**Name:** Kerry Bosworth  
**Date:** Nov 18, 2019  
**Assignment:** 07

# Don’t get into a pickle with your exceptions

**Introduction**

This week we spent time learning how to use pickle when writing (dumping) and reading (loading) data to a file. This method obscures the data as well as takes up less space. We also increased our knowledge of error handling, including raising a custom exception with try and except statements.

**Programming Assignment**

To complete the assignment, it was necessary to not only read one item that was dumped into a pickle, but more than one. I researched how to load the object so that all values could be printed to the screen. Like readline, you could keep calling the pickle and it would pick up where it left off.

**For example:**

When the file has these entries (on how to build a shed):

Buy blueprint, plan  
Dig holes, prep  
Build deck, build

And I have this code:

list\_of\_data = pickle.load(objFile)

print()  
list\_of\_data = pickle.load(objFile)  
print()

It will display the first two values only:

Buy blueprint, plan  
Dig holes, prep

Initially I did some experimentation and then researched how to do this on the web. I did not quickly find the answer and it took several different search keywords. This page looked promising, but it contained information we already covered:

<https://www.tutorialexample.com/fix-python-pickle-load-typeerror-file-must-have-read-and-readline-attributes-error-python-tutorial/>

Other pages like this were interesting, but more for reference only.

<https://docs.python.org/2/library/pickle.html>

Finally, this page presented a viable solution. One of the answers used the append function and this felt familiar because we used it in previous assignments. Once I saw it, I had high confidence that it should work. Incorporating append allowed me to achieve my goal.

<https://stackoverflow.com/questions/35067957/how-to-read-pickle-file>

Here is that section of code in my program. Coincidentally, this also included a try and except block as well.

**def** read\_data\_from\_file(file\_name):  
 *"""  
 Desc - Removes an item input by the user into the table* **:param** *file\_name: (file) file that will be read* **:return***: list  
 """* list\_of\_data = []  
 **with** (open(file\_name, **"rb"**)) **as** objFile:  
 **while True**:  
 **try**:  
 list\_of\_data.append(pickle.load(objFile))  
 **except** EOFError:  
 **break** objFile.close()  
 **for** row **in** list\_of\_data:  
 print(row[0] + **" ("** + row[1] + **")"**)  
 print()

The main body of my program is fairly simple.

*# Presentation ------------------------------------ #  
# TODO: check to see if the file exists*FileHandling.does\_file\_exist(strFileName)  
  
*# TODO: read the contents of the binary file*print(**"These are the steps documented to build the shed:"**)  
FileHandling.read\_data\_from\_file(strFileName)  
  
*# TODO: ask for input*step = input(**"Enter a step: "**)  
phase = input(**"Enter a phase (plan, prep, build, finish: "**)  
lstSteps=[step, phase]  
  
*# TODO: store the list object into a binary file*FileHandling.save\_data\_to\_file(strFileName,lstSteps)  
  
*# # TODO: read the data from the file after updates have been saved*print(**"Here is the list after your additions:"**)  
FileHandling.read\_data\_from\_file(strFileName)

The bulk of my exceptions were in the function called **does\_file\_exist**. Researching exceptions online was easy. These were good reference pages:

<https://docs.python.org/3/library/exceptions.html#built-in-exceptions>  
<https://www.pythonforbeginners.com/error-handling/exception-handling-in-python/>

And there were good pages with some tutorials.

<https://pyblog.in/programming/python/python-exception-handling/>  
<https://www.edureka.co/blog/python-try-except/>

I called this function first thing when the program starts to see if there is a file. If there is no file present in the directory, it will ask the user if they want to create one. If ‘y’, the file will be created, and it will continue to the next function in the main body of the program.

**class** FileHandling:  
 **def** does\_file\_exist(file\_name):  
 *"""  
 Desc - Checks to see if a file exists* **:param** *file\_name: (file) file to be checked:* **:return***: nothing  
 """* **try**:  
 objFile = open(file\_name, **"rb"**)  
 objFile.close()  
 **except** FileNotFoundError:  
 *#print("File does not exist.")* choice = input(**"File does not exist. Create one? y/n \n"**)  
 **try**:  
 **if** choice.lower() == **'y'**:  
 objFile = open(file\_name, **"wb+"**)  
 objFile.close()  
 **else**:  
 **raise** CustomError()  
 **except** Exception **as** e:  
 print(e, e.\_\_doc\_\_, type(e), sep=**'\n'**)  
 sys.exit(1)

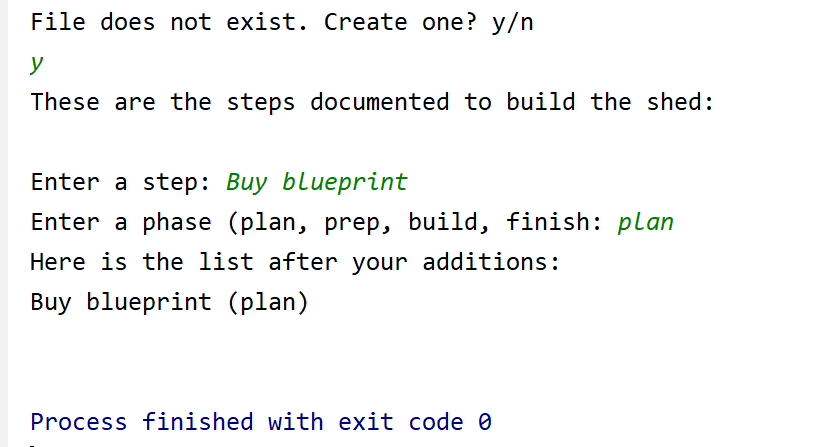
If the user inputs any other key sequence it will raise a custom error stating that the program will now exit,

**class** CustomError(Exception):  
 **def** \_\_str\_\_(self):  
 *""""This is a custom error when when the user does not want to proceed and exit the program"""* **return "The program will exit now."**

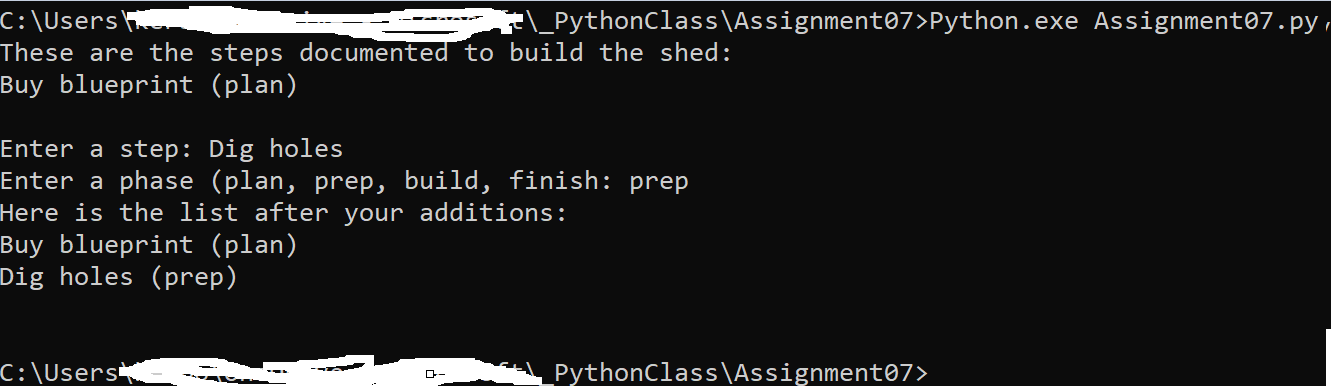
**For example:**

File does not exist. Create one? y/n  
n  
The program will exit now.  
None  
<class '\_\_main\_\_.CustomError'>

These are examples of the program running successfully. It creates a file if none exists (see Figure 1) and if one does (see Figure 2), it displays the list and prompts the user to add another.



**Figure 1:** Run of program in PyCharm



**Figure 2:** Run of program in command line

**Summary**

When creating this simple program, I first addressed how to pickle the file. It’s another concept that we can leverage as we continue to do our assignments to store, read and display data. Then I went back and added in exception handling to round out the program. The program successfully allows a user to create or add to a file with steps on how to build a shed. The last part of the assignment was to mirror this word doc on a GitHub webpage using simple mark down language. Here is the link to the webpage: <https://yukkutobu777.github.io/IntroToProg-Python-Mod07/>.