Francis Hourigan

February 21st, 2023

IT FDN 110 A Wi 23: Foundations of Programming: Python

Assignment 06

My To Do List with Functions

Introduction

This "To Do List" assignment was not difficult in the code blocks that we needed to insert. The main issues I faced were when I was dealing with the code organized into areas of concern and needed to make sure all the arguments and parameters were correct. I spent a significant amount of my time this week just debugging my code errors and going back through each code block until I found the sources of the errors. Using someone else's starter code was still helpful because I still don't think I could even begin to write a comprehensive script like that off the top of my head. However, the format of the script is not completely foreign to me because in my work with ArcGIS software I have seen both Python and the ArcGIS ARCADE scripting languages use this type of organization. I often use globals and functions to create a custom pop-up window in a web map and that is precisely why I wanted to take this class and get better at the syntax for both languages.

Classes of Functions

Using your own custom functions in your script saves you time in writing the same block of code multiple times. If you organize some code into a custom function, then you remove potential typing errors and bugs. Additionally organizing your functions into classes can make writing your code faster in that you can use pre-exiting classes of functions to execute a particular block of code. The parameters for the function are the only changes that you need to make when adding to your new script.

Breaking Down the Steps

I needed all the help videos for this assignment for the tips on how to connect the code and get all the parameters correct. The code blocks that were given in the demo code are the same ones we have been using since the beginning of the course, so those are straight forward. But now, it is the organization and matching of variables and functions with their parameters, that I find the most challenging.

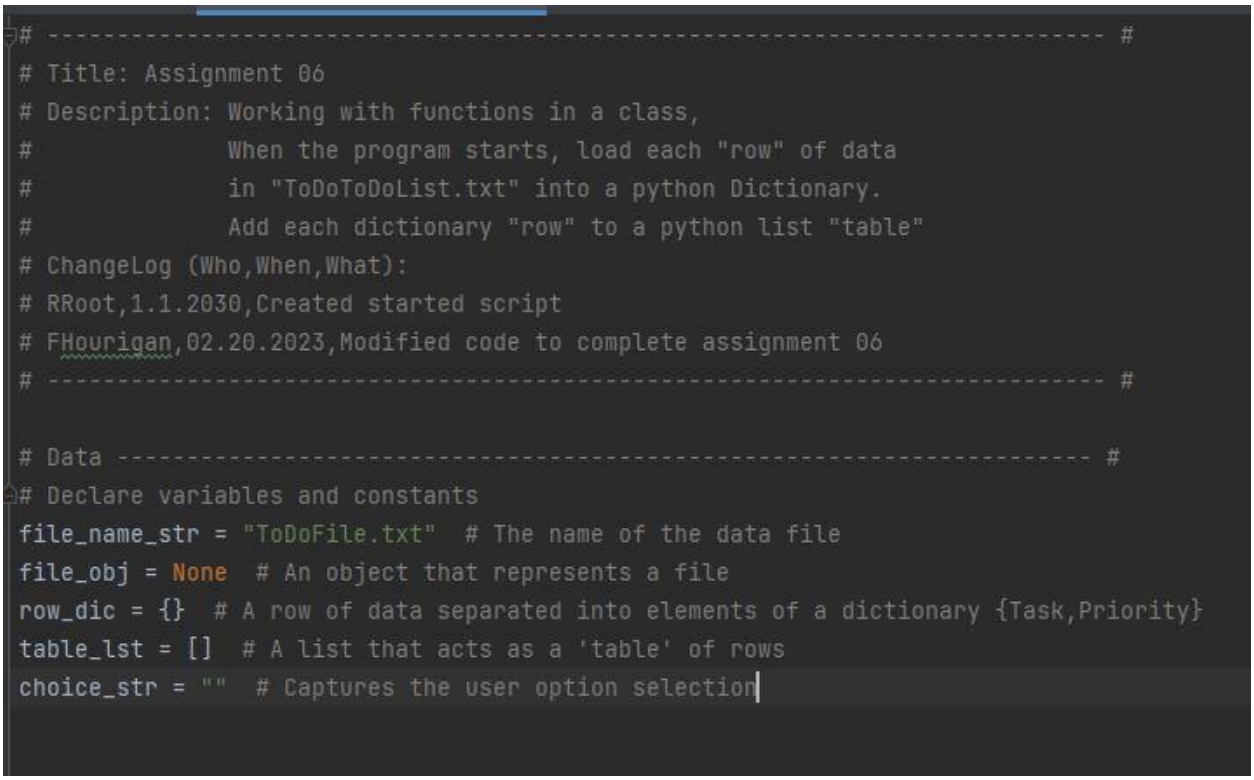
Adding to the Starter Code

I used a combination of the starter code and the assignment demos from last week to finish the script. The areas that I got tripped up had to do with trying to return something that was not a string or not putting the correct parameters into my functions. In addition, debugging my code with the debugger was not all that helpful yet since I was not sure what the debugging messages were trying to tell me.

Additionally, there were some issues with needing the data type to be string and I have not quite wrapped my head around when that is the case.

My To Do List with Functions Script

First, I created the ToDoList.txt file in the same directory as all my other assignment 06 files. This is so I can use the relative file path in my code. Then I added the code to the areas where we were told to #Add Code Here.



```
# ------------------------------------------------------------------------ #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# FHourigan,02.20.2023,Modified code to complete assignment 06
# ------------------------------------------------------------------------ #

# Data ------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # 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 6.1: The Script Header and the Data Section

```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(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows

    @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 to add more data to:
        :return: (list) of dictionary rows
        """
        row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
        list_of_rows.append(row)
        return list_of_rows
```

Figure 6.2: The Processing Section – Selections 1 & 2

```python
@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

@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 6.3: The Processing Section – Selections 3 & 4

```python
# Presentation (Input/Output)  -------------------------------------------- #


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

    @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

    @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 6.4: The Presentation Section – Menu Options and User Selection

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

    @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("Please Enter a New Task: ")).strip()
        priority = str(input("Give Your Task a Priority Level: ")).strip()
        return task, priority

    @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 Have You Completed? ")).strip()
        print()
        return task
```

Figure 6.5 The Processing Section – Current Data, Add Data, and Remove Data

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


# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file data

# 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)
        print("Data Saved!")
        continue  # to show the menu

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

Figure 6.6 The Main Body of the Script

```
C:\UWIntroPython\_PythonClass\Assignment06\venv\Scripts\python.exe C:\UWIntroPython\_
******* The current tasks ToDo are: *******
Homework (High)
Dishes (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 New Task: Pack for work trip
Give Your Task a Priority Level: High
******* The current tasks ToDo are: *******
Homework (High)
Dishes (Low)
Pack for work trip (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 Have You Completed? Dishes

******* The current tasks ToDo are: *******
Homework (High)
Pack for work trip (High)
*******************************************
```

Figure 6.7 The Script Executed in the PyCharm Shell – Part 1

```
Which option would you like to perform? [1 to 4] - 2

Which Task Have You Completed? Dishes

******* The current tasks ToDo are: *******
Homework (High)
Pack for work trip (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: *******
Homework (High)
Pack for work trip (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 6.8 The Script Executed in the PyCharm Shell – Part 2

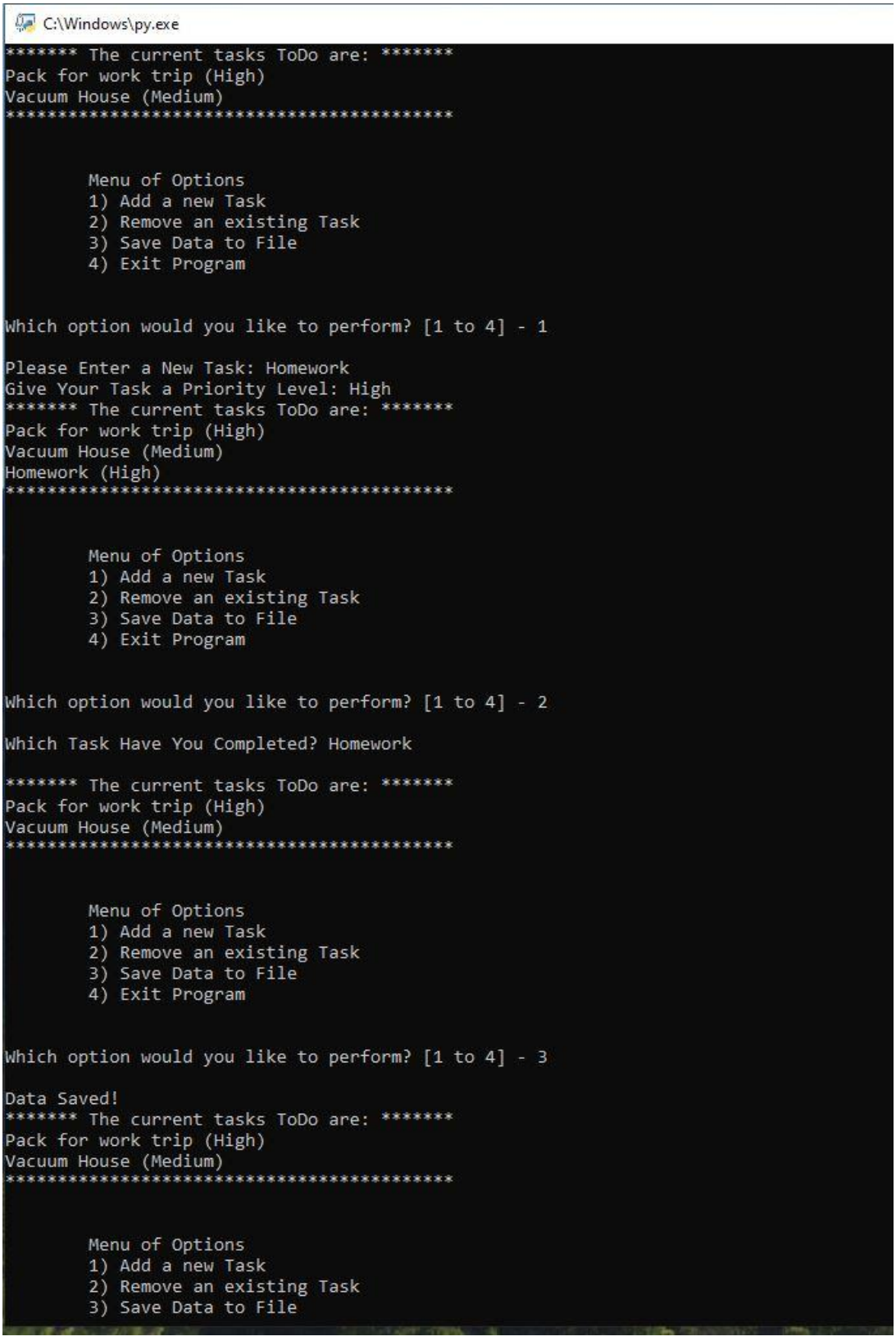


Figure 6.9 The Command Prompt Window – Part 1

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

Data Saved!
******* The current tasks ToDo are: *******
Pack for work trip (High)
Vacuum House (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
```

Figure 6.10 The Command Prompt Window – Part 2

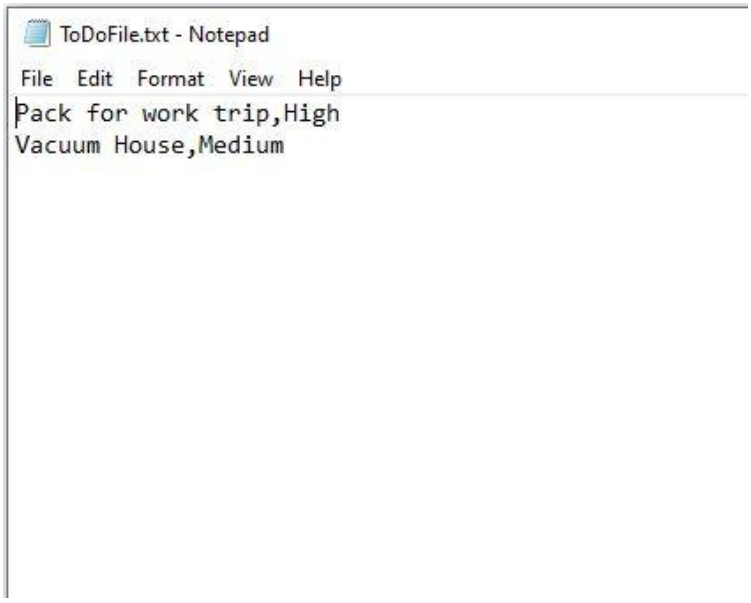


Figure 6.11 The Final Text File

Summary

I understand and can appreciate the concept of organizing your code into manageable sections. The Areas of Concern allows you to see the data, variables, functions you will be using throughout the script in the data section. The processing section which contains the functions organized into classes further

simplifies the need to reinvent the wheel by letting you reuse code blocks and only typing the reference to them. Finally, the presentation section gives you a focused look at the way your code will appear to the user in either the IDE or the Command Prompt window. The text and syntax organization also works well in PyCharm with the debug tool because you can stem through each section and make sure the pieces of code are working as you intend them to. I am still very new at Python and even this simple ToDo List code seems too advanced for me to complete without the starter code and assistance videos. However, the underlying principals and the way that the code works as a whole within the IDE is becoming clearer. Since GitHub is a useful resource for hosting and sharing code, I would assume that most developers reuse the custom functions from each other. I will likely get used to borrowing the code of others before I can sit down and write a python script from start to finish.