Cindy Li

Nov 19, 2021

IT FDN 100

Assignment 6

<https://github.com/cindy-x-li/IntroToProg-Python-Mod07>

# Error Handling and Pickling Demonstrations

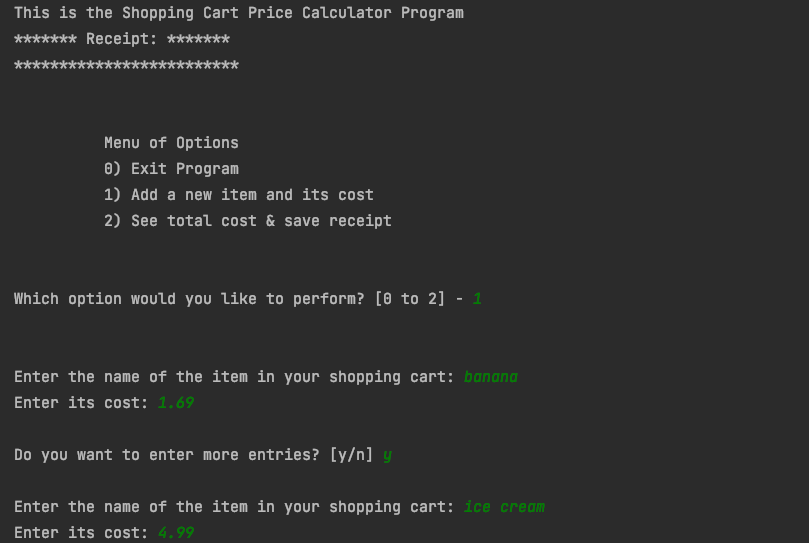
## I. Introduction

The seventh assignment is to research error handling and pickling works and then create programming script that demonstrates these concepts. My response to the assignment is to create a program called the shopping cart price calculator. This program allows the user to input an item name and its cost as dictionary rows into a table/list of data. Afterwards it pickles the data, saves it to a binary file, unpickles it and presents it back to the user.

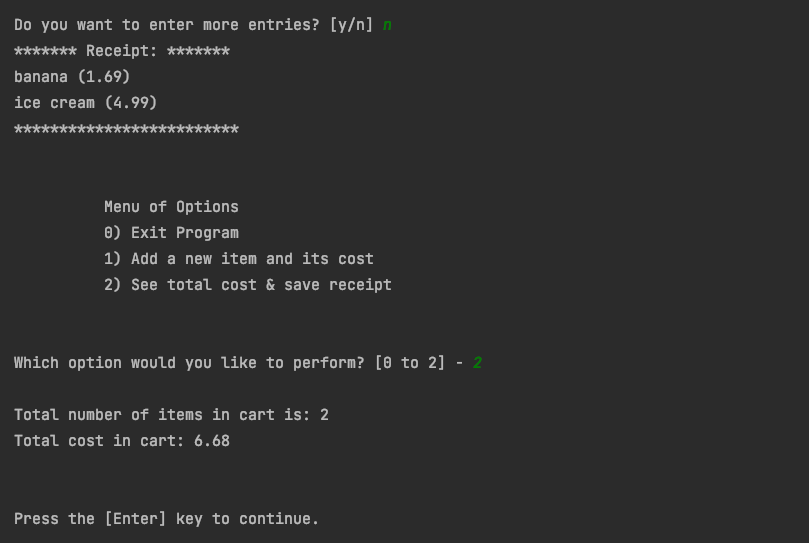
## II. Error Handling

When creating a program with user interactions, there are often times where the user can enter in a variable that causes the program to run into an error and stop working. Python will display an error message, which is also known as raising an exception, to allow the developer to understand what lines caused the program to crash. However, this error message provides little meaning to the user and s/he does not know where they went wrong. Error handling is when the developer anticipates the situations where the user’s entry will raise an exception and handle it in a way that is useful to the users of the program.

The shopping cart price calculator program uses error handling to evaluate the user’s input of the shopping item and its cost prior to adding it to the table of data, also known as the receipt. First, let’s look at the case where the user successfully enters the name of an item and its cost via prompts in Figure 2.1 and then the receipt is generated in Figure 2.2 with the entries.

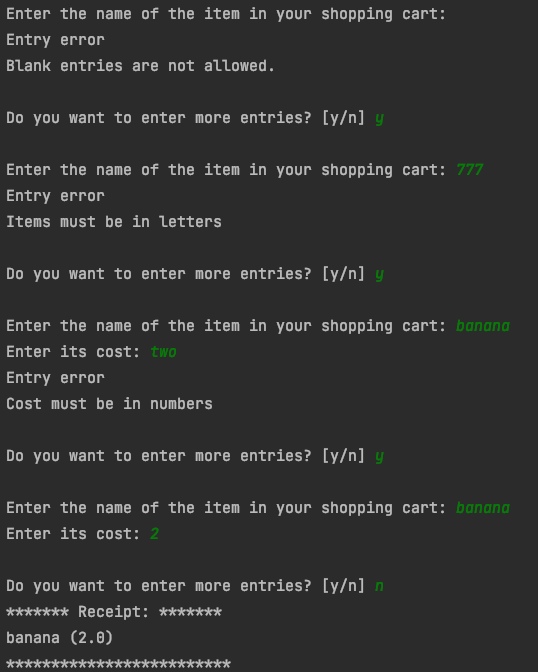


***Figure 2.1: Entering inputs into the program***



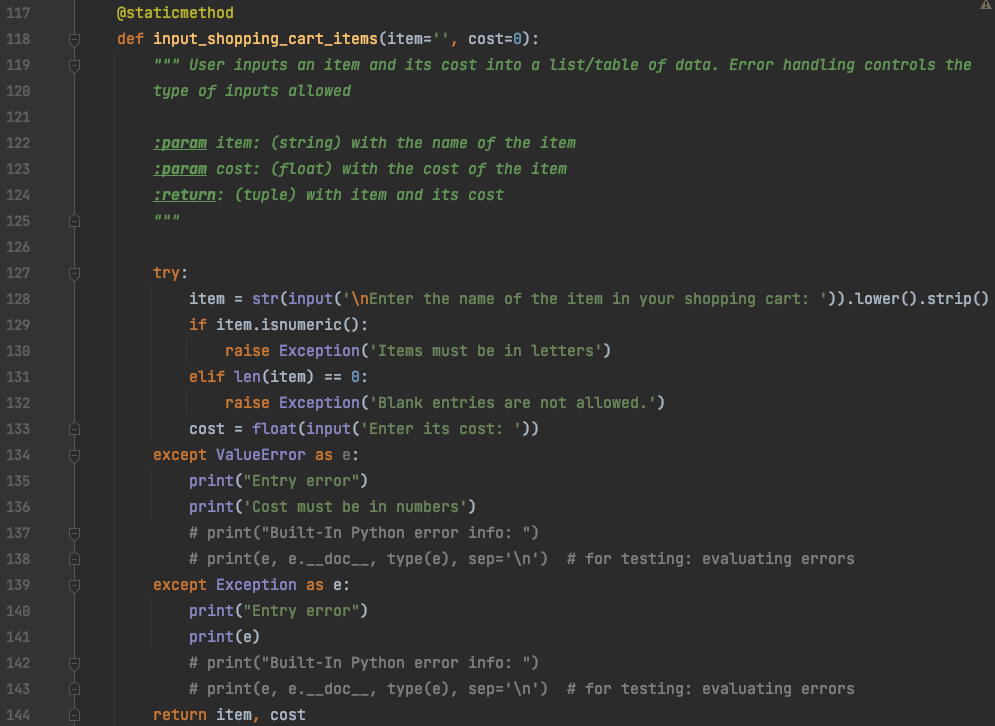
***Figure 2.2: Entries successfully added and a receipt is generated***

Now, let us look at an example, in Figure 2.3, in which the user is entering entries that are not accepted by the program and encounter entry errors by entering a blank entry, using numbers instead of letters for the name of the item and writing out the cost in letters than using numbers. These are three different situations, yet the program is able to trap these exceptions and provide an error message that is understandable to the user. Finally, the user understands the rules and adds an entry to the table.



***Figure 2.3: Error handling***

The code to perform error handling of user inputs for the shopping cart occurs in the def\_input\_shopping\_cart\_items function (Figure 2.4). For the item name input, two custom exceptions are raised for when the user enters numbers instead of letters and when there is a blank entry (lines 129 to 132). A custom exception is created for each case, because there is not a specific Python exception type. Nonetheless these entries are undesirable and cannot be included in the receipt. For the cost input, the input is float type, so when letters are entered instead it would cause a ValueError, a specific Exception type. As a result, this error can be handled using the except clause with ValueError, instead of raising a custom Exception.

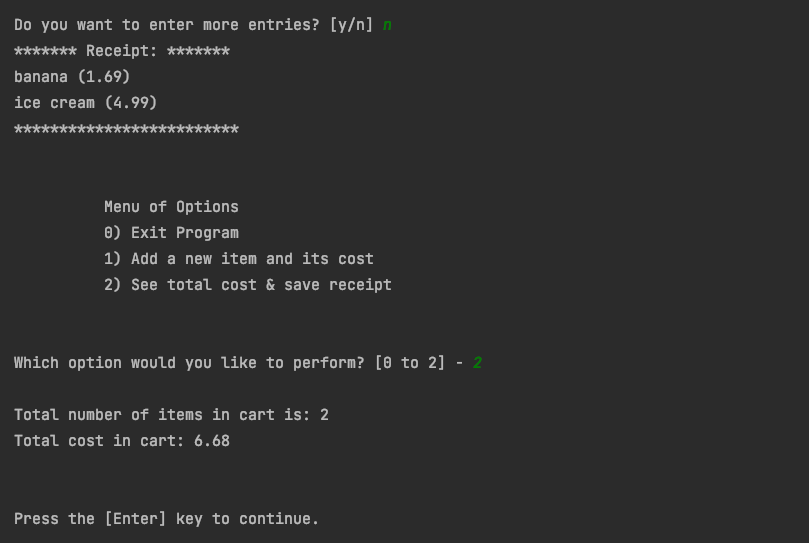


***Figure 2.4: The shopping cart items function within the IO class***

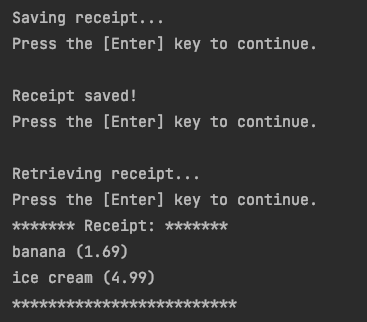
## III. Pickling

After the user finish putting in entries and the receipt is generated, the data is preserved in a binary file via processes known as pickling and unpickling. These processes are also called serialization and deserialization. It is faster for the computer to write, read and append to binary files, because this information is saved to the computer’s native language of 1s and 0s.

For this program, when option two is selected, the user first sees information generated from the data entry (Figure 3.1) and then the data is saved and retrieved (Figure 3.2). In Figure 3.2, each of the three phases is broken up by an enter key in order for the user to see what is happening to the data. Finally, the retrieved data is displayed in the same table prior to saving it.

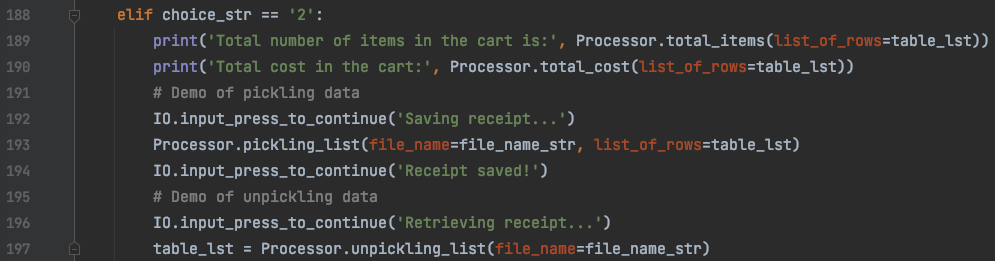


***Figure 3.1: UI for the total cost calculator***



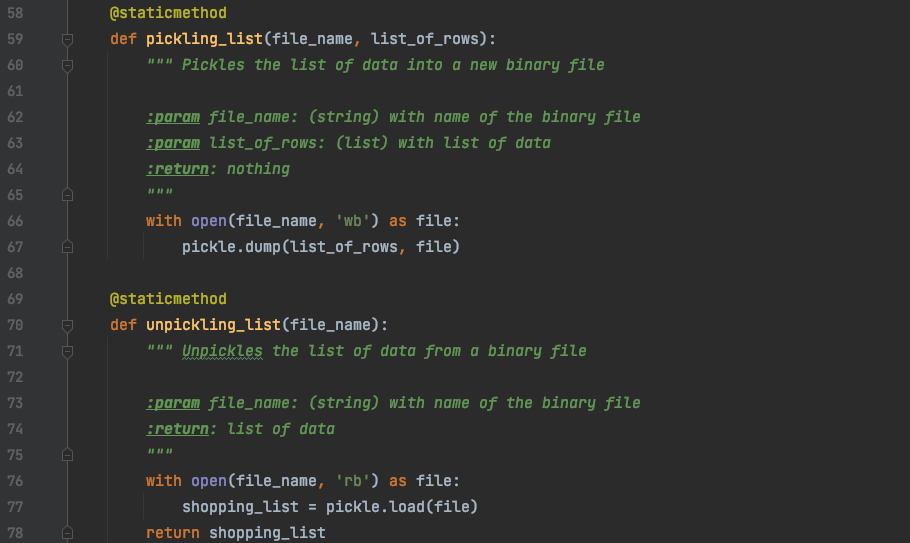
***Figure 3.2: The shopping cart items functions***

Figure 3.3 shows what the main program looks like when option 2 from the menu is selected.



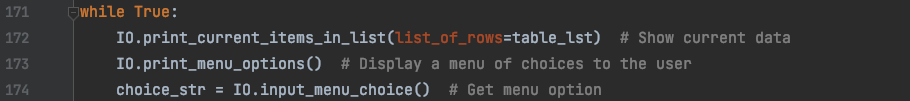
***Figure 3.3: Option 2 in the Main Program***

In order to performing pickling and unpickling, the pickle module must be imported first into the file. This allows the various functions of the modules to be used within the script. The specific functions for performing the pickling and unpickling occurs in the Processor class (Figure 3.4). Pickling data function requires two parameters: the name of the binary file where the data will be saved and the list of data. As seen in Figure 3.4, in lines 66-67, when these arguments are passed in, the file is opened using the keyword “wb” for write binary, and the table of data is “dumped” into the file via the pickle.dump() function. The with keyboard opens and closes the file. To retrieve the data, one opens the file where the data is stored, using the keyword “rb” for reading binary, and use the pickle.load function. The list of data is captured via the local variable, shopping\_list (76-77).

****

***Figure 3.4: Pickling and unpickling functions***

In the main program, the unpickling\_list custom function returns a list that is captured by the global variable, table\_list (Figure 3.3, line 197) and the list is printed into the receipt table at the start of the while loop (Figure 3.5, line 172). I specifically saved all the data into a single table, because the pickle functions can only load and unload one list at a time. So in order to avoid confusion on the number of dump and load functions that has to be written to access all the data, one single list was created to capture that information.



***Figure 3.5: Script for the initial user interface***

Once the user exits the program, s/he can view the binary file, “ShoppingCart.dat” and see that the saved entries are obscured (Figure 3.6).

## Macintosh HD:private:var:folders:yw:13gbt77n1hs35zwm545dn46h0000gn:T:TemporaryItems:Screen Shot 2021-11-27 at 5.25.44 PM.png

***Figure 3.6: Binary file entry***

## VI: Summary

The shopping cart price calculator demonstrates error handling via the user inputs and pickling by saving and retrieving the receipt, a table of dictionary rows of data. I learned two things from this assignment. First, it is difficult to imagine all the possible errors a user can create and capture it. Second, there are some limitations to using the pickle module. As a result, in creating a custom demonstration, I had the freedom to design the program to avoid pickling multiple lists.