Jay Plemons

Professor R. Root

Foundations of Programming: Python

GitHub repository (external link)

25 May 2020

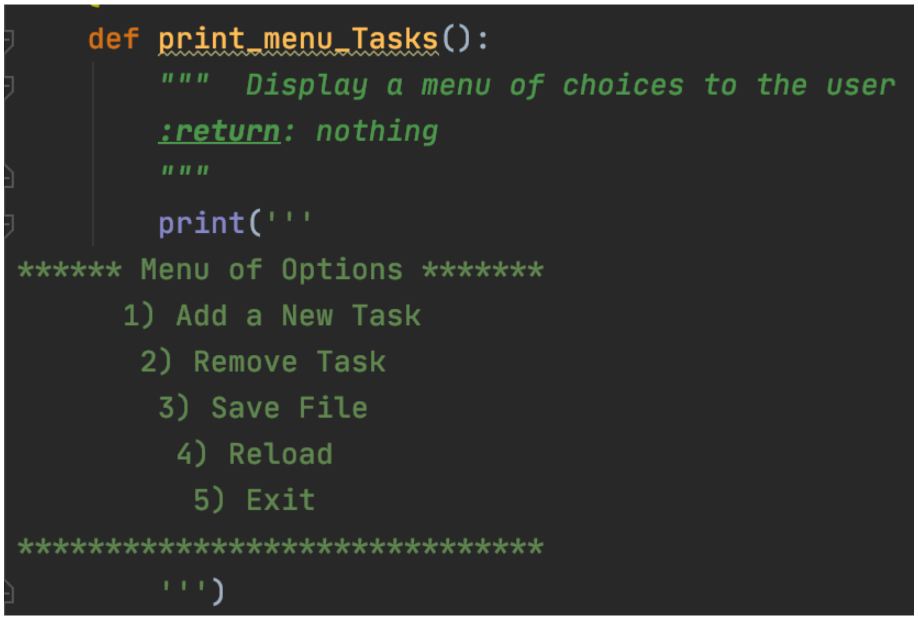
To-Do List Script II

Introduction

This assignment is a continuation from the To-Do script in assignment 05. This is a script that reads from a To-Do List text file then either adds or removes tasks to the file. The first version of this script introduced **Lists** and **Dictionaries**. This version builds on that by including **Classes** and **Functions**. Functions allows the piece of code to be typed once but used in many places throughout the script.

Drafting the Script

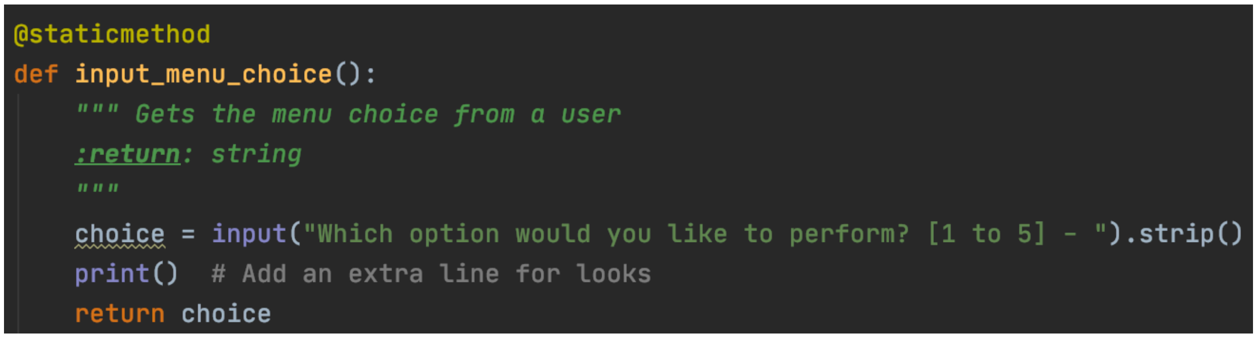
The script is broken into several sections - Data, Processing, Input/Output, and the Main Body, or Core, of the script. The Data section declares the variable and constants to be used in the script. The Processing and Input/Output sections hold the Functions in a Class. One example of how this works is with the function "print_menu_Tasks" (figure 6.1) creates the menu to be displayed to the user.



```
    def print_menu_Tasks():
        """  Display a menu of choices to the user
        :return: nothing
        """
        print('''
****** Menu of Options *******
      1) Add a New Task
       2) Remove Task
        3) Save File
         4) Reload
          5) Exit
******************************
        ''')
```

Figure 6.1 – Menu of Options

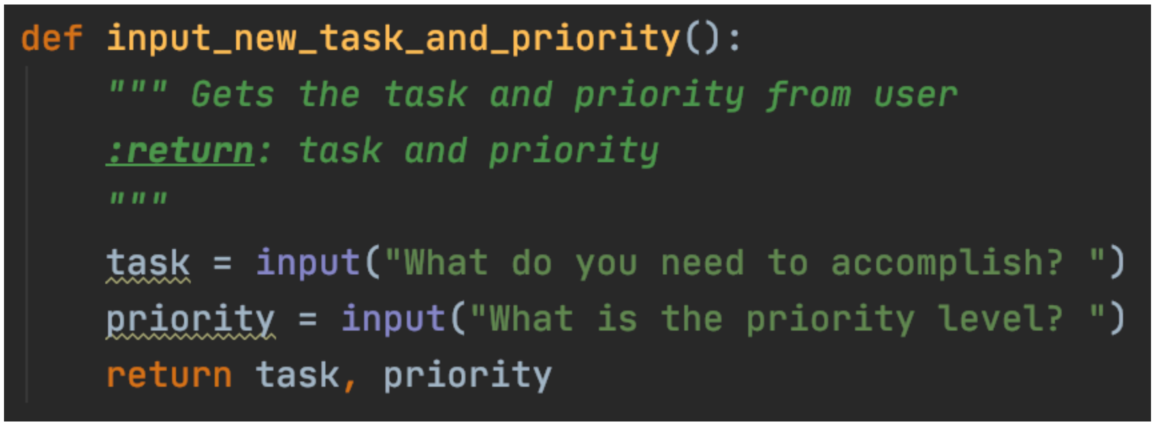
In the core of the script, a while loop (figure 6.2) calls the functions "print_current_tasks_in_lost," "print_menu_Tasks," and "input_menu_choice" (figure 6.2) from the Class "IO."



```
@staticmethod
def input_menu_choice():
    """ Gets the menu choice from a user
    :return: string
    """
    choice = input("Which option would you like to perform? [1 to 5] - ").strip()
    print()  # Add an extra line for looks
    return choice
```

Figure 6.2 – input_menu_choice

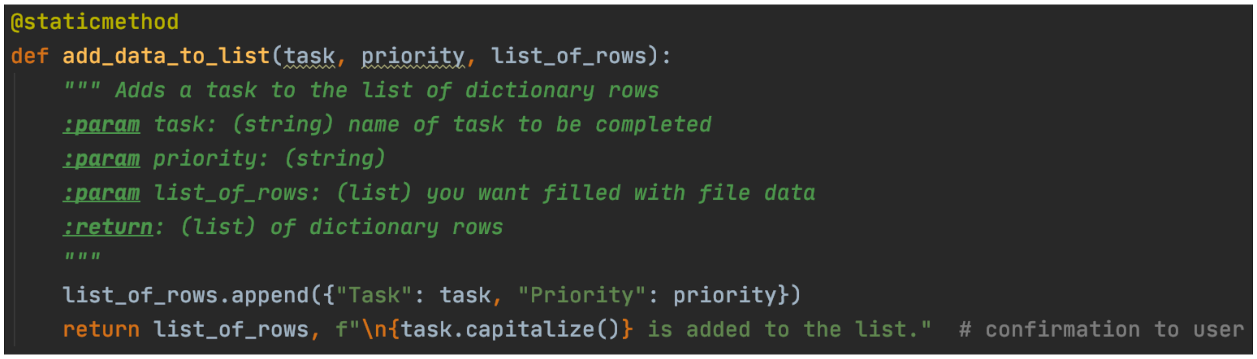
Then input returns the choice given by the user to determine which function to call next. If the user selects "1" to add a task, the function "input_new_task_and_priority," (figure 6.3) also from the "IO" Class will be called. This function will return the task and priority level given by the user.



```python
def input_new_task_and_priority():
    """ Gets the task and priority from user
    :return: task and priority
    """
    task = input("What do you need to accomplish? ")
    priority = input("What is the priority level? ")
    return task, priority
```

Figure 6.3 - input_new_task_and_priority

From here, the function "add_data_to_list" (figure 6.4) will be called from the Class, "Processor." This function will take the input from the user and process it into the list and dictionary containing the rest of the tasks and priorities.



```python
@staticmethod
def add_data_to_list(task, priority, list_of_rows):
    """ Adds a task to the list of dictionary rows
    :param task: (string) name of task to be completed
    :param priority: (string)
    :param list_of_rows: (list) you want filled with file data
    :return: (list) of dictionary rows
    """
    list_of_rows.append({"Task": task, "Priority": priority})
    return list_of_rows, f"\n{task.capitalize()} is added to the list."  # confirmation to user
```

Figure 6.4 – add_data_to_list

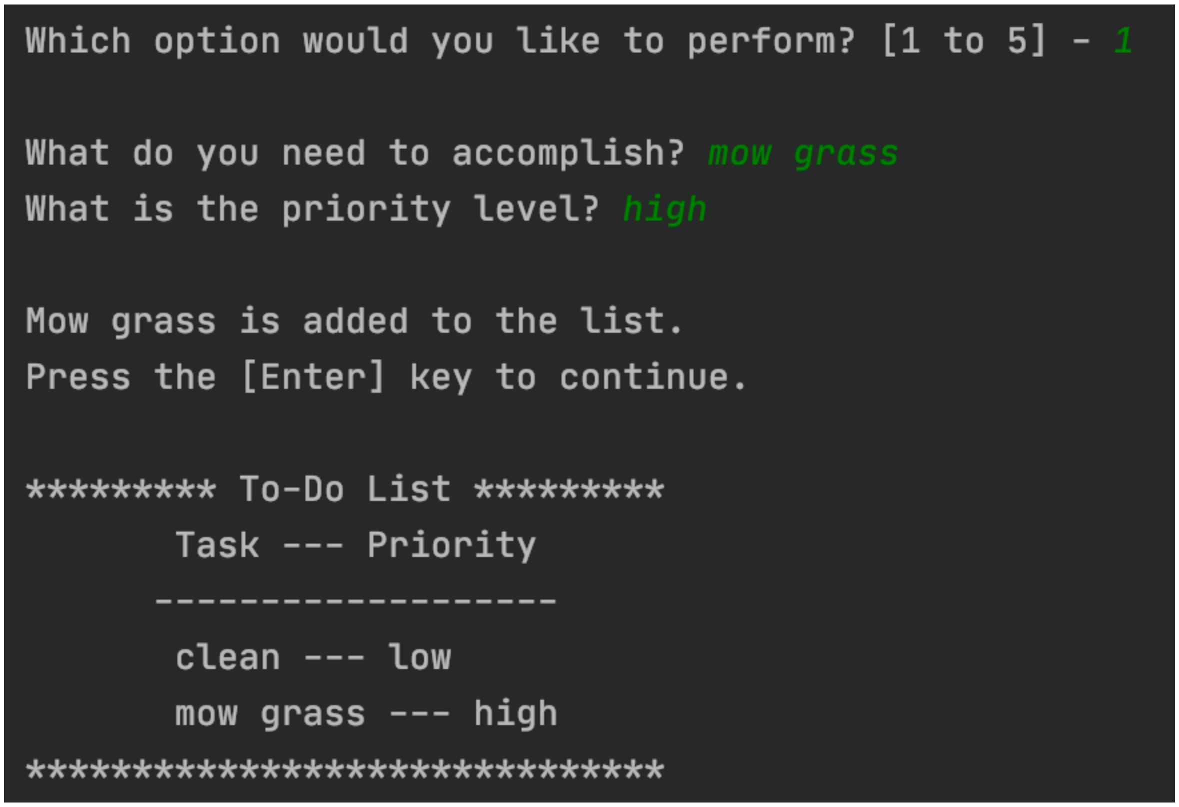
From here, the information can be either displayed, saved to the computer's hard drive, or further edited with the removal and addition of tasks and priorities.

There are many benefits to using functions. First there is the organization of having the same type of code together in like groups. For code that could be used in multiple different ways, it only needs to be written once and having just the name of the function called when it is needs.

Also, when it comes to editing the code and fixing bugs, it is easier to have each piece of code contained in a function as opposed to being spread throughout the script and repeated.

The Result

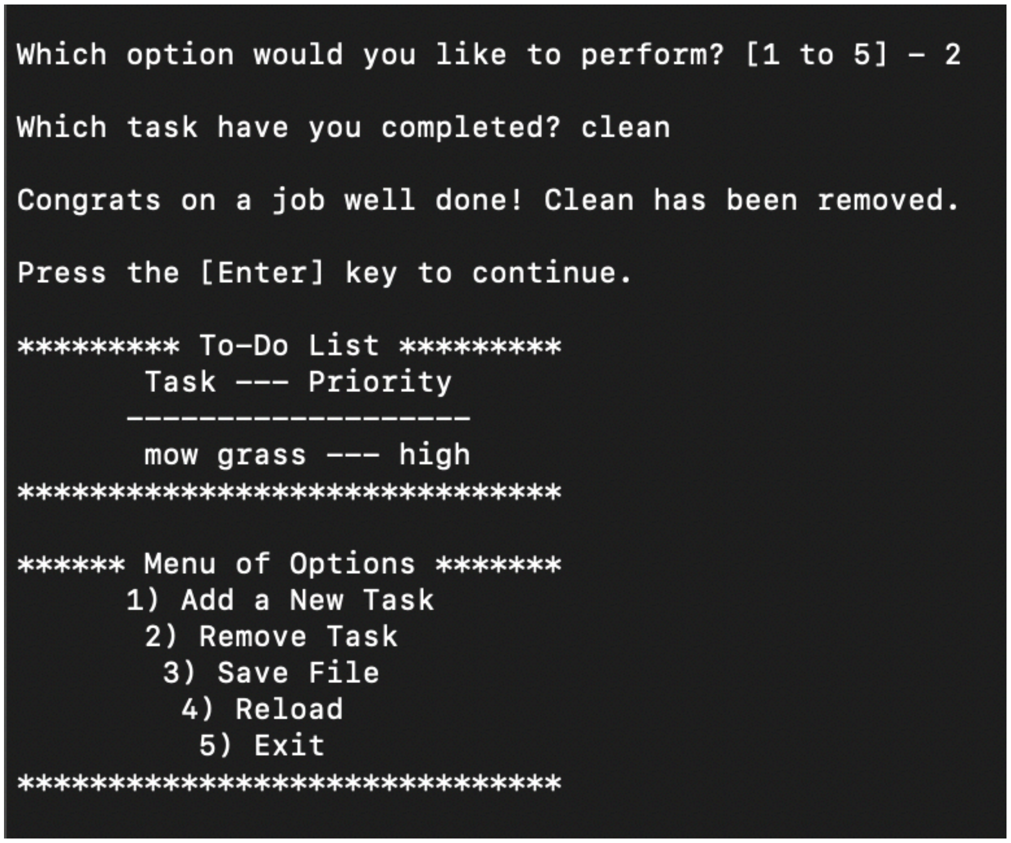
Whether this program is run in PyCharm or Terminal, the same text file will be accessed, and updated just as before. The code is first run in PyCharm to add a couple tasks (figure 6.5).



```
Which option would you like to perform? [1 to 5] - 1

What do you need to accomplish? mow grass
What is the priority level? high

Mow grass is added to the list.
Press the [Enter] key to continue.

********* To-Do List *********
       Task --- Priority
      -------------------
       clean --- low
       mow grass --- high
******************************
```

Figure 6.5 – PyCharm

The text file is then saved and run in Terminal (figure 6.6). A task is removed, and the file is saved again. It does not matter the Python interpreter through which the code is run, the file can be accessed and manipulated.



```
Which option would you like to perform? [1 to 5] - 2

Which task have you completed? clean

Congrats on a job well done! Clean has been removed.

Press the [Enter] key to continue.

********* To-Do List *********
       Task --- Priority
      -------------------
       mow grass --- high
******************************

****** Menu of Options *******
      1) Add a New Task
       2) Remove Task
        3) Save File
         4) Reload
          5) Exit
******************************
```

Figure 6.6 – Terminal

Conclusion

This is script that will read a text file from the hard drive and display its contents to the user. The user can then add tasks and remove tasks once complete. The script can be run from various interpreters all reading and saving the information to the same text file. While it is possible to write such a script using lists and dictionaries, the addition of functions and classes is quite an improvement. I am excited to see how much more improvement to writing scripts lies in the last month of this class.