Fernando E Hernandez

December 2nd, 2019

Working With Classes

Assingment08

# preface

## In this assignment we were given a starter file to edit with the main goal of adding custom classes in order to organize functions properties. One thing we did a bit different was that we did not directly copy the starter file *or text* onto an editor. We decided to work with sections of the code separately and test them until the program could provide us with a good execution. We will now proceed to illustrate all errors as we debugged the code.

# the “duder”

## As we type our selection (1) for displaying items we find this error *(Fig. 1)*. Notice how the messages describes how the ‘Product’ object does not have any attributes. Certainly from the picture we can see that we are missing the “duder” or the double underscore for the Constructor (special methods).

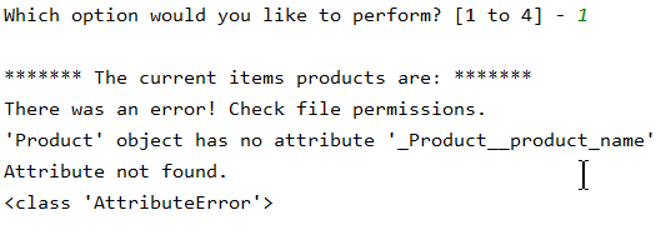


Figure 1

## Looking back at the code we precisely capture the error right in front of the “product\_name” parameter as we missed the underscore *(see Fig. 2)*.

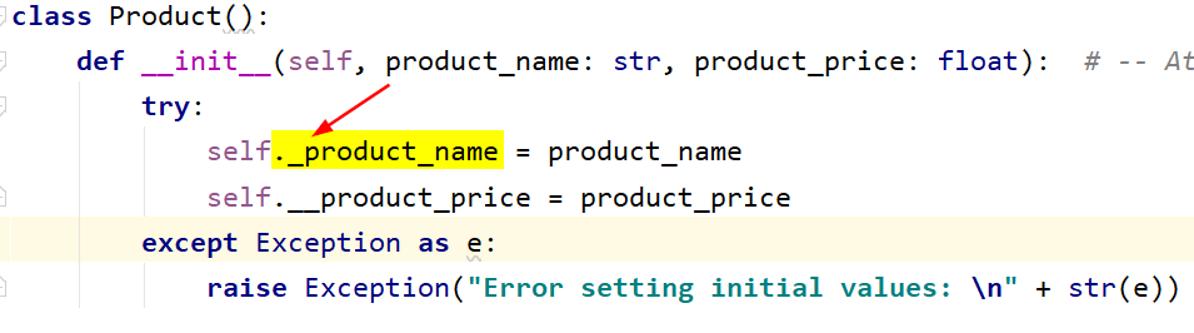


Figure 2

# the empty file

## As we make our selection again (1) for displaying the items we can see that the products item list is empty and the “index” is out of range *(see Fig. 3)*. The code is looking for two values from a file but apparently it finds nothing.

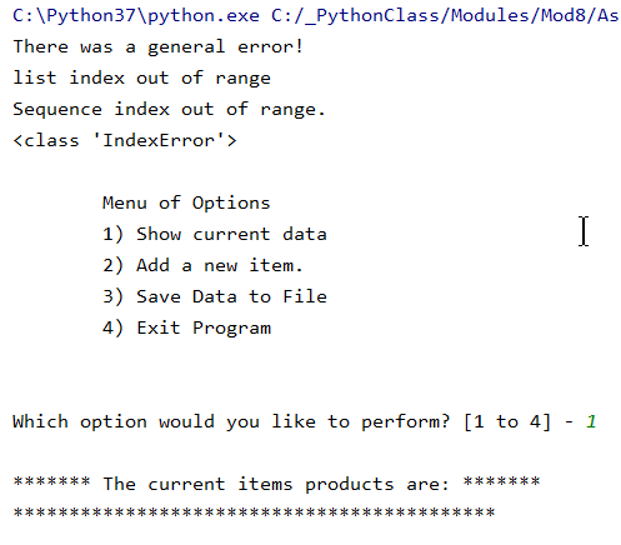


Figure 3

## Consequently, after running the debugger we observed that all the code found was practically nothing as it returned ‘\n’ *(see Fig. 4)*. This led us to believe that somehow data was missing from the file. Surely enough, as we opened the “products.txt” file we spotted a big empty space *(see Fig. 4)*.

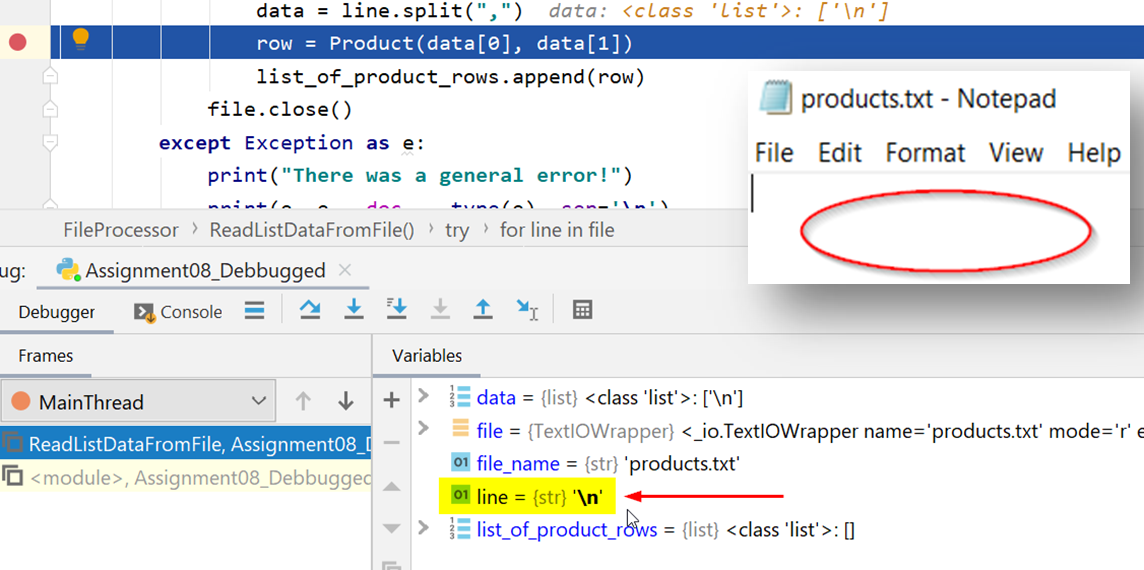


Figure 4

# testing for product “price”

## For this part of code testing we decided to test the product price by giving it a string input *(see Fig. 5)*. As soon as we type the price without any numbers we immediately get a message about not being able to convert string to float. This was a good indication that the program code was doing its job. Changing the input to numbers would then display the product and its price correctly *(see Fig. 6).*

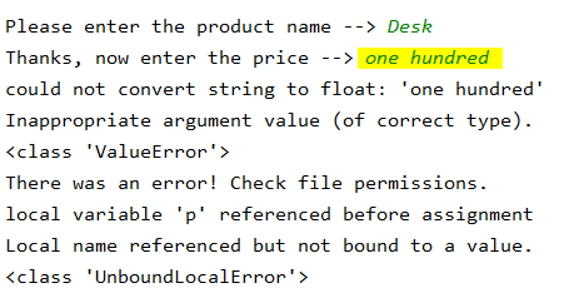


Figure 5

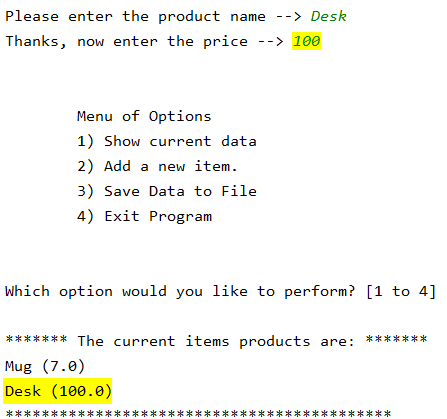


Figure 6

# testing for product “name”

## In this particular test things didn’t work as expected. As you can observe just below *(Fig. 7)* the program took the floating inputs as values and saved. A lot of changes were made to the program code but those couldn’t provide the answers we were looking for.

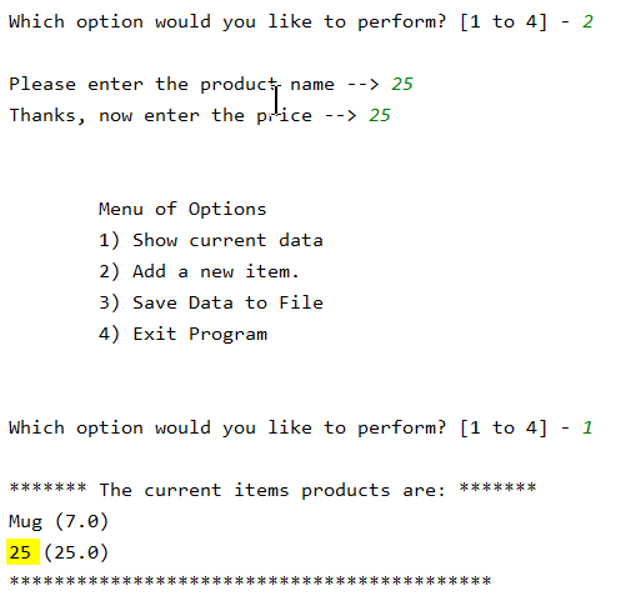


Figure 7

## However, we tested this particular section separate from the main code *(see Fig. 8)* and noticed that it would work without a problem. The “product\_name” from the code wasn’t taking any numbers as strings. In fact, it would return the message “Names cannot be numbers” as planned.

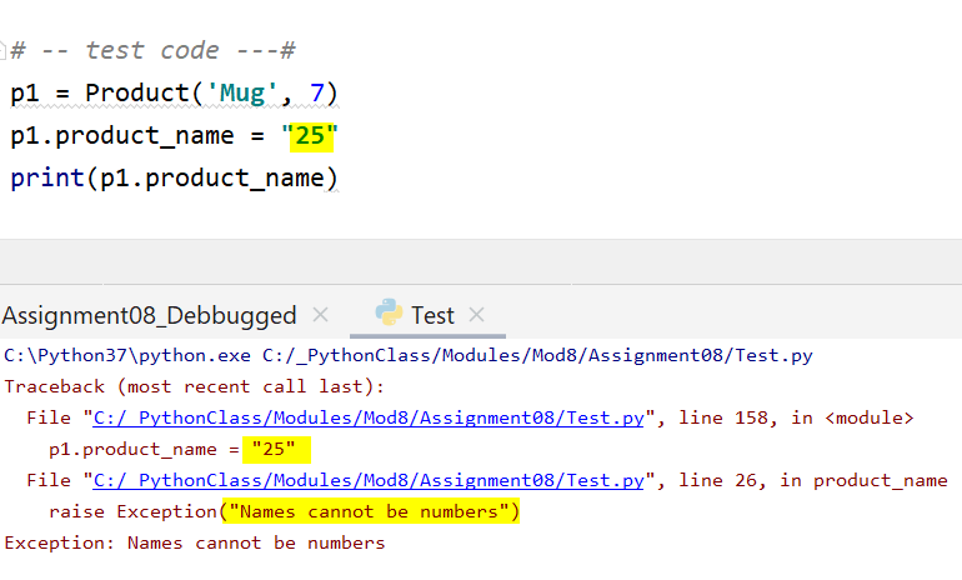


Figure 8

# closeout

## Testing for program code logic is definitely not an easy task and at times it may seem like code strength can be somewhat subjective. Classes and Methods are a great way of making programing code very intuitive, organized, and ready for multi processes. After having completed this exercise we have learned that Python is an “object-oriented” programming language; especially after having used the “classes” and their properties.