Nikita Daharia

May 24, 2022

Foundations of Programming: Python

Assignment 06

https://github.com/nikitadaharia/IntroToProg-Python-Mod06

Working with Functions and classes:

Introduction:

This paper aims to demonstrate the steps I undertook to create a python script that enables a user to modify a new script that manages a "ToDo list" and add more functions to it to organize the code. The latest version of PyCharm Community version is used for this assignment on a Mac OS.

Writing the python script and output:

Create a new sub-folder called Assignment06 inside of the _PythonClass folder (you created in Module 01) in the Documents folder on a Mac OS. Create a new project in PyCharm that uses the _PythonClass\Assignment06 folder as its location.

The script was started by defining the global variables used throughout the code.

```
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
strToDoFile = "ToDoFile.txt"  # Name of data file
objFile = None  # Object that represents a file
dicRow = {}  # Row of data separated into elements of a dictionary {Task,Priority}
lstTable = []  # List that acts as a 'table' of rows
strChoice = ""  # Captures the selection of user options
strPriority = ""  # Captures the user priority data
strStatus = ""  # Captures the status of processing functions
strTask = ""  # Captures the user task data
```

Figure 1. Snippet of the python script defining the global variables

The first class in the script is the class Processor that consists of functions that process the data. It consists of functions that read, add, remove and write data to the file. Code was added after explaining the parameters to add, remove and write data to the list.

```
@staticmethod
def add_data_to_list(task, priority, list_of_rows):
    """
    :param task: inputted task that user wants to add to list
    :param priority: inputted priority user wants to add to list
    :param list_of_rows: the list of dictionary rows tasks and priorities are appended to
    :return: the updated list of rows with the added tasks
    """
    list_of_rows.append({"Task": task, "Priority": priority})
    return list_of_rows, 'Success'
```

Figure 2. Snippet of the python script to add data to the file.

```
@staticmethod
def remove_data_from_list(task, list_of_rows):
    """
    :param task: inputted task the user wants to delete the task/priority pair for
    :param list_of_rows: the list of dictionary rows tasks and priorities are saved in
    :return: the updated list_of_rows with the inputted task row removed (if it was in it),
            whether the entered task was successfully removed or whether it was not found
    """
```

Figure 3. Snippet of the python to remove data from the file.

```
@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """
    :param file_name: the name of the file that the data will be written to
    :param list_of_rows: the list of dictionaries the data is saved in
    :return: list_of_rows that has just been written to the file
    """
```

Figure 4. Snippet of the python script to write data from the file.

The presentation section of the script consisted of the input() and print() functions. This was followed by the task and priority function which a user was used to enter a task and a priority for the task which was then returned.

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

    @staticmethod
    def print_menu_Tasks():
        """  Display a menu of choices to the user
        :return:
        """
        print(
            '''
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Reload Data from File
        5) Exit Program
        ''')
        print()

    @staticmethod
    def input_menu_choice():
        """ Gets the menu choice from a user
        :return: string
        """
```

Figure 5. Snippet of the python script showing an output function to display a menu of choices to the user.

Lastly, main body of the program was written to remove a pre-existing task, save data to the file, reload data from file and finally exit the program.

```
elif strChoice == '2':  # Remove an existing Task
    strTask = IO.input_task_to_remove()  # user inputs task to remove
    lstTable, strStatus = Processor.remove_data_from_list(
        strTask, lstTable)  # row of entered task is removed
    IO.print_current_Tasks_in_list(lstTable)  # displays updated list of current tasks
    IO.input_press_to_continue(strStatus)
    continue  # to see the menu
```

Figure 6. Snippet of the main body of the python to remove a pre-existing task saved in the file.

```
elif strChoice == '3':  # Save Data to File
    strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
    if strChoice.lower() == "y":
        lstTable, strStatus = Processor.write_data_to_file(
            strToDoFile, lstTable)  # write data to file & save return variables
        IO.input_press_to_continue(strStatus)
    else:
        IO.input_press_to_continue("Save Cancelled!")
    continue  # to see the menu
```

Figure 7. Snippet of the main body of the python save data to the file.

```
elif strChoice == '4':  # Reload Data from File
    print("Warning: Unsaved Data Will Be Lost!")
    strChoice = IO.input_yes_no_choice("Are you sure you want to reload data from file? (y/n) -  ")
    if strChoice.lower() == 'y':
        lstTable, strStatus = Processor.read_data_from_file(strToDoFile, lstTable)
        IO.print_current_Tasks_in_list(lstTable)
        IO.input_press_to_continue(strStatus)
    else:
        IO.input_press_to_continue("File Reload Cancelled!")
    continue  # to see the menu
```

Figure 8. Snippet of the main body of the python to reload data from the file.

Run the Script by right clicking on the file and choosing Run.

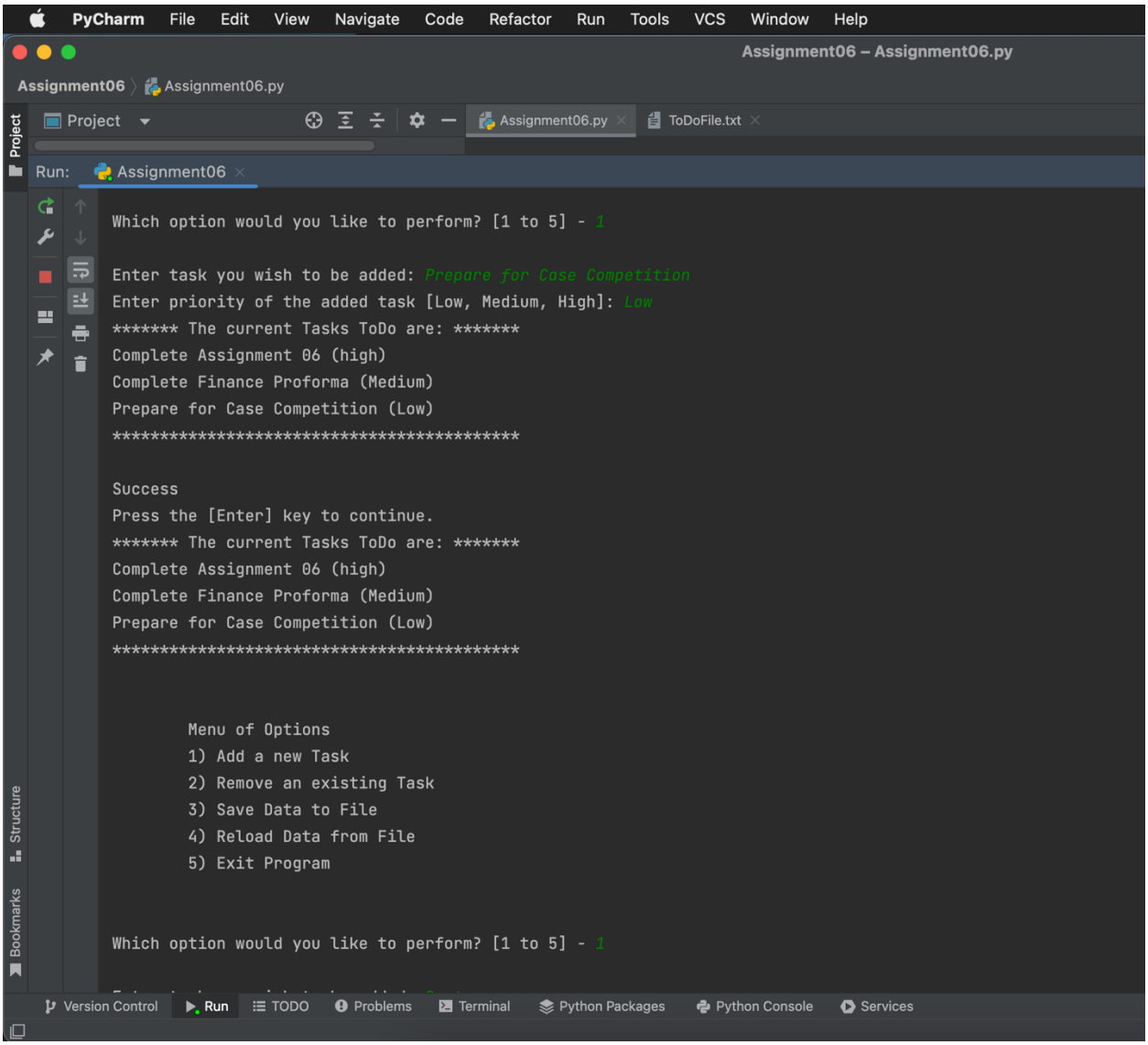

Figure 9. Snippet of output displayed in PyCharm Shell after running the python script.

Run the script on the Terminal window

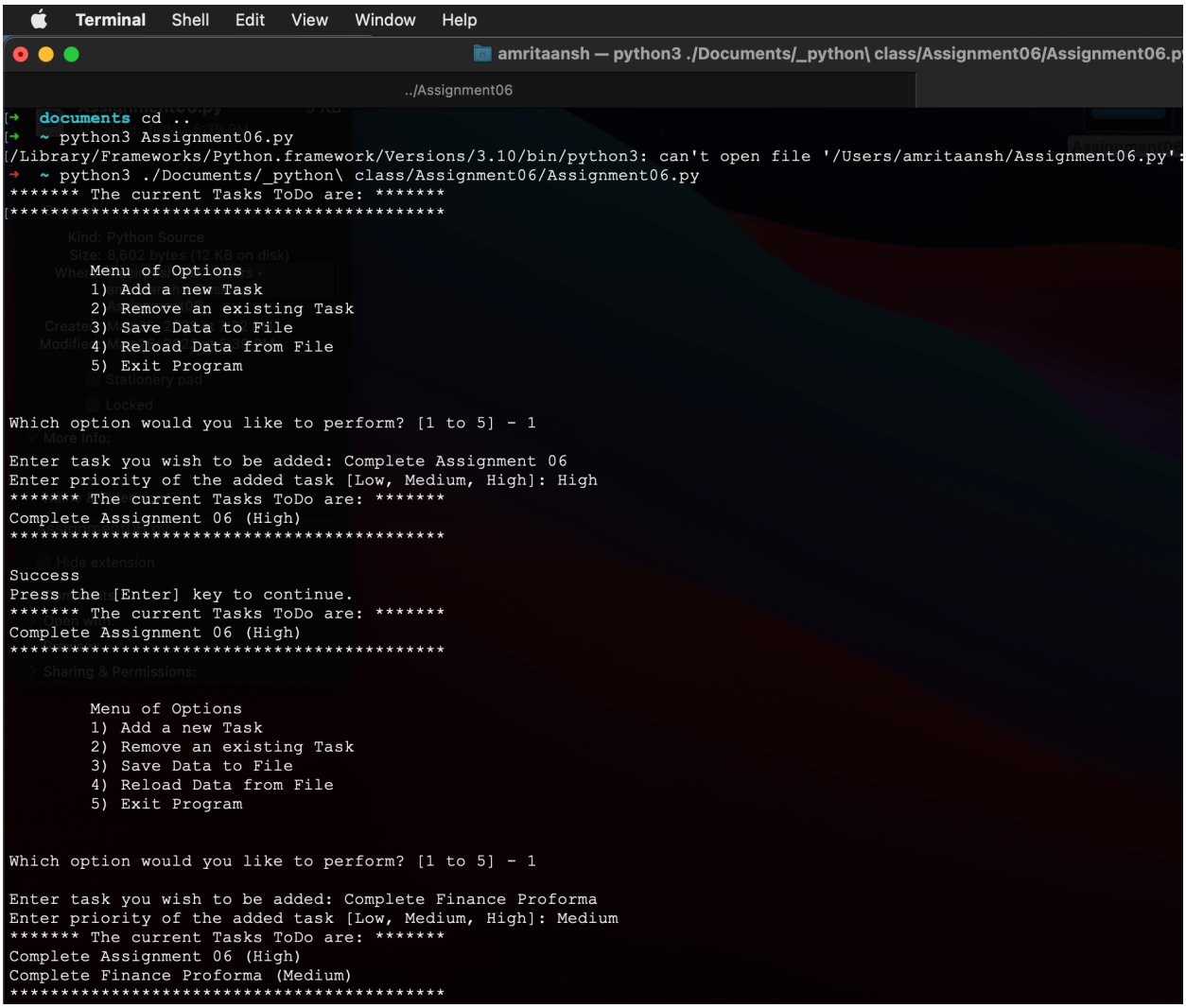

Figure 10. Output displayed in Terminal window after running the python script.

Verify that it code worked by locating the text file and opening it in a text editor.

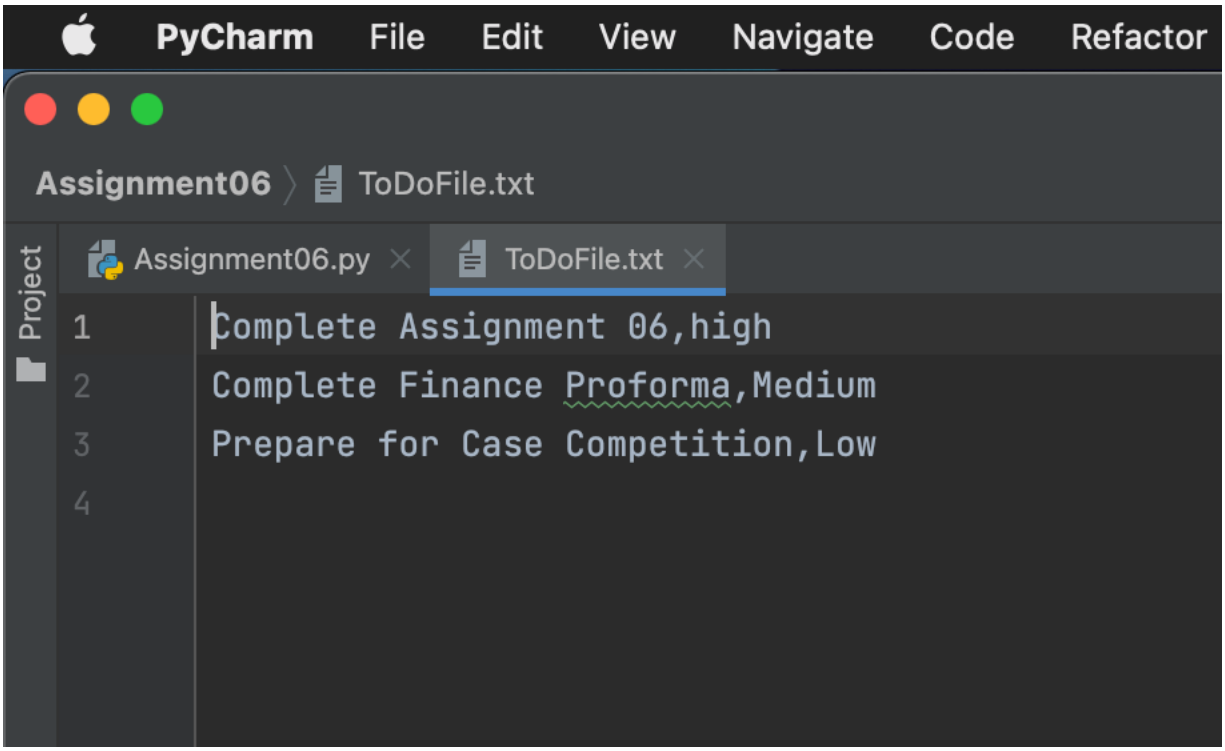

Figure 11. ToDoList.txt Window as by the user.

Summary:

Python is a simple yet powerful language programming language that runs on Windows, Linus/Unix, and Mac OS. I used PyCharm to create a python script that script that enables a user to modify a new script that manages a "ToDo list" using classes and functions. The script was run both in PyCharm and Terminal . Finally, the code was verified by locating the ToDoList.txt text file and opening it in a text editor and it successfully updated the list saved by the user.