Aaron Lanphear

March 1 2021

IT FDN 110 B Wi 21: Foundations of Programming: Python

Assignment 7

<https://aaroncl13.github.io/IntroToProg-Python-Mod07/>

# Pickling and Error Handling

## Introduction

This week we learned about pickling and exception error handling. This assignment was a little more freeform, in that we could create what we wanted to demonstrate how these things work.

## Pickling

Pickling is a way for us to store data in binary format, thus saving space on our computers. If the file were to be opened by a human on the hard drive, the majority of text would like gibberish. However, the computer knows how to read it. We can create lists, or dictionaries, store them in a binary file and pull them back into the program.

## Error Handling

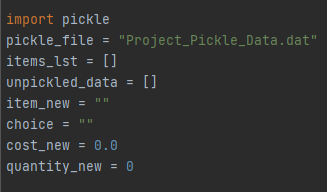
Error handling allows to catch errors that would normally derail our program. Doing this allows us to do two important things. First, we can present the issue back to the user in a user-friendly format; something that makes more sense than what Python would normally give them. Second, it keeps our program running. Without the error handling, the program would break and cease it’s operation.

## My Code – The Project Budget Tracker

To demonstrate both of these, I will walk you through the script I created this week. This is a simple version of a project budget tracker, but should give you a sense of how pickling and error handling work.

### Importing the Pickle Module and Declaring Variables

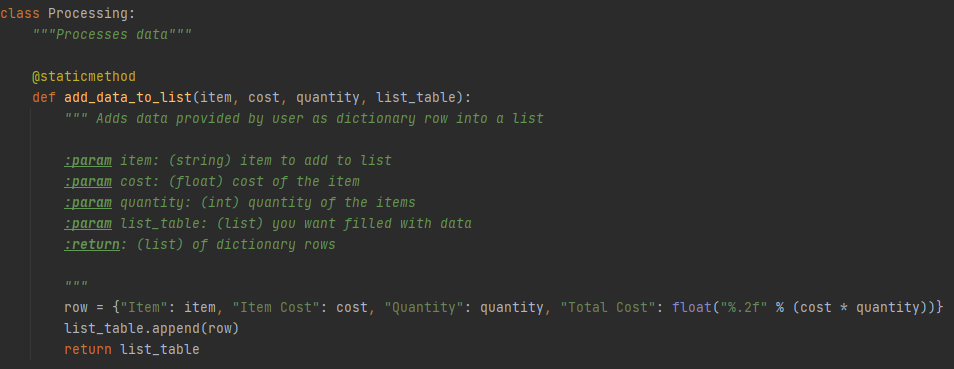
To get started, I imported the Pickle module and declared my variables (Figure 1). This sets them as their appropriate types (i.e., string, list, float, and int) and allows me to name my file.



***Figure 1: Declaring the script’s variables***

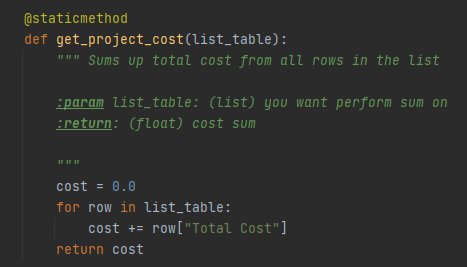
### The Processing Class and Functions

From there, I created my first class for processing the data, which contains 5 functions. The first of those is for adding data to the list (Figure 2). This function also allows me to do a little math and format it into a dollar figure.



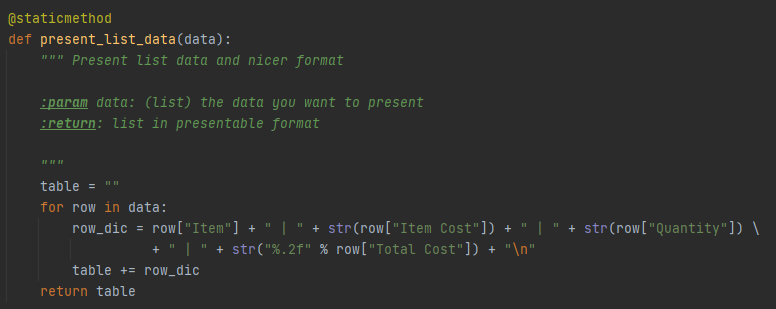
***Figure 2: The add\_data\_to\_list function***

Next, I created a function for getting the total project cost. This simply creates a cost float variable and adds the total cost from each row, from the list you provide, to it. It returns the total back to the program (Figure 3).



***Figure 3: The get\_project\_cost function***

In order to present the pickled data back to the user in a more presentable way, I created a function to do just that (Figure 4). It iterates through each row, even formatting the Total Cost of each into a dollar figure.



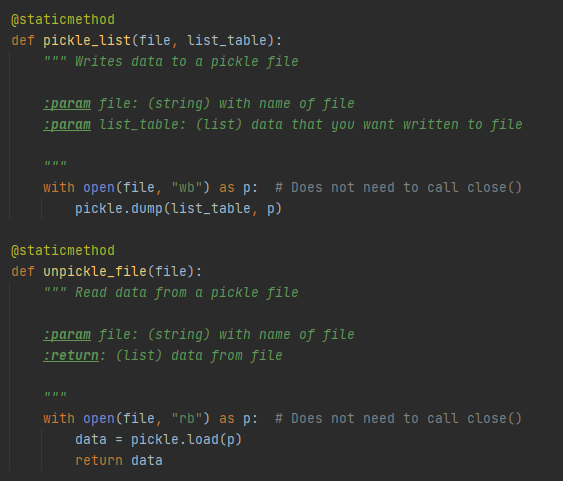
***Figure 4: The present\_list\_data function***

The two functions to follow were created to pickle and unpickle the data (Figure 5). As you can see, pickling is pretty simple.

To write the data, you “dump” it into the file. To read the data in the file, you “load” it. In both cases I’ve decided to open the file using the “with” statement. Because this statement also automatically closes the file, I don’t need to include the “close” function.

The pickle function takes two parameters: the file to create or write to and the list of data to write to it.

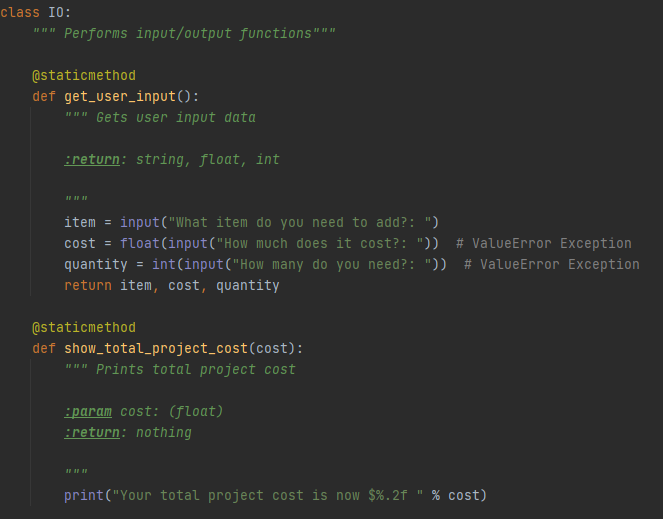
The unpickle function takes one parameter: the file to read; and returns the data from the file.



***Figure 5: The pickle and unpickle functions***

### The IO Class and Functions

I also created two input/output functions, within the IO Class. One of these gets user input and returns it to the program, while the other prints back total project cost (Figure 6).



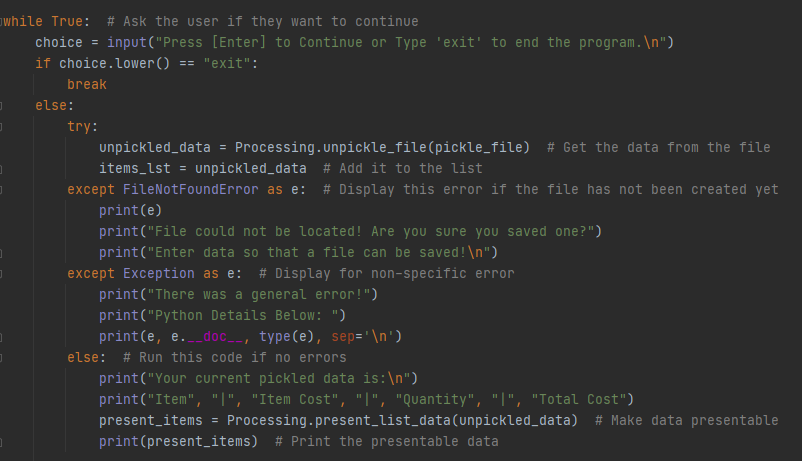
***Figure 6: The IO Class and functions***

## The Main Program

The main program code is contained within a while loop. There are two main sections. The first unpickles the data, adds it to the items list and presents it back to the user. The second gets user input for new items, adds it to the list, gets and displays project cost, and pickles the new list.

### The First Section

To handle errors, you put the code within try blocks. These allow you to “try” the code, but run through error handling options if there is a problem (Figure 7).

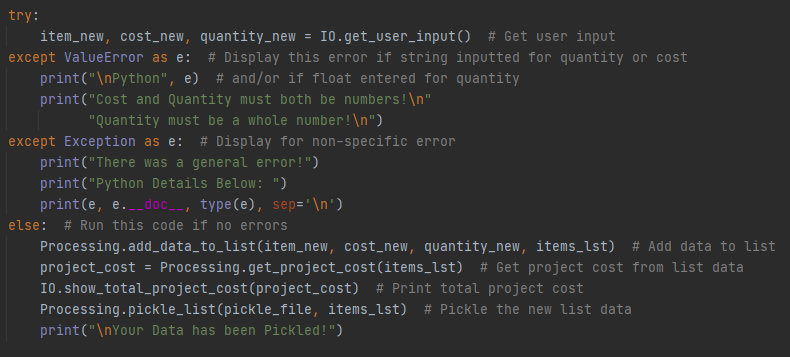


***Figure 7 : The first section of the main code, which includes error handling***

The error handling included here, first runs through the specific “FileNoteFoundError” exception, which is the most likely to cause a problem. If this happens, user-friendly descriptions are printed back to the user, and the program continues back to the top of the while loop. I’ve included a general exceptions catch, in case of unforeseen issues. If no issues are found, the code after “else” will run.

### The Second Section

This section is similar to the first, only that the specific error handling is different (Figure 8). This time around we are looking for value errors. These would be from the user, if for instance they entered a string where the program expected a float or int. Rather than the program failing, it would go back to the top of the while loop and give the user another chance. Again, if no error is found then the else statement will run.



***Figure 8: The section of the main code, which also includes error handling***

## Summary

This week was all about Pickling and Error Handling. This hopefully displayed how useful pickling can be for storing large amounts of data and how error handling can prevent your program from breaking do to preventable errors.