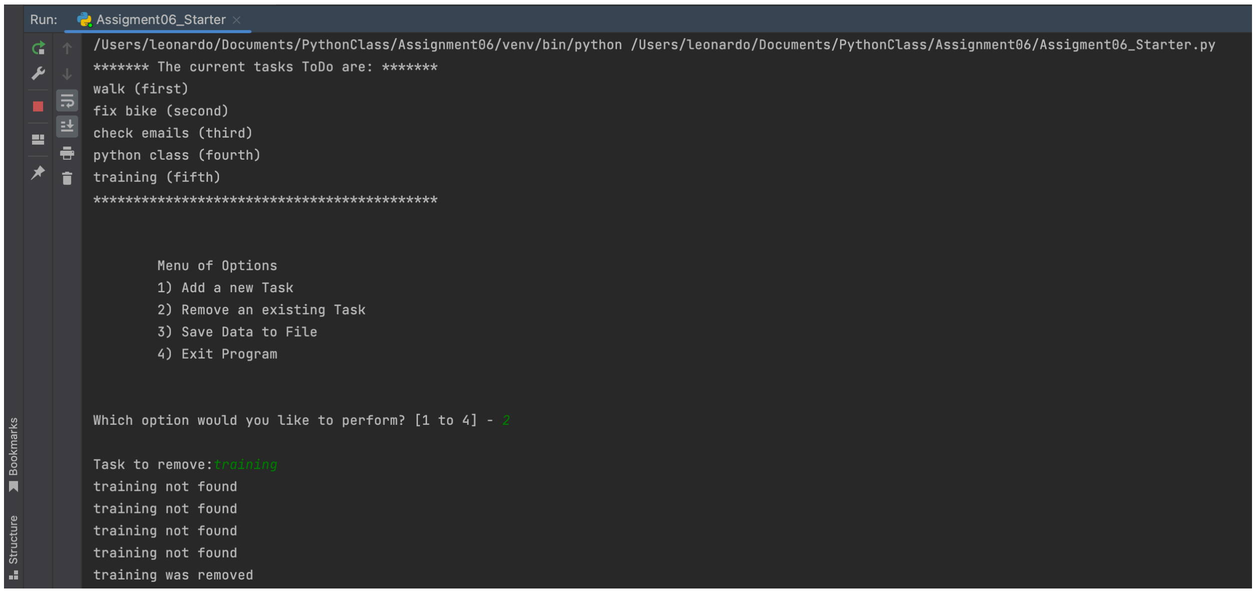


```python
    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        """ Writes data from a list of dictionary rows to a File

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        # TODO: Add Code Here!

        objFile = open(file_name, "w")  # writing into the .txt file
        for row in list_of_rows:
            objFile.write(str(row["Task"]) + "," + str(row["Priority"]) + "\n")  # save the new data
        objFile.close()
        return list_of_rows
```

Figure 5: Code to save the data in the plain text file

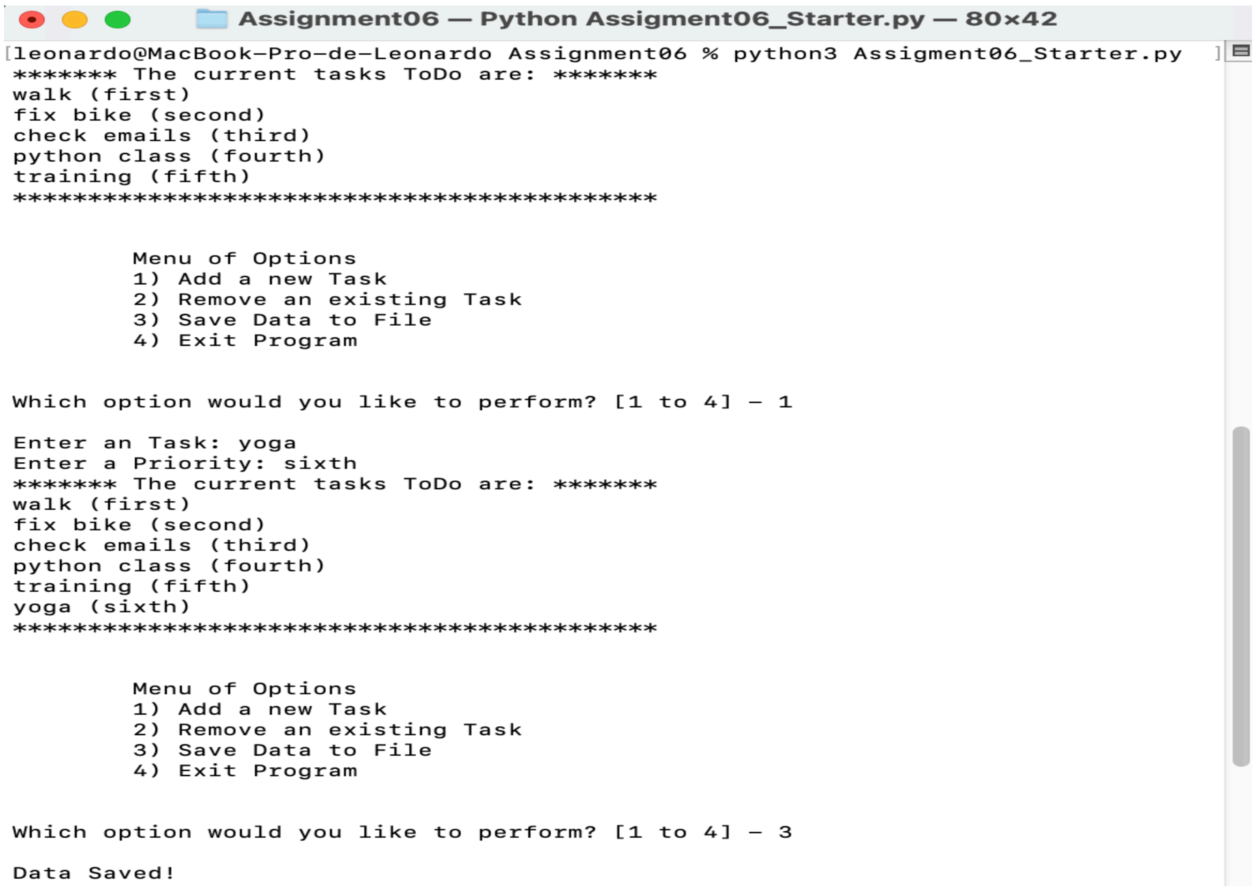
I.2. Running the code in PyCharm & Command Terminal

Then I validated the sccript in PyCharm and the Command Terminal. I ran the program without problems (Figure 6 and 7).



```
/Users/leonardo/Documents/PythonClass/Assignment06/venv/bin/python /Users/leonardo/Documents/PythonClass/Assignment06/Assigment06_Starter.py
******* The current tasks ToDo are: *******
walk (first)
fix bike (second)
check emails (third)
python class (fourth)
training (fifth)
*******************************************



        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] - 2

Task to remove:training
training not found
training not found
training not found
training not found
training was removed
```

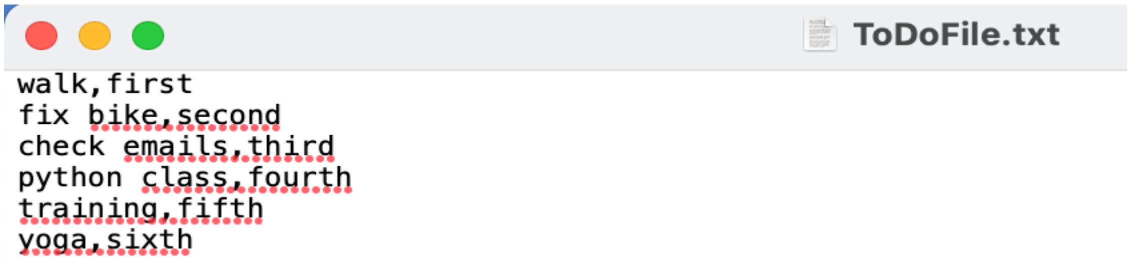
Figure 6: The code running in PyCharm



```
leonardo@MacBook-Pro-de-Leonardo Assignment06 % python3 Assigment06_Starter.py
******* The current tasks ToDo are: *******
walk (first)
fix bike (second)
check emails (third)
python class (fourth)
training (fifth)
*******************************************


        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 an Task: yoga
Enter a Priority: sixth
******* The current tasks ToDo are: *******
walk (first)
fix bike (second)
check emails (third)
python class (fourth)
training (fifth)
yoga (sixth)
*******************************************


        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

Data Saved!
```

Figure 7: The code running in the Command Terminal

Finally, I opened the plain text file *ToDoFile.txt* and it can be demonstrated that the code works by saving the information correctly (Figure 8).



```
walk,first
fix bike,second
check emails,third
python class,fourth
training,fifth
yoga,sixth
```

Figure 8: Checking the data in .txt file

Summary

In this document I explain the steps I went through to modify the code that manages a To Do List, using functions and simple classes. Before starting, it was important to review the resources recommended by the professor to learn and deepen about functions, specifically about local and global variables, parameters, and arguments (reading, watch the videos and review the websites). In this way, I completed the script, and it ran correctly.