George Kim

January 29, 2019

Foundations of Programming: Python

Assignment 04

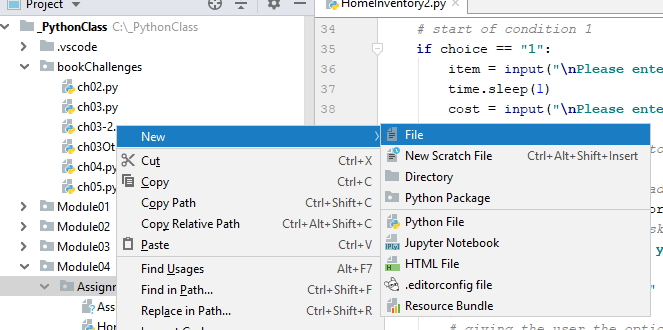
Home Inventory Incorporating Tuples

# Introduction

In this assignment, we are building on our previous assignment of the Home Inventory python script. We are asking the user for new entries and storing that data in a 2-dimensional Tuple. We then ask the user if they would like to save the data or not. This is a stepping stone to create our own application, and I will walk you through my thought process of how I accomplished this project.

**Starting the Script**

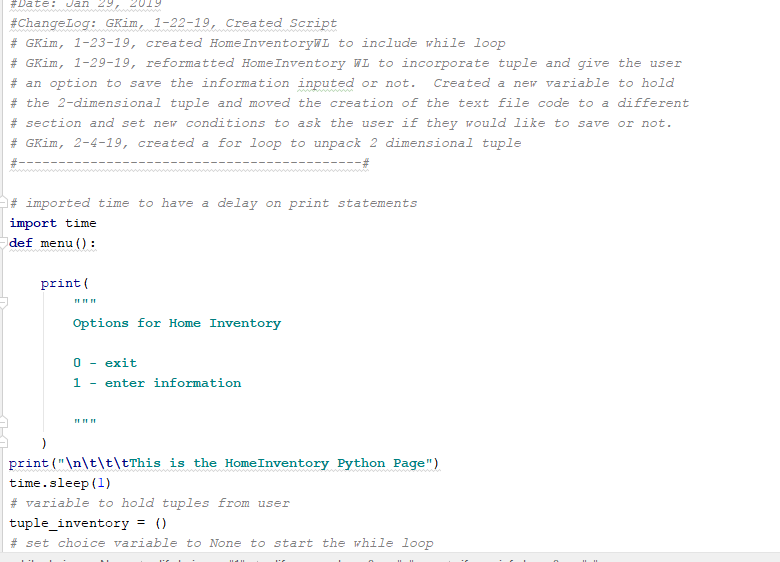
I began this project like we did in the last assignment(**Figure 1**). Download the Module 4 zip file and save it to \_PythonClass. We then start a new project on PyCharm, Assignment 4, and name our python file. I chose to name mine HomeInventory2.py. Now since this project builds off the last assignment, there isn’t that much to do but I did make some adjustments to account for some errors. I copied my last code and pasted it on the new python file.



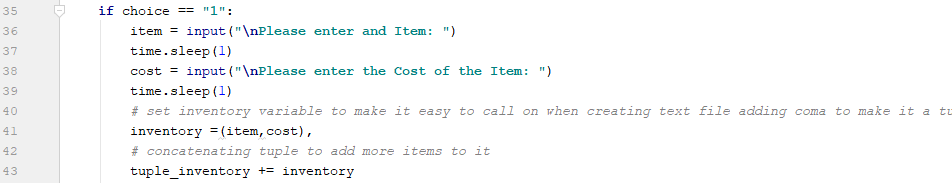
***Figure 1. Creating a file***

**Additions to Previous Script**

As you can see in **Figure 2**, our code starts off the same as the last assignment that had the while loop. The difference in the few lines of code as compared to the last assignment is that there is a new variable, *tuple\_inventory = ()*. This is the introduction to Tuples, which are *immutable.* That means that they can not be changed. This can make things challenging in what you can and can’t do to Tuples. In this case, it is an easy addition to the previous assignment as I will show you if **Figure 3.**

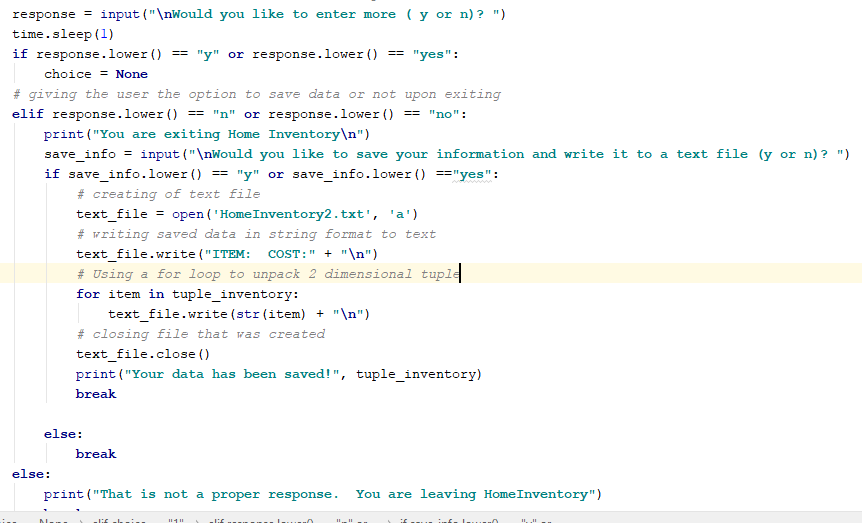


***Figure 2. The addition of the Tuple variable***



***Figure 3. Concatenating the Tuple, and making sure there is a coma in the inventory variable***

In order to to save multiple pieces of data that is given to us by the user, we have to concatenate the tuple with the concatenating operator , + (**Figure 3**). Remember, tuples are immutable, but we can augment it and create a **brand new tuple** with each new *item* and *cost* (saved in *inventory* variable) that we want to add.

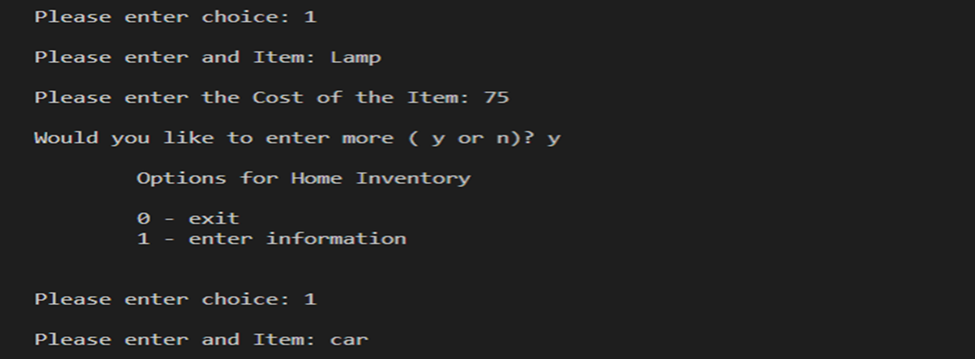


***Figure 4. Created another addition to response moved text file creation. Also a for loop is created to unpack the tuple***

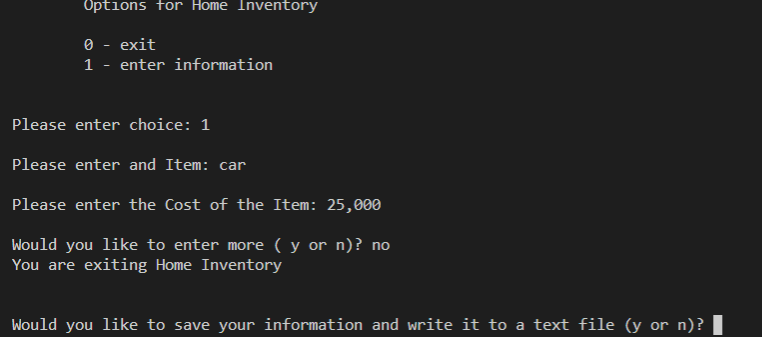
I realized from the previous assignment that the user could enter “y” or “yes” and I wanted to correct for that. In the conditional from the “if” statement, I wrote: “response.lower() == “y” or response.lower() == “yes” ”, which is going to capture the input from the user , whether they use capital letters or not (**Figure 4**). This “if” statement lower cases the input and has python look for these specific inputs or “y” or “yes” to go back into the loop to add more data. You will see also that I did the same thing for “no” and I moved the text command codes to this section of the script. What the assignment asked for was upon leaving the program after the user had inputted the data, if they would like to save and write to the text file. I nested the input code response, for “no”, under the options (if statement) to go back into the loop. Putting the codes for the text file here, will get the data input from the tuple and write it to the text file. I included a header also of ITEM and COST to give the user and any other user an idea of what each tuple will contain. I also included a “for loop” to unpack the 2 dimensional tuple and write on a new line on the text document. It took me a little practice and brute force to get this to work, but I was able to get the item in the tuple and then write it to new lines on the text file. I iterated through the tuple using the for loop to write it to the text file, I then put the *text\_file.close()* out of the “for loop” to close it when the iteration was done. This ensured to capture all the data in the text file.

**Output of the Script**

The output works great and you can see the items being saved and written to the text file like the last assignment (**Figure 5 and 6**). I let the loop run a couple of times and enter “no” to exit the loop and then “yes” to save and write the file to a text doc.

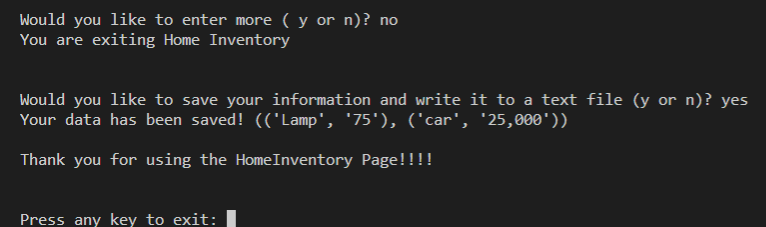


***Figure 5. Output of first loop***

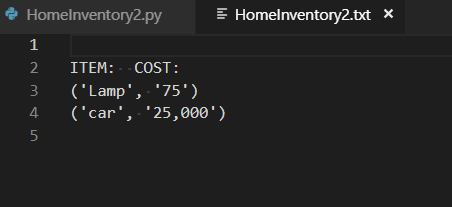


***Figure 6. Output of second loop***

In **Figure 7** below, the next steps of the program run, and shows the saved data that will be written to the text doc in **Figure 8.**



***Figure 7. Shows saved information to the user***



***Figure 8. Text document with saved data which is unpacked from the tuple for loop.***

**Summary**

Going through the steps of writing our application is very beneficial. I can see how adding and moving code gives you practice, but you also see a better way of writing your code and making the necessary adjustments. I found a few errors in my code and were able to fix them with the “y” or “yes” inputs. Getting specific in your code and seeing potential errors early on will make the process of developing an application a bit easier in the future. Just a bit more practice like this, and going in the step by step process of writing code will help develop and improve on those skills.