Lyndsey Holmes

September 7, 2020

Intro to Programming (Python)

Assignment 08

List of Product Objects

# Introduction

Objects are a collection of data and functions that act on the data. An object has properties and methods. The methods are static functions. For this assignment, instead of using a list or dictionaries, a list of product objects is required. The user must be provided a menu of options: including display data, add products, save data and exit the program. This code allows a user to pull data from a file and save data from a file. It first requires reading the data from a file and saving it to a list of product objects.

## Starter Script

Initially provided was a starter script to include main functions of the procedure. The purpose was to open a file, read in the data into a list of product objects, allow a user to modify the list, and finally save the new information back to the file and exit the program. The starter script provided pseudocode as shown in Figure 1: Pseudocode. It provided details including global constants, the class: Product, the class: FileProcessor and the class: IO. Additionally, there are some details regarding the static methods to be

# ------------------------------------------------------------------------ #

# Title: Assignment 08

# Description: Working with classes

# ChangeLog (Who, When, What):

# RRoot,1.1.2030, Created started script

# RRoot,1.1.2030, Added pseudo-code to start assignment 8

# <Your Name>, <Today's Date>, Modified code to complete assignment 8

# ------------------------------------------------------------------------ #

# Data -------------------------------------------------------------------- #

strFileName = 'products.txt'

lstOfProductObjects = []

class Product:

    """Stores data about a product:

    properties:

        product\_name: (string) with the products' name

        product\_price: (float) with the products' standard price

    methods:

    changelog: (When,Who,What)

        RRoot,1.1.2030, Created Class

        <Your Name>, <Today's Date>, Modified code to complete assignment 8

    """

    pass

    # TODO: Add Code to the Product class

# Data -------------------------------------------------------------------- #

# 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

        <Your Name>, <Today's Date>, Modified code to complete assignment 8

    """

    pass

    # TODO: Add Code to process data from a file

    # TODO: Add Code to process data to a file

# Processing  ------------------------------------------------------------- #

# Presentation (Input/Output)  -------------------------------------------- #

class IO:

    # TODO: Add docstring

    pass

    # TODO: Add code to show menu to user

    # TODO: Add code to get user's choice

    # TODO: Add code to show the current data from the file to user

    # TODO: Add code to get product data from user

# Presentation (Input/Output)  -------------------------------------------- #

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

# TODO: Add Data Code to the Main body

# Load data from file into a list of product objects when script starts

# Show user a menu of options

# Get user's menu option choice

    # Show user current data in the list of product objects

    # Let user add data to the list of product objects

    # let user save current data to file and exit program

Figure 1: Psuedo-code

used under each class. The final step involves details regarding the purpose of the main body of the script. The purpose of the main script is to piece together the static methods. Upon running the script, the first two actions involve reading the file in to a list of objects and displaying a menu to the user. Then you must wait for the user’s response regarding which menu item will be selected. After the user provides input regarding which menu item is required, the main function has a conditional statement to determine which action to take based on the user’s input.

## Product Object

The first step to develop this script is to create the product object. The product object would be called by many other of the class methods. Two variables are product\_name and product\_price and are initialized and set to ‘None.’ The next step required is to define the product objects to the variables set to ‘None.’ As you can see in Figure 2: List of Objects, Part 1, Line 31 shows the development of the product objects.

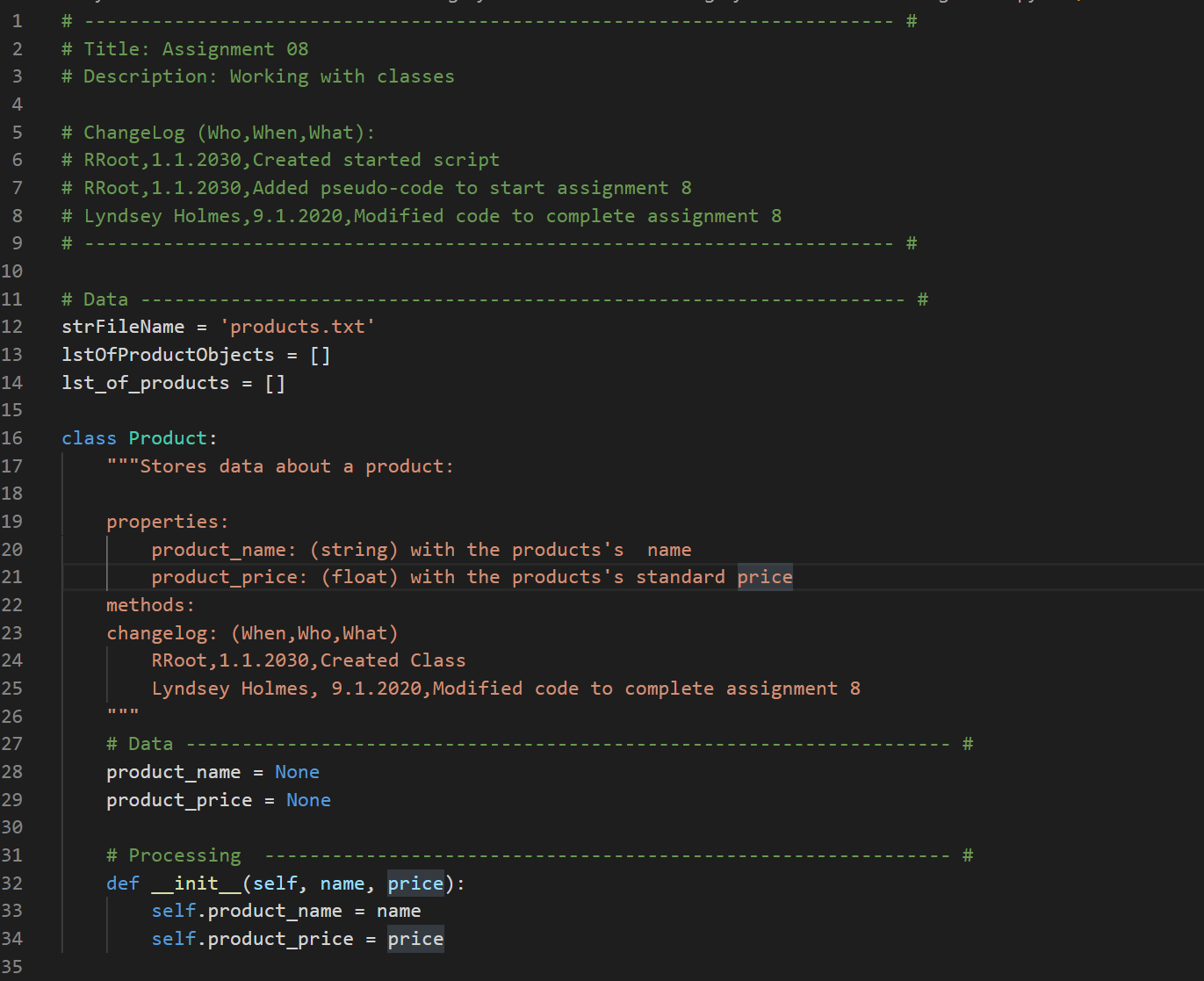


Figure 2: List Of Objects, Part 1

## File Processor

As shown in Figures 2 and 3: List of Objects, Part 2 and 3, the file processing class involves reading the data from a file and saving the data to a file. While reading the data from a file, the data is saved in to a product object in the static method “read\_data\_from\_file().” Using a for loop, the data is split by the comma delimiter and saved to name and price. Then the product objects are created and appended to the list of product objects. In line 74 of Figure 4: List of Objects, Part 3, the name and price are stripped of spaces and end of line characters so as not to interfere with displaying the data and saving data back to a file. In line 61 of Figure 3: List of Objects, Part 2, the end of line character is added back to each row as it is saved back in the file, “Products.txt.”

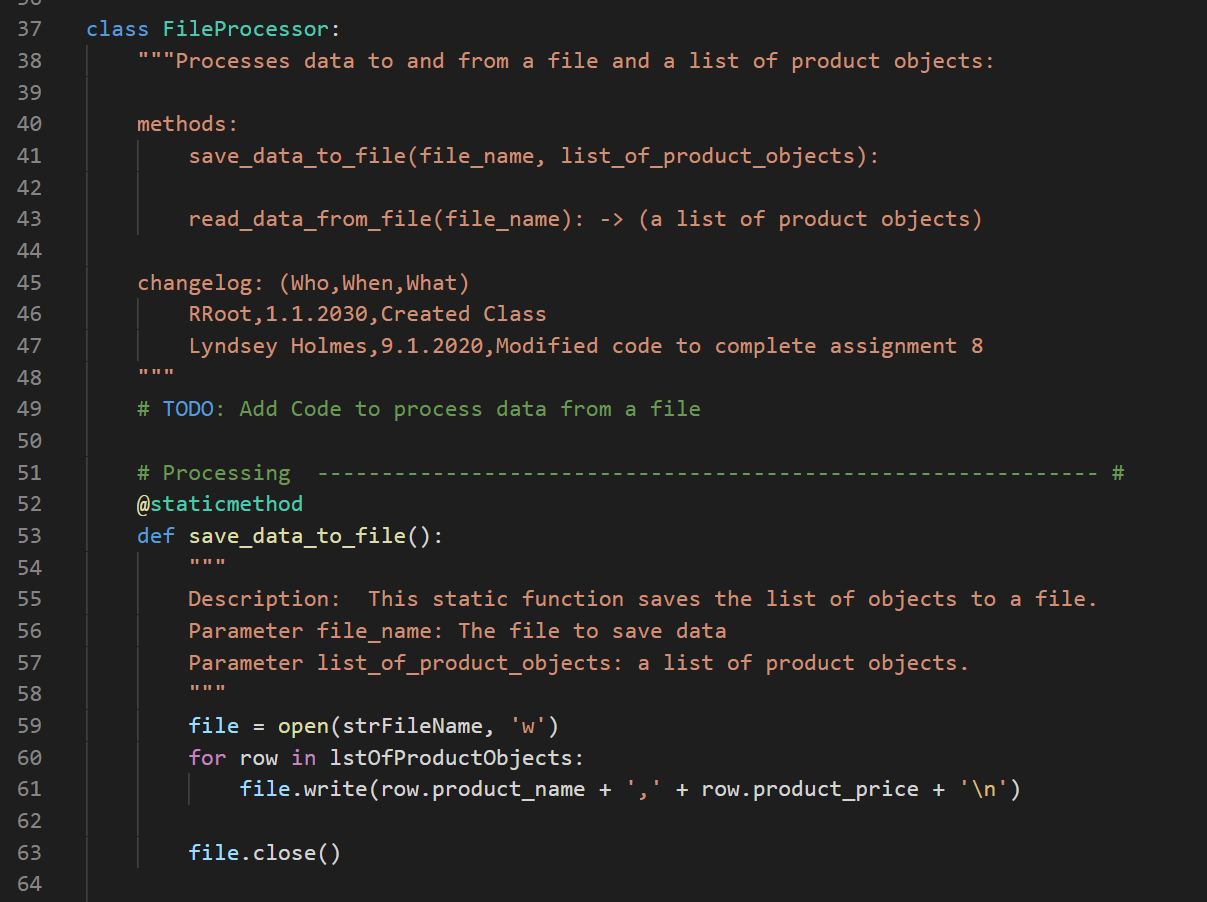


Figure 3: List Of Objects, Part 2

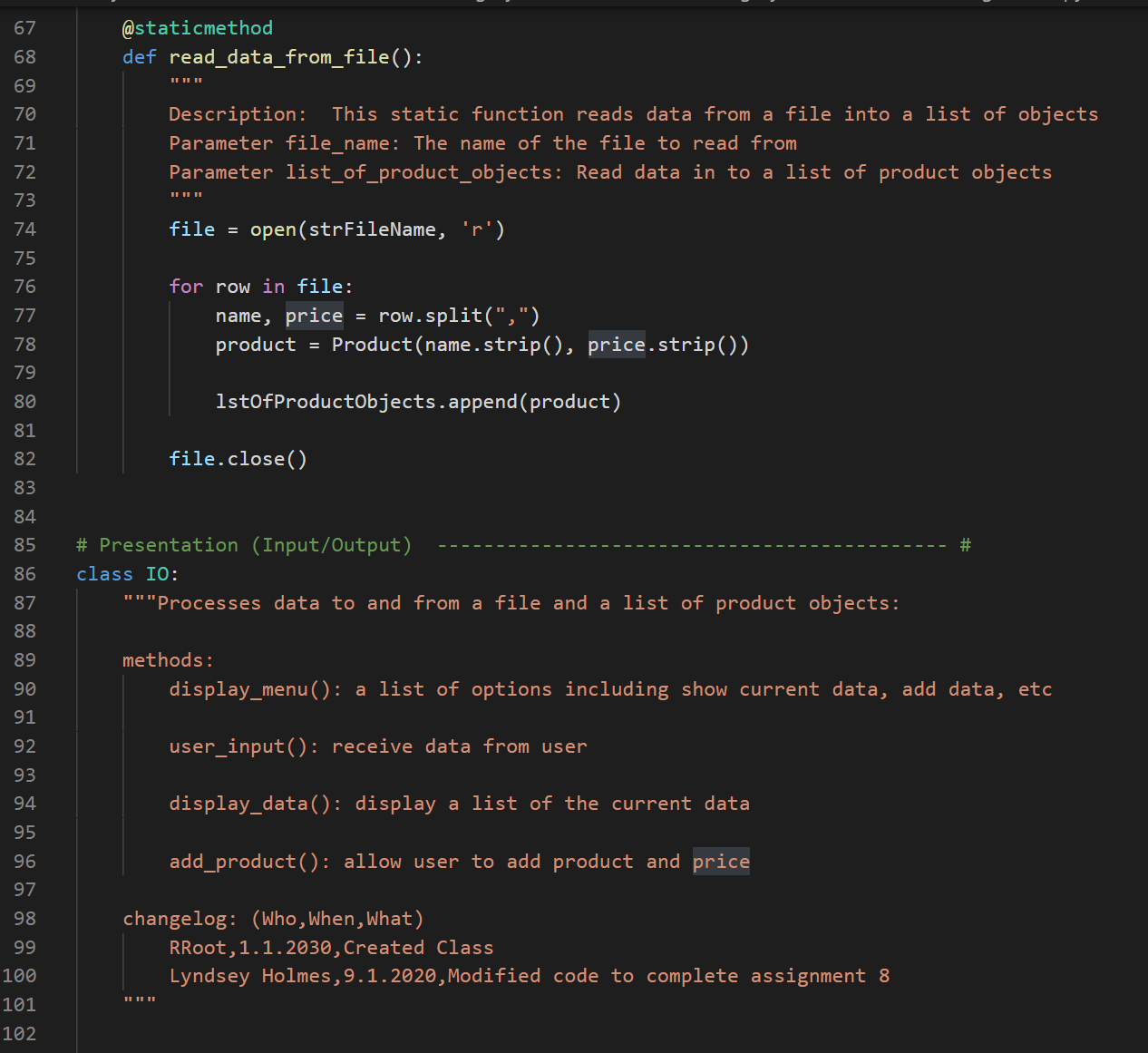


Figure 4: List Of Objects, Part 3

## Input/Output class

In Figure 4: List of Objects, Part 3, the document string includes details about the methods to be used. The methods are display a menu, receive user input of the menu choice, display current data available from file (including added data), and add data. Figure 5 shows the displayed menu is a list of print functions. Figures 5 and 6 show that the user input is a little more involved. A try-except function is required to ensure that user error will not cause a Python error. A user may not understand a Python error, so more description from the developer is required to explain that a letter or character other than the numbers 1-4 are unacceptable input. The try-except function prevents the script from ending due to user error. A counter was used to keep a menu available on the screen. If the user has trouble with submitting the required input, then the menu will no longer be available on the screen. So, a conditional statement checks the count and if the count is 3 or more, the menu function is called to display menu for convenience.

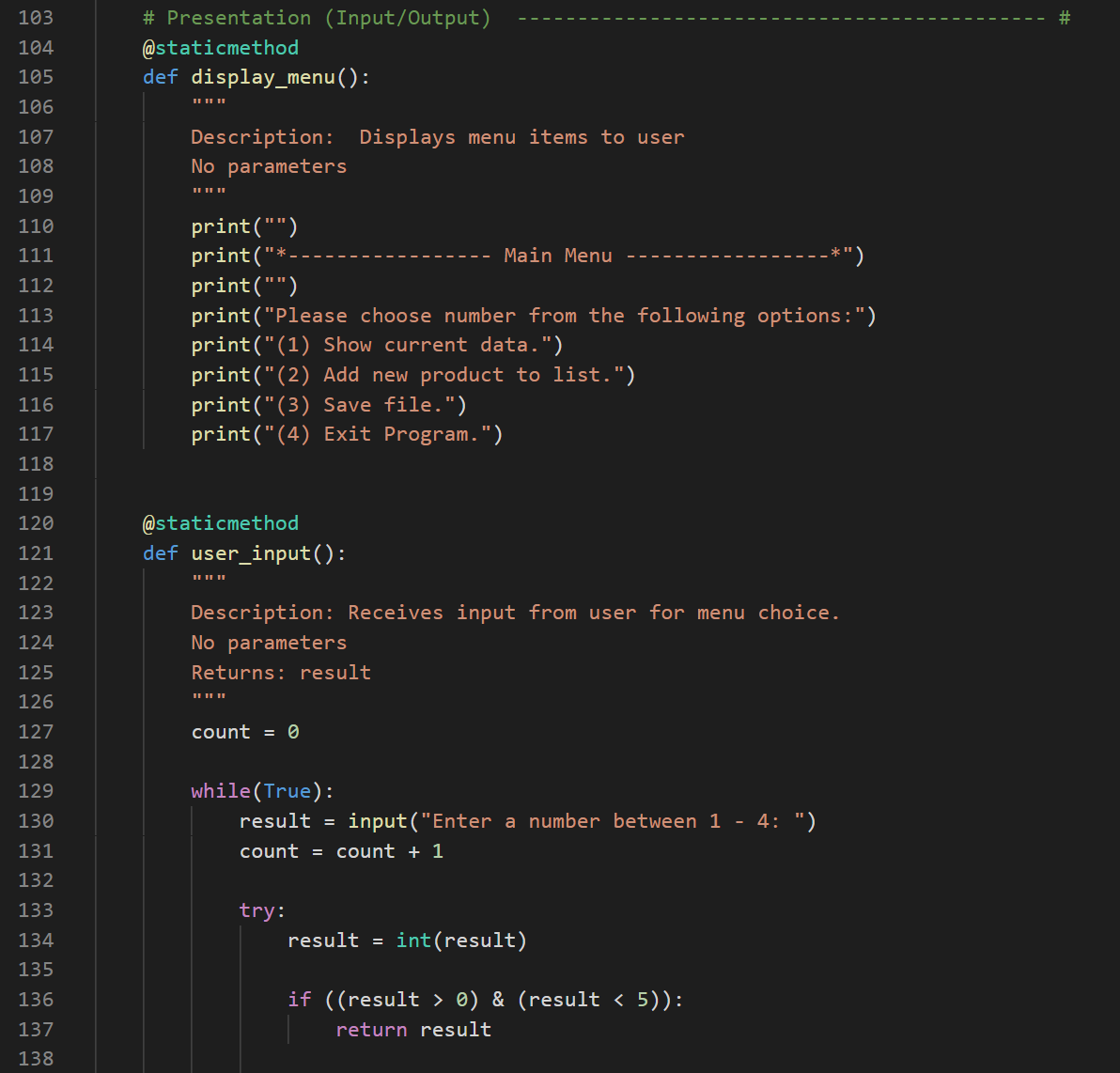


Figure 5: List Of Objects, Part 4

Figures 6 and 7 provide details regarding displaying data method and adding a product method. These both require the use of the product objects. If a print function were used to display the list of product objects, then Python would print the stored address of the products in memory. That is not useful information to the user of this script. Instead a loop is required to loop through the product objects and print each product.name and product.price with a comma between the two. In order to add a new product, first the user input is needed and then that data is stored into the product.name and product.price and then appended to the list of product objects. This is shown in Figures 6 and 7.

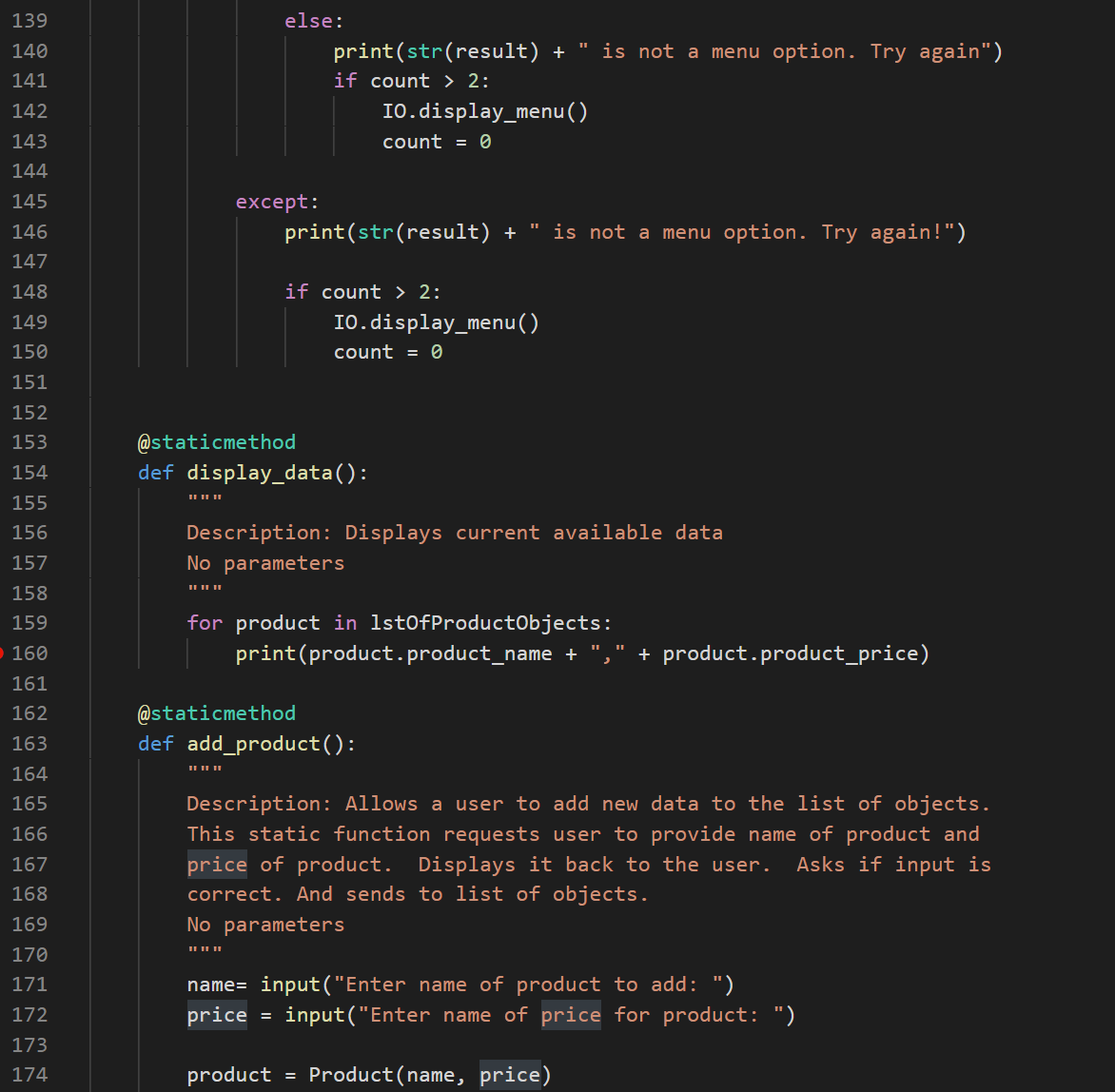


Figure 6: List Of Objects, Part 5

## Main Body of Script

The main body of the script, shown in Figure 7, List of Objects, Part 6, pieces the classes and their static methods together. Since none of the methods are called initially, they will not run without the main body of the script directing the functions. The first step of the main body is to call the read data function. The next step is calling the menu function and user input function. These functions are looped in an endless loop so that the script does not end before the user decides to end it. Once the user input is received, a conditional statement is required to choose which function to call. If the user chooses to exit (option 4), then the loop will break and the script will end.

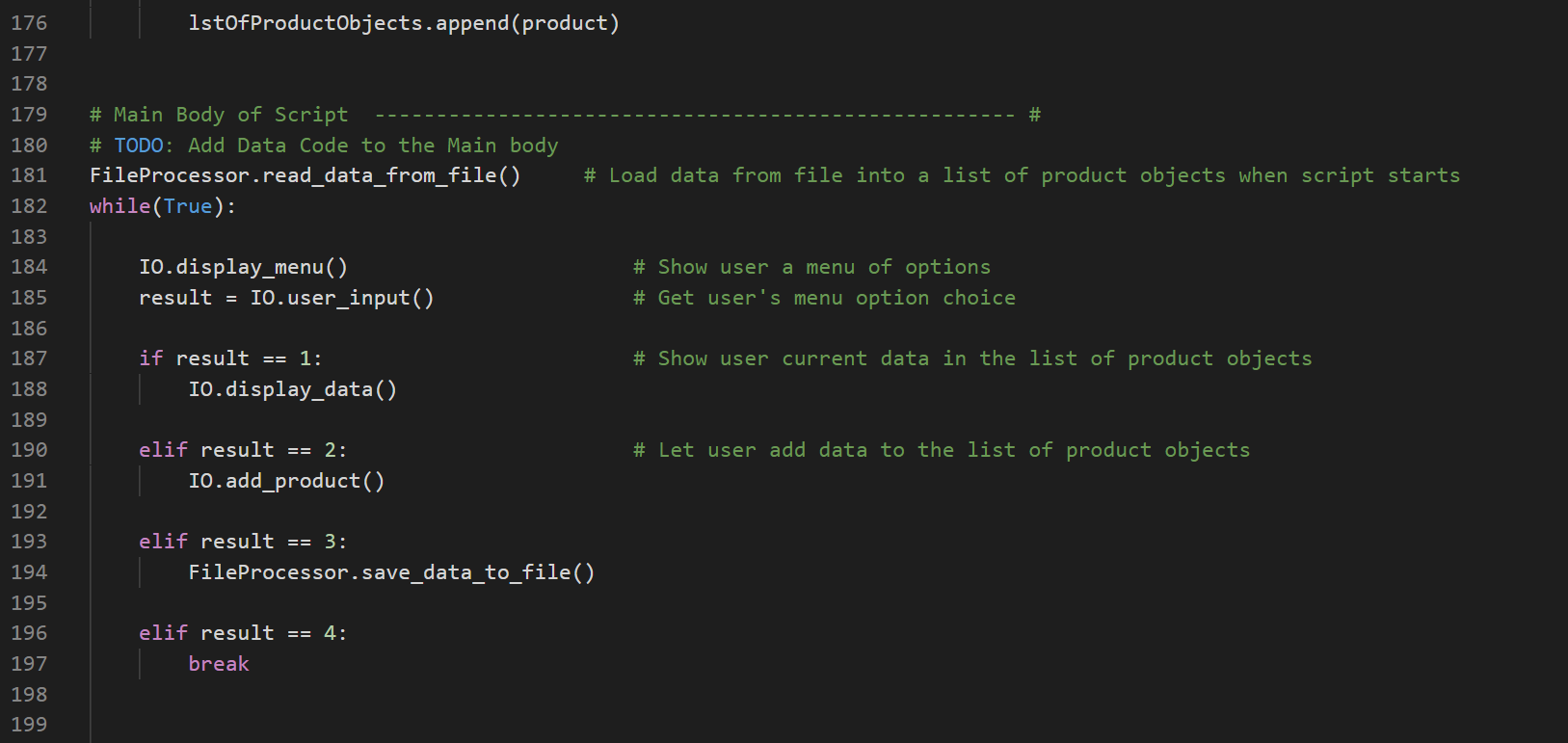


Figure 7: List Of Objects, Part 6

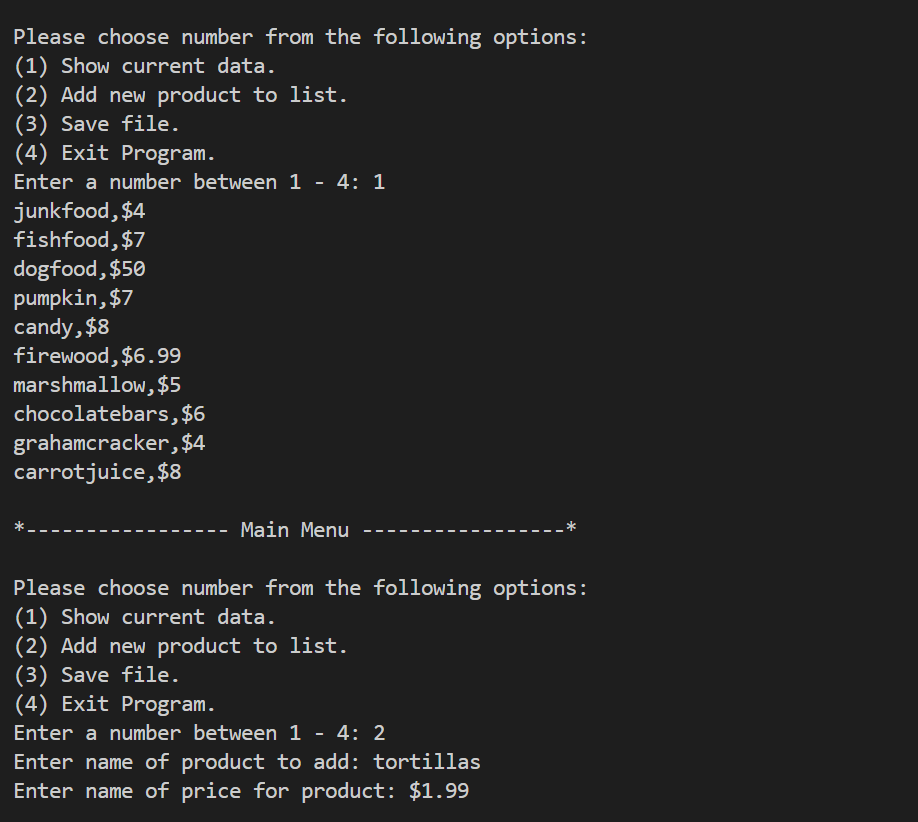


Figure 8: Show Current Data and Add data

## Running the Script

Figures 8, 9, and 10 show the script while running. As seen in Figure 8, first the menu is displayed. The user chooses item (1), which displays the current available data from the file. The menu is displayed again for the user to make another choice. When the user chooses item (2), the user is asked to enter the name of a product and price of the product. In Figure 9, the menu is displayed again. To show that the new item was added, the user selected item (1). Tortillas for $1.99 was added to the list. As shown in figure 10, the menu was displayed again. The user saved the data to the file and exited the script. Figure 11 displays the data copied directly from the text file.

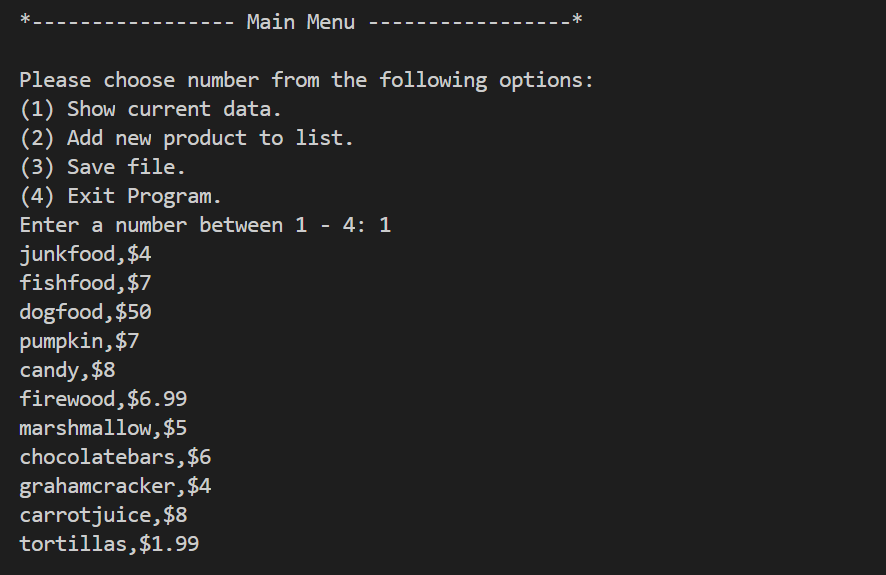


Figure 9: Showing added Data

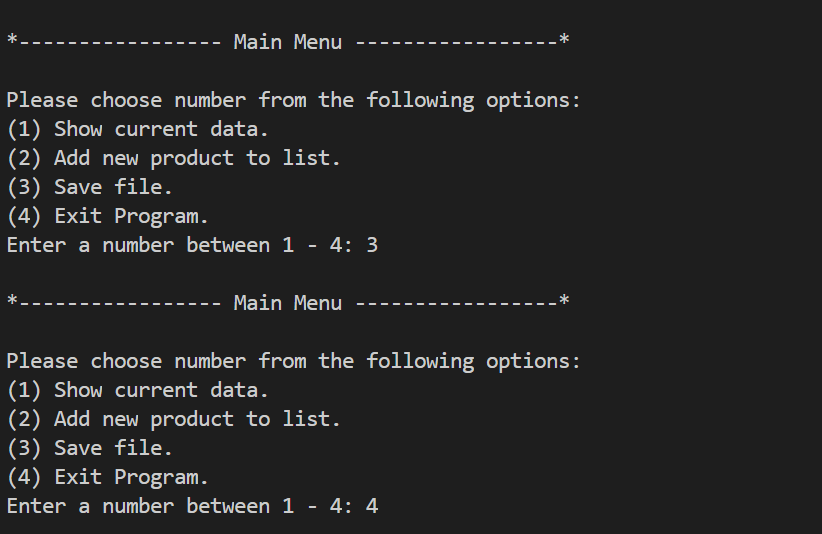
****

Figure 10: Save file and Exit



Figure 11: Product.txt

# Summary

In summary, a dictionary would have been too advanced for this simple script. Using a list of objects allows for setting properties to products and displaying them to a user or saving them to a file. The use of loops and conditionals are still required. Also, instead of using functions, the methods applied to the classes organize the code and are accessible throughout the script.