Kyle Banfill

3/15/2022

Foundations of Programming: Python

Assignment 08

<github website>

Objects and Classes

Introduction

In this module, we learned to create and utilize classes and objects by breaking down their parts. We also explored more of GitHub’s functionality.

Questions

1. What is the difference between a class and the objects made from a class?

Classes use code directly while object instances use code indirectly.

1. What are the components that make up the standard pattern of a class?

The components of a standard class pattern are Fields, Constructors, Properties, and Methods.

1. What is the purpose of a class constructor?

Constructors are used to set the initial values of Field data.

1. When do you use the keyword "self?"

You use “self” when identifying different instances of a class.

1. When do you use the keyword "@staticmethod?"

When you want to include methods called directly from the class.

1. How are fields and attributes and property functions related?

Property functions validate values that get assigned to fields and attributes.

1. What is the difference between a property and a method?

Properties are functions that manage attribute and field data; Methods are other functions within a class.

1. Why do you include a docstring in a class?

IDEs can display tooltips to show the dev’s notes regarding the class.

1. What is the difference between Git and GitHub?

Git is software on our computer that allows us to track and manage version history. GitHub is the online website where we can host our Git stuff.

1. What is GitHub Desktop?

GitHub Desktop is an app that allows us to interact with GitHub from our local computers.

Our Products and Price List

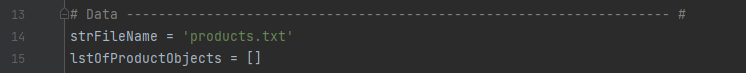
Text

Description automatically generated

Text

Description automatically generated

For this module, we created a product and price list. Its construction is very similar to what we did in module 6 for our task and priority list, so a lot of components will look similar.



First thing we do is initialize values that will exist outside our classes. For this script, the file name is important to list here.

Text

Description automatically generated

Here we define our object class. We’re looking to create two key variables, product\_name and product\_price. Originally, I had made the attributes as “self.\_\_product\_name” and “self.\_\_product\_price”, but I was finding when debugging that it wasn’t validating correctly with the getter and setter. I admit that I don’t completely understand the interaction and why it wasn’t working correctly in my code, but after changing it so that it referenced the code directly, it was validating correctly.

Text

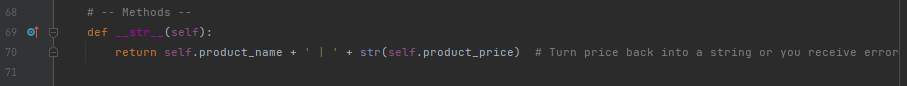
Description automatically generated

Here, I’ve set up the getter and setter for product\_name. This does use the double underscore in order to make the values private. I’ve included structured error handling so that it raises an exception if the value is numeric. We can see where it raises to later in the code when we get to the main body.

Text

Description automatically generated

Same deal for the next property. You may notice that in our attributes, we didn’t force the product\_price to be a float value despite us indicating that was our intent. If we set things up correctly, we can do kill two birds with one stone by setting valid values to float here while also setting up logic to catch invalid values if they can’t convert to float with our try-except.



Here we set up our method. We’ve set it up so that it returns product\_name and product\_price together with a bar separating as a unified way to both write and read data with later code. I’ve converted the product\_price, which should be a float, to a string because we can’t concatenate float values.

Text

Description automatically generated

Next is our processing segment. We begin defining our FileProcessor class. The save data and read data methods were included in the starter, but I included a remove data as well.

Text

Description automatically generated

Our first method in this class is the save data method. We largely took the code from assignment 6 to create this, but we’ve updated it to include the method we created earlier in our Product class. We’ve also included a try-except just in case something goes wrong with saving the file.

Text

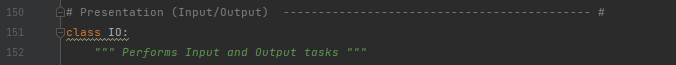
Description automatically generated

Next is our read data method. Again, we’ve largely taken code from assignment 6, but updated it to match our created classes and added a try-except.

Text

Description automatically generated

The last method in this class is the remove data method.



Next, we define the class we’ll use for presentation: IO.

Text

Description automatically generated

A lot of these methods will look very similar to assignment 6 as we’re going to be using the same menu format.

Text

Description automatically generated

Here is our input menu choice method.

Text

Description automatically generated

Here is our output. Again, we convert our float values to string just to make sure there aren’t any errors in displaying it.

Text

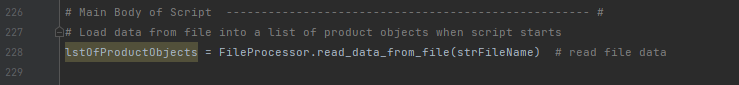
Description automatically generated

Here is our input new product method. I originally had this set to change the price input into float, but thanks to us using our new getters and setters, they’ll undergo that validation once we pass them through our Product function.

Text

Description automatically generated

Here is our input product to remove method.



In the main body of our script, we start by reading the file data so that it loads into memory.

Text

Description automatically generated

Here, we’ve included the try-except *inside* our while loop so that if there’s an error when the user inputs values, we’ll catch it with the except inside the loop, which will move us to the beginning of the loop. This way, if a user does something like mistakenly types their product twice, or mixes up product and price, the script doesn’t just crash completely.

Text

Description automatically generated

Here is the main body of our loop which looks just like it did in assignment 6.

Graphical user interface, text

Description automatically generated

And, finally, the except block inside the wile loop. We could use some cleaner verbiage if we were really trying to impress, but I figured it would be helpful to make note that the user will be heading back to the beginning loop.

Conclusion

This module was particularly challenging. Objects are a tricky concept to learn, but important for clean coding with lots of variable setting. We also explored how attributes, properties, and methods all interact with each other to normalize our variable construction.