Matt Dunn
August 13, 2023
IT FDN 110 A
Assignment 06
https://github.com/madunn5/IntroToProg-Python-Mod06

Functions – Improving the To-Do List Script

Introduction

In this paper I will be go over the steps I took to finish the starter code that was provided by Professor Root in order to create a program that functions list a To-Do List. This assignment builds off what we did in Week 5, but instead use functions to do most of the processing work. This was a great exercise in seeing the power of functions, and how it cleans up your code so that someone that is reading the code for the first time can find the Class and subsequent functions being used to process the code.

Creating & Testing the Program

The first step I took when creating the code was to look over the starter code provided by Professor Root. It was very clearly outlined with the expectations for this program, and I decided to work on it one function at time, since most of the rest of the code was already taken care of.

The first section of the code acts a log of information about what the code should accomplish, as well as any changes made to the code so that anyone can come in and get an idea of what is going on. Professor Root started the log, so I added my name, the date, and a brief summary of what I did. In a professional setting it would probably be better to be more detailed about the steps, but since the backbone of this code was already created for me, I decided to just be more concise with my log description.

```
1  # ------------------------------------------------------------------------ #
2  # Title: Assignment 06
3  # Description: Working with functions in a class.
4  #              When the program starts, load each "row" of data
5  #              in "ToDoToDoList.txt" into a python Dictionary.
6  #              Add each dictionary "row" to a python list "table"
7  # ChangeLog (Who,When,What):
8  # RRoot,1.1.2030,Created started script
9  # MDunn,8.9.2023,Updated code to finish project
10 # ------------------------------------------------------------------------ #
```

Figure 1. Top of the code where detail can be added about what the code should accomplish and any changes that have been made to it.

The next part of the code is the declaration of variables. Like Assignment 5, the majority of these were already outlined and didn't require much additional work.

```
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file
file = None  # An object that represents a file
row = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ""  # Captures the user option selection
```

Figure 2. Declaring variables in their own section to have an overview of what to expect down below in the actual code.

The rest of this document I will explain each function and what the expectation of each function is to do. There were two Classes outlined in the code provided by Professor Root, Processor and IO. The Processor Class holds all the functions that have to do with processing data, and the IO Class holds everything that deals with the inputs and outputs of the code. I'll start with the functions in the Processor Class.

The first function is read_data_from_file. This function takes two parameters, file_name and list_of_rows, and returns a list of the dictionary rows. Conveniently, functions in python allow you to outline the parameters (param) and the return value in the function by using triple quotations (the green part of the below screenshot). This will be common throughout all the functions shown in this assignment. Also, list_of_rows will be another commonly used param because that acts as the list of dictionary rows that represent the tasks in the ToDo list.

```
# Processing  ------------------------------------------------------------ #
4 usages
class Processor:
    """  Performs Processing tasks """

    1 usage
    @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 in file:
            task, priority = line.split(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows
```

Figure 3. Defining the function read_data_from_file. This function will read the data currently in the ToDoFile.txt file.

The next function, add_data_to_list, does just that – adds data to the list. The major difference here compared to the previous function is that there are three parameters for this function: task, priority, and list_of_rows. List_of_rows we are already familiar with, but task and priority will be defined in a later function, but they relate to the input values that the user gives to the program.

```
42      @staticmethod
43      def add_data_to_list(task, priority, list_of_rows):
44          """ Adds data to a list of dictionary rows
45
46          :param task: (string) with name of task:
47          :param priority: (string) with name of priority:
48          :param list_of_rows: (list) you want to add more data to:
49          :return: (list) of dictionary rows
50          """
51          row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
52          list_of_rows.append(row)
53          print()  # Add an extra line for looks
54          print(task, 'has been added to the list, but the list has not saved yet!', '\n')
55          return list_of_rows
```

Figure 4. Defining add_data_to_list. This is the first function that has inputs for parameters, which will be defined in the IO Class.

The next function is remove_data_from_list. Just like in Assignment 5, this function defines how to remove a task from the list. In fact, it's my same exact code from before (much like the previous two functions) just reworked so that the code works within this function. This function only takes two params: task and list_of_rows.

```
57      @staticmethod
58      def remove_data_from_list(task, list_of_rows):
59          """ Removes data from a list of dictionary rows
60
61          :param task: (string) with name of task:
62          :param list_of_rows: (list) you want filled with file data:
63          :return: (list) of dictionary rows
64          """
65          found = False
66          for line in list_of_rows:
67              if line['Task'].lower() == task.lower():
68                  list_of_rows.remove(line)
69                  print()  # Add an extra line for looks
70                  print(task, 'has been removed from list, but the list has not been saved yet!', '\n')
71                  found = True
72                  break
73          if found:
74              pass
75          else:
76              print()  # Add an extra line for looks
77              print('Task not found in the To Do List.', '\n')
78          return list_of_rows
```

Figure 5. Defining remove_data_from_list, which will remove a specified task from the ToDo List.

The final function in the Processing Class is write_data_to_file, which we will act as the "save" function for this program. This function takes a new param, file_name, which is essentially just the name of the file, as well as list_of_rows.

```
80      @staticmethod
81      def write_data_to_file(file_name, list_of_rows):
82          """ Writes data from a list of dictionary rows to a File
83
84          :param file_name: (string) with name of file:
85          :param list_of_rows: (list) you want filled with file data:
86          :return: (list) of dictionary rows
87          """
88          file_name = open(file_name_str, 'w')
89          for row in list_of_rows:
90              file_name.write(row['Task'] + ',' + row['Priority'] + '\n')
91          print()  # Add an extra line for looks
92          print('Data Saved!', '\n')
93          return list_of_rows
```

Figure 6. Defining write_data_to_file, which is the final function in the Processing Class.

The IO Class is next, which represents all the input and output tasks required for this code. These functions in general are a little less challenging when it comes to the coding, but they are arguably just as important (if not the most important) because these inputs/outputs are involved in all the Processing Class functions as well.

The first function of the IO Class is output_menu_tasks, which will print the menu of options shown to the user at the beginning of each loop. This function has no parameters and simply acts as a print statement of the menu.

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

    1 usage
    @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
```

Figure 7. Defining output_menu_tasks which prints a menu for the user to use.

The next function is input_menu_choice, which represents the menu choice 1-4 that the user will give at the beginning of each loop. This function also has no parameters, and simply acts to capture the user's input, which is returned at the end of the function.

```python
    @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
```

Figure 8. Defining input_menu_choice which will capture the user's input choice based on the menu of options.

The next function, output_current_tasks_in_list, is the only function of the IO Class that has a param, which is list_of_rows that was used in each of the Processing Class functions. This function uses a for loop to loop through the items currently in list_of_rows and print the task and its priority to the user.

```python
    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("******* The current tasks ToDo are: *******")
        for row in list_of_rows:
            print(row["Task"] + " (" + row["Priority"] + ")")
        print("*" * 40)
        print()  # Add an extra line for looks
```

Figure 9. Defining output_current_tasks_in_list which will show the user what is currently in their ToDo List.

The next function is input_new_task_and_priority, which takes the user's inputs for task and priority and returns them. Task and priority may sound familiar, as they are referenced in some of the functions in the Processing Class.

```python
    @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
        """
        task = input('Please enter a task: ')
        priority = input('Please choose from Low, Medium or High Priority: ')
        return task, priority
```

Figure 10. Defining input_new_task_and_priority which takes an input for task and priority from the user.

The final function of the IO Class is input_task_to_remove, which will ask the user for the task they wish to remove from the list.

```
150     @staticmethod
151     def input_task_to_remove():
152         """  Gets the task name to be removed from the list
153
154         :return: (string) with task
155         """
156         task = input('What task would you like to remove?: ')
157         return task
```

Figure 11. Defining input_task_to_remove, which prompts the user for the task they want to remove from the ToDo list.

Now that all the Classes and their subsequent functions have been created and defined, it's time to work them into the main body of the script! This was already defined by Professor Root ahead of time, so if everything was built correctly in the Classes then the code should work perfectly! I did make a change at the beginning of the code, which was to add in a try/except function so that if the ToDoFile.txt doesn't currently exist then the program will let the user know that file currently does not exist and to add some tasks.

```python
# Main Body of Script  ------------------------------------------------------ #


# Step 1 - When the program starts, Load data from ToDoFile.txt. If no text file currently exists
# then it will let the user know to add some tasks
try:
    Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file data
except:
    print('There is currently no ToDo List file saved. Please add some tasks!', '\n')

# 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)
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print("Goodbye!")
        break  # by exiting loop

```

Figure 12. Main body of the script incorporating all the functions.

Last step now is to test the code! I started from scratch, so there was currently no text file when I started the test. My goal is to add two tasks, Homework and Yardwork, to the script and then remove Homework at the end since I'm now down with this assignment.

```
/Users/Matt/Assignment06/bin/python /Users/Matt/Documents/_PythonClass/Assignment06/Assignment06_Starter.py
There is currently no ToDo List file saved. Please add some tasks!

******* The current tasks ToDo are: *******
***************************************


        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

Please enter a task: Homework
Please choose from Low, Medium or High Priority: High

Homework has been added to the list, but the list has not saved yet!

******* The current tasks ToDo are: *******
Homework (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
```

Figure 13. Running the program in PyCharm to test.

```
Please enter a task: Yardwork
Please choose from Low, Medium or High Priority: Low

Yardwork has been added to the list, but the list has not saved yet!

******* The current tasks ToDo are: *******
Homework (High)
Yardwork (Low)
***************************************


        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

What task would you like to remove?: Homework

Homework has been removed from list, but the list has not been saved yet!

******* The current tasks ToDo are: *******
Yardwork (Low)
***************************************


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

Figure 14. Successfully added Homework and Yardwork, before removing Homework from the list.

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


Data Saved!

******* The current tasks ToDo are: *******
Yardwork (Low)
*****************************************


        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!

Process finished with exit code 0
```

Figure 15. Saved the data and exited the file successfully.

Figure 16. Text file after being saved.

Running the Program in the Terminal

The last thing to test for this assignment was to the run the program in the Mac Terminal.

```
Last login: Sun Aug 13 17:10:59 on ttys000
(base) Matt@Matthews-MacBook-Air ~ % cd /Users/Matt/Documents/_PythonClass/Assignment06/
(base) Matt@Matthews-MacBook-Air Assignment06 % python Assignment06_Starter.py
******* The current tasks ToDo are: *******
Yardwork (Low)
*****************************************


        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

Please enter a task: Buy more dog food
Please choose from Low, Medium or High Priority: Medium

Buy more dog food has been added to the list, but the list has not saved yet!

******* The current tasks ToDo are: *******
Yardwork (Low)
Buy more dog food (Medium)
*****************************************


        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

What task would you like to remove?: Yardwork

Yardwork has been removed from list, but the list has not been saved yet!

******* The current tasks ToDo are: *******
Buy more dog food (Medium)
*****************************************
```

Figure 17. Running the code in the Mac Terminal by navigating to the folder first and then running the program. I add one task before removing a different one.

```
        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!

******* The current tasks ToDo are: *******
Buy more dog food (Medium)
***************************************


        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!
```

Figure 18. Saving then closing the file.

Figure 19. Updated text file with the inputs from the Terminal run.

Summary

This assignment was a great example in how we can uses Classes and Functions to simplify the organization of our code and make the overall result easier to read and run more efficiently on larger scale projects. I can really see my skills starting to develop, and I look forward to getting to expand further on next week's assignment.