Nathaniel Smith

May 4, 2020

Foundations of Programming: Python (IT FDN 100 A Sp 20)

Assignment03

Python: Writing to a File

# Introduction

In this paper I discuss the process taken to write the code for my script called HomeInventory.py. The script asks the user to enter the name of a household item, then it asks the user to provide the estimated value, it saves the input data to a text file, and then prints to the screen that the data was saved.

# Script Header

As with writing all scripts, I started HomeInventory.py with a script header, see lines 1 through 12 in Figure 1. The purpose of the script header is to describe what the script does and when it was created. And just as I have done, it’s also a good place to store the change log for the script.

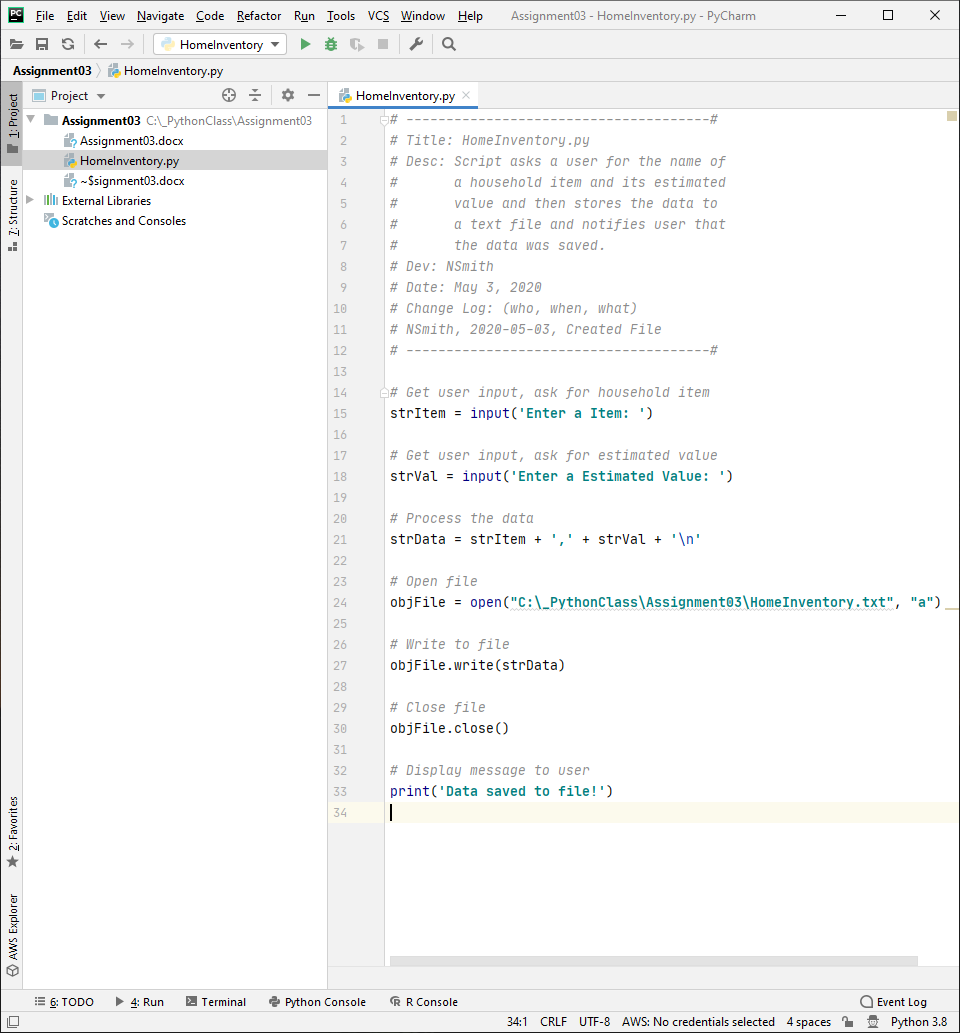


Figure 1. HomeInventory.py Script

# Pseudocode

Next, I wrote the pseudocode for my script. Pseudocode is a way of writing the code in text form. It is sort of an outline for the code. The pseudocode is not the final code, instead it is used to aid in writing the code. The pseudocode should be written so that it is easy to follow and include all the steps that are needed for the code to work. Without it, it can make writing the code hard to do. After writing the pseudocode, I was ready to write my Python script. However, I could have just as easily used my pseudocode to write the same program in any programming language. My pseudocode can be read in Figure 1. HomeInventory.py ScriptFigure 1; it includes all the lines that are commented out in my script file excluding the header script lines.

# The Code

Following in the order of my pseudocode I first needed to ask the user for input. First, I asked the user to “Enter a Item”. Next, I asked the user to “Enter a Estimated value” (notice that in both cases I used the letter ‘a’ instead of ‘an’ because I was following the assignment instructions). Both of these tasks were accomplished by using the input() function. I also used the assignment operator, =, to assign the values entered by the user to the variables strItem and strVal. See lines 15 and 18 of HomeInventory.py in Figure 1.

After getting the data from the user, I processed the data into a new string, strData to include both variables strItem and strVal separated by a comma. I used the assignment operator, =, and the concatenate operator, +, to achieve this. See the assignment statement in line 21 of HomeInventory.py in Figure 1. Note that I used \n at the end of the concatenated string. I did this so that if the script runs more than once, the new data will be added to the next line rather than being appended to the first line of text.

In this code I also used a function that I had not used before, the open() function. In line 24 I used the open() function to create and open a text file, HomeInventory.txt, and combining this with the assignment operator the file object was assigned to the variable objFile. Notice that I used an “a” inside the parenthesis of the open() function. This opens the file for writing and will append any data to the file. I could have chosen some other options as shown in Figure 2 which I found on the Python docs website. (<https://docs.python.org/3/library/functions.html#open>) (external site).

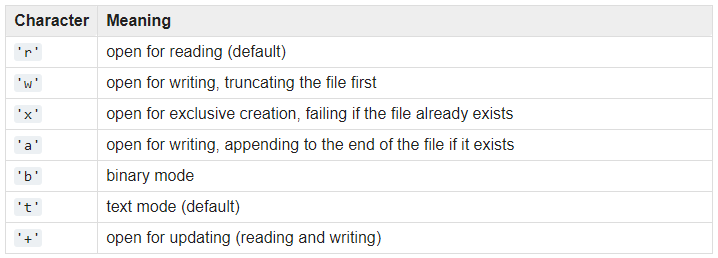


Figure 2. Character Table for open() Function

I was curious about the class type of the variable objFile so during the testing of the script I added the following statement just after the objFile assignment (the statement is not included in my final script and thus is not shown in Figure 1).

print(type(objFile))

Python returned the following.

<class '\_io.TextIOWrapper'>

I am unfamiliar with this class type as it is different from any of the class types I have seen thus far. I will need to do more research on this, but what I have learned so far from the Python docs website is that this class type uses text mode (strings) as opposed to binary mode (0’s and 1’s). I do not yet understand what this means in practice.

Now that I have received data for the user, processed the data, and opened the file, it’s time to write the data to file. This is done with the write() function using dot notation i.e. there is a dot, ., between objFile and write(). See line 27 of the script in Figure 1. The write() function takes the value of strData and writes it to the object file objFile.

The next step is to close the file. This is done using the close() function which is very similar to the write() function in that it also uses dot notation. The close() function however closes the file so that it will no longer be open and writeable. See line 30 of Figure 1.

The final step is to display a message to the user that the data was saved using the print() function. See line 33 of Figure 1.

# Testing the Script

I first ran the script using the Interactive Development Environment (IDE) software from JetBrains called PyCharm. The result is shown in Figure 3. The code works as intended!

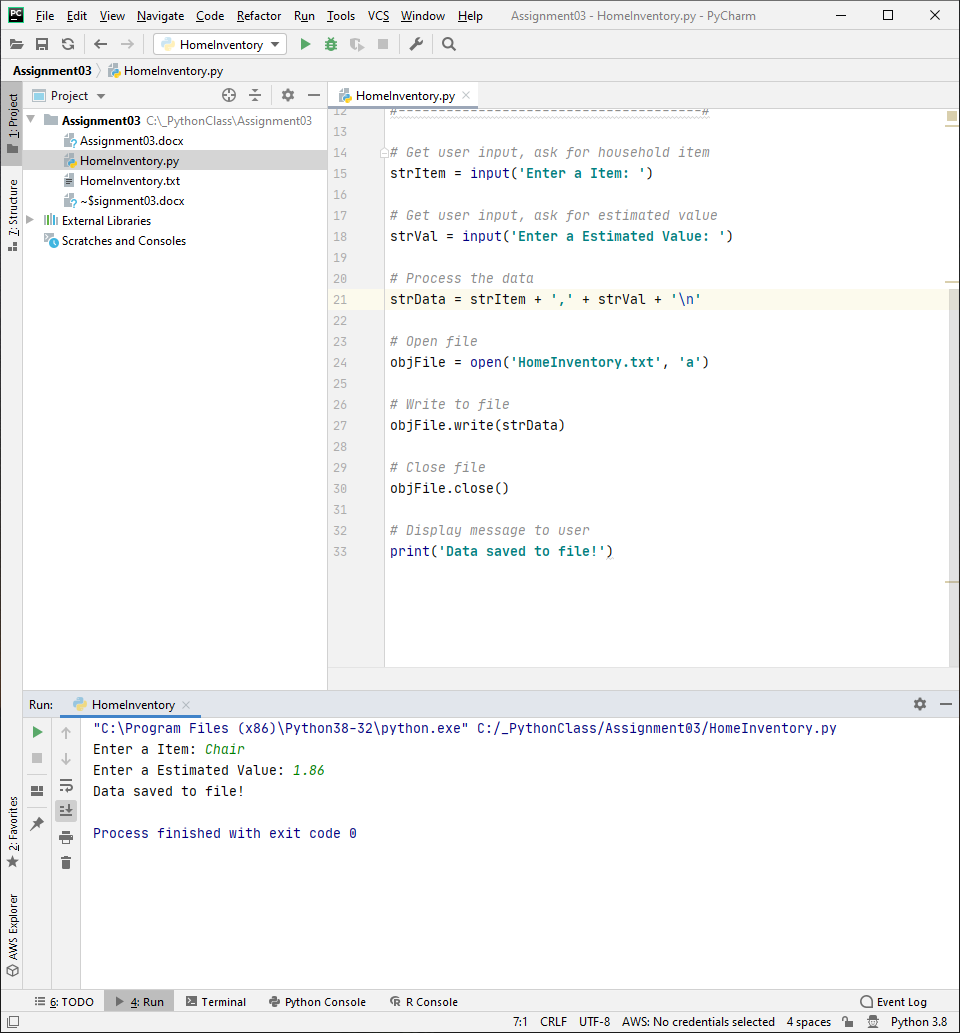


Figure 3. Running HomeInventory.py in PyCharm

Now that I had it running in PyCharm, I also tested my script from Windows Command Prompt as shown in Figure 4. Success! Only thing to check now is the text file.

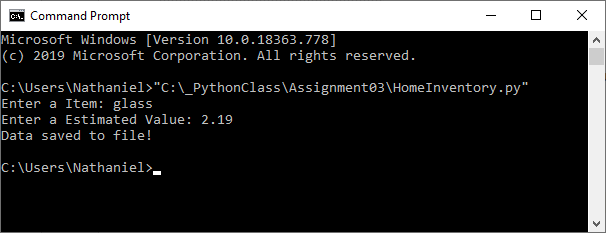


Figure 4. Running HomeInventory.py in Windows Command Prompt

The final goal of running the HomeInventory.py script is to write data to a text file. After running the script in PyCharm and then running the script in Command Prompt I ended up with the text file shown in Figure 5. Complete success!

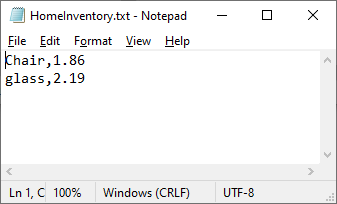


Figure 5. HomeInventory.txt Opened in Notepad

# Summary

In summary, I wrote a script that asks a user to enter an item, enter an estimated value for the item, and saves that data to a file. In this paper I discussed the functions and operators used to achieve this and provided the results of running the script. The script could easily be improved by adding a while loop so that the user is repeatedly asked for household items and values so that a list of items could be easily created. Currently, the script would require the user to rerun the script to add another item which would be quite laborious if adding many items. Another improvement would be to use conditional statements to check that the user entered text and did not accidentally hit the enter key, because as it is written now the user could hit the enter key twice and the program would end without getting any useful data.