

Modifying a Script to Add Functions

Introduction

This document presents the steps taken to modify a script to add functions. This assignment is a continuation of Assignment05 to manage a to-do list. The script allows a user to select from a menu of options. The data being managed is a task and its priority. If data is inputted and saved, the data is stored in a .txt file. PyCharm is used to edit and test the script. IDLE is also used to test the script. There are four sections within the script: data code, process code, presentation code, and the main body of the script.

Data Code

The first section in the script is for declaring variables and constants. This tells the reader/editor what will appear in the script to follow. The None keyword can be used to indicate the absence of a parameter value. Data type is also defined in the data code. A dictionary, list, and string are defined with curly brackets, square brackets, and quotation marks, respectively. See Figure 1 below for the script within the data code.

```
# Data ----- #  
# Declare variables and constants  
file_name = "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 1: Load the Data
```

Process Code

The second section is the process code. This includes all the custom functions. In this assignment, functions must be added to add data, remove data, and write data to a file. Each function includes a docstring which explains the purpose of the function and identifies the inputs and outputs. The functions are grouped into a class. The class and function can then be called later in the script.

Add Data

The append() function is used to add data to a list of dictionary rows. The function returns a value captured as a variable. This allows the variable to be used multiple times without calling the function again. The script below in Figure #2 can be used to add data.

```

@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 filled with file data:
    :return: (list) of dictionary rows
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    list_of_rows.append(row)
    return list_of_rows

```

Figure 2: Add data to a list of dictionary rows

Remove Data

The `remove()` function is used to remove data from a list of dictionary rows. The user is allowed to select the task to be removed. The name of the task to be removed becomes an argument passed into the function. A for loop is used to progress through the rows in the dictionary. An if statement is used to check if the task in the current row is equal to the task to be deleted. The script in Figure 3 can be used to remove data.

```

@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
    """
    for row in list_of_rows:
        if row["Task"].lower() == task.lower():
            list_of_rows.remove(row)
    return list_of_rows

```

Figure 3: Remove data from a list of dictionary rows

Write Data

The `write()` function is used to write data from a list of dictionary rows to a file. The file is opened in “write” mode. A for loop is used to progress through the rows in the dictionary and write them individually. The tasks and priorities are accessed by their key and are separated by a comma. Figure 4 below provides script to write data.

```

@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
    """
    file = open(file_name, "w")

```

```

for row in list_of_rows:
    file.write(row["Task"] + ", " + row["Priority"] + "\n")
file.close()
return list_of_rows

```

Figure 4: Write data from a list of dictionary rows to a File

Presentation Code

The third section in the script is the presentation code. This section of code is used to interact with the user. The inputs from the user define the arguments to be passed into the process functions. In this assignment, functions must be added to get tasks and priorities to be added and removed from the list.

Get Tasks to Add

The input() function is used to gather tasks and priorities from the user to add to the list. The strip() function removes unwanted spaces. Two values are returned, task and priority. See Figure 5 below for the commands to get new tasks and priorities from the user.

```

@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 = str(input("What is the task? - ")).strip()
    priority = str(input("What is the priority? [high | low] - ")).strip()
    print() # Add an extra line for looks
    return task, priority

```

Figure 5: Get task and priority values to be added to the list

Get Tasks to Remove

The input() function is used to get tasks to remove from the list. One value is returned, the task to be removed. The script in Figure 6 is used to ask the user which task they want to remove.

```

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

    :return: (string) with task
    """
    task = str(input("Which TASK would you like removed? - ")).strip()
    print() # Add an extra line for looks
    return task

```

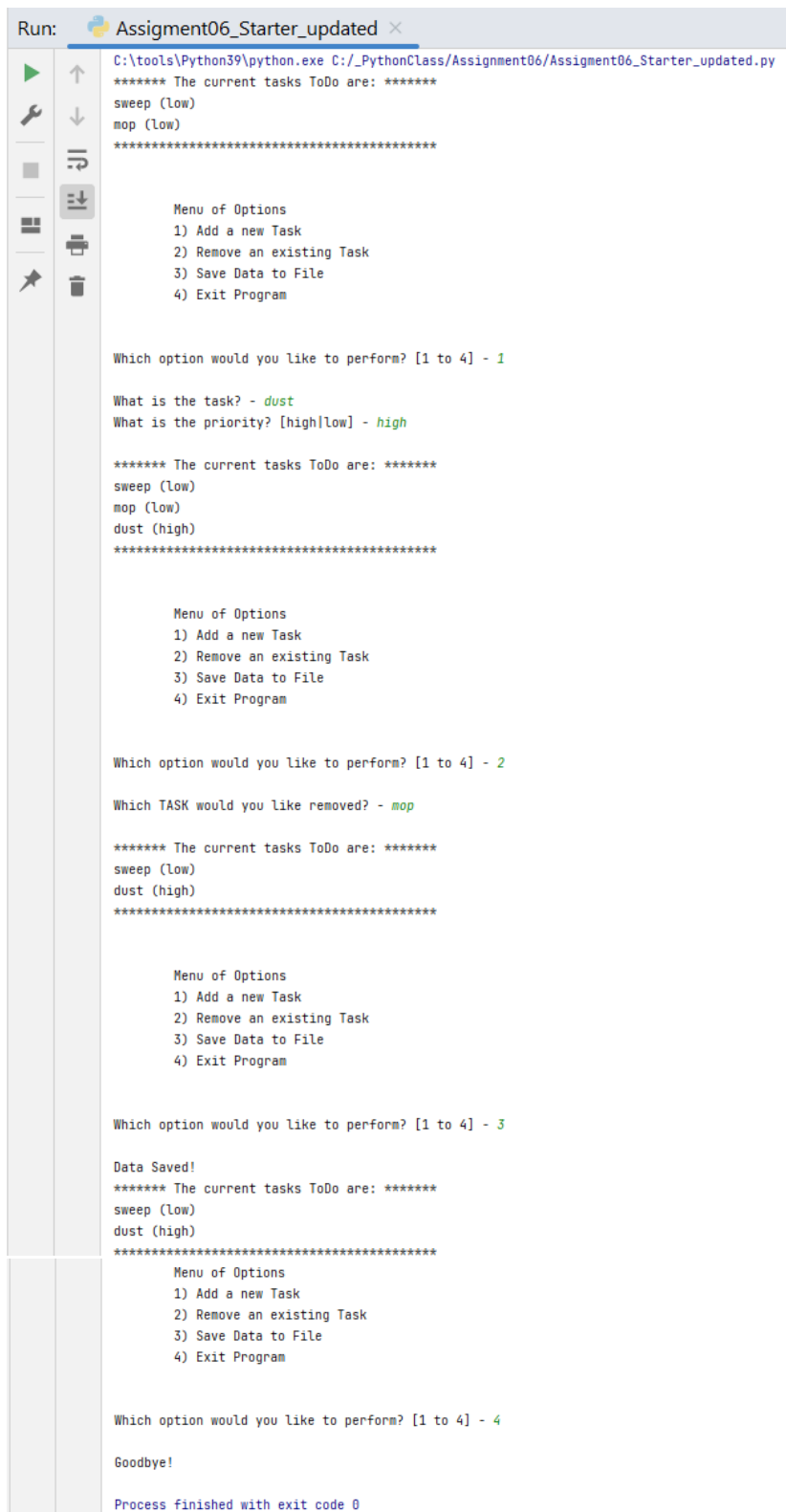
Figure 6: Get the task name to be removed from the list

Main Body of Script

The fourth and last section in the script is the main body. The main body controls when to use the functions defined in the process code and presentation code. The main body drills through the class and selects the function based on the user menu selection. The program is exited with the break command.

Testing the Script

Figure 7 below shows the script working in Pycharm.



```
Run: Assignment06_Starter_updated x
C:\tools\Python39\python.exe C:/_PythonClass/Assignment06/Assignment06_Starter_updated.py
***** The current tasks ToDo are: *****
sweep (Low)
mop (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

What is the task? - dust
What is the priority? [high|low] - high

***** The current tasks ToDo are: *****
sweep (Low)
mop (Low)
dust (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] - 2

Which TASK would you like removed? - mop

***** The current tasks ToDo are: *****
sweep (Low)
dust (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] - 3

Data Saved!
***** The current tasks ToDo are: *****
sweep (Low)
dust (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] - 4

Goodbye!

Process finished with exit code 0
```

Figure 7: Script Running From PyCharm

Figure 8 below shows the script working in a shell window.

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\_PythonClass\Assignment06\Assignment06_Starter_updated.py =====
***** The current tasks ToDo are: *****
sweep (low)
dust (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

What is the task? - cook
What is the priority? [high|low] - low

***** The current tasks ToDo are: *****
sweep (low)
dust (high)
cook (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

Which TASK would you like removed? - sweep

***** The current tasks ToDo are: *****
dust (high)
cook (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] - 3

Data Saved!
***** The current tasks ToDo are: *****
dust (high)
cook (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!
>>> |
```

Figure 8: Script Running From a Shell Window

Figure 9 below shows the data saved to a text file.

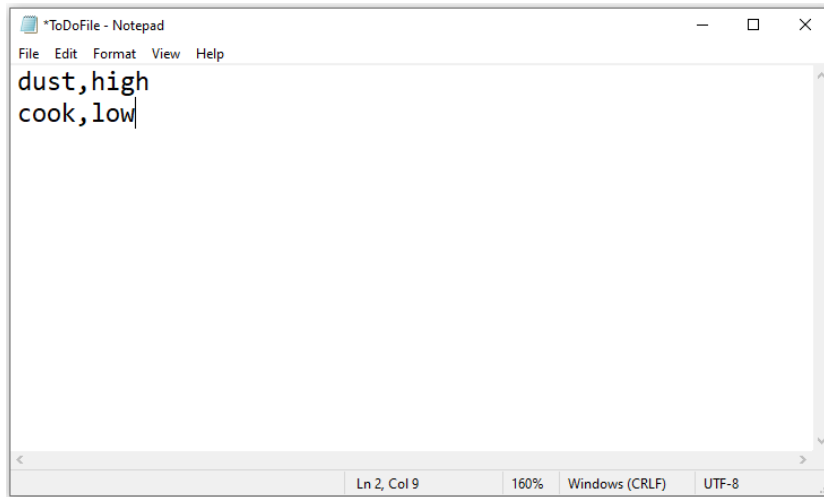


Figure 9: Verifying that the File has Data

Summary

This assignment provides practice making functions and modifying an existing script. Functions allow the code to be separated into data, processing, and presentation code. Functions can also be grouped into classes. These techniques help organize the script and improve readability. This script also incorporates returning data for later use.