Lauren Parker

December 6, 2020

IT FDN 110 A Au 20: Foundations Of Programming: Python

Assignment 08

GitHub

Learning to Use Classes

# Introduction

This week’s assignment was to create a script showcasing three classes, Product, FileProcessor, and IO. Our professor provided us with starter code that we would need to read, understand and then add code to make the program work. The following showcases the steps I took in order to get this program up and running.

# Step 1: Importing Starter Code & Reviewing Pseudo Code:

The first thing I did was create a new file in PyCharm and imported our professor’s starter code for Assignment08. I reviewed the code and took note of what the code did, and where I would need to add additional code. This was denoted by the keyword TODO, which you can see in figure 1.

Text

Description automatically generated

***Figure 1. Example of TODO tasks highlighted in green.***

# Step 2: Getters, Setters, & Error Handling

After reading over the script, I knew that I would need to add additional code to three separate classes in order to get the program to run. The program would present a menu to its user, providing four options to choose from, presenting the current data, adding data, saving data, and exiting. There were three classes already defined by our professor. Product, the first class, which would store data about the product using getters and setters. FileProcessor, the second class, where I would need to define static methods to write data to a file and save data to a file. The last class, IO, would perform input and output tasks showcasing the menu and its options to the user. This class would also contain static methods.

The standard class pattern includes fields, constructors, attributes, properties, and methods. I knew that I would need to add attributes, or virtual fields, to the Product class. I would need to add properties for each attribute, which would include a getter and a setter that would aid in error handling. Figure 2.1 and 2.2 shows the full code for the Product class.

Text

Description automatically generated

***Figure 2.1: Code for Product Class part 1.***

***Text

Description automatically generated***

***Figure 2.2: Code for Product Class part 2.***

# Step 3: Defining Static Methods for FileProcessor

For the processing section of the code, I needed to create two different static methods. One that would read data from a file and one that would write data to a file. Figure 3.1 and 3.2 shows the code for both static methods, their doc strings, and the error handling used to keep the code running smoothly.

Text

Description automatically generated

***Figure 3.1: Defining the first static method for FileProcessor.***

Text

Description automatically generated

***Figure 3.2: Defining the second static method for FileProcessor.***

# Step 4: Input/Output

The next portion of the code was listed under the presentation IO class, for input and output code. This code would process the user’s menu choice as well as the product name input and product price input. There were four static methods defined. One for each option in the menu. Figure 4.1 and 4.2 shows the code for each.

Text

Description automatically generated

***Figure 4.1: Static methods in IO Class part 1.***

***Text

Description automatically generated***

***Figure 4.2: Static methods in IO Class part 2.***

# Step 5: Calling Functions in Main Body of Script

The next portion of code is considered the main body of the script. It would be used to call the previously defined functions from the above classes. I used a try-except block to add in additional error handling as you can see in figure 5. I used dot notation to drill into the IO and FileProcessor class in order to access the static methods. You can see an example of this in figure 5.

Text

Description automatically generated

***Figure 5: Main body of script.***

# Summary

In conclusion, we have reviewed how I went about setting up the code for assignment 08. By utilizing classes I was able to break up my code into organized sections for data, processing, presentation and main body code.