Kelsey Kawaguchi

Dec. 2, 2019

IT FDN 100

Assignment 08

<GITHUB URL>

Recording Products – Working with Classes

Introduction

This paper explains the logic behind how I developed the code on how to work with classes. The goal and purpose of the assignment was to use classes to collect data: product name and product price.

Methodology

Figure 1 displays initial “working with classes” code. The first portion of the code identifies variables.

The next section of the code begins with identifying the class Product. Within the class, the first thing that is identified are the constructors or functions. In this case, the function \_\_init\_\_ lists the initial values of the field data. The initial values set are the product\_name and product\_price and the value type is also identified as a string or float.

The next section lists the attributes, which identifies that the product\_name and product\_price will be a string and float, respectively. There is an error handling message included to inform the user that if the values input are not a string and float, then the user must input those values as such.

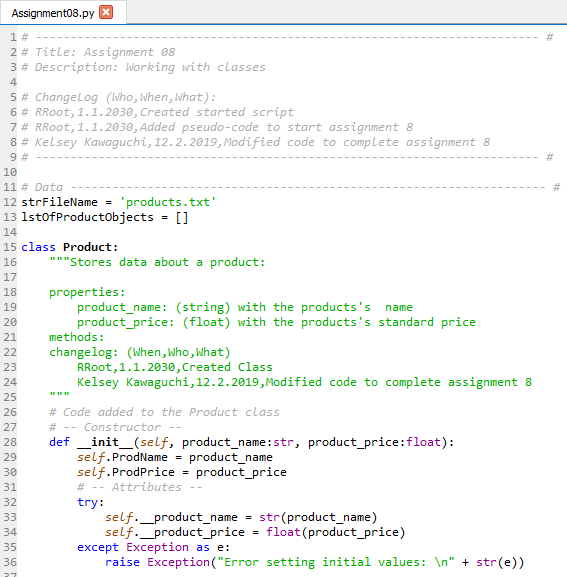


Figure 1 – class Product part 1

The next part of the code as shown in Figure 2 is a continuation of the data portion of the code. The next section identifies the properties of the product name and product price, by identifying “getters” and “setters”. The getters are input to get and format the data per the coder’s preference, and the setters are input to set the data. The setter code is for validation and error handling.

In Figure 2, the getters identify that the product name and the product price will be a string and float, respectively. The purpose of the setter is to validate that the value read in the program is a number or is not a number. If the value is not as expected, an error is raised to the user.

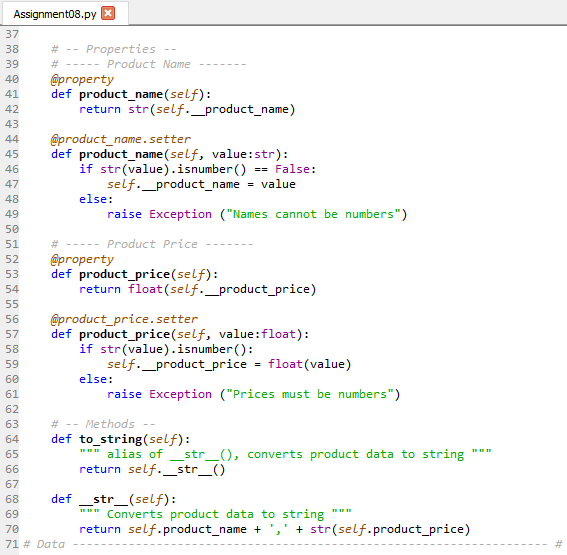


Figure 2 – class Product part 2

Under the methods portion in Figure 2 of the code, the to\_string method returns a string value. The \_\_str\_\_ then returns the product name and product price per the format shown above.

The next section of the code as shown in Figure 3 includes the processing FileProcessor class. This class processes data to and from the file.

The first function created is to save the data to a file. In the first line, the data input is identified along with the data type. The success\_status as a default is set to false. The code will begin by opening the file in write mode. In the next line, for a product in the list of product objects, the new list of rows will be written to the file. Next, the file will close. If done correctly, the success status will be true.

The except line displays some error handling if an error occurs.

The final expectation, if everything is run successfully, is for the function to return with a success status.

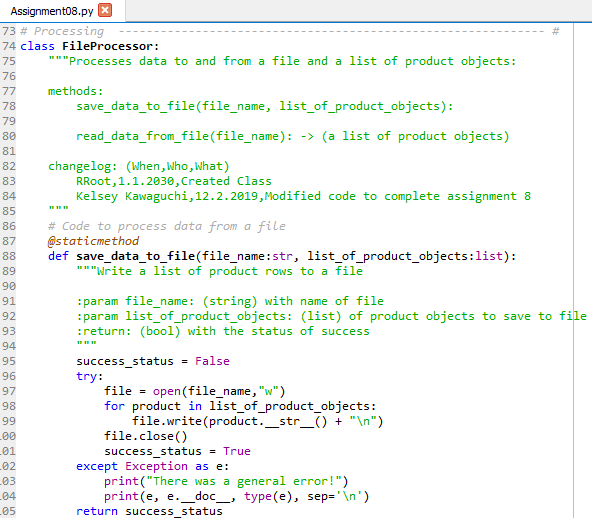


Figure 3 – Processing Part 1

Figure 4 is a continuation of the processing code. The next function in the processing phase is to read the data from the file. As identified in line 109, the file name should be a string. Line 115 then identifies that the list of product rows will be a list. From line 116 through line 122, the code calls for the file to be read, and for each line in the file, data will be identified as separate based on the location of the commas. For each row, the Product class will take the data[0] (product name) and data[1] (product price) as input. Line 121 of the code will then append the list of product rows in the file based on the row, then the file will close.

Lines 123 through 125 include error handling.

If successful, then the read data from file will return with a list of product rows.

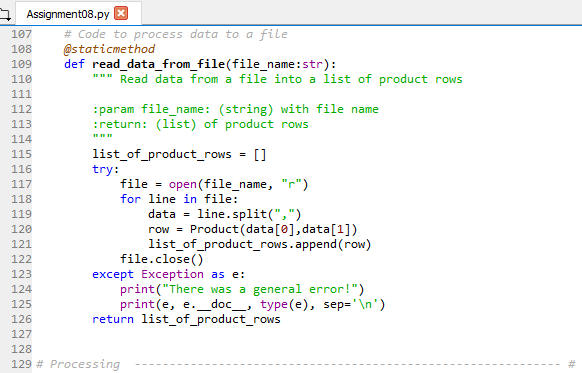


Figure 4 – Processing Part 2

Figure 5 is the Presentation (input/output) section of the code and what the user will be interfacing with. The class is called IO. The first function of the class prints the options menu to the user.

The next function is the input menu choice option, which prompts the user to select one of the menu options.

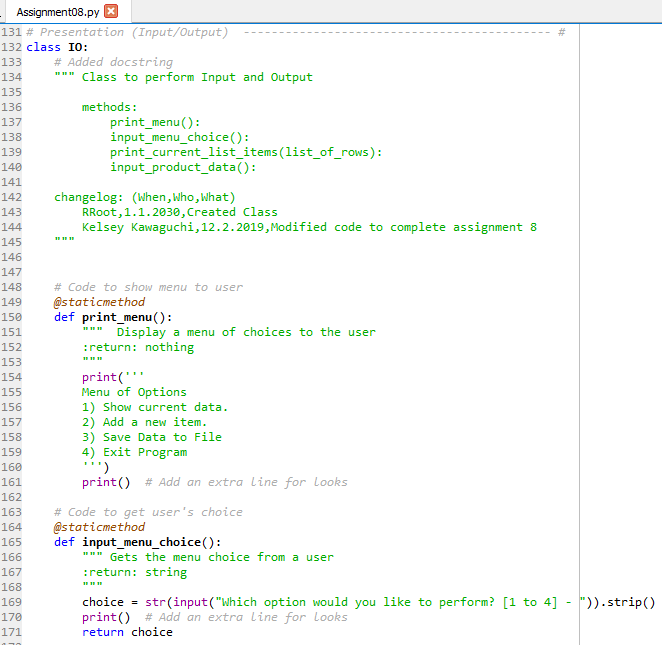


Figure 5 – Presentation (input/output) Part 1

Figure 6 is a continuation of the presentation section of the code. The next function calls the rows in the list of rows to be printed out to the user in the command window. The final function is used to input product data by prompting the user to input a product name and product price, which both are striped of spaces. The variable p is then identified by calling the Product class and inputting the product name and the product price as variables. Line 197 and 198 are for error handling, and line 199 is to return the product row.

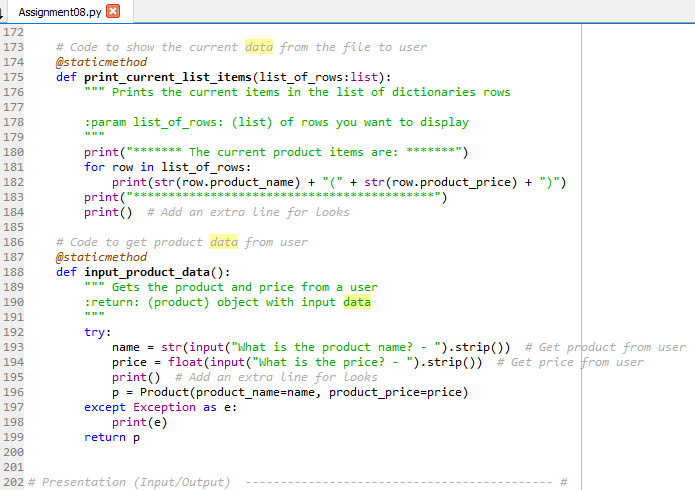


Figure 6 – Presentation (input/output) Part 2

The last section of the code is the Main Body of the Script, which calls on the classes and functions created in the previous code. The first part of the code identifies the variable ListOfProdObj to call on the FileProcessor class and read the data from the strFileName.

The next part includes a while loop that calls on the Input/Output menu. The main part of the code then uses the input values input by the user in the class IO section of the code, and if the user inputs 1, then the current list of items in the ListOfProdObjs is printed. If 2 is selected by the user, then the ListOfProdObj variable will be appended with the input product data function that lives in the class IO. If the user selects 3, then the FileProcessor class save\_data\_to\_file function is called, and the file name and list of product objects are used as inputs. Lastly, if 4 is selected, then the program will break.

Lines 229 through 231 are created for error handling.

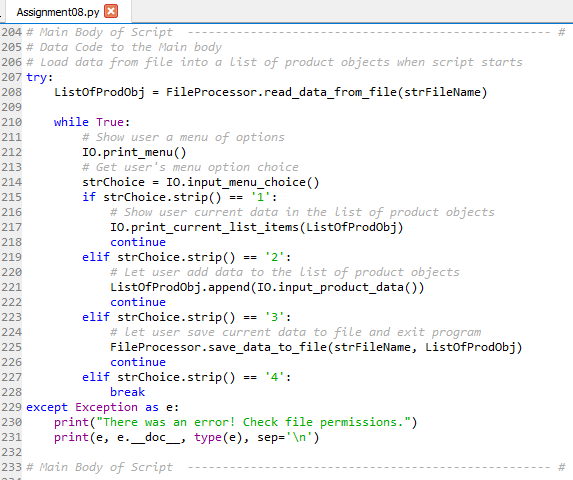


Figure 7 – Main Body of Script

Command Results

Figure 8 shows the working code in the command window view. As seen, the code successfully records the data input by the user.

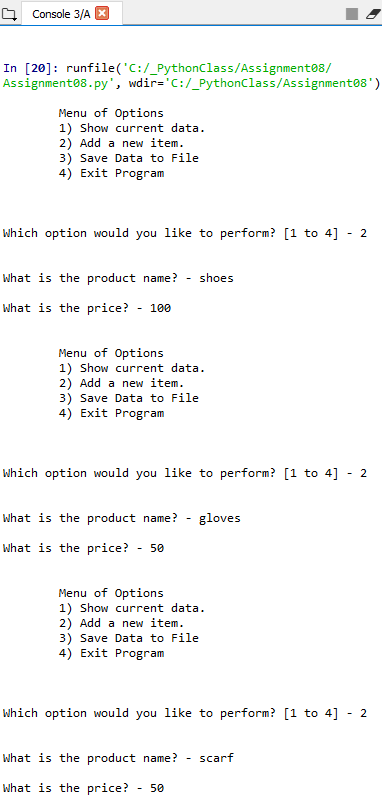


Figure 8 – Data Input

Figure 9 shows that the data input by the user was successfully stored in the current product list. As shown below, the user is able to save the data to the file. Lastly, the user is also able to successfully exit the program as 4 is input.

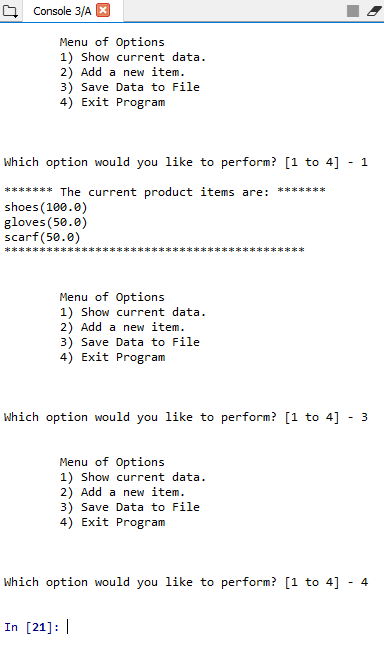


Figure 9 – Output and Save data then Exit Program

Figure 10 is a screen shot of the assignment 8 folder structure to prove that the products text file was created in the same folder as the code.

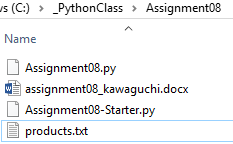


Figure 10 – products.txt file proof of location

Figure 11 shows the data input by the user is successfully saved to the products.txt file.

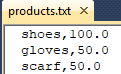


Figure 11 – Recorded Product Data

Summary

In Conclusions, I was also able to successfully collect and save product names and prices, while implementing the use of classes in a program.