Nick Hertlein

March 11, 2021

IT FDN 110 B: Foundations Of Programming: Python

Assignment08

GitHub URL: https://github.com/nhertlein/IntroToProg-Python-Mod08

Objects & Classes

# Introduction

This document will illustrate the knowledge gained from the eighth lecture and the media portion of the assignment. In addition, I will cover the modification of an existing Python script that uses classes and objects to create a list of products and prices. This assignment was pretty tricky for me as the classes and methods are pretty new to me. Using the “setter” and “getter” decorators were a new concept that took me some experimenting to figure out. Overall this was a very helpful module as it explained how to use classes in a way I have seen in other programs but did not understand.

# Writing the Script

For this week’s assignment I made a project in PyCharm in the Assignment08 folder of the C:\\_PythonClass directory. As we were starting with a “starter” file for this assignment the first step included updating the header change log. Additionally, I added another default variable strChoice to capture the user’s choice from the menu similar to previous programs.

The next section of the script was defining the class Product. This class was used to define an object product that had two properties associated with it: product\_name, and product\_price as defined in the starter script. The first step in populating the class was to make an initial constructor to automatically populate the properties associated with the product object on initialization. I made the properties private by using a double underscore before the name of the attributes. This was not really necessary but it was a good excuse to use this feature and see how it works. I also specified the type for each attribute in the object so the intention is clearer for someone in the future to understand the script. Inputs were also converted to the correct type (string or float) just to be sure they are in the correct format (Figure 1).

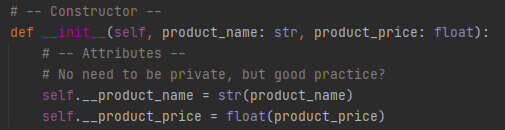


Figure 1. Initial constructor for class Product

Defining the properties of the class is straight forward as the properties are only providing access to the private attributes. The attribute product\_name was converted to a title text as well to keep consistency between the objects entered (Figure 2).

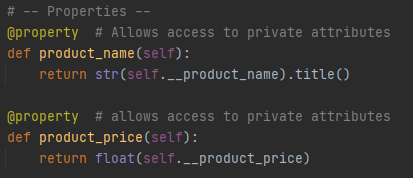


Figure 2. Defining properties of class Product

In order to change the class properties we need to have a “Setter” for the attributes which is covered in the last section of code for the class (Figure 3).

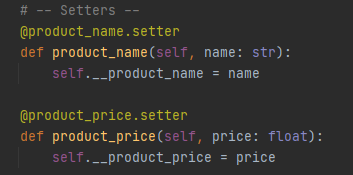


Figure 3. Defining setters of class Product

Now that the Product class is defined so we can create product objects we can move to the FileProcessor class used to get data from a file and save data to a file. The FileProcessor class contains @staticmethod’s so the functions can be called without instantiating the class first. The first method of this class is saving data to a file which is a function we have used in the past and I mimicked what we have done here previously. One thing I did differently is use the “with” statement which handles closing the file when the operation is complete (Figure 4).

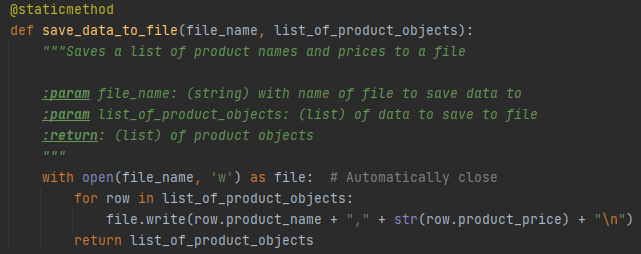


Figure 4. Save data to file method

The next method in the FileProcessor class is for reading data from a file. For this method I used the “with” statement again for the same reasons as above. The part that was different with the reading data from file method compared to what we have done previously is we were putting the information into an object and appending that to the list instead of making a dictionary row or similar. I initially tried making this into a dictionary row like we had done previously which did not work out so well. Saving the input as an object makes a lot of sense, but it took me a little bit to get the format right and understand what I was doing wrong (Figure 5).

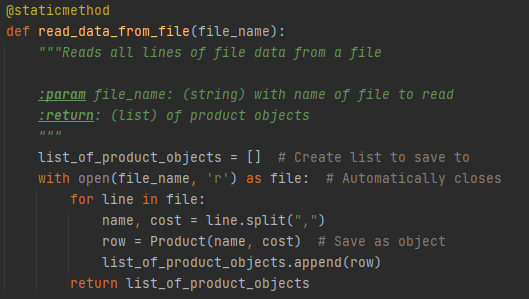


Figure 5. Read data from file method

The next class we had to populate was the IO class which was similar to ones we have used in the past so I will only cover the differences. For the static method in the IO class to add product data I used a while loop for each input to keep asking the user for an entry until an entry that meets the criteria is entered. The product\_name is only entered if the value is not numeric, otherwise an exception is raised and the user is notified with an error and an opportunity to try the entry again. I used the condition for the input to not be numeric because I wanted to allow alphanumeric entries which .isalpha() did not allow. Of course after making this I realized there is an option to use .isalnum() which would have worked too, but I suppose the method I used allowed me to research using the “not” statement. The product\_price input will set a ValueError if something that cannot be converted to a float is entered. The user will be notified if a ValueError is raised and they will get another opportunity to enter a correct value until the conditions are met. After both inputs are received the inputs are assigned to a Product object and returned as a new row to be added to the product list (Figure 6). Having the error checking at the input also helps ensure that the correct value types will be passed to the other classes/methods so each step does not need to have error handling.

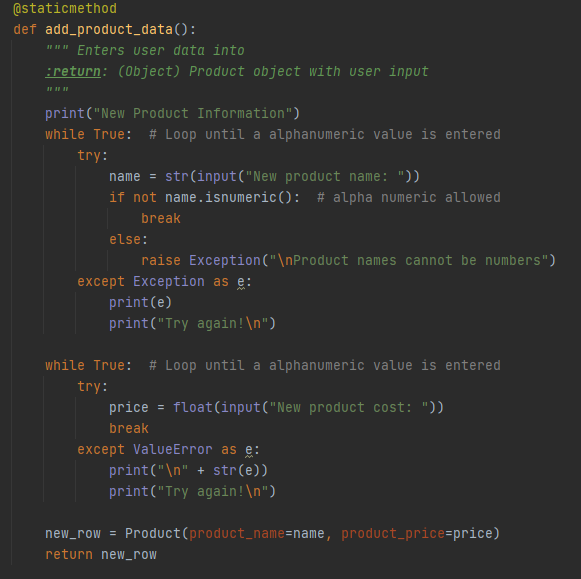


Figure 6. Add product data method

The final portion of the script is the main body portion which is used to call the other functions and execute them. The first step in the main body portion of the script was to read in data from the file. As the starter for this assignment did not have a file to work from I had to get the program working to know what format things would be saved in to read it (this is a good thing as it forced me to figure it out). Right off the bat I realized as I did not have a demo file to read in, I needed to have a way to determine that and handle the issue without crashing the script. For this I made a try except block to notify the user if a file was not found, or the file found was unable to load (Figure 7).

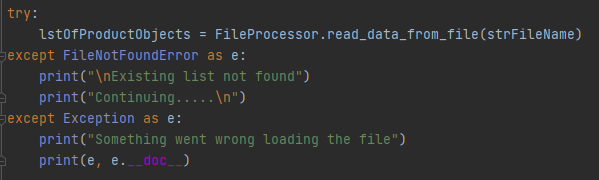


Figure 7. Try except block for loading in a file

After the file is loaded in (or not, if not available) the program moves to a while loop which is where all the user selections and actions take place until the user chooses to exit the program. With each loop through the program the user is displayed the menu and the users choice is returned through an IO method similar to previous programs. If the user selects option 1, a list of the current products are displayed through an IO method if the list is not empty. It did not seem helpful to display an empty list to me. If the user selects option 2, another IO method is called which allows the user to enter the product attributes as covered above. The product object output from the method is then added to the current running list of product objects. If the user selects option 3, the file is saved using a FileProcessor method covered above. If the user selects option 4, the user is notified the program is exiting and the while loop is broken.

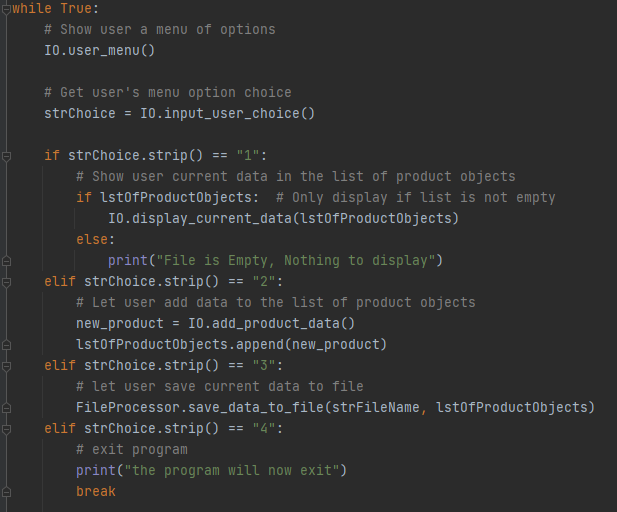


Figure 8.

# Summary

This module was intended to be an expansion of previous assignments with the addition of using a class to export an object. The class for creating the object has a constructor which was executed every time an object was created and also allowed access to its attributes and the ability to modify them. I found this module to be fairly difficult as this was a new concept to me, but it helped me understand classes and decorators, etc. a lot more. I was also able to apply this knowledge to one of my MATLAB scripts at work for a GUI which is built off a class. Knowing what the heck some of the @... meant and what the difference between the public and private variables meant allowed me to make some modifications to the functionality to better suit my needs by creating a method to execute a function and allow different variants of the function to be called which helped clean up the code by not having to repeat similar blocks with slight tweaks for various user options selected. It’s always nice to put what you learn to use ☺