Hyojin Kwak

December 7, 2020

Foundations of Programming

Assignment08

https://github.com/hihojin/assignment08/tree/main

**\_\_str\_\_() and self() in Classes**

**Introduction**

This week, we learned about error handling such as ‘try-except’ and another terminology: pickling, which is a technique to save as a binary file. Assignment07.py is a new script that is organized with Processor class and its functions, asking user about their grocery items and costs which are ultimately saved in the dictionary lists of a table. Below paragraphs will show a step-by-step information of how I wrote codes to create the script.

**Creating the script**

Under class product, I formatted the class in order of constructor, properties and methods. Constructor was simply set with two attributes of self.\_\_name and self.price. I did not forget to name a constructor with \_\_init\_\_(self, name, price). I made total of four properties, by making a getter and a setter for each attribute. In getter: @property, I formatted self.\_\_price and self.\_\_price with .title() so that when they are inputted by the user later, the first letter would be automatically capitalized. In setters, I raised exceptions and added validations for values so when values are correctly inputted by the user it will pass the values to attributes or it will raise the error message of Exception error messages.

Graphical user interface, text, application

Description automatically generated

*Figure 1. class product constructor*

Text

Description automatically generated

*Figure 2. properties and methods in class product*

Method parts was also done with simple coding with return of self.name and str(self.price) with comma separation in between.

In FileProcessor class @staticsmethods are used since it is indeed a processing section of codes. When to read data from file, try except was used so that when it fails to load data from file it will show an error message saying ‘could not load data from file’ or it will read line by line in the file and brings [name, price] of products and return the list of products of objects.

Text

Description automatically generated

*Figure 3. class FileProcessor, read\_data\_from\_file(file\_name)*

To write data in the text file, I first opened a file with append mode. Since data I want to write is in list of product object table, I used for loop so that it can be written line by line, or list by list. When data is written to the file successfully, it will return ‘saved!’ message and ‘not saved!’ when data was not saved.

Text

Description automatically generated

*Figure 4. save data to text file function*

Class IO is where mainly input functions are used to get input from the user and return the value of name, price, and their menu choices. Similar functions from the week 6’ script was used.

Text

Description automatically generated

*Figure 5. class IO functions*

Text

Description automatically generated

*Figure 6. main body of codes*

While loop was used throughout the main body and returned values of choice, name, and price were used with class FileProcessor and class product methods.

**Getting Output**

Text

Description automatically generated

*Figure 7. Output in Terminal*

Rectangle

Description automatically generated

*Figure 8. Output in products.txt*

Text

Description automatically generated

*Figure 9. Output in PyCharm*

Text

Description automatically generated

*Figure 10. Output in PyCharm cont’d*

The script ran smoothly both on PyCharm and on terminal. Error messages popped up correctly and told the user about the error with the proper exception message.

**Summary**

I had some hard time trying to append the objects to the list table at first, but I figured it is very similar to adding items to list without objects. As the course is getting to the end, I feel more certain that I would need to continue practicing coding and get familiarize myself to use classes and function more often.