Jennifer D. Smith

March 15, 2020

Foundations of Programming: Python

Assignment 08

GitHub: <https://github.com/greeneyedsoandso/IntroToProg-Python-Mod08>

GitHub Page: <https://greeneyedsoandso.github.io/IntroToProg-Python-Mod08/>

Using Objects with Classes

# Introduction

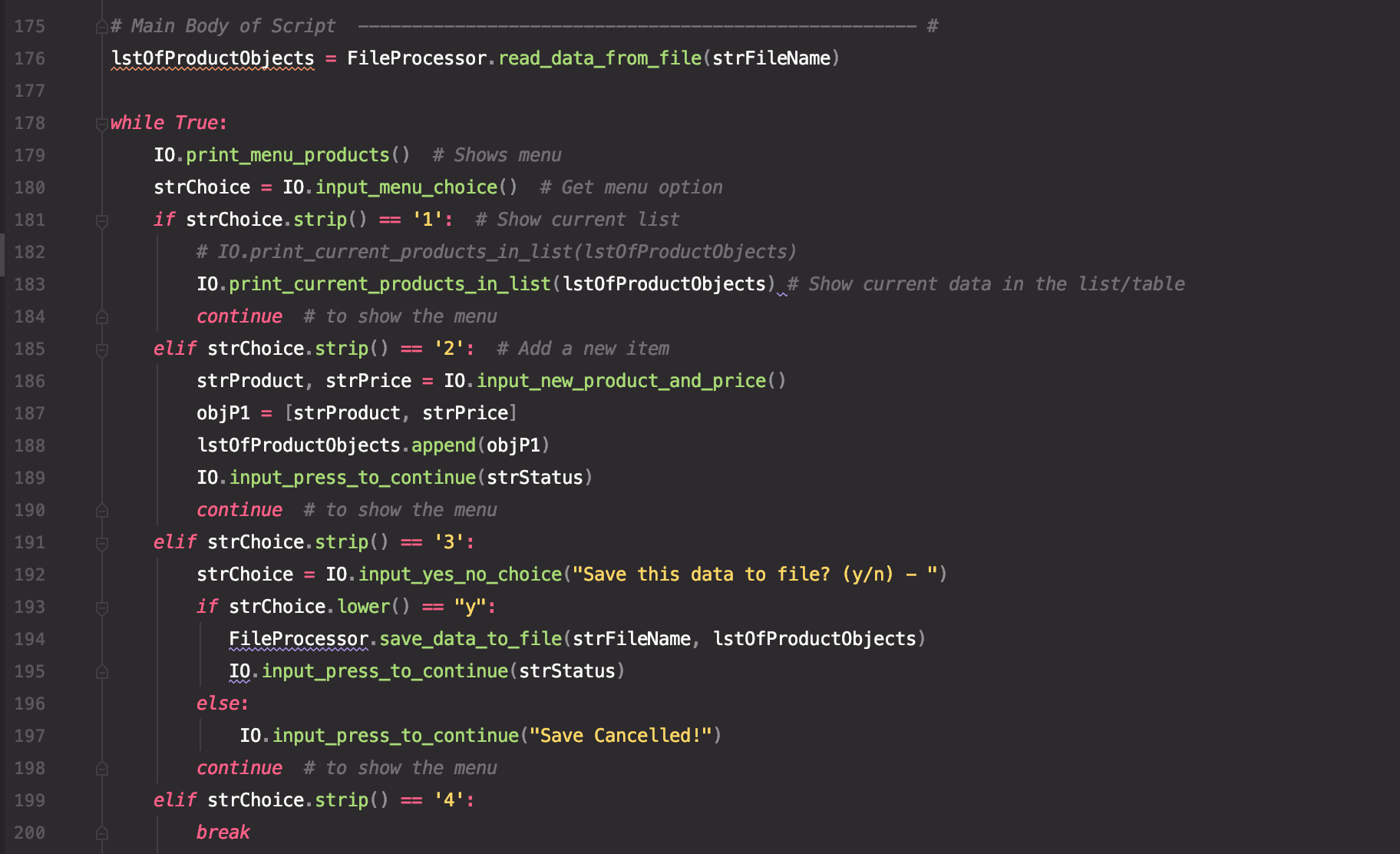
In this module, we built on what we learned previously about classes and started to create objects from classes. At first objects seem like a lot of lines of code to do things we already know. But they are a powerful way to separate concerns and make future changes less painful.

### False starts

This was a very difficult concept for me. I could see the various pieces but really wasn’t understanding how they fit together, and more importantly why. I restarted this project three or four times and had trouble even after watching the module review video. Ultimately, the way I was able to move forward was to start out by ignoring the objects part of the rubric, and then refactor to work the objects in.

### Refactoring for objects

First, using the starter file, I got the desired functionality working without the objects. These were all tasks we had done before––read file, add items, review items, write file, a simple menu. I stored the product and price in a variable (Figure 1).

 Figure 1 Using functions and classes, but not Class objects

For example, in the FileProcessor class, I was reading out lists from the file (Figure 2).

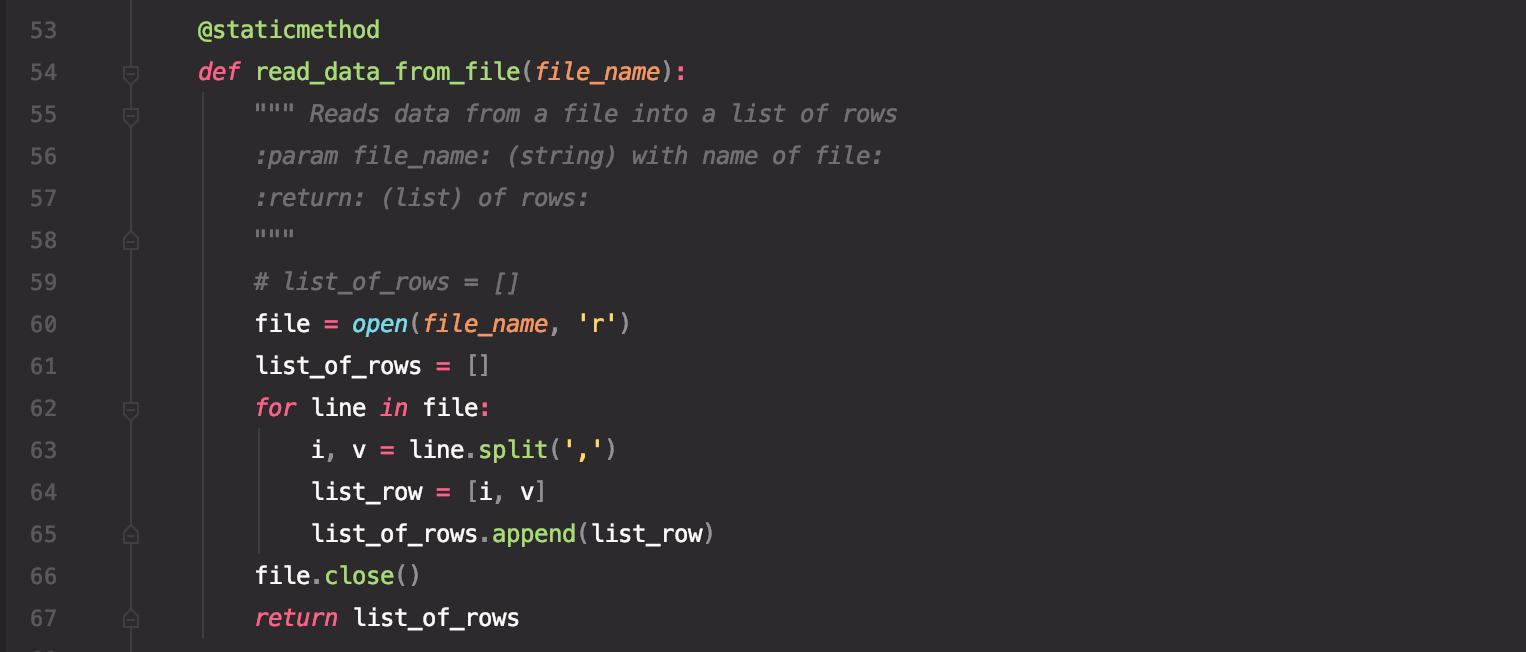


Figure 2 Making list rows

This was the first place I wanted to start using the Product class, so I filled in the \_\_init\_\_ and \_\_str\_\_ methods (Figure 3). Now I could start using objects to store and manipulate data. 

Figure 3 Setting up methods

In the new version of FileProcessor.read\_data\_from\_file, I could eliminate the list\_row variable and replace it with a Product object (Figure 4).

 Figure 4 Adding the Product class into the function

The next area to simplify was IO.print\_current\_products\_in\_list (Figure 5).

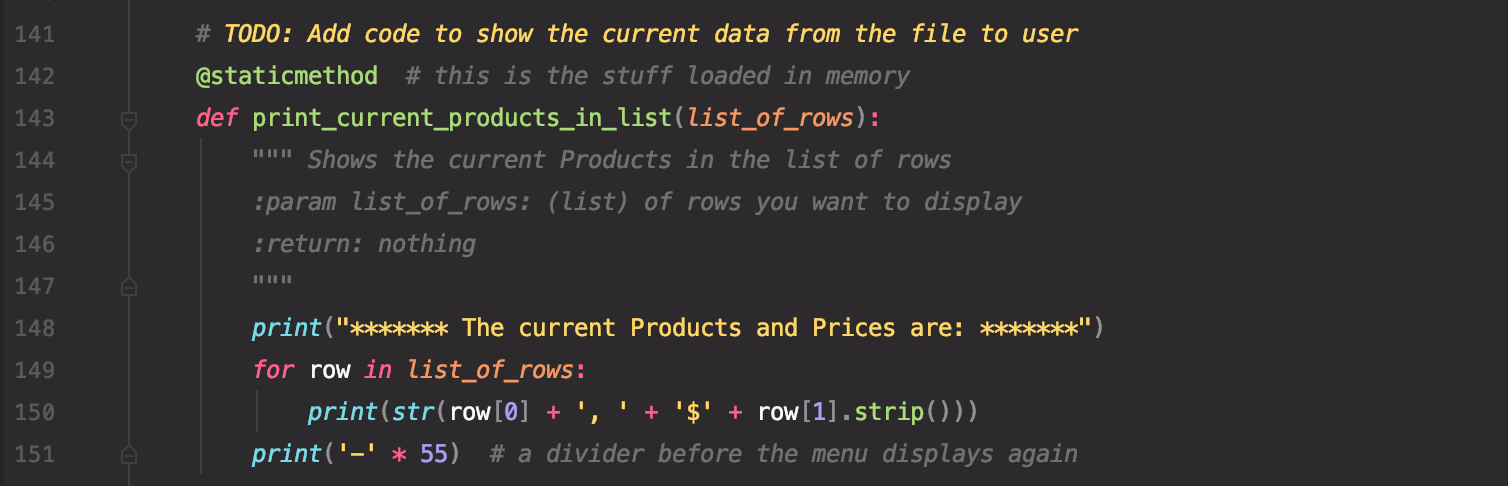


Figure 5 A very detailed string for presentation

Now that my rows were Product objects, the print statement could be very simple (Figure 6). (More on that later.)

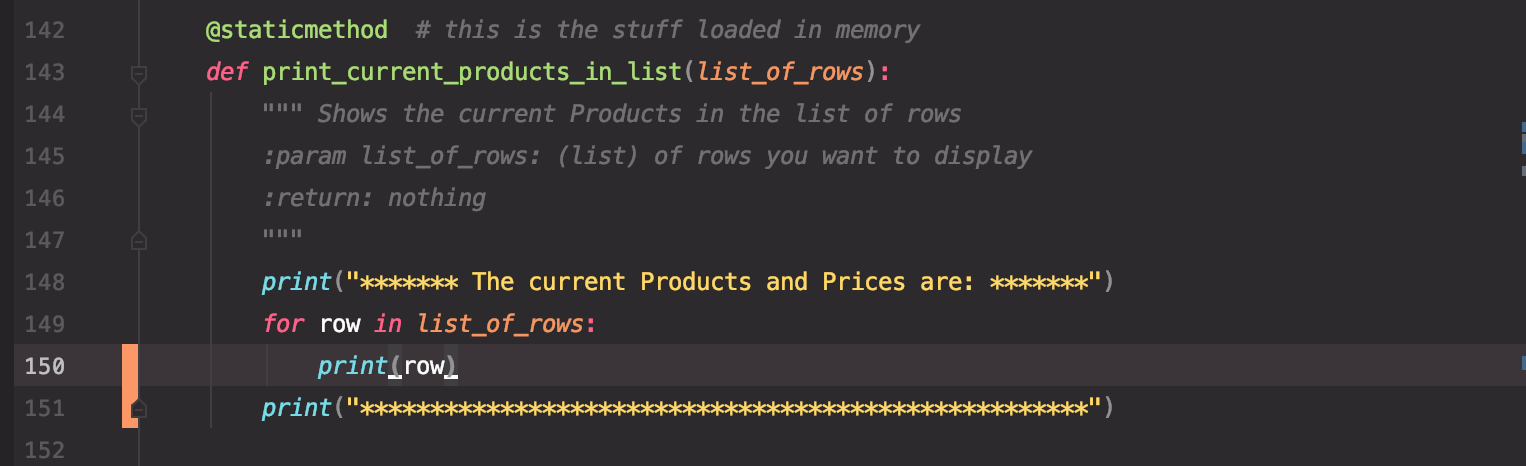


Figure 6 Just the rows

IO.input\_new\_product\_and\_price got simpler as well (Figure 7).



Figure 7 Now with fewer variables

And that made menu option 2 half as long (Figure 8). (I was getting an error at this stage, which I later fixed when I added exception handling.)



Figure 8 Simplification thanks to objects

After all that simplification, menu option 1 to review data didn’t look as good as it used to. I lost all my spacing and formatting with the dollar sign. And I definitely didn’t want that formatting cruft back in my nice csv data file. So, I decided to try adding a method to make things pretty for display (Figure 9).



Figure 9 The pretty printing method

While I’m not sure that’s a perfect separation of concerns, I’m pleased with how it satisfied my desire for human-readable text and machine-readable data (Figure 10).

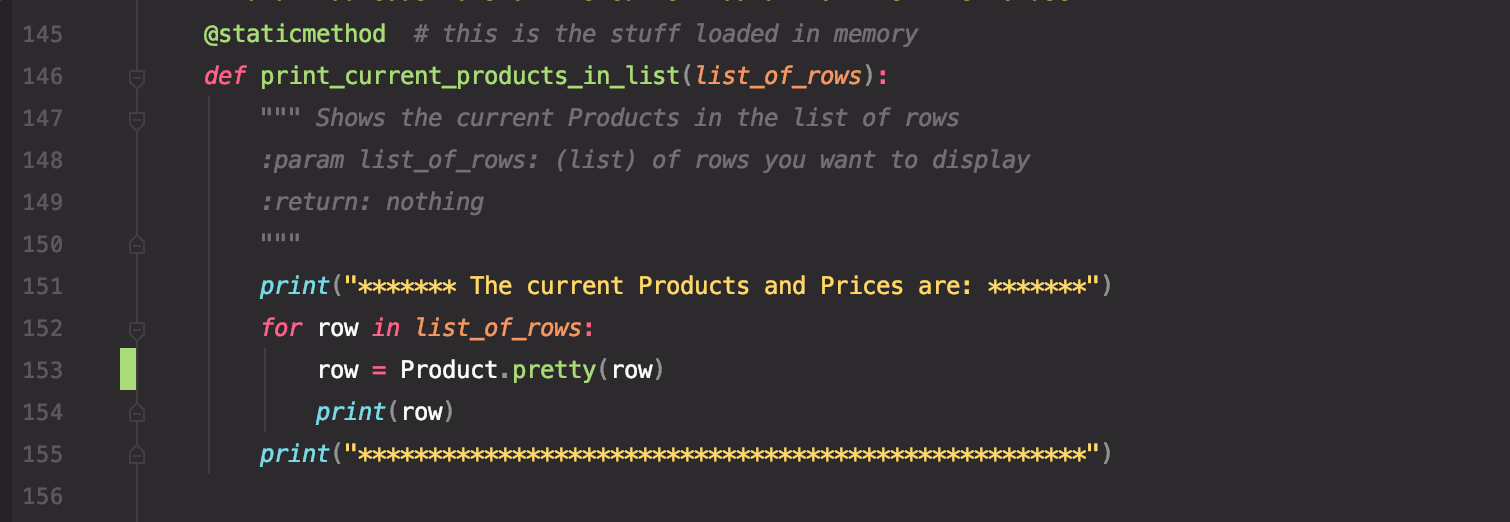


Figure 10 Printing rows, the pretty way

Once everything seemed to be working, I reviewed the getter/setter properties––I had created them in early attempts but because I wasn’t using the class successfully, I wasn’t able to get the exceptions to fire. With the class working properly, it was time to set up some validation (Figure 11).



Figure 11 Getter/setters in the Product class

In the process, I also realized that there was no reason to have two separate variables storing IO.input\_new\_product\_and\_price, and that’s why I was error-ing out when I tried to enter more than two new items (Figure 12).

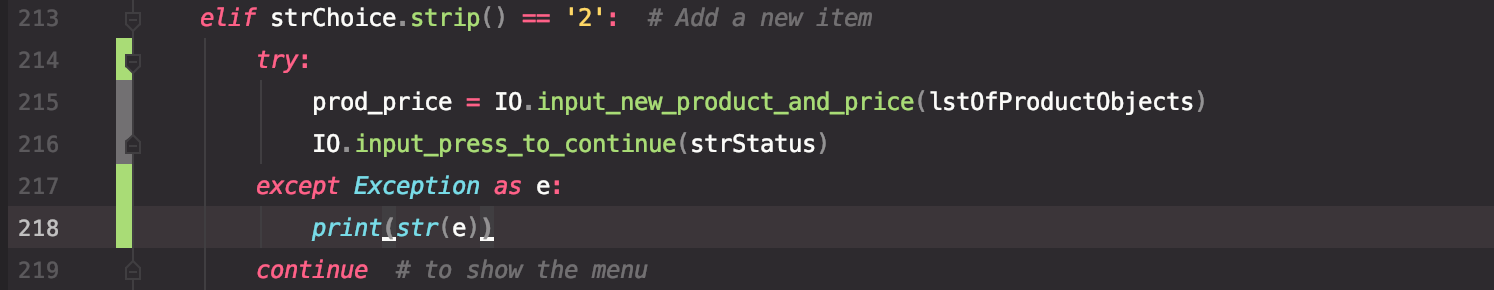


Figure 12 One variable to rule them all

The final version of the main body of the script was much more concise (Figure 13).

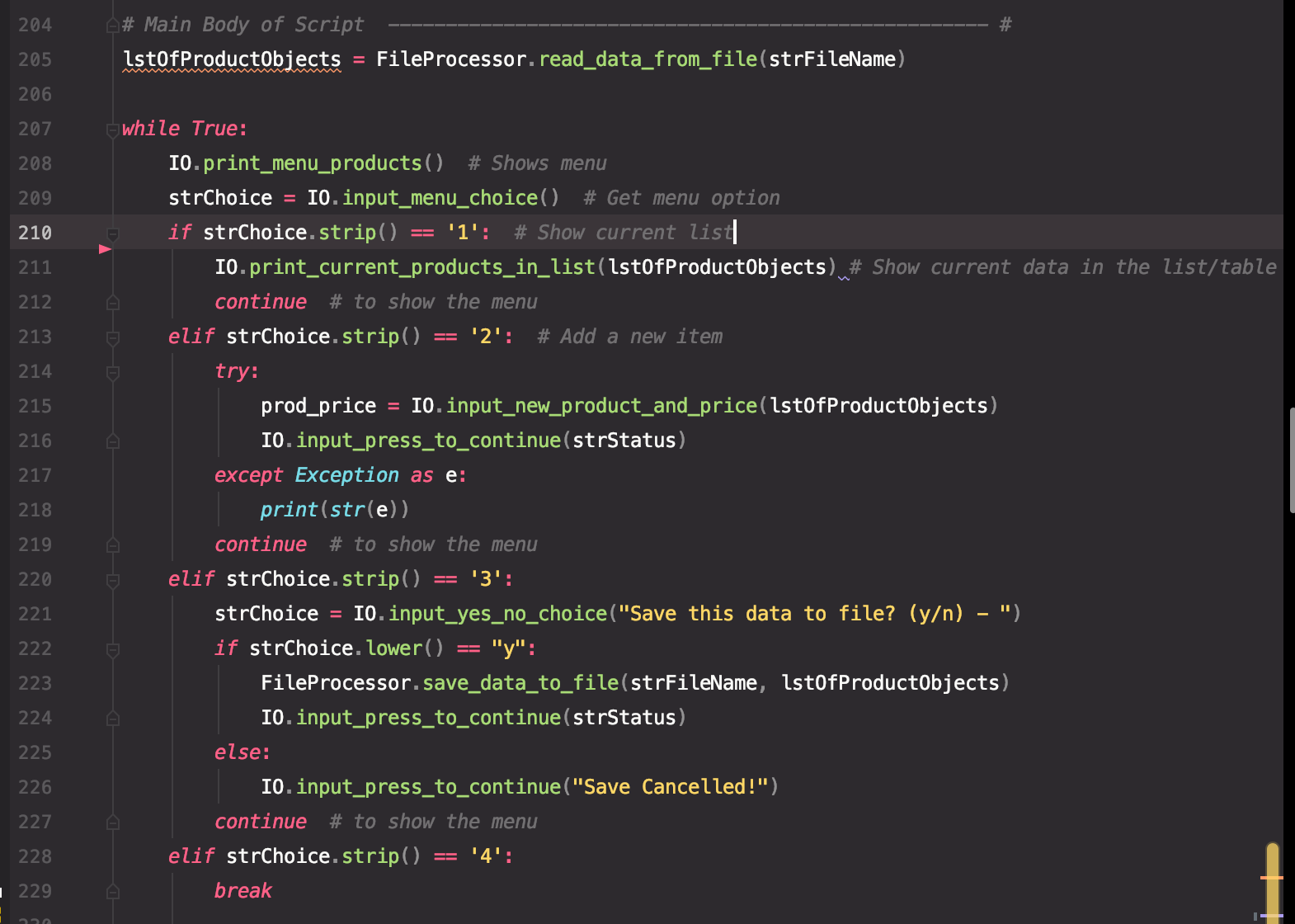
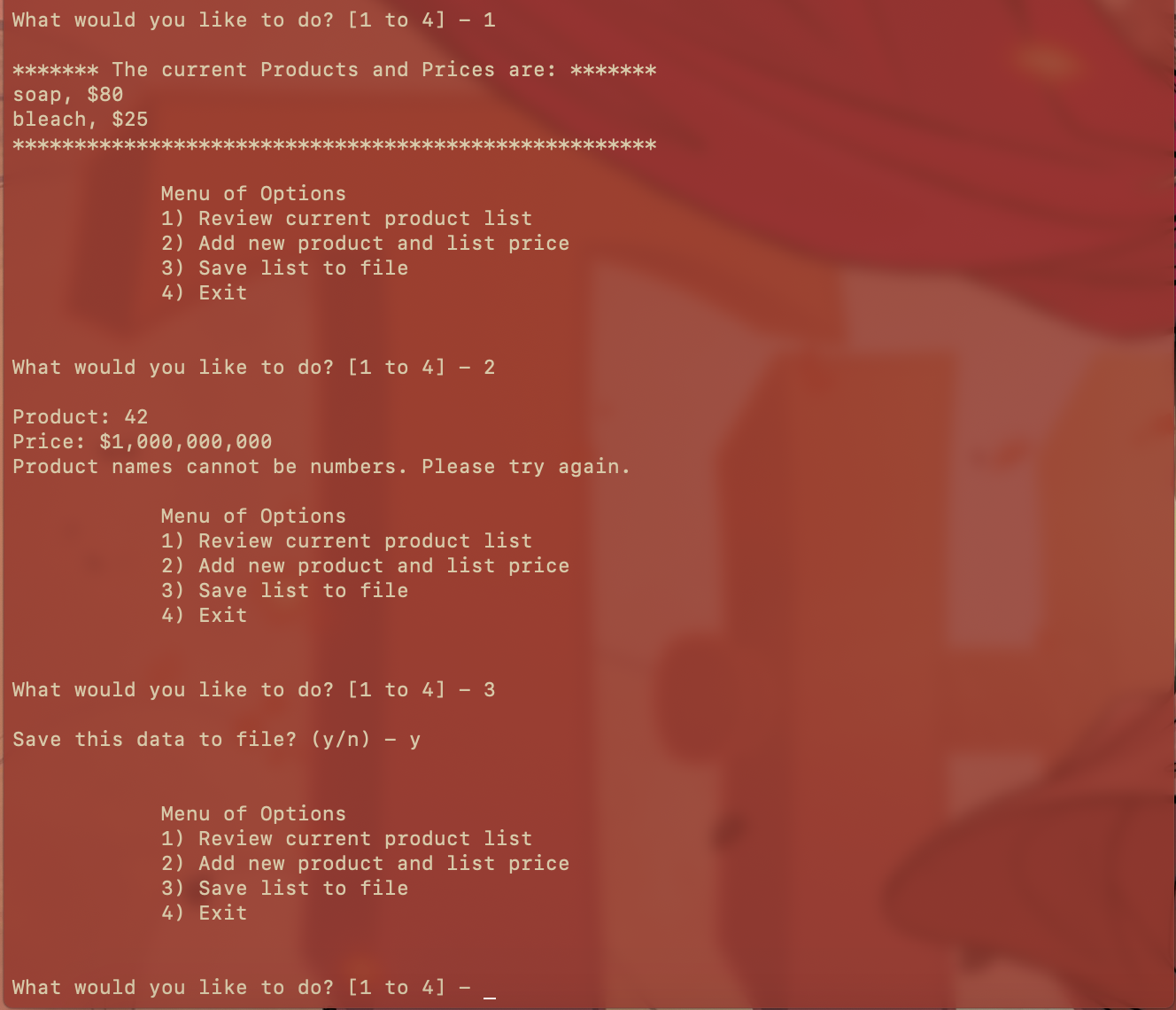


Figure 13 The cleaner new main body

I made a few cosmetic tweaks, and everything worked as expected (Figures 14-16).



Figure 14 Running in Terminal

 Figure 15 Running in Terminal, continued

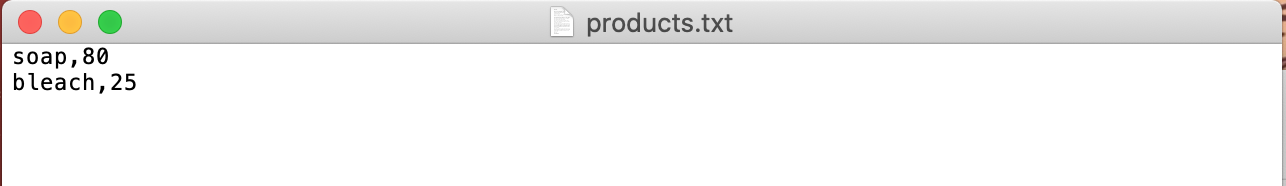


Figure 16 The text file containing only good data, in csv form

# Conclusion

I started working on this script with a superficial understanding of using class objects. It turns out that during my first several tries, I had created the class well, I just didn’t get how to actually use it. By the time I figured out how to use it to transform data presentation, it really started to make sense and I could start to see how it streamlined the main body of the script and would make global changes easier.