Justin Calhoun

Professor Summer Rae Elasady

Foundations Of Programming: Python

20 October 2018

Assignment 03 - Household Items

This assignment tasks us with creating a python script that asks a user to input the name and price of a household item, and then to write it to a file. I knew from the beginning that I wanted this application to have both an interactive and a command line mode, so that a user could either choose to do repeated data entry, or add a new item as a one-shot from the CLI. So, to begin, I mocked up the expected program flow with pseudocode.

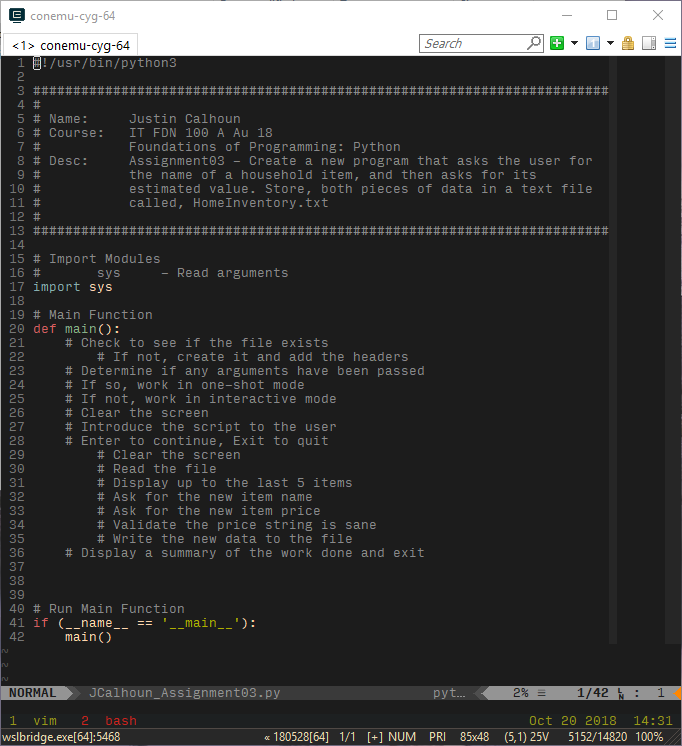


Figure 1: Pseudocode

I first built the logic for choosing between the two modes. If there are any arguments at all, we should assume the user wants to do a one-shot addition from the CLI. If not, we assume interactive mode.

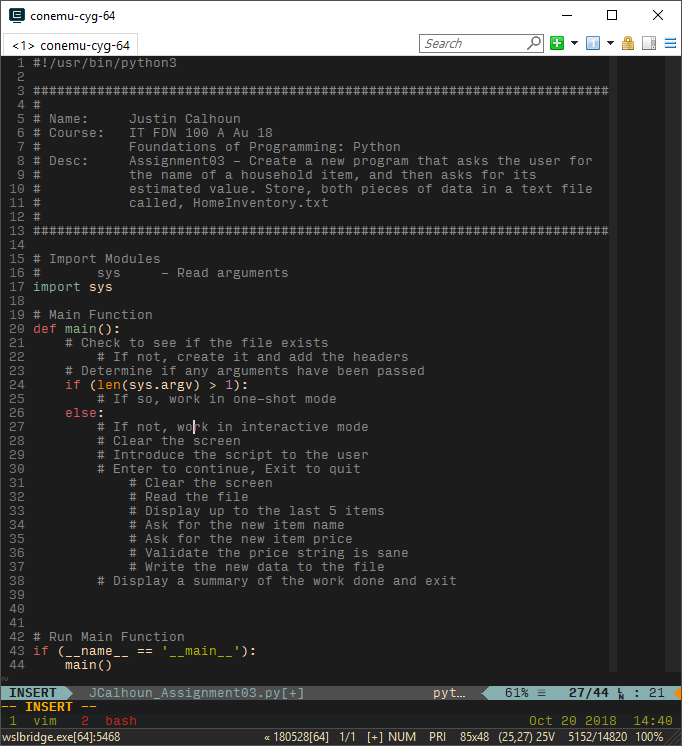


Figure 2: Simple Logic

Before continuing on with the CLI mode, I decided it would be best to write the Interactive mode, as I suspected there would be code that I could reuse later. After writing a friendly introduction with instructions for the user, and providing them a moment to bail-out if they don’t really want to run this program right now, I start the main interactive loop. I used a simple while (True), because I will eventually use break to end the loop if the user types “exit”.

For the main interactive loop, I decided to keep things simple and focused on the current task. I clear the screen, then show the user the last few entries to the HomeInventory.txt file, if there are any, by reading the file into a list object, and using a set of conditionals to get the proper printing behavior for up to five entries. I made the decision to add a header row upon file first creation (which I will cover later), so we slice the first entry off the list for both conditional tests. Then, we can use len() to see if the remaining list is greater than five items, and print the last five; zero items, and print that there is no data; or in the leftover case of 0-5, just print everything other than the header.

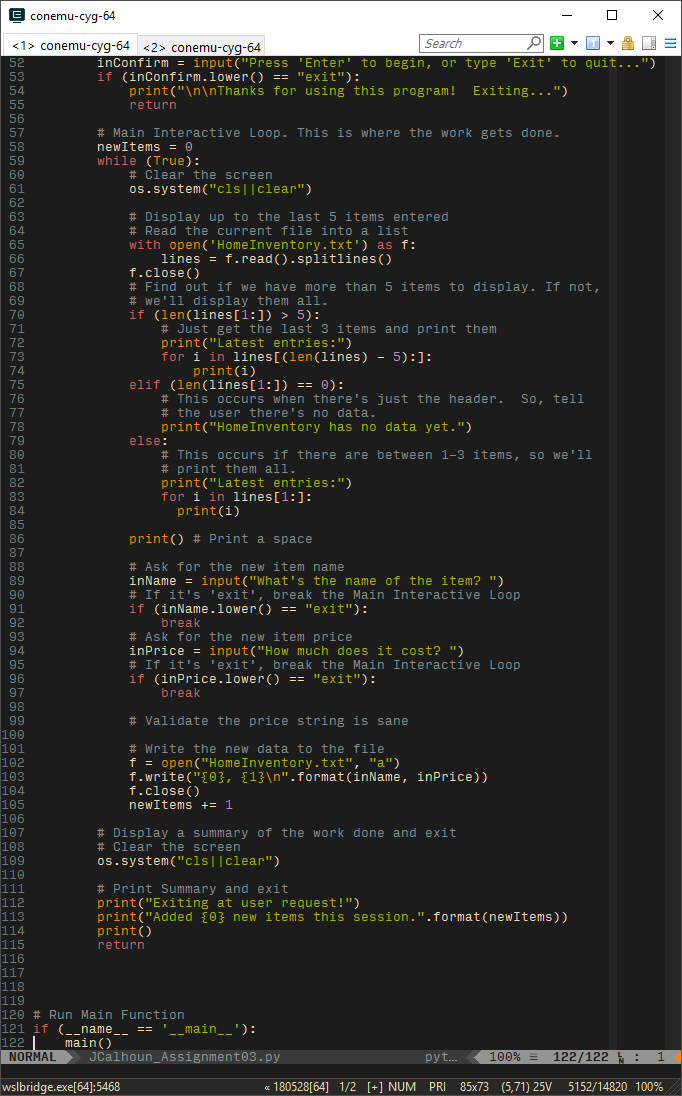


Figure 3: Displaying the latest entries to HomeInventory.txt

As the final part of the main interactive loop, we ask for the name and price. For the name, I really don’t care, the user can call it whatever they like. For the price, though, I want some validation and standardization, so these values might be consumable by another application at some point. To achieve this, I imported the regex module, re, and built a very simple regex object, "^([0-9]+\.[0-9]{2}$|^[0-9]+)$". Read simply: there must be one or more digits followed by a period and then exactly two digits, or one or more digits alone, and nothing else in the string.

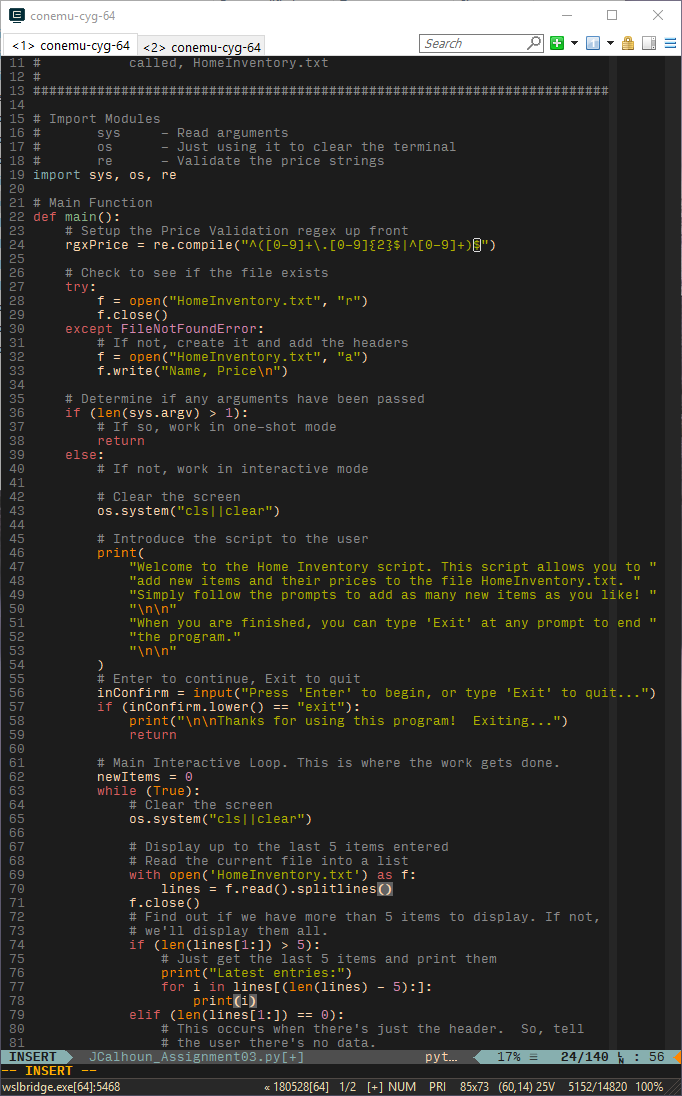


Figure 4: Regex object

Then, once we ask for the price, we can first check that it’s a string we like, and if so, convert it to a float and back to a string formatted with two decimal places. If it’s exit, we exit, and for all other situations, just ask again.



Figure 5: Input and validation

As the final part of the main interactive loop, we open the HomeInventory.txt file, write the value, and close the file. This ensures the user’s data is always cleanly committed and reduces the chances of some outside event interrupting our access to the file and causing lost work. We’ve also been running a count of how many items we’ve entered so far this session, so it’s a good time to increment the counter.



Figure 6: Write the data

Once the user is ready, they can type “exit” at any prompt and the main interactive loop will end. We clear the screen, give a quick summary of the number of items added, and end the program.



Figure 7: Summary and exit

Returning to the top, it was time to fix up the file handling. I wanted the creation of the file to be automatic, with no need for the user to copy or create it when they install/copy the script. I accomplished this with a simple try statement, in which I attempt to open the file, and if it is not there, we’ll create it fresh and insert a simple header of “Name, Price”. As currently set up, the file created and filed by this script is a standard CSV, and can easily be consumed by other applications, or opened directly in a spreadsheet application such as Excel.

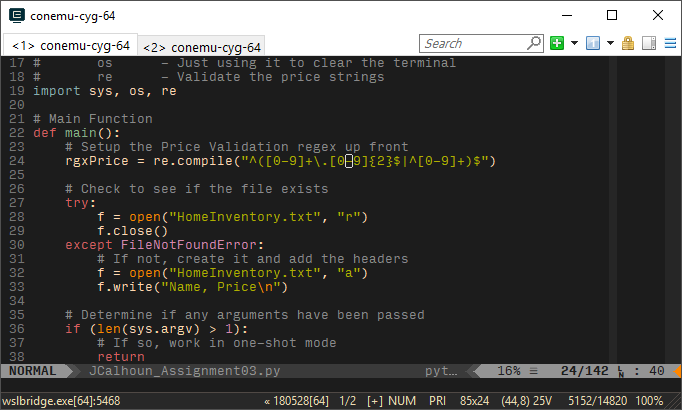


Figure 8: Create the file as needed

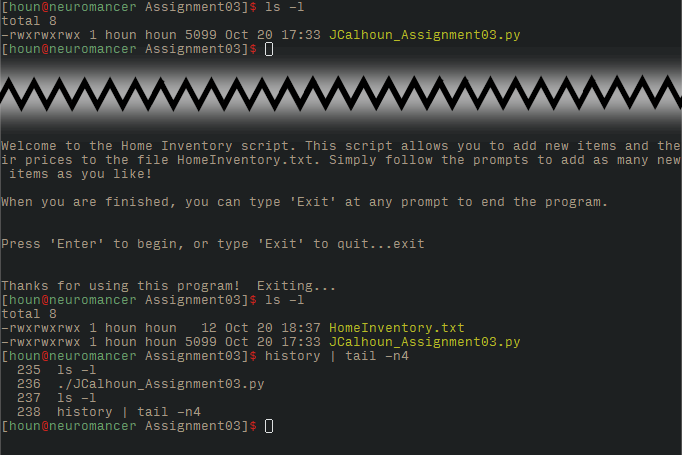


Figure 9: Showing creation of HomeInventory.txt on first run

With interactive mode complete, I returned to the top of the script to flesh out CLI mode. Expanding on the initial check of “are there any arguments at all”, I perform a series of nested if statements. If at any point the input is incorrect, the script will fall through to a print statement with instructions for how this mode is intended to be used. We check to ensure there are only two arguments passed (after the name/path of the script itself), and then we check that the price is in the correct format much the same as in interactive mode. If everything is correct, the file will be opened and the data appended, a short message to the user will be displayed, and the script will exit.



Figure 10: CLI mode code

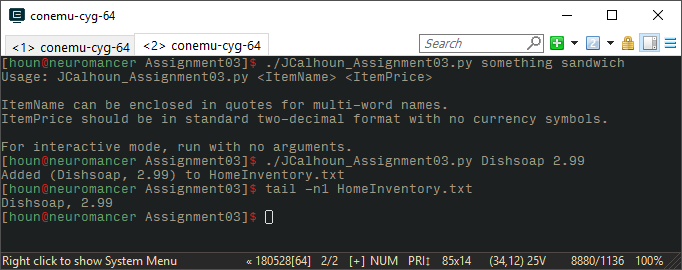


Figure 11:CLI mode in action

At this point, I feel I’m done with the script. Next steps would be to look for ways to modularize certain bits of repeated code into their own functions, but I’m not certain that there would be much gain for the current design and logic. Were this to be expanded with additional features and data, though, there would be a far stronger case. For this task, as defined, though, what I’ve got should provide a good user experience in either mode.