Aisha Shafique
6.2.2022
Foundations of Programming: Python
Assignment 8
https://github.com/Ai-Shaf/IntrotoProg-Python-Mod08

Class Objects and Static Methods

So far in class, we have covered classes with static method functions. In Module 8, we look at how classes are used to create object instances, as well as static methods. Assignment 8 begins with starter code that is primarily composed of pseudo-code for completing the assignment where we will create three classes and then a main script to run out program.

Writing the Script

Product Class & Object Instances

We can create a class and initialize it with a construction function, which sets attributes when class objects are initially created. We can use accessor and mutator functions to establish the properties for the attributes, rather than directly listing attributes to the constructor function. We can also overwrite the built in __str__() method so that we can customize what the string value for our object would be. In this manner, when we create an instance object of a class, we have control over the properties that we would want that object to have.

In the case of our assignment, we are creating a product class and our eventual aim will be to create a list of product objects, which we will be able to read from file, save to file, modify, and display to user.

Therefore, we start off by defining our data class as Product() (using the python convention of having the class begin with a capital letter). Listing one below illustrates my code for the product object class.

As we can see, I start off by setting up the __init__() function, which takes the first parameter as self, required for all instance methods, and then it has parameters for product name and product price. Our product instance will be defined by both of these values.

My getter functions, under the @property decorator, simple format the product name and product price. In the case of the product name, the first letter of each word is capitalized and for the product price, it is simply returned. Because these are instance methods, they begin with self.__property_name(). The property name begins with two underscores to symbolize a hidden variable.

```python
class Product(object):
    """Stores data about a product:

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

        product_price: (float) with the product's standard price
    methods:
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        AShafique, 6.2.2022,Modified code to complete assignment 8
    """

    # ---Constructor--- #
    def __init__(self, product_name, product_price):
        self.product_name = product_name
        self.product_price = product_price

    # ---Overwrite the string method--- #
    def __str__(self):
        return self.product_name + ' | ' + self.product_price

    # ---Properties of Class: Accessor and Mutator--- #
    # ---Product Name Getter--- #
    @property
    def product_name(self):
        return str(self.__product_name).title()

    # ---Product Name Setter--- #
    @product_name.setter
    def product_name(self, value):
        if str(value).isnumeric() == False:
            self.__product_name = value
        else:
            raise Exception("Product name cannot be numeric!")

    # ---Product Price Getter--- #
    @property
    def product_price(self):
        return str(self.__product_price)

    # ---Product Price Setter--- #
    @product_price.setter
    def product_price(self, value):
        if bool(float(value)) is True:  # check if string value can be
#converted to float
            self.__product_price = value
        else:
            raise Exception("Product price must be a float!")
```
Listing 1: The Product Class

The getter functions check to see if the values going into the functions are what we want. For name, I check to make sure it is not numerical, and for the price I make sure that it is something that can be converted into a float. Once the functions check for this, they set the value to the product name and product price, and then that passes into the constructor function as attributes for the class.

Last, but not least, I overwite the __str__() method for the class so that it returns the product name and product price separated by a ‘|’. When later I make my list of objects, I will already thus have my product name and price formatted by calling the string method.

FileProcessor Class & Static Methods

My second class, FileProcessor, uses only static methods to save to file and read to file. Listing 2 provides the code used for this class.

```
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
        AShafique,6.2.2022,Modified code to complete assignment 8
    """
    @staticmethod
    def save_data_to_file(file_name, list_of_product_objects):
        with open(file_name, 'w') as file:
            for row in list_of_product_objects:
                file.write(row.__str__() + '\n')
            file.close()

    @staticmethod
    def read_data_from_file(file_name):
        list_of_product_objects = []
        try:
            file = open(file_name, 'r')
            for each in file:
                name, price = each.split('|')
                name = name.strip()
                price = price.strip()
                product_obj = Product(name, price)
                list_of_product_objects.append(product_obj)

        except FileNotFoundError as e:
            print("File not found. You can add data and create file.")
        except Exception as e:
            print("No data.")
        finally:
               return list_of_product_objects
```
Listing 2: The FileProcessor class

As we can see, I use the decorator @staticmethod above each of my two functions.

For my save function, I open the file and use a file handle as an instance of that class I'm calling, and then iterate over my list of product objects and write them into the file. I use the string method for each row element so that my product name and price are saved together in one go, as formatted previously in the string method. The file name I wish to save the data to and list of objects I'm drawing data from are the two arguments I pass into the two parameters for this function. I close the file at the end.

For the read function, I iterate over each line from the file and split the rows, separating out name and price. Then, I can strip the '\n' from the data and create a product object from the Product() class, passing the name and price variables I'm reading as the attributes of the instance created from the Product class.

I then append each of these product objects into the table of lists.

I use try except blocks to capture errors such as FileNotFoundError and other exceptions that may occur if there is no data in the file, or if the file does not exist.

I return my list using *finally.* This is because no matter whether a file exists, I still want an empty list returned. That way, when I use my global list variable in the main script to capture the output from this read function, when there is no file, it will still pass an empty list and retain the global list variable (designated as lstofProductObjects) as a list. Otherwise, no file would lead to nothing being past to lstofProductObjects, overwriting its list type and preventing it being used as a list in subsequent code (essentially breaking the program).

Class IO – Input/Output & Static Methods

In this last class I have a variety of functions that take user input and store it or display it and display menu options and data to the user. For these, I also use static methods as I am not using this class to create object instances. Listing 3 showcases the code for this class, including the doc string that was to be added as part of the assignment.

```
class IO:
    """ Takes input from user and outputs processed data:

    methods:
        display_menu() -> prints out menu of options for user to choose from
        menu_choice() -> returns the user's choice from menu
        display_file_data(file_name) -> displays the data from the file
        display_table_data(list_table) -> iterates over list and displays the
line items
        get_user_data() -> returns inputted user data
    changelog: (Who, When, What)
        RRoot, 1.1.2030, Created Class
        AShafique, 6.2.2022, Added code to complete assignment 8
    """

    @staticmethod
    def display_menu():
        print(
            """
            Menu of Options:
                1 - Display current data in table
                2 - Add data
                3 - Save data to file
                4 - Read data from file
                5 - Clear data from list
                6 - Exit
            """
        )

    @staticmethod
    def menu_choice():
        choice = input("Enter choice from menu (1,2,3,4, 5, or 6): ")
        return choice

    @staticmethod
    def display_file_data(file_name):
        print('Product Name' + '|' + 'Product Price')
        try:
            table_lst = FileProcessor.read_data_from_file(file_name)
            for each in table_lst:
                print(each)
        except FileNotFoundError as err:
            print("File or data not found.")
            print(err)
        except TypeError as err:
            print("Empty list is not iterable")
            print(err)
        except Exception as err:
            print(err)

    @staticmethod
    def display_table_data(list_table):
        try:
            print('Product Name' + '|' + 'Product Price')
            for line in list_table:
                print(line.__str__())
        except TypeError as e:
            print(e)
        except Exception as e:
            print(e)


    @staticmethod
    def get_user_data():
        try:
            name = input("Name a product to enter: ")
            price = (input("Name a price for the product: ")).strip('$')
            data = Product(name, price)  # creates an instance of the Product
class, using user inputted data
        except Exception as ee:  # if the user inputted data does not conform
to the class properties, it throws an exception
            print("Product name must be letters and product price must be
numerical(decimals ok).")
        else:
            return data  # if there are no exceptions, the function returns
the string for the object instance
```

Listing 3: Class IO

The first function simply displays a menu of options. The second function, display_file_data(), reads the file and saves the returned list. It then iterates over each element of the list and prints it out. Try/Except blocks are used to catch errors for no file found, for type error, which can result if the list is empty, and for other errors, for which Exception is used as a catch-all. Running the program many times trying to find bugs helped identify the expected errors that I then use multiple except blocks to capture and handle.

I use similar code for display_table_data(), except I do it directly from the table list that I will be appending new data too. I added this function so the user can take a look at the unsaved data s/he is collecting. I print a header, then I iterate over the list and display the string method of each object. I catch the TypeError (in case the list is empty, then it cannot iterate), and a general catch-all Exception error.

My last function in this class gets user data and returns it as an object. I save the name and price (where I strip $ signs from the price) and then add them as attributes for my data instance from the Product class.

When I add them as attributes, I can potentially trigger an error if the values do not meet the conditions set up in the Product class. If the name value is all numbers, then it will fail to be added. If the price value is alphanumeric, that too will fail, as the price has to be convertible to a float. If conditions are not met, the errors from the except block in this function or the raised errors from the getter and setter functions can get triggered and display. In this function, I capture those exceptions and first print my message reminding the user that s/he must not have a numeric name and the price does need to be a float.

If no exceptions are raised, then the function returns the data object.

Main Script

The main script starts by loading the data from file. If there is no file, it prints the relevant handled error. If there is a file, then it captures it in product_list and then iterate over the list, appending each item into the lstofProductObjects.

I then start my while loop – while True, it displays the menu, asks the user for a choice, and then executes the condition corresponding with that choice.

With choice 1, I run display_data_table() function. Of course, because it is a method in the IO class, I have to first enter in the IO class, followed by a dot and then the method. I use the lstofProductObjects as the argument to be passed in, as that is what I want to have displayed.

If user chooses 2, then user adds in name and price for a product. I use the IO.get_user_data() function to capture this data. If no exceptions occur, then it will end by displaying the updated table of lists using the same display_data_table() method.

For option 3, I run my FileProcessor.save_data_to_file() function and save the data, printing “Data Saved!” if it runs without exception.

For choice 4, the user sees the data from the file, shown by calling the function IO.display_file_data().

The user may wish to clear the data from the file or from the table. With option 5, the user can clear the table (and if s/he wants to clear the saved file, s/he can simply save the empty list to file by entering 3 again).

```python
# Load data from file into a list of product objects when script starts
try:
    lstofProductObjects = FileProcessor.read_data_from_file(strFileName)
except AttributeError as e:
    print("Empty data.")
except TypeError as e:
    print("No data yet to iterate.")
except Exception as e:
    print("Error: ")
    print(e)


while True:
    # Show user a menu of options
    IO.display_menu()

    # Get user's menu option choice
    strChoice = IO.menu_choice()

    # Show user current data in the list of product objects
    if strChoice == '1':
        IO.display_table_data(lstOfProductObjects)

    # Let user add data to the list of product objects
    elif strChoice == '2':
        strData = IO.get_user_data()
        if bool(strData) == True:
            lstOfProductObjects.append(strData)
            IO.display_table_data(lstOfProductObjects)
        else:
            print("Item not added. Check inputted product name and price.")
        continue

    # let user save current data to file and exit program
    elif strChoice == '3':
        try:
            FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
            print("Data Saved!")
        except FileNotFoundError as error:
            print("File not found. Data not saved.")
            print(error)
        except Exception as error:
            print("Some error. Data not saved.")
            print(error)
        continue

    # Display data from file to user
    elif strChoice == '4':
        IO.display_file_data(strFileName)

    # Clear data from list
    elif strChoice == '5':
        lstOfProductObjects.clear()

    # Allow user to exit the program
    elif strChoice == '6':
        strExit = input("Exit without saving? 'y' or 'n': ")
        if strExit.lower() == 'y':
            print("Exiting program. Data NOT saved.")
            break
        else:
            continue

    else:
        print("Choose only 1, 2, 3, 4, or 5!")
```
Listing 4: The Main Script

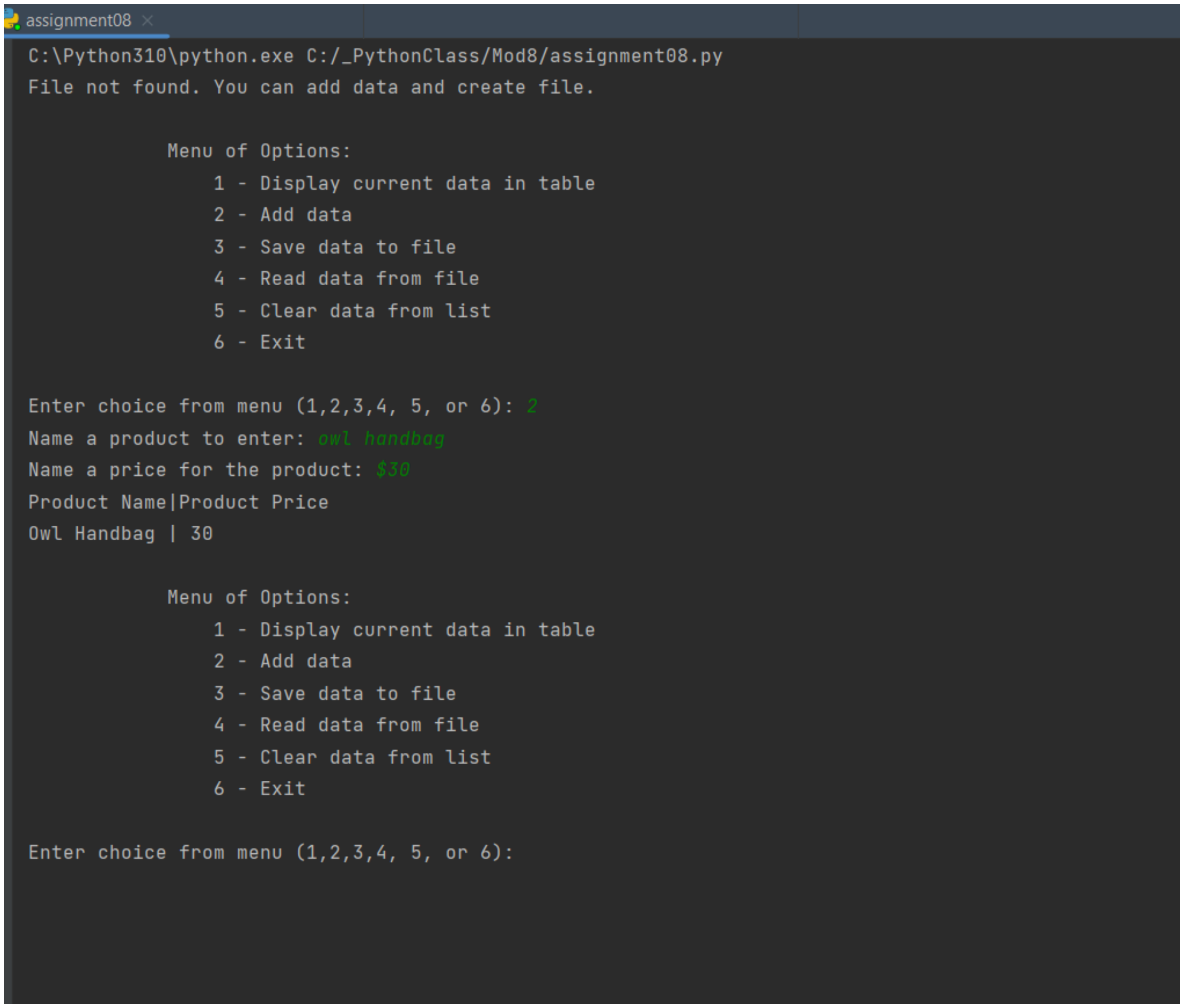
The last option, 6, allows the user to exit, after making sure that user wants to exit without saving the data. If the user continues to exit, the program breaks from the loop and ends

I have an else statement for in case the user chooses options other than 1-6.

Running the Script

Figure 1 below show the code running in PyCharm when no file exists. Since the program first tries to load the data, when no file is found to load from, it runs the error message "File not found. You can add data and create file." I then choose to add data, so I enter in owl handbag for product name and $30 for product price. The program strips the $ sign and returns the added items in a table before returning to the main menu.

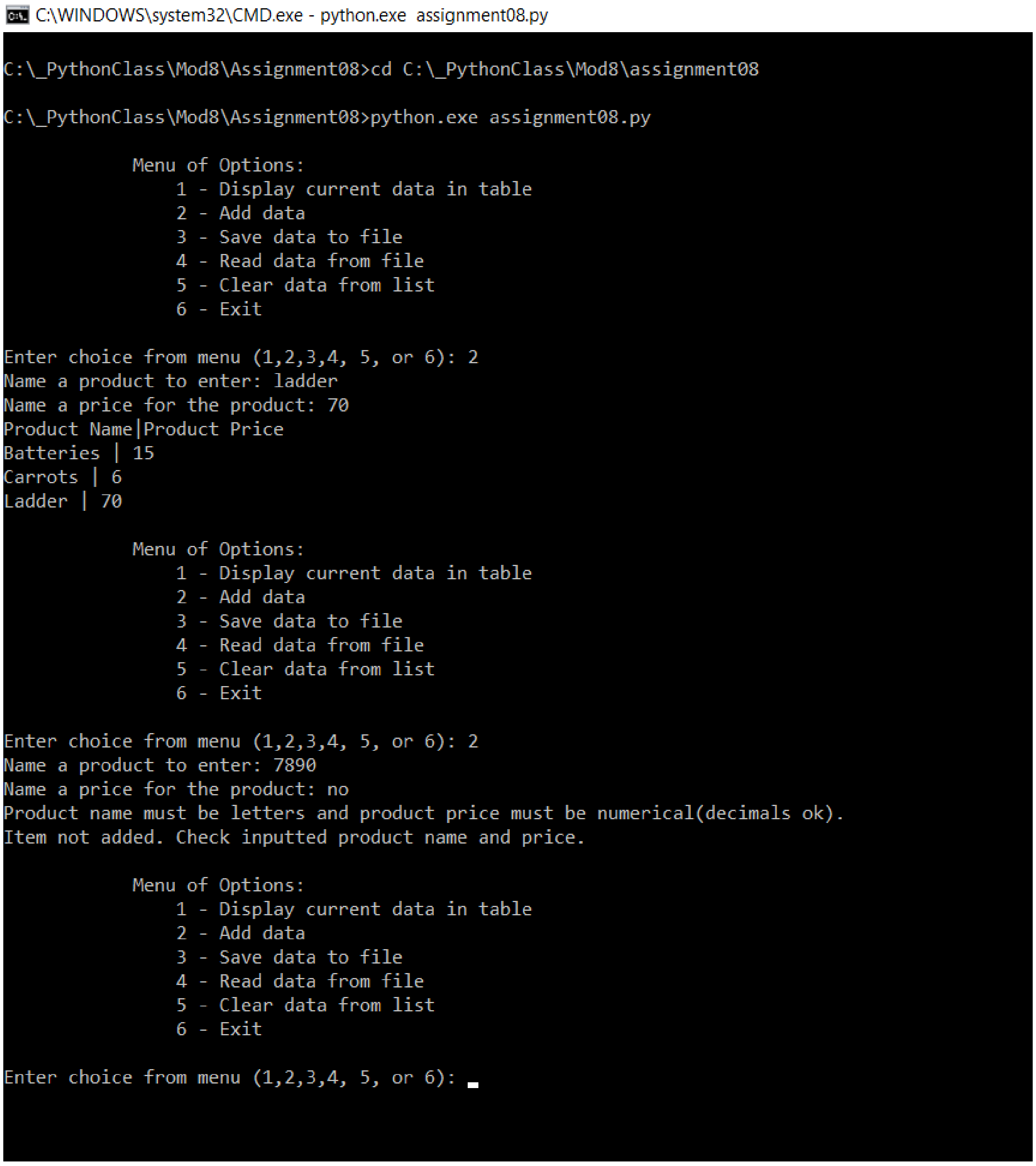
Figure 2 shows the program being run in the Windows OS console. This time, I begin with the text file existing. The figure shows user items added and appended to list. Then, user tries to add item with numbers for name and letters for price –this leads to an exception and as per exception handling in the code, it reminds user to use letters for name and numbers for price.



assignment08

```
C:\Python310\python.exe C:/_PythonClass/Mod8/assignment08.py
File not found. You can add data and create file.

            Menu of Options:
                1 - Display current data in table
                2 - Add data
                3 - Save data to file
                4 - Read data from file
                5 - Clear data from list
                6 - Exit

Enter choice from menu (1,2,3,4, 5, or 6): 2
Name a product to enter: owl handbag
Name a price for the product: $30
Product Name|Product Price
Owl Handbag | 30

            Menu of Options:
                1 - Display current data in table
                2 - Add data
                3 - Save data to file
                4 - Read data from file
                5 - Clear data from list
                6 - Exit

Enter choice from menu (1,2,3,4, 5, or 6):
```

Figure 1: Assignment 8 running in PyCharm



```
C:\WINDOWS\system32\CMD.exe - python.exe  assignment08.py

C:\_PythonClass\Mod8\Assignment08>cd C:\_PythonClass\Mod8\assignment08

C:\_PythonClass\Mod8\Assignment08>python.exe assignment08.py

            Menu of Options:
                1 - Display current data in table
                2 - Add data
                3 - Save data to file
                4 - Read data from file
                5 - Clear data from list
                6 - Exit

Enter choice from menu (1,2,3,4, 5, or 6): 2
Name a product to enter: ladder
Name a price for the product: 70
Product Name|Product Price
Batteries | 15
Carrots | 6
Ladder | 70

            Menu of Options:
                1 - Display current data in table
                2 - Add data
                3 - Save data to file
                4 - Read data from file
                5 - Clear data from list
                6 - Exit

Enter choice from menu (1,2,3,4, 5, or 6): 2
Name a product to enter: 7890
Name a price for the product: no
Product name must be letters and product price must be numerical(decimals ok).
Item not added. Check inputted product name and price.

            Menu of Options:
                1 - Display current data in table
                2 - Add data
                3 - Save data to file
                4 - Read data from file
                5 - Clear data from list
                6 - Exit

Enter choice from menu (1,2,3,4, 5, or 6):
```

Figure 2: Assignment 08 running in Windows OS Console

Summary

In this assignment, we learned to make objects from a class by creating a “Product” class. We created a constructor function, which set the attributes for the class and then we created getter and setter functions to manage the attributes for the class. The __*str*__ function is overwritten with a function that returns the formatted attributes for the Product class object instances. We created a class for processing files, using static functions for saving and reading files. A third class, also with static functions, we used for functions made for inputting and outputting data, object or otherwise. The main script is then used to load the data and run a loop to give users options and execute the user’s choices. Throughout the code, we use try/except blocks to handle expected errors.