Henry Chisolm

September 8, 2021

Document Write-up:

Assignment 08

**Adapt and write a script from pseudo-code provided.**

**Introduction:**

The purpose of this assignment was to take a script that had pseudo-code and develop a functioning program that adhered to the outline of the pseudo-code and provided error handling.

**Scope of Assignment:**

The scope of this assignment is really dictated by the pseudo-code that was provided. It outlined a program that would read and write to a file from user inputs of a product with it’s price. Not many more parameters were given other than classes and sketches for function categories. The code and pseudo code for this assignment are similar to Assignment 06, so a lot of the code is borrowed and adapted from that assignment. This assignment will be analyzing the class of functions provided and the code to run them.

(1) Class Product

(2) Class FileProcessor

(3) Class IO

(4) Body of the Script

(1) Class Product:

This class was established to store data about a product including both it’s name in string form and it’s price in float form. To do this we utilize the class to define objects or in this case products with the attributes of name and price. We start out by utilizing a constructor which is a function that automatically runs when we create an object in this class. In this case we are using it to set the attributes with product\_name and product\_price

Graphical user interface, text, application, email

Description automatically generated(fig. 1 code illustrating constructor)

In this case we use “\_\_init\_\_()” which automatically passes and argument into it or in this case it sets product\_name to a string value and product\_price to a float value.

Next up we code properties to manage and attribute our data in this case both a getter and a setter. This is done for both of our properties name and price.

A setter allows for validation of values and error handling , where values passed into this function of they are valid are assigned the attributes of the function. In this case we have a setter for both the name and price attributes. The setter “self.\_\_product\_name = value sets the value to a string and takes in and value that is a string. Likewise the setter in price does the same but takes in the product\_price as a float value.

Text

Description automatically generated

The getter in this case lets me add code to attribute data, with product\_name returning a string and product\_value returning a float value. Finally, a function of method \_\_str(self) is used that returns the data in a form we want and can utilize.

Chart, scatter chart

Description automatically generated

So when we finally invoke the class Product we get the data of name and price in the format that we desire.

(2) Class FileProcessor:

This class is used to handle the processing portion of our program or in this case process data from a file and save data to a file. In this case it is done with a read\_data\_from\_file and save\_data\_to\_a\_file functions.

Graphical user interface, text, application

Description automatically generatedText

Description automatically generated

The read\_data\_from\_file looks for the file (in this case later attributed to “product.txt”) and if there is any data in that file loads it into the program separated out. Our save\_data\_to\_file writes the data we have inputted in our program and saves it to the “product.txt” file

(3) Class IO:

This class is created to perform input and output tasks by the program. This was heavily adapted from the code in assignment 06. This is done by creating 4 functions in this class all using the @staticmethod which enables it to return object of the class.

In this case the pseudo-code wants a menu of choices to be displayed (print\_menu\_options()), receive input from menu choice (input\_menu\_choice()), print the current data that has been pulled from the “product.txt” (print\_current\_product\_list) , and add new product information (input\_add\_new\_product\_and\_price).

Graphical user interface, text

Description automatically generated

When called upon the IO.specific function will run that specific task.

(4) Main Body of the Script:

Graphical user interface, text, application

Description automatically generated

The script starts out by loading data from a file int a list of product objects when the script starts. This is accomplished by setting a variable that was given to us initialized at the beginning of the script lstOfProductObjects to the FileProcessor class read\_data\_from\_file function to the variable strFile which we have calling our product.txt file. So anything from that file will be loaded into the program. Next set up a while loop where when true we get we get the user’s menu choice.

Text, letter

Description automatically generated

We do this by setting a variable to Our IP class function input\_menu\_choices which captures 1-4 and then brings the resultant action. If 1 is called the the IO class fucnction of print\_current\_product\_list runs bringing our inputted data back (in all honesty I couldn’t get this portion of the program to work). Option 2 get the IO class function input\_add\_new\_product\_and\_price to add data to our list of product objects. 3 allows us to save data to out “products.txt file” by calling our FileProcessor class save\_data\_to\_file function. Option 4 just offers a break and exits the program. After all of the options but 4 the program loops around back to the start menu.