Jace Riley

Randal Root

IT FDN 100 A

11/20/19

**Error Handling and Pickling**

**Introduction**

This document is to be used as a training aid on how to error handle in python, how pickling works and is to be used within any Python GUI. It may also be utilized to help further personal research.

**How to error handle within Python**

There are essentially two types of errors in Python: Syntax Error and Exception. A syntax error is when the Python parser is unable to understand a line of code. This could be something as simple as a period in the wrong place, a space in the wrong place, or a misspelled function. An exception is an error that is detected while the code is executing. Python has quite a few built-in exceptions. A decent list of some of them can be found at “<https://www.tutorialsteacher.com/python/error-types-in-python>”

There is a way that a programmer can handle these errors within their code. This is through a try-except statement. A try clause is executed, and if it succeeds the except clause is skipped. If the try statement does not work then it immediately goes to the except clause. A basic version of the try except statement is in Figure 1a below, where the function is stated first and the try and except will be tested on that function. In this case the function is to divide two numbers. Figure 1b shows the output of this.

def divide():  
 x = float(input("Enter a value to divide: "))  
 y = float(input("Enter a value to divide by: "))  
 result = x//y  
 print ("The answer is: ", result)  
  
# Demonstrate a generic try and except handling of an error  
try:  
 divide()  
except:  
 print("An exception occurred.")

Figure 1a



Figure 1b.

A slightly more advanced version of the try except statement includes the actual error clause that is being broken in the code. An example of that is in Figure 2a where the “ZeroDivisionError” is being tested. Figure 2b has the output.

print("\nEnter a zero as your second number.\n")  
try:  
 divide()  
except ZeroDivisionError:  
 print("Sorry! You are dividing by zero.\n")  
except:  
 print("Something else went wrong.\n")

Figure 2a.

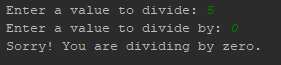


Figure 2b.

The next test, in Figure 3a, is utilizing the finally aspect of the try, except statements. This will be executed regardless of the try statement raising and error or not. Figure 3b contains the output which as you will be able to tell, no value was entered and it cast an error. It would be possible to list each possible error type through except statements.

try:  
 divide()  
except ZeroDivisionError:  
 print("Sorry! You are dividing by zero.")  
except:  
 print("Something else went wrong.")  
finally:  
 print("The 'try except' is finished.")

Figure 3a.

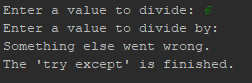


Figure 3b.

**How to research and utilize the pickle library.**

The pickle library is utilized when converting an object into characters, called serializing or to pickle, or for reading those characters into an object called deserializing, or depickling. It is possible for any object in python to be pickled. The best place to obtain information about libraries are typically from that library’s documentation, or the python libraries documents page. In the case of the Pickle library, it is located at <https://docs.python.org/3/library/pickle.html>. The importance of checking this is it may contain information that is not listed in a typical website teaching about the library. In the case of the pickle library, from the website listed above we learn that the pickle module is not secure, and to only unpickle data that you trust. The other benefit of this website is that it gives you the information of all the classes in the libraries. Other websites are great for learning how to use the code, but this one gives you the information in an unopinionated manor.

Next we will demonstrate the use of pickling in a similar code to the home inventory code we wrote before. The first thing that was done was to define three functions: “enterData,” in Figure 4a, “readData,”in Figure 4b, and “saveData” in Figure 4c.

def enterData():  
 print('Type in a "Name" and value for your household items')  
 try:  
 name = str(input('Enter an Item Name: '))  
 except:  
 print("Sorry, incorrect entry type, please enter a string as the item.")  
 try:  
 value = int(input('Enter a Value: '))  
 except ValueError:  
 print("Sorry, incorrect entry type, please enter a integer value.")  
 itemList.append(name + ', $' + str(value))  
 print(itemList)

Figure 4a

The enterData function goes through the error handling discussed earlier.

def readData():  
 objFile = open(fileName, "rb")  
 list\_of\_data = pickle.load(objFile)  
 objFile.close()  
 print(list\_of\_data)

Figure 4b

def saveData():  
 print('Would you like to save your Data?')  
 # Input is cast as a variable to allow for option to save or not.  
 prompt = input('Enter "y" or "n"')  
 if prompt.lower() == 'y':  
 print('saving data to file...')  
 file = open("HomeInventory.dat", "wb")  
 pickle.dump(itemList, file)  
 file.close()  
 else:  
 print('You have exited without saving')

Figure 4c

Figure 4b is ultimately showing a depickling of data. The way that this code is written, the data needs to be saved through pickling first as in Figure 4c.

The main body of the code that this runs through is listed in Figure 5.

while True:  
  
 # The print statement gives the options to the user.  
 print('\n' + 'Please enter from these menu options:' + '\n' + '1) Add Data to List' + '\n' +  
 "2) Display Current Data" + '\n' + '3) Save to File' + '\n'+ '4) Exit' + '\n')  
 # Casting the input as a variable to allow it to be used through the options given.  
  
 option = input('Which option would you like to perform? (1 to 4): ')  
  
 # Option 1 is to append the name and value entered by the user into the in memory list.  
 if option == '1':  
 enterData()  
  
 # Option 2 is to give the user back the information for review to see if they want to save with option 3.  
 elif option == '2':  
 readData()  
  
 # Option 3 is to allow the user to either save the data to a text file or exit without saving.  
 elif option == '3':  
 saveData()  
 elif option == '4':  
 print("You have exited the program")  
 break

Figure 5.

First thing printed are the options. Option 1 runs through the enterData function which the results of that are listed in figure 6a. Once the data is entered it is important to save the data to the file which represented is in figure 6b. In this code the save action is pickling the information into a file. Once the file is saved you can read the data, which depickles that file. This is shown in Figure 6c, and the raw picked data is shown in figure 6d.

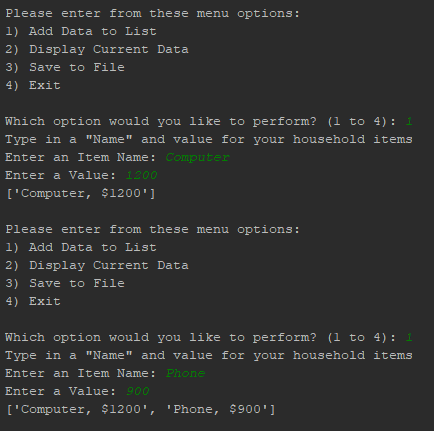


Figure 6a.

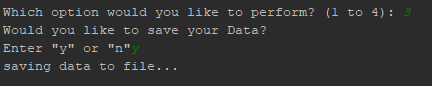


Figure 6b.



Figure 6c.

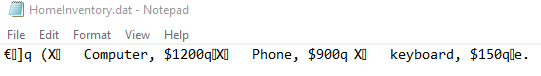


Figure 6d.

**Useful Websites.**

Some websites that I found useful were, and why they were useful are below:

<https://www.w3schools.com/python/python_try_except.asp>

I love w3schools, due to the ability to run the examples yourself. It is also useful because it gives options to be able to look at other programming languages.

<https://www.geeksforgeeks.org/try-except-python/>

geeks for geeks is another one that I have enjoyed for a while in all my code research. It is a little more substantive with giving suggestions for other areas of the coding language to research.

<https://www.tutorialsteacher.com/python/error-types-in-python>

Tutorialsteacher is a great option for building presentations or teaching others. It goes into each topic fairly in depth, but very concise and readable.

<https://www.journaldev.com/15638/python-pickle-example>

Journaldev was one of my favorite websites to search. It was very informative, and had a gif that showed running portions of the code to see how exactly each line interacted with the code.

<https://docs.python.org/3/library/pickle.html>

This website I think is one of the most important. It is the direct library documentation within Python. It tells you all the nuances of the back end of the code as well. An important aspect I find is that it can give you warning of bugs, or dangers in running that code.

**Summary**

This was a fun exercise to run training of code, which I know teaches you more about the code than simply reading documentation. You have to think a little outside of the box to be able to walk people through code, and how it is being utilized. It also impacted how I structured the code.