Adrian Ronsse

August 27, 2021

Foundations Of Programming: Python

Assignment 07

**Pickling files and Structured Error Handling**

GitHub repository: https://github.com/adrianronsse/ITFdn110-Mod07

GitHub website: https://adrianronsse.github.io/ITFdn110-Mod07/

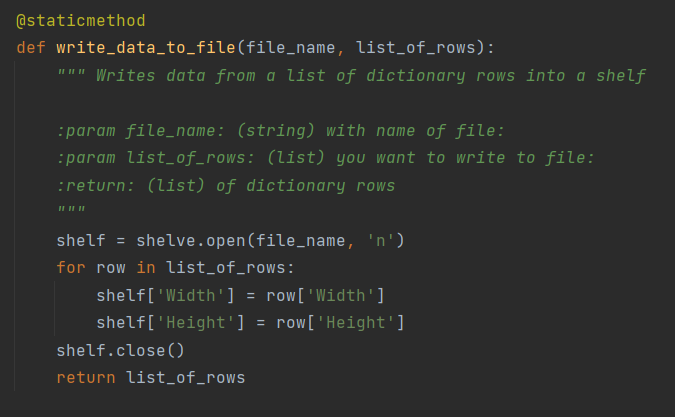
**Introduction:**

This assignment covered pickling data into a binary file (.dat) and using structured error handling to display messages to the user and keep the program running to a voluntary completion. I decided to model the code after the work we had done in Assignment06, hoping to create a basic structure for collecting, storing, and loading dimensional data that I could use in my work as an estimator for a commercial glass and door installer. My intent was to store and load data in a format such that I can later add additional code to perform calculations on the individual data entries and thus produce the quantities of framing component stock lengths used for each location.

**Writing the code:**

*Note: Since we covered function creation and implementation in the previous assignment, this section will focus on the assigned tasks of demonstrating pickling and structured error handling rather than walking throught the entire code.*

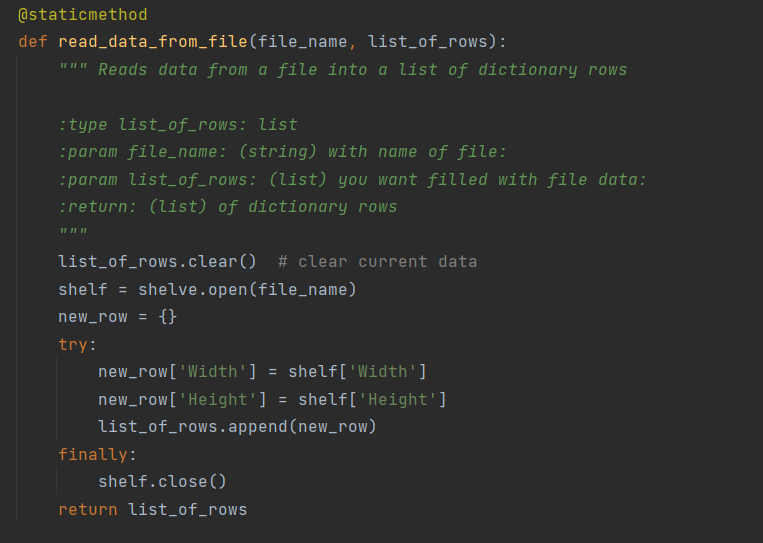
Step 1 – Create function to pickle data and write to binary file

I decided that I wanted to store dictionaries in the binary file rather than lists, because I intend to collect a third piece of data in the future that will determine whether certain calculations are performed to the first two items associated with that new piece of data. In Chapter 7 of Python Programming for the Absolute Beginner, I learned that in order to store data in dictionary format, I need to use a module called “shelve”, which builds on the pickle module. I invoke that module below for the write function: 

*Figure 1 – Writing a list of dictionaries into a binary file using the shelve module*

Step 2—Create a function to load shelf data from the binary file back into the program

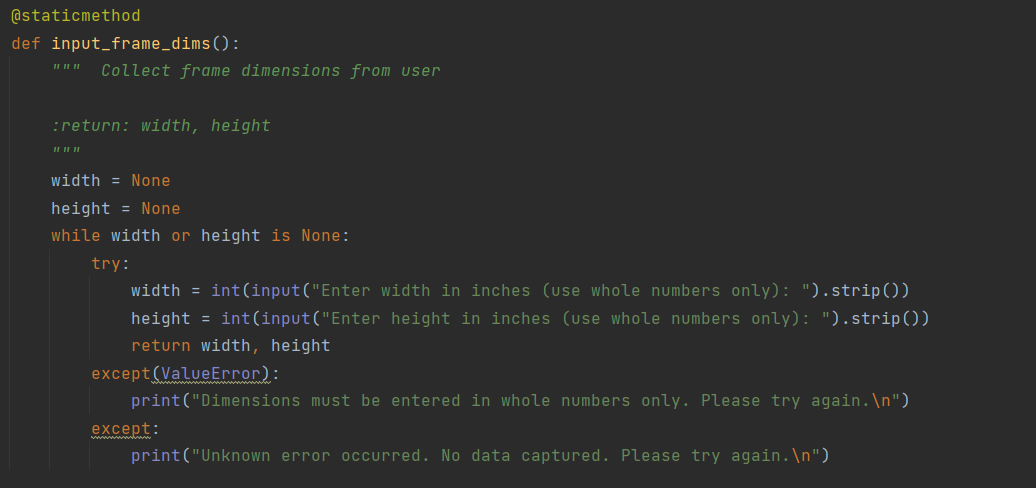
This step proved too difficult for me. I wanted to load all of the data in the file back into the program as a list of rows, but I was only able to come up with code that loads the last piece of information in the file for each of the keys “Width” and “Height”. I searched online for a solution but was unable to solve this, so I am hoping to receive feedback to this assignment that will show me how to do this properly. Here is the code I was able to come up with:



*Figure 2 – Reads data from binary file into a dictionary, but only captures the final data entry in the file*

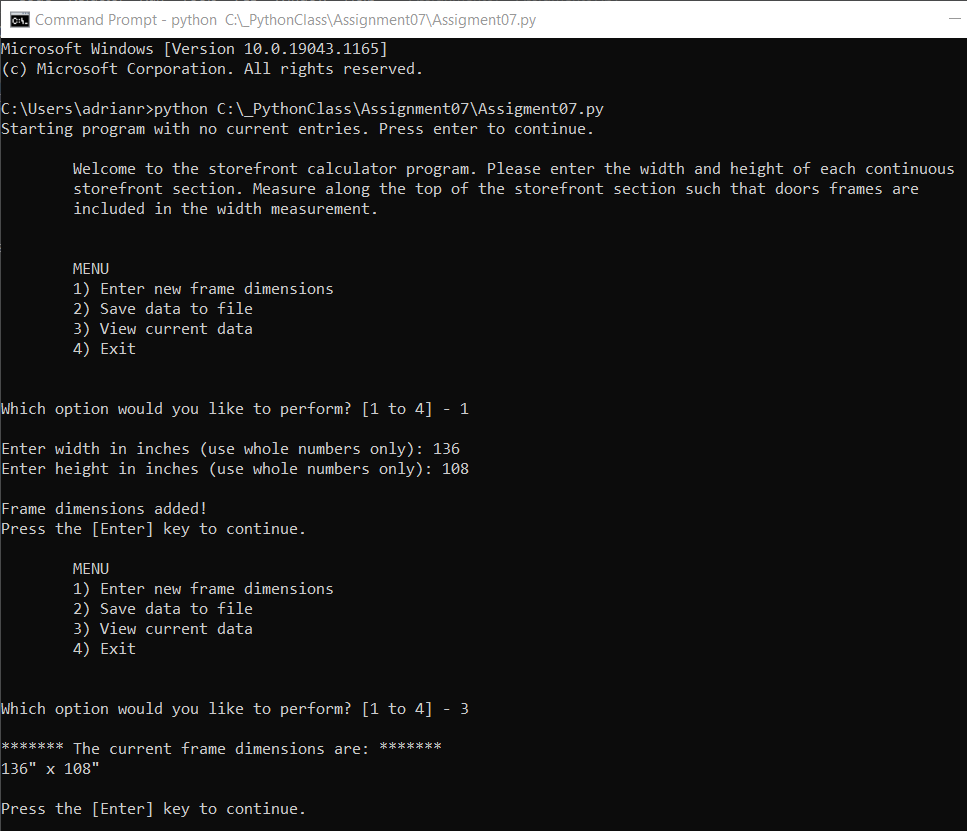
Step 3: Demonstrate structured error handling

This part of the code makes sure that the program can keep running even if the user enters data in the wrong format or if some other unforeseen error were to occur.

*Figure 3 – Structured error handling*

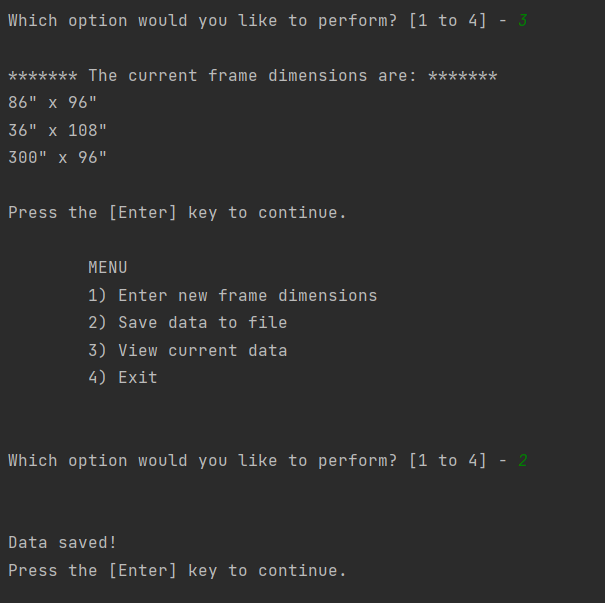
**Testing the code:**

After preparing the code, I needed to test it to make sure it was working as expected. First I opened the program in the command console to test. Here I see that I am able to log dimensions and view that data:



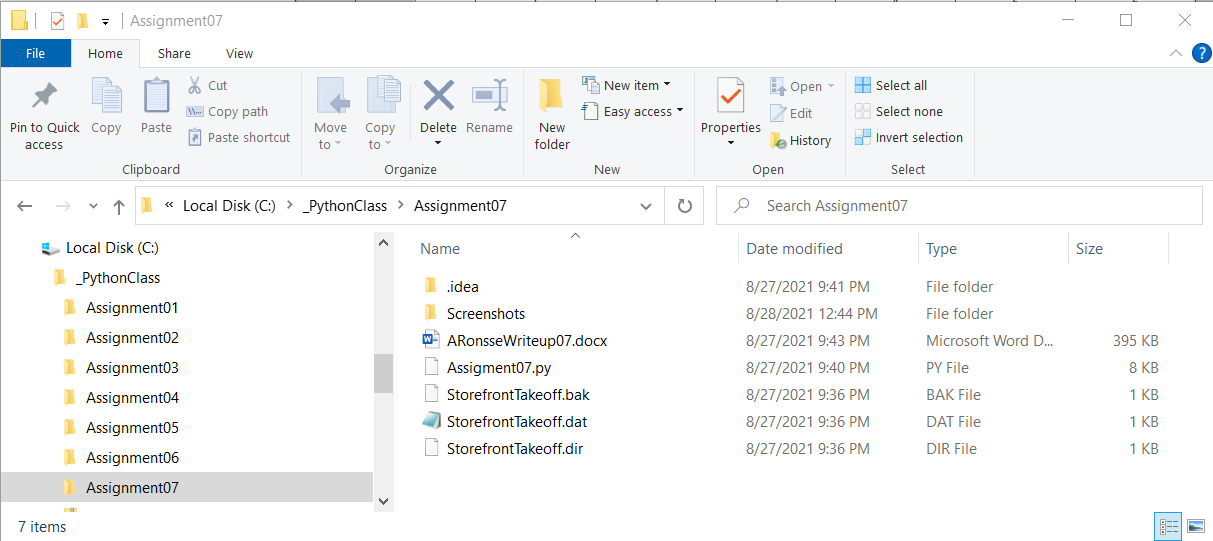
*Figure 4 – Command console test*

Then I tested it in PyCharm as well and found I was also able to save the file as intended.



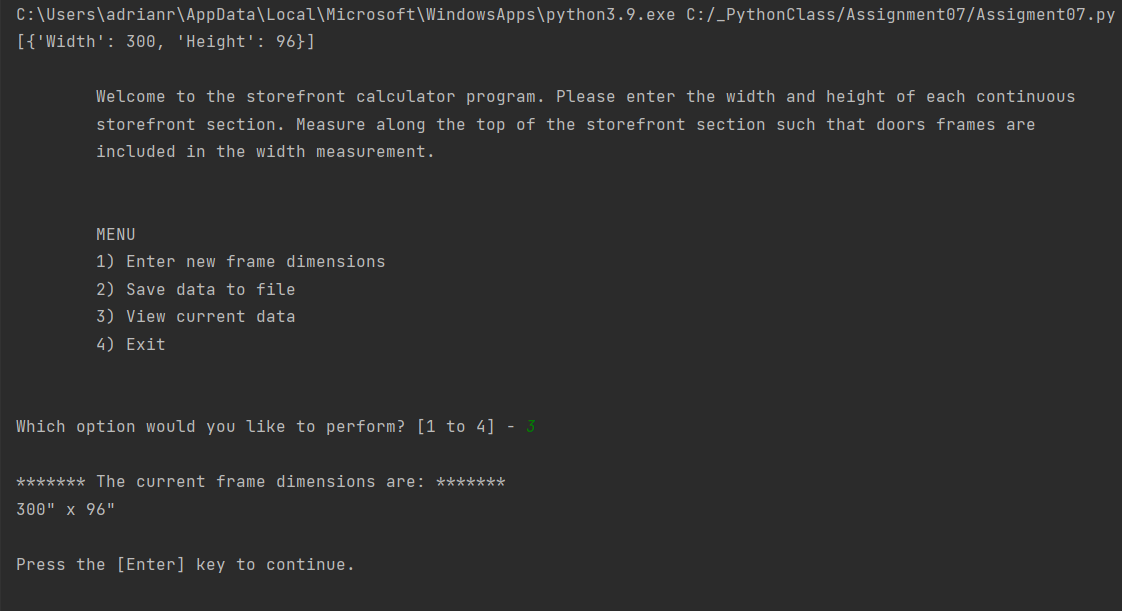
*Figure 5 – PyCharm test: data saves successfully*

I can confirm too, that the file has populated in my folder along with some other files that support it:



*Figure 6—Files saved in folder*

Unfortunately when I stop and restart the program, I find that I have only loaded the final data entry:



*Figure 7 – PyCharm test: only partial data load*

**Summary:**

In this coding exercise I was able to write binary data to a file using the shelve module, which is a way to pickle dictionaries. I was also able to load some of that data back to the file, although was not successful in fully reconstructing the dictionary list. I also used structured error handling to help ensure that the program would run smoothly and not quit unexpectedly.