Nathaniel Smith

May 11, 2020

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

Assignment04

Python: Script with Persistent Menu

# Introduction

In this paper I discuss the process taken to write the code for my script called HomeInventory.py. This has the same name as the script from the previous assignment, but this version is an improved version that provides the user with a menu that has three options: 1) add data to a list, 2) display current data, and 3) exit and save to file. This script makes use of while loops, for loops, if-elif-else statements, and lists. I will carefully explain the usage of each in my code throughout this paper.

# Script Header

As with writing all scripts, I started HomeInventory.py with a script header, see lines 1 through 16 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. From the change log you can see that on May 10 I changed a variable name in my code from list to listTable. I made the variable name change because list() is one of Python’s built-in functions and I did not want to create any confusion in the future by reusing a built-in name as a variable name. Although my code worked using list as a variable name; it is not good practice to do so.

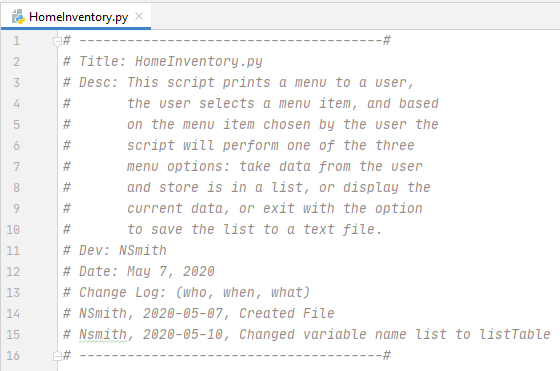


Figure 1. HomeInventory.py Script Header

# Pseudocode

Next, I wrote the pseudocode for my script (well not really, it was provided by the instructor, Randal Root). 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. Good pseudocode can be used to write code in any programming language not just Python. Below is the pseudocode for this assignment.

*# Step 1 # Display a menu of choices to the user*

*# ("Add Data to List", "Display Current Data", "Exit and Save to File") # Step 2 # Add a new item to the List(Table) each time the user makes that choice*

*# Step 3 # Display the data in the List(Table) each time the user makes that choice*

*# Step 4 # Exit the program and save the data to a text file when the user makes that choice*

# The Code

The code is shown in Figure 2 with the script header omitted. In the following sections of this paper I will explain the code in groups based on the four pseudocode steps mentioned above. Often, in my explanations I will reference a numbered line of code; these numbers can be found on the left column of Figure 2.

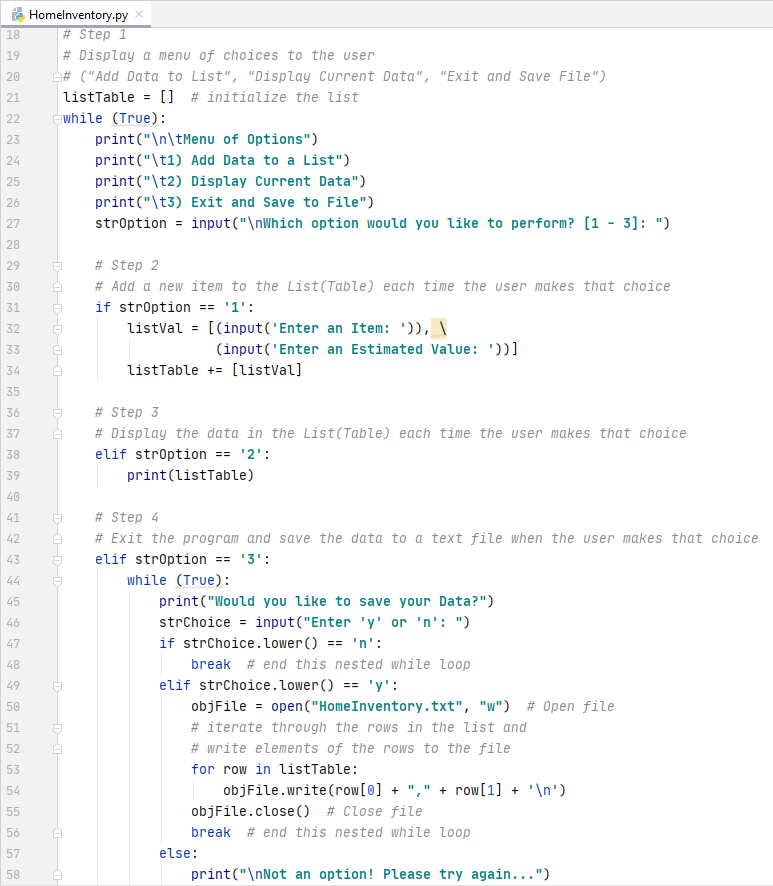


Figure 2. HomeInventory.py Script

# The Code: Step 1

In step one I initialized the list that will be used to store the data provided by the user with the following statement:

listTable = []

listTable will be updated each time the user is prompted to input data. Later I will explain why I had to initialize listTable at the beginning of the script.

The next statement is a while loop statement that is always true i.e. it is an infinite loop. This was achieved with the following statement:

while (True):

The rest of the script resides within the body of this while loop; notice that all the code below this while statement is indented. The indentation is required and it ‘tells’ Python that the code belongs in the body of the while loop.

The next four lines of code (lines 23-26) print the menu of options. The code is shown below.

print("\n\tMenu of Options")

print("\t1) Add Data to a List")

print("\t2) Display Current Data")

print("\t3) Exit and Save to File")

There is not much going on here. These are simple print statements. The use of \n results in a newline, and the use of \t results in a tab. The default end option for a print statement is a newline, so each of these print statements results in a separate line of text on the screen.

Line 27 prompts the user to choose one of the menu options. The user input is stored into the variable strOption. The line of code is shown below.

strOption = input("\nWhich option would you like to perform? [1 - 3]: ")

This is the completion of pseudocode step 1. Seven lines of code in total were used to provide a list of menu option to the user (only five lines if you ignore the list initialization and the while statement as the menu could be printed without them, but it would be very difficult to get the rest of the code to work if they were missing).

# The Code: Step 2

Step 2 asks the user for input each time option 1 is chosen from the menu of options. The user will be asked for two inputs that are stored in a list. If option 1 is not chosen then this step will not execute. The code for step 2 is shown below.

if strOption == '1':

listVal = [(input('Enter an Item: ')), \

(input('Enter an Estimated Value: '))]

listTable += [listVal]

The code starts with an if statement. The if statement is checking if the value of strOption (this variable is storing the menu option value entered by the user in step 1) is equal to the string ‘1’. If True, the block of code in the if statement will execute.

The user will be asked to ‘Enter an Item’ and then to ‘Enter an Estimated Value’. These two inputs will be stored as two elements of the list variable listVal.

listTable is then augmented with the list stored in listVal and the result is a 2-dimensional list. Each time this block of code executes, the list stored in listVal will be added to listTable. In this way, listTable will keep a running tab of the all items and values the user inputs. Earlier I mentioned that I initialized listTable at the beginning of the script for a reason yet to be explained. Well if it was initialized inside the while loop it would be set to an empty list each time the while loop executed. This means that the data entered in previous iterations of the while loop would be set to an empty string right before step 2 is executed and therefore listTable would not keep a running tab of the items input by the user. listTable had to be initialized outside of the while loop in order for this to work as intended (there is likely another way to do this with an if statement within the while loop, but this seemed the simplest way to me).

# The Code: Step 3

Step 3 was the easiest part of code to write of all the steps in this script. It is only two lines long and it is simple to understand; if strOption is equal to ‘2’ then the computer will print to screen the contents of list listTable, e.g. if the user selected option 2 from the menu of options in step 1, an inventory list will be printed to screen. The code is shown below.

elif strOption == '2':

print(listTable)

This section of code starts with the elif statement. This elif statement is part of a larger if-elif-elif-else statement, all of which are indented the same amount as seen in lines 31, 38, 43, and 68 in Figure 2. The block of code below the elif statement, in this case just one print statement, will execute if the variable strOption is equal to the string ‘2’. This is true if the user chooses option 2 from the menu of options. If the user has indeed chosen option 2, the script will print the list listTable to the screen. An example of printing the listTable is shown Figure 6.

If the user has not yet input any data, i.e. the user has yet to choose option 1, an empty list represented by a pair of left and right square brackets will be printed to the screen, see Figure 3. This occurs because listTable was initialized to the empty list at the beginning of the script.



Figure 3. Empty List Printed to Screen

# The Code: Step 4

This is the final section of code and it was the most difficult to write. It includes the final elif and the else statements of the larger if-elif-elif-else statement starting on line 31 of the script. In addition, this step includes a while loop, an if-elif-else loop, and a for loop. It comprises of 17 lines of code.

Let’s take a closer look at what the code does. In step 4, the script checks if the user entered ‘3’ at the menu of options, and if so, will ask the user if the data should be saved to file before exiting the script. The user can save the data to file by entering ‘y’ for yes at the prompt (I also added some code to accept ‘Y’ for yes). Or, the script can exit without saving data if the user enters ‘n’ for no (I also added some code to accept ‘N’ for no). If the script makes it to step 4 of the code it means the user did not enter either a ‘1’ or a ‘2’ at the menu prompt i.e. neither the if statement nor the first elif statement evaluated to True. If for some reason the user did not enter either a ‘1’, ‘2’, or ‘3’, no worries, the script will execute the else statement which notifies the user that a valid menu option was not chosen and the user will be presented with the menu screen again.

Below is the code for step 4 with comments for easy reading.

elif strOption == '3':

while (True):

print("Would you like to save your Data?")

strChoice = input("Enter 'y' or 'n': ")

if strChoice.lower() == 'n':

break # end this nested while loop

elif strChoice.lower() == 'y':

objFile = open("HomeInventory.txt", "w") # Open file

# iterate through the rows in the list and

# write elements of the rows to the file

for row in listTable:

objFile.write(row[0] + "," + row[1] + '\n')

objFile.close() # Close file

break # end this nested while loop

else:

print("\nNot an option! Please try again...")

break # end the while loop, i.e. end the script

else:

print("\nNot an option! Please try again...")

Step 4 begins with an elif statement to check if the user has entered ‘3’. If the user has entered ‘3’ the elif expression is True, and an infinite while loop will execute. The body of the while loop contains code that prints “Would you like to save your data?” using the print() function and prompts the user for input with “Enter ‘y’ or ‘n’: ” using the input() function. The input from the user is stored in the variable strChoice.

Now we are at line 47 of the code and the beginning of an if-elif-else statement. The if statement will check if the user entered ‘n’ (or ‘N’), the elif statement will check if the user entered ‘y’ (or ‘Y’), and if neither of those expressions evaluate to True the script will move to the else statement at line 57 and will print “Not an option!, Please try again…” and the script will return to the while loop at line 44 and present the user with the option to choose ‘y’ or ‘n’ again before exiting the program.

Let’s dive deeper into this block of code. The expression of the if statement checks if the user entered ‘n’. But because the expression uses the string method lower(), the expression will also evaluate to True if the user enters ‘N’. If True, the break statement executes on line 48, which terminates the current while loop starting on line 44 and moves to the next statement which is another break statement on line 59. The break statement on line 59 terminates the while loop starting at line 22 and thus exits the script without saving data to file.

If the user has not entered ‘n’ or ‘N’ the script will move to the elif statement on line 49. Here it check if the user entered either ‘y’ or ‘Y” because again I used the lower() method. If the expression evaluates to True, a file with the name HomeInventory.txt will be opened for writing as indicated by the “w” option in the parentheses of the open() function and this is stored in the object file variable objFile.

The next two lines of code, lines 53 and 54, contain a for loop. The for loop will iterate for each element in the list listTable. Each element in listTable is another list that contains two elements representing an item and estimated value entered by the user in step 2. In other words, the for loop will iterate the loop for each list in listTable, and for each iteration a line will be written to the object file objFile. Each line written to file will include an item followed by a comma followed by the estimated value followed by a newline so that each pair of item and estimated value will be written on a separate line of the text file. See line 54 of the code in Figure 2 for the syntax.

Here are some notes on the notation used on line 54. Notice that the string being written to file is calling the variables row[0] and row[1]. row contains 2 elements: item and estimated value. Item, being the first stored element of the sequence, is located in index position 0, and is referenced by the subscript 0 in brackets. The estimated value, being the second stored element of the sequence, is stored in index position 1, and is referenced by the subscript 1 in brackets.

After the for loop has iterated through all rows i.e. all lists within listTable, objFile is closed using the built-in method close(). This is followed by the break statement at line 56 which terminates the while loop starting at line 44. The next statement to execute is another break statement at line 59 which terminates the while loop starting at line 22 and the script exits.

# 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 4. The code works as intended!

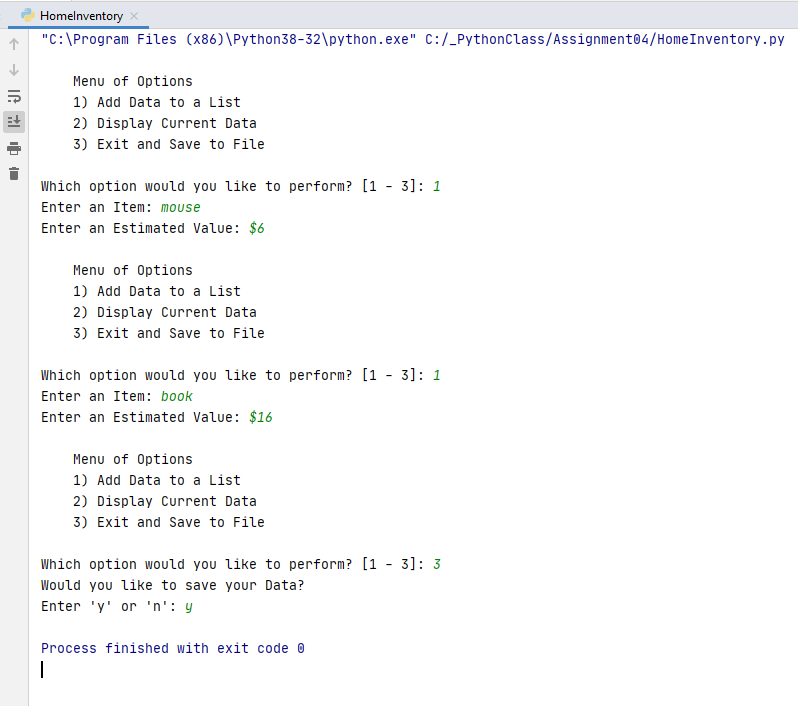


Figure 4. Running HomeInventory.py in PyCharm

I checked the text file after running the code in PyCharm and it did work, see Figure 5.

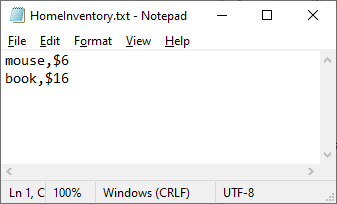


Figure 5. HomeInventory.txt After Running Script in PyCharm

Now that I had it running in PyCharm, I also tested my script from Windows Command Prompt as shown in Figure 6. Success!

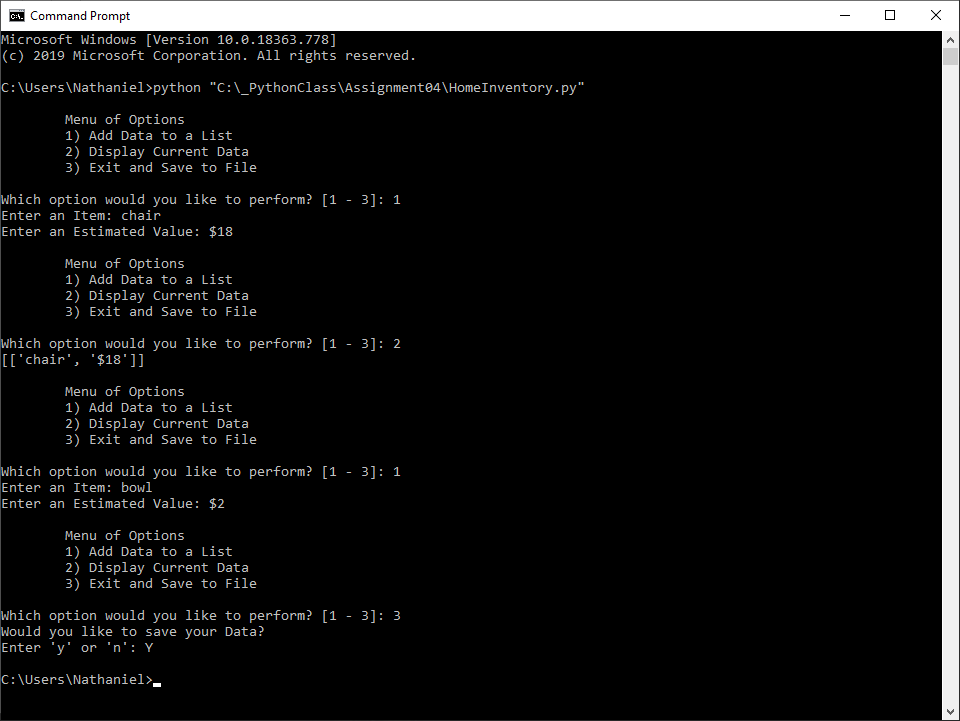


Figure 6. Running HomeInventory.py in Windows Command Prompt

After running the script in Command Prompt, I ended up with the text file shown in Figure 7. Complete success!

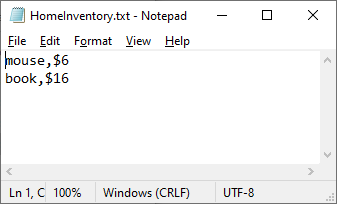


Figure 7. HomeInventory.txt After Running Script in Command Prompt

# Summary

In summary, I wrote a script that provides a user with a menu of three options. The user chooses an option from the menu, and based on the option chosen by the user, the script will either run code that allows the user to add data to a list, display the current data in the list, or exit the script with an option to write the data in the list to a file. I have also been successful in writing the script so that it does not error if the user chooses an option that is not available; instead the script will notify the user of the poor choice and allow the user to reenter a choice. The hardest part of writing this script was trying to remember where I was in the script because of the 5 indents. The pseudocode did its job of keeping me on track while writing the code. I can see now that as code gets more complicated, it is essential that good pseudocode is written before starting to code.