Deja Monet

Randal Root

Foundations of Python

31 August 2022

GitHub

Assignment 08: Objects and Classes

Introduction

In this assignment, I will review standard patterns of classes, creating object instances in classes, and calling this data from functions in other classes.

Completing the Assignment: Reading and Editing the Pseudo-code

In this assignment, no clear output was provided, and part of the challenge was understanding the pseudo-code in order to complete the assignment. The pseudo-code showed three classes, **Product**, **FileProcessor**, and **IO**, and the main body of the script. Each class called for certain properties and methods to be added by the student. From the pseudo-code in the main body of the script, it became clear that the assignment was to process user input "products" and "product prices" by displaying a menu of choices to the user and processing input data based on the user's choice from this menu.

I decided to use Jupyter Lab in order to organize the classes and functions, and to be able to troubleshoot errors with any of the classes in blocks. I began by declaring the variable **user_choice** as an empty string. I then defined a class **Errors** with the functions **user_input**, **no_numeric**, and **too_many_characters** in order to make sure the user entered appropriate inputs in the script (**Figure 1**).

```
class Errors:
    def user_input():
        print("Something went wrong with the user input...")
    def no_numeric():
        print("You must enter a number!")
    def too_many_characters():
        print("You entered too many characters!")
```

Figure 1. Errors that have been defined by the programmer.

The pseudo-code for the **Product** class used the properties **product_name** and **product_price**, so I initialized these variables as an empty string and the float value 0.00, respectively. I then defined the function **add_data_to_list**, stored the user input data in a temporary array **row**, appended **row** to the pre-defined variable **lstOfProductObjects**, and alerted the user that this process was successful (**Figure 2**).

```python
class Product:
    """
    Stores data about a product:

    properties:
        product_name: (string) with the product's  name

        product_price: (float) with the product's standard price
    methods:
        add_data_to_list(): processes user input data into a temporary array that is then appended a list of arrays
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        dejam,220829:
            - modified code to complete assignment 8
            - created add_data_to_list function
    """
    #product_name and product_price are initialized
    product_name = ""
    product_price = 0.00

    #add data to list of product objects
    def add_data_to_list(product_name, product_price):

        #add user input data to array
        row = [product_name, product_price]

        #add array to list of arrays
        lstOfProductObjects.append(row)

        #alert user their data has been successfully processed
        print("Your data has been added to the list!")

        #return updated list of rows
        return lstOfProductObjects
```

Figure 2. The completed Product class with the defined function add_data_to_list.

The next class, **FileProcessor**, needed defined functions to save the user input data to a file and to load data from a file. I defined the function **load_data_from_file** within a **try/except** statement, such that when the file does not exist, the script alerts the user of this and continues to the main menu instead of exiting (**Figure 3**).

```python
# Processing  ------------------------------------------------------------ #
class FileProcessor:
    """
    Processes data to and from a file and a list of product objects:

    methods:
        save_data_to_file(file_name, list_of_product_objects):

        read_data_from_file(file_name): -> (a list of product objects)

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        dejam,220829:
            - modified code to complete assignment 8
            - created load_data_to_file and save_data_to_file functions
    """
    #data is loaded from file
    @staticmethod
    def load_data_to_file(strFileName):
        try:
            lstOfProductObjects.clear() #clear current data from list
            file_load = open(strFileName, "r")

            #array is loaded as rows into lstOfProductObjects table
            for line_cycle in file_load:

                #paired product_name and product_price are identified by the comma between them
                line_split = line_cycle.split(",")
                product_name = line_split[0].strip()
                product_price = line_split[1].strip()

                #row is the product name and product price identified above (with any leading/trailing spaces removed)
                row = [product_name, product_price]

                #row is appended to lstOfProductObjects
                lstOfProductObjects.append(row)

            file_load.close()
            print("Data loaded from file!")
            return lstOfProductObjects

        except FileNotFoundError:
            print("No data to load from file!")
```

Figure 3. The function load_data_to_file in the FileProcessor class.

The function **save_data_to_file** uses a for loop to cycle through the arrays in **lstOfProductObjects** and writes the product name and product price as strings to the file **products.txt** (**Figure 4**).

```python
#data is saved to file
@staticmethod
def save_data_to_file(strFileName, lstOfProductObjects):

    #products.txt is identified as file_obj
    file_obj = open(strFileName, "w")

    #write user input to-do items to products.txt
    for row in lstOfProductObjects:
        file_obj.write(row[0] + ", " + str(row[1]) + "\n")

    #products.txt is closed
    file_obj.close()

    #user is alerted that their data has successfully been saved
    print("Your data has been saved!")
```

Figure 4. The function save_data_to_file in the FileProcessor class.

The next class was the **IO** class, which receives and the user input data and prints outputs when the user makes that choice. I defined four methods in this class: **main_menu** that prints the main menu to the user; **input_menu_choice** that captures the user's choice from the main menu; **output_current_data** that prints the current data to the user when the user makes that choice; and **input_product_data** that uses the product_name and product_price variables from the Product class to capture and return the user input data for these respective variables (**Figure 5**).

```
# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """
    Receives user input and processes output to user

    properties:
        lstOfProductObjects: a list of Python arrays that contain the user's input product_name and product_price
    methods:
        main_menu(): prints main menu to user
        input_menu_choice(): receives the user's choice from the main menu and returns this choice
        output_current_data(): prints the user's current data when the user makes that choice
        input_product_data: receives the user's input product_name and product_price
    changeLog:
        dejam, 220829:
            - created docString
            - created functions listed above in "methods" to complete assignment
    """

    #show main menu to user
    @staticmethod
    def main_menu():
        print("""
        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program
        """)

    #return user's choice from main menu
    @staticmethod
    def input_menu_choice():
        user_choice = str(input("Which option would you like to perform? [1 to 3]: ")).strip()
        return user_choice

    #show the current data from the file to user
    @staticmethod
    def output_current_data(lstOfProductObjects):
        if(lstOfProductObjects != []):
            print("Here is the current product data: ")
            for row in lstOfProductObjects:
                print(row[0] + ", " + str(row[1]))
        else:
            print("There is no data to display!")

    #get product data from user
    @staticmethod
    def input_product_data():
        #retrieve user input product name
        Product.product_name = input("Enter a product name: ")

        #retrieve user input product price
        Product.product_price = float(input("Enter the product's standard price: "))

        #return user input product name and product price
        return Product.product_name, Product.product_price
```

Figure 5. The completed main_menu, input_menu_choice, output_current_data, and input_product_data functions of the IO class.

Next, I needed to complete the main body of the script by calling the appropriate functions. The first step was to load any data when the script is run, so I called the **load_data_to_file** function from the **file_processor** class with the input **strFileName**. I then created a while loop to continuously show the user the main menu and process their choice until they chose to save their data and exit the script. I added a series of if statements such that if the user does not choose an option from the menu by entering too many characters or not entering a number, an error is shown and they are returned to the main menu.

To process the user's choice, I called **IO.output_current_data** when the user chooses to see their current data, **IO.input_product_data** and **Product.add_data_to_list** when the user chooses to enter more data to the list of product objects, and **FileProcessor.save_data_to_file** when the user chooses to save their data and exit the script (**Figure 6**).

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

#load data from file into a list of product objects when script starts
FileProcessor.load_data_to_file(strFileName)

#continue to return to main menu until user chooses to save and exit script
while(True):
    #show user a menu of options
    IO.main_menu()

    #get user's choice from menu of options
    user_choice = IO.input_menu_choice()

    #return to main menu if user does not enter a number
    if(user_choice != "1" and user_choice != "2" and user_choice != "3"):
        Errors.user_input()
        if(user_choice.isnumeric() == False):
            Errors.no_numeric()
        if(len(user_choice) != 1):
            Errors.too_many_characters()

    #show user current data in the list of product objects
    if(user_choice.strip() == "1"):
        #show user the current data
        IO.output_current_data(lstOfProductObjects)

    #Let user add data to the list of product objects
    if(user_choice.strip() == "2"):
        #ask for user input product name and product price
        product_name, product_price = IO.input_product_data()

        #add user input product name and product price to list of product objects
        Product.add_data_to_list(product_name, product_price)

    #Let user save current data to file and exit program
    if(user_choice.strip() == "3"):
        #save current data to file
        FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)

        #exit script
        print("Exit script!")
        break
```

Figure 6. The completed main body of the script.

With the script completed, I could begin testing the script in an IDE and Anaconda Prompt.

Completing the Assignment: Testing the Script in Jupyter Lab

I began by testing the script in Jupyter Lab. I started with no existing products.txt file in the directory. I then directed the script to try to show the current data, then entered the string “abc” when prompted to choose an item from the main menu. I then added the products apple and carrots with the prices 2 and 3 respectively, then saved this data and exited the script (**Figure 7**).

```
No data to load from file!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  1
There is no data to display!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  abc
Something went wrong with the user input...
You must enter a number!
You entered too many characters!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  1
There is no data to display!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  2
Enter a product name:  apple
Enter the product's standard price:  2
Your data has been added to the list!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  2
Enter a product name:  carrots
Enter the product's standard price:  3
Your data has been added to the list!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  3
Your data has been saved!
Exit script!
```

Figure 7. Part 1/2 of testing the script in Jupyter Lab.

I then ran the script again in Jupyter Lab, called the script to show the current data, added the product grapes with the price 6, then saved the data and exited the script (**Figure 8**).

```
Data loaded from file!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  1
Here is the current product data:
apple, 2.0
carrots, 3.0

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  2
Enter a product name:  grapes
Enter the product's standard price:  6
Your data has been added to the list!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]:  3
Your data has been saved!
Exit script!
```

Figure 8. Part 2/2 of testing the script in Jupyter Lab.

With the script working correctly in Jupyter Lab, I moved on to testing the script in Anaconda Prompt.

Completing the Assignment: Testing the Script in Anaconda Prompt

To test the script in Anaconda Prompt, I first changed the working directory to the directory where the script was saved. I called the Jupyter Lab .ipynb script using ipython command, then showed the data loaded from the products.txt file. I then added the product ice cream with the price 4, and the product milk with the price 3 (**Figure 9**).

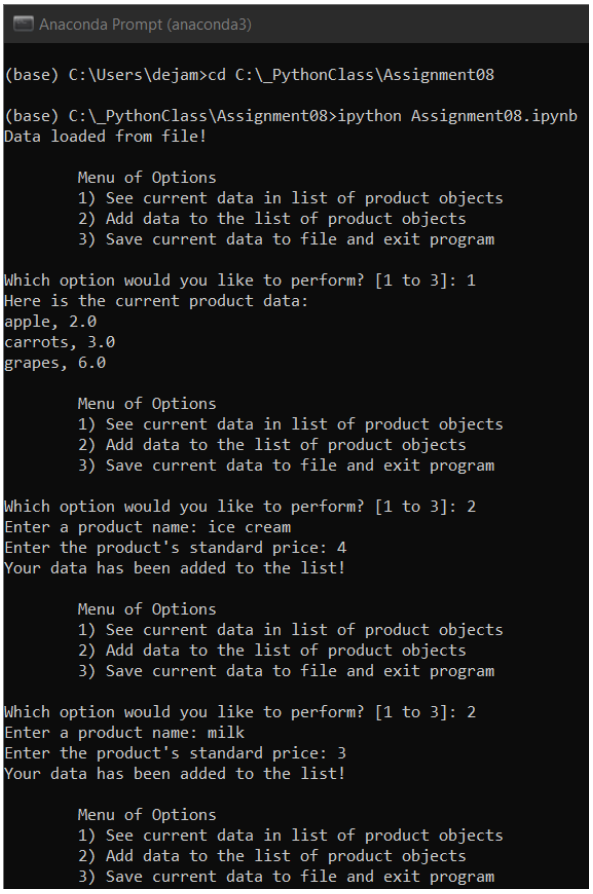

Figure 9. Part 1/2 of testing the script in Anaconda Prompt.

I then displayed the current data, then saved the current data and exited the program (**Figure 10**).

```
Which option would you like to perform? [1 to 3]: 2
Enter a product name: milk
Enter the product's standard price: 3
Your data has been added to the list!

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]: 1
Here is the current product data:
apple, 2.0
carrots, 3.0
grapes, 6.0
ice cream, 4.0
milk, 3.0

        Menu of Options
        1) See current data in list of product objects
        2) Add data to the list of product objects
        3) Save current data to file and exit program

Which option would you like to perform? [1 to 3]: 3
Your data has been saved!
Exit script!

(base) C:\_PythonClass\Assignment08>
```

Figure 10. Part 2/2 of testing the script in Anaconda Prompt.

With the script working properly in both Jupyter Lab and Anaconda Prompt, the assignment was complete.

Conclusions

In this assignment, I reviewed the standard design pattern of classes in coding languages. I also reviewed creating and calling data from object instances in classes.