Luis Villa
March 20, 2021
Foundations of Programming, Python
Assignment 06
https://github.com/luisv052/ITFoundations-Intro-to-Python-Mod6/tree/main/docs

Organizing with Classes and Functions

Introduction

In this paper I will be going over the process in modifying the assignment 06 starter script with my previous assignment 05 script to create a new script that has all the same features but organized into classes and using functions so that it is easier to read and to test. We will first review some new concepts that were needed. We will then examine the steps I took to craft the script, edit the "ToDoList" text file, and publish the script online. The purpose of this paper is to properly document my procedure in creating a script that showcases classes, functions, the python debugging tool, and organization.

Understanding the Concepts

The goal of this assignment to enhance the code from assignment 05 with betterorganization. For a successful run at this we will be using many of the same concepts and methods as last week's assignment but with the added challenge of using functions, classes, and the python debugging tool.

Functions and Classes

In the processing portion of assignment 05 all I did was load up the data in the text file and then rest of the code was written in the IO section by building the list and dictionary combo, using variables, and a lot of if statements/ for loops. This time around the script was be modified by separating the processing, IO, and main body using functions that serve their bucketed purpose. Professor Root introduces functions by writing "Functions are a way of grouping one or more statements. In Python, you must define a function before you can use code to call the function. Calling the function executes the statements in the function." [Randal Root, Programming with Python module 06]. Classes were then established for processing and IO with each one containing their perspective functions. The main body of the script then consisted of calling up those already defined functions. Figure 1 below shows several examples of functions in the processing class.

```python
# Processing  ------------------------------------------------------------ #
class Processor:
    """  Performs Processing tasks """

    @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(",")
            dicRow = {"Task": task.lower().strip(), "Priority": priority.lower().strip()}
            list_of_rows.append(dicRow)
        file.close()
        return list_of_rows, 'Success'

    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        if priority[0].lower().strip() == "l" or priority[0].lower().strip() == "m" or priority[
         0].lower().strip() == "h":  # keeps priority to three distinct values
            row = {"Task": task.lower().strip(), "Priority": priority[0].lower().strip()}
            list_of_rows.append(row)
            print("\n" + task.lower().strip() + "," + priority[0].lower().strip() + " added!")
        else:
            print("\nInvalid input, please choose from [L]ow [M]edium or [H]igh")
        return list_of_rows, 'Success'

    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        count = 0  # counter started to index hierarchy to avoid duplicate print statements
        for item in list_of_rows:
            if item["Task"] == task.lower().strip():
                list_of_rows.remove(item)
                count += 1
        if count > 0:
            print("\n" + task.lower().strip() + " has been removed")
        else:
            print("\nInvalid Task Selection")
        return task, 'Success'  # returns table with task removed

```

Figure 1: Processing functions with some doc strings

PyCharm Debugger

One purpose behind using functions and classes was to make the script easier to read and easier to test along the way. When everything comes together there might still be some bugs and here is where tools like the PyCharm Debugger shine. When running the code using this tool you are able to go through the code step by step while receiving feedback along the way and you are also able to set breakpoints that act as a pause during the run. Figure 2 demonstrates the tool running and giving feedback on the code.

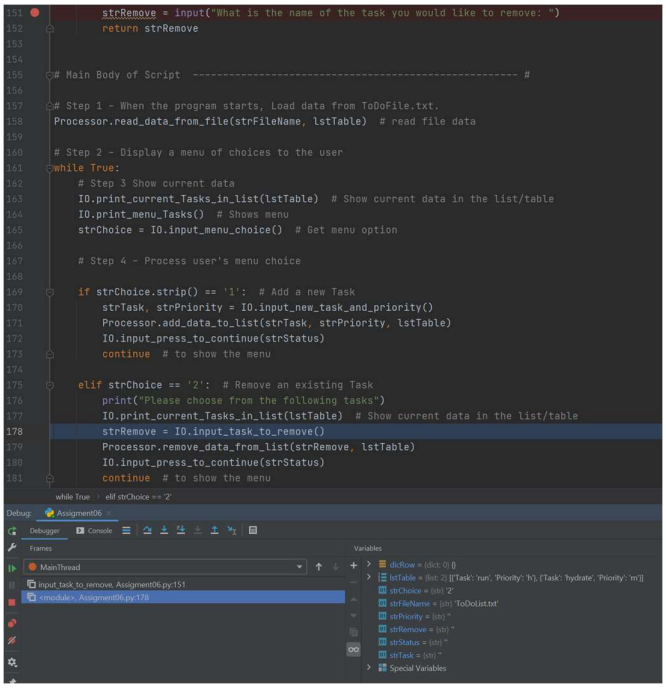

Figure 2: Example of PyCharm Debugger

Completing the Script

Design Intent

This week's assignment is basically giving assignment 05 a makeover with functions and classes. Which means that most of the pseudo code is already made available through the starter file that Professor Root supplies. It also means that user friendly additions used in assignment 05 now must be separated into those that belong in processing and those that belong in IO. Updating the pseudo code to determine where these additions will go gives the programmer a better idea of what goes where before the importing process begins. Shown Below in figure 3 is the Pseudo-Code used that adds comments to better demonstrate the structure and design intent for Processing.

```python
# ------------------------------------------------------------------------------ #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoList.txt" into a python Dictionary.
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# RRoot,1.1.2030,Added code to complete assignment 5
# ------------------------------------------------------------------------------ #

# Data ---------------------------------------------------------------------- #
# Declare variables and constants
strFileName = "ToDoFile.txt"  # The name of the data file
objFile = None   # An object that represents a file
dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
strChoice = ""  # Captures the user option selection
strTask = ""  # Captures the user task data
strPriority = ""  # Captures the user priority data
strStatus = ""  # Captures the status of an processing functions

# Processing  --------------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @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, 'Success'

    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        # TODO: Add Code Here!
        # keep tasks and priorities uniform by stripping and low casing inputs
        # keep priorities to three distinct values Low, Medium, and High
        # restart the loop if user makes an invalid choice for priority
        # inform the user when a change has been made
        return list_of_rows, 'Success'

    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        # TODO: Add Code Here!
        # restart the loop if user makes an invalid choice for task
        # inform the user when a change has been made
        return list_of_rows, 'Success'
```

Figure 3: Modified Template

Writing and Testing

After my goals were made clear I proceeded to make use of the new concepts in their respectful sections. This code required heavy debugging time because the separation of processing and IO was unfamiliar. This separation, however, led to a pleasurable testing phase. During testing different combinations of inputs were used to purposely try and crash the code to ensure as many routes as possible were covered. The completed code can be seen below in figure 4 with the resulting text file in figure 5.

```python
# ------------------------------------------------------------------------- #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoList.txt" into a python Dictionary.
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# RRoot,1.1.2030,Added code to complete assignment 5
# LVilla,03.20.21,Modified code to complete assignment 6
# ------------------------------------------------------------------------- #

# Data ------------------------------------------------------------------- #
# Declare variables and constants
strFileName = "ToDoList.txt"  # The name of the data file
dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
strChoice = ""  # Captures the user option selection
strTask = ""  # Captures the user task data
strPriority = ""  # Captures the user priority data
strStatus = ""  # Captures the status of an processing functions
strRemove = ""  # Captures task to be removed


# Processing  ------------------------------------------------------------ #
class Processor:
    """  Performs Processing tasks """

    @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(",")
            dicRow = {"Task": task.lower().strip(), "Priority": priority.lower().strip()}
            list_of_rows.append(dicRow)
        file.close()
        return list_of_rows, 'Success'

    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        if priority[0].lower().strip() == "l" or priority[0].lower().strip() == "m" or priority[
         0].lower().strip() == "h":  # keeps priority to three distinct values
            row = {"Task": task.lower().strip(), "Priority": priority[0].lower().strip()}
            list_of_rows.append(row)
            print("\n" + task.lower().strip() + "," + priority[0].lower().strip() + " added!")
        else:
            print("\nInvalid input, please choose from [L]ow [M]edium or [H]igh")
        return list_of_rows, 'Success'
```

```python
    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        count = 0  # counter started to index hierarchy to avoid duplicate print statements
        for item in list_of_rows:
            if item["Task"] == task.lower().strip():
                list_of_rows.remove(item)
                count += 1
        if count > 0:
            print("\n" + task.lower().strip() + " has been removed")
        else:
            print("\nInvalid Task Selection")
        return task, 'Success'  # returns table with task removed

    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        file = open(file_name, 'w')
        for item in list_of_rows:
            file.write(item["Task"] + ',' + item["Priority"] + '\n')
        file.close()
        return list_of_rows, 'Success'


# Presentation (Input/Output)  ---------------------------------------- #
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    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 an existing Task
        3) Save Data to File
        4) Reload Data from File
        5) Exit Program
        ''')
        print()  # Add an extra line for looks

    @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 5] - ")).strip()
        print()  # Add an extra line for looks
        return choice
```

```python
    @staticmethod
    def print_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("*****************************************")
        print()  # Add an extra line for looks

    @staticmethod
    def input_yes_no_choice(message):
        """ Gets a yes or no choice from the user

        :return: string
        """
        return str(input(message)).strip().lower()

    @staticmethod
    def input_press_to_continue(optional_message=''):
        """ Pause program and show a message before continuing

        :param optional_message:  An optional message you want to display
        :return: nothing
        """
        print(optional_message)
        input('Press the [Enter] key to continue.')

    @staticmethod
    def input_new_task_and_priority():
        """
        :return: strings
        """
        task = input("Enter a Task: ")
        priority = input("Enter it's priority [L]ow [M]edium or [H]igh: ")
        return task, priority

    @staticmethod
    def input_task_to_remove():
        strRemove = input("What is the name of the task you would like to remove: ")
        return strRemove
```

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

# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(strFileName, lstTable)  # read file data

# Step 2 - Display a menu of choices to the user
while True:
    # Step 3 Show current data
    IO.print_current_Tasks_in_list(lstTable)  # Show current data in the list/table
    IO.print_menu_Tasks()  # Shows menu
    strChoice = IO.input_menu_choice()  # Get menu option

    # Step 4 - Process user's menu choice

    if strChoice.strip() == '1':  # Add a new Task
        strTask, strPriority = IO.input_new_task_and_priority()
        Processor.add_data_to_list(strTask, strPriority, lstTable)
        IO.input_press_to_continue(strStatus)
        continue  # to show the menu

    elif strChoice == '2':  # Remove an existing Task
        print("Please choose from the following tasks")
        IO.print_current_Tasks_in_list(lstTable)  # Show current data in the list/table
        strRemove = IO.input_task_to_remove()
        Processor.remove_data_from_list(strRemove, lstTable)
        IO.input_press_to_continue(strStatus)
        continue  # to show the menu

    elif strChoice == '3':  # Save Data to File
        strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
        if strChoice[0].lower() == "y":
            Processor.write_data_to_file(strFileName, lstTable)
            print("\nFile Saved!")
            IO.input_press_to_continue(strStatus)
        else:
            IO.input_press_to_continue("Save Cancelled!")
        continue  # to show the menu

    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[0].lower() == 'y':
            Processor.read_data_from_file(strFileName, lstTable)
            IO.input_press_to_continue(strStatus)
        else:
            IO.input_press_to_continue("File Reload Cancelled!")
        continue  # to show the menu

    elif strChoice == '5':  # Exit Program
        print("Goodbye!")
        break  # and Exit
```

Figure 4: Final script in Pycharm

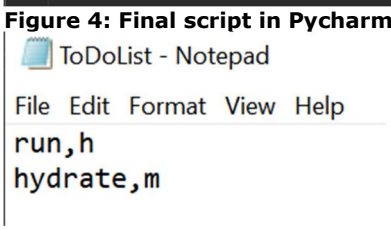

Figure 5: Resulting text file

Running the Script in Command Prompt

At this point the script runs great in the Pycharm IDE (integrated development environment) and the text file is created with the test inputs, it is now time to test it in the command shell. Command Prompt was launched by typing “cmd” in the command menu. Once that was done the file was called by making use of the drag functionality in Microsoft OS, where you can drag the file name from the folder directly to the Command Prompt screen. From there the on-screen instructions were followed, and the program completed its objective seen below in figure 6 and 7. I will also be demonstrating what happens if some invalid options are used.

```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\luise>Python C:\FoundationsPython\ClassModules\module06\assignments\Assigment06.py
******* The current Tasks ToDo are: *******
run (h)
hydrate (m)
*****************************************


        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


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

Enter a Task: AsSiGnment 06
Enter it's priority [L]ow [M]edium or [H]igh: HIGH

assignment 06,h added!

Press the [Enter] key to continue.
******* The current Tasks ToDo are: *******
run (h)
hydrate (m)
assignment 06 (h)
*****************************************


        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


Which option would you like to perform? [1 to 5] - 6

******* The current Tasks ToDo are: *******
run (h)
hydrate (m)
assignment 06 (h)
*****************************************


        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


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

Enter a Task: clean
Enter it's priority [L]ow [M]edium or [H]igh: Med

clean,m added!

Press the [Enter] key to continue.
******* The current Tasks ToDo are: *******
run (h)
hydrate (m)
assignment 06 (h)
clean (m)
*****************************************


        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
```

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

Please choose from the following tasks
******* The current Tasks ToDo are: *******
run (h)
hydrate (m)
assignment 06 (h)
clean (m)
*****************************************

What is the name of the task you would like to remove: run

run has been removed

Press the [Enter] key to continue.
******* The current Tasks ToDo are: *******
hydrate (m)
assignment 06 (h)
clean (m)
*****************************************


        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


Which option would you like to perform? [1 to 5] - 3

Save this data to file? (y/n) - Yes

File Saved!

Press the [Enter] key to continue.
******* The current Tasks ToDo are: *******
hydrate (m)
assignment 06 (h)
clean (m)
*****************************************


        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


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

Goodbye!

C:\Users\luise>_
```

Figure 6: Script running in Command Prompt

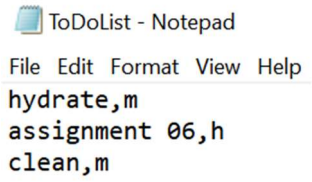

Figure 7: Text File Appended

Publish and Add a GitHub WebPage

The last step is to upload the file to Github which will serve a role in publishing and making your code public, this way it can be reviewed by peers and will be open for improvement. The file for this project can be found through this link (https://github.com/luisv052/ITFoundations-Intro-to-Python-Mod6/tree/main/docs) external site. This week we will also be starting a webpage through GitHub but for now it is just an introduction to creating it (https://luisv052.github.io/ITFoundations-Intro-to-Python-Mod6/) external site.

Summary

This assignment introduced us to a more organized way of programming. Although after it was done it looked neater and everything was easier to test and modify, getting there was quite challenging. Switching to the function mindset had a steep learning curve but I believe it was worth the effort.