Matthew Lord
2023.02.26
IT FDN 110 A Wi 23: Foundations of Programming: Python
Assignment 06
https://github.com/mlord000
GitHub Site: https://github.com/mlord000/mlord000.github.io

Adapting Pre-Written Code Using Classes, Functions, and Separation of Concerns for File Input/Output

Introduction

In this assignment we received pre-written code in which we are to modify and complete in order for the script to operate as intended. The code is presented with the separation of concerns design principle and we will adapt, modify, and complete this code to open, read, and save data from a file for future reference.

Program Setup

The first steps I took with this assignment was to review the current code. Part of reviewing the code for me was to visually follow the steps the code is taking as it is run. Following the data flow and referring to the methods and classes used allowed me to locate the areas where additional code was needed and for what purpose. Most of these code blocks were already indicated in the pre-written code with a text comment reading: "# TODO: Add Code Here!" As in the example below (Listing 1).

```
...

 row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
 # TODO: Add Code Here!
 return list_of_rows

...
```

(Listing 1: Identifying where to place new code)

Reviewing the code in this way was helpful in gaining a basic understanding of the flow of control. Even though I didn't feel as though I understood every detail of the provided code, I did feel like I had a general idea which lended itself to a way forward.

Developing the Script

After reviewing the code I determined the best way forward for me would be to run the code and deal with any errors as they came up, including handling the input/output and formatting. This appeared to work well in this case. It became clear that for much of the code, the functionality was very similar to what we worked on in the previous assignment (Assignment 05). I found myself copying and pasting code from the previous assignment and making changes where needed, including added functionality and input handling.

An example of this is in the function: write_data_to_file() (listing 2)

```
   @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
        """
        # TODO: Add Code Here!
        return list_of_rows
```

(Listing 2: A function to be written)

For this method I copied Step #6 from assignment five (listing 3):

```
    # Step 6 - Save tasks to the ToDoToDoList.txt file
    elif (strChoice.strip() == '4'):
        # open the file to save the data to
        objFile = open(strFile, "w")
        # format the lstTable row by row
        for row in lstTable:
            dicRow = row  # assign the row to a dictionary
            # save the data, formatted as CSV
            objFile.write(dicRow["task"] + "," + dicRow["priority"] + "\n")
        objFile.close()  # close the file
        # let the user know that the data has been saved
        print("The data has been saved!")
        continue  # continue to the while/menu loop
```

(Listing 3: Code from a previous assignment which appeared could work in the current assignment)

I changed the variables used to fit the pre-written code for our current assignment and it seemed to do what is needed for the script to work properly. A big difference however is that the code from Assignment 05 was in the main body of the script and after using it in Assignment 06, this code is now part of a function within the "class Processor:" (listing 4):

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

    ...

    @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 saved to file:
        :return: (list) of dictionary rows
        """
        # open the file to save the data to
        file_obj = open(file_name, "w")
        # format the list_of_rows, row by row
        for row in list_of_rows:
            dic_row = row  # assign the row to a dictionary
            # save the data, formatted as CSV
            file_obj.write(dic_row["task"] + "," + dic_row["priority"] + "\n")
        file_obj.close()  # close the file
        print("Data Saved!")
        print()  # Add an extra line for looks
        return list_of_rows

    ...
```

(Listing 4: Adding and updating variables in the code to work for a function)

Using this process of running the code and focusing on only the errors which came up (whether they were actual errors, formatting issues, or input/output controls) became a fairly fluid way for me to feel comfortable in handing the assignment. References code previously written also helped.

Conclusion

In this assignment we transferred tasks of a script which we had used previously into an incomplete and pre-written python script which contains code blocks distinguished by the separation of concerns principle and incorporated class and functions. This build on what our experience with coding had previously been allowing us the opportunity to reuse our own pre-written code to incorporate it.