Dmitriy Alexandrov

November 20th, 2019

IT FDN 100 A

Assignment07

Python Error Handling and Pickling

Research

Per the assignment instructions, I first researched pickling and error handling in Python. Here are the following resources that I used for each:

Pickling:

https://docs.python.org/2/library/pickle.html - I looked in the Python documentation for a description of pickling to get an in-depth understanding of what the module did and its usages. This was a very useful resource for learning.

https://www.geeksforgeeks.org/understanding-python-pickling-example/ - I liked this resource because it gave code examples of how pickling works and what the outputs would look like. It also gave some advantages of using pickling as well which helped me further understand the usage of it.

https://www.benfrederickson.com/dont-pickle-your-data/ - I liked this resource because it showed the down sides of pickle and why you should not use it in your scripts.

Error Handling:

https://www.pythonforbeginners.com/error-handling/ - I found this error handling for beginners' resource useful because it helped me understand how error handling worked in Python as well as how to use error handling in my own code in the future with examples.

https://realpython.com/python-exceptions/ - Was another resource that I used and found very helpful because as it explained error handling in Python it gave examples and command outputs with each example to make it easier to understand and visually see what the outputs would be.

Pickle In Python

Per the definition from the Python library, "..."Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream is converted back into an object hierarchy.

In other words, data is converted into a bit stream before it is written to a file and this bit stream contains the information needed to create the data in the "unpickling" process.

Why use Pickle?

You might be wondering as I was, why would we even use this concept of pickling to begin with? What does it give us? Well during my research on the GeeksForGeeks site, they identified that pickling has 3 major advantages:

1. **Recursive objects**: Pickle keeps track of the objects it has already serialized, so later references to the same object won't be serialized again.
2. **Object sharing**: Similar to self-referencing objects; pickle stores the object once and ensures that all other references point to the master copy.
3. **User-defined classes and their instances**: Pickle can save and restore class instances transparently.

Why should you not use Pickle?

While looking up the definition of pickle on the Python documentation site, I came across a red banner with a warning message in it that stated,

"Warning The pickle module is not secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source."

I figured this wasn't just a common small warning, and decided to do some more research on why you should not use Pickle and what are some disadvantages of it. This is where I came across Ben Frederickson's site on why you should not Pickle your data and his main points were that using pickle is slow, a huge security risk. The slow factor is in comparison to other data serialization methods like JSON, or cPickle (C-based Pickle) but the security risk that was brought up in the Python documentation was that unpickling unknown binary code could result in malicious code execution.

Error Handling In Python

As your Python script is running and as a user interacts with your scripts there is an inevitable chance that an error will occur.

Whether that error is due to something that the user has done (exception) or whether it is an issue with the script itself (syntax).

Different errors will result in different outcomes in the code execution process. A syntax error will result in the program crashing all together without a chance to recover due to the error existing in the code itself and Python not being able to understand it.

While exception are errors that are put in by a developer to help a user understand what they have done wrong or out of the bounds of the way the script should be working. This is usually done through a try-except method in Python where a developer will have the script to attempt to interpret user input for example, and if that input results in an error than an except block will be used to let the user know what they have done incorrectly.

How does Error Handling work?

Per the PythonForBeginners site, "...error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks. If an error is encountered, a try block code execution is stopped and transferred down to the except block." In other words, as a script is being executed if an error occurs that isn't due to the Python interpreters ability to understand the code (syntax error) then an exception is generated (if Python knows about this type of error or if the developer created a custom exception) and the code continues to run after that.

Why is Error Handling important?

Error Handling is useful for all sorts of reasons spanning from identifying bugs in a script to helping a user of a script use that script properly. As a developer is it important to include error handling in your code to allow others that might interact with your script to understand the intended functionality of the script.

Meet "Pickle Rick 3000" – A Look at Pickling/Unpickling and Error Handling

As an avid "Rick and Morty" fan, when I was learning about the concept of Pickling/Unpickling data all I could think about was the Pickle Rick episode.
And thus was born the script that would help me practice using pickle and error handling in the same spot.

The script provides the user with the following options:

```
Welcome to the Pickle Rick 3000:
 Here are your options:
    1: View My Data
    2: Add Data
    3: Pickle My Data
    4: Unpickle My Data
    5: Exit
What option do you want to choose? [1 - 5]
```

The first two options are easy to understand and were covered in the previous assignments. That being that the user is returned the data stored in a list in option 1, and the user has the ability to input an item name and value for that item to be added to a list in option 2.

But in option 3, the user has the ability to pickle the data that they have stored in the list to a file. This is done with the following code:

```
try:
    outputFile = open_file()
    if not outputFile.endswith(".dat"):
        raise BinaryFileExtensionError()
except Exception as e:  # Custom Exception thrown
    print("An error has occurred...")
    print("The Custom Error info:")
    print(e)
    print("Please try again. \n")
    continue
```

```
else:
    print("\nPickling your data...")
    objF = open(outputFile, "wb")
    pickle.dump(lstData, objF)
    objF.close()
    print("Your data has been pickled in the " + outputFile + " file. \n")
```

You will see here that this incorporates both the use of Pickle and Error Handling in a single set of code. This is done by "try"-ing to get that user input with the "open_file" function and taking the return and seeing if it ends with ".dat".
If it does not, then a custom error is generated called "BinaryFileExtensionError". Since this is a custom generated error that isn't native to Python it is created as a separate class above in the script and looks like this:

```
class BinaryFileExtensionError(Exception): # custom exception if the file extension
is not .dat
    def __str__(self):
        return "File Extension is invalid, expecting .dat"
```

If the user does not input a file name with a ".dat" extension then they will receive a message telling them this. This is what the error message looked like as the script was running:

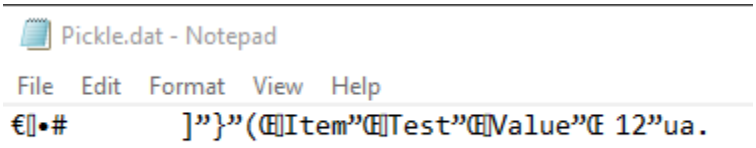
```
What is the file name that you would like to open? *make sure it's a .dat file* Pickle
An error has occurred...
The Custom Error info:
File Extension is invalid, expecting .dat
Please try again.
```

But if the user did as they were told and put in a file name with a ".dat" extension, then the script took the data that was stored in a list and dumped the bit stream value of the data into a file with the name that the user specified. This is what that functionality looked like to the user:

```
What option do you want to choose? [1 - 5] 3
What is the file name that you would like to open? *make sure it's a .dat file* Pickle.dat

Pickling your data...
Your data has been pickled in the Pickle.dat file.
```

And here is what the output looked like in the file that the Pickle module dumped the bit stream into:



This is a single example from the PickleRick3000 script that shows the use of both pickling and error handling. The rest of the script also had the reverse process of unpickling data and more error handling functionality.

Summary

Learning about how to use the pickle method in Python along with Error Handling has been great. It was important to not only understand the concepts of how each work but to also get some hands-on experience with trying to incorporate each of them into a script. Lastly, being able to teach what you had learned was also a different challenge in itself because you had to be able to represent what you had learned to try and show someone else. But most importantly, having fun in the whole process was the real key to success.