Nick Hertlein

March 04, 2021

IT FDN 110 B: Foundations Of Programming: Python

Assignment07

GitHub URL: https://github.com/nhertlein/IntroToProg-Python-Mod07

Files & Exceptions

# Introduction

This document will illustrate the knowledge gained from the seventh lecture and the media portion of the third assignment. In addition, I will cover the creation of a Python script that shows how to pickle and unpickle data, and how to use exceptions to handle errors gracefully. The program will read in some great one liner quotes from Arnold Schwarzenegger. The list of quotes will then be pickled and unpickled and displayed to verify the process worked correctly. The next section of the script had an ability to induce problems to get errors based on the great “rockumentry” Spinal Tap. This assignment was a little different as it was open-ended I didn’t go off the rails too far. No particular reason for using Arnie quotes or Spinal Tap as inspiration. My assignment is late again ☹

# Online Media to Research Pickling and Exception Handling

For the assignment we were asked to do some online research about pickling and exception handling. From the search results I went to a thread on pickling on the StackOverflow forums which I always find helpful. The link to the thread is [pickle - Understanding Pickling in Python - Stack Overflow](https://stackoverflow.com/questions/7501947/understanding-pickling-in-python) (external site) and I found it pretty helpful. There were some basic comments which are good for understanding the function, and as always there are the more technical answers which are good for understanding what is going on in the background of the module. The other place I looked for information on pickling is the actual Python documentation [pickle — Python object serialization — Python 3.9.2 documentation](https://docs.python.org/3/library/pickle.html) (external site). The official documentation is very detailed and was good to read after getting a better understanding from the class lecture as well as from the StackOverflow thread. My main takeaway is that pickling is a good way to store data in a format that not easily readable, but of course it is not encrypted so if you need that level of protection this is not the method to use.

For exception handling research I used the same sources as I always find them helpful. On StackOverflow I looked at the thread [Manually raising (throwing) an exception in Python - Stack Overflow](https://stackoverflow.com/questions/2052390/manually-raising-throwing-an-exception-in-python/24065533#24065533) (external site). This thread had a lot of good information on the right and wrong ways to use exceptions. From my experience exceptions are really important to keep the program from just crashing unexpectedly. The other important aspect is being able to tell the user what exactly went wrong so they can find/fix the bug or know what to change for the next time they run the script.

# Writing the Pickling Portion of the Script

For this week’s assignment I made a project in PyCharm in the Assignment07 folder of the C:\\_PythonClass directory. This assignment was a bit different in that it did not have a “starter” so I created a blank script and copied the header over from the previous assignments and filled it out.

Under the header I imported the pickle module so it was available for use in the script. The first block of code I used for declaring variables. This script didn’t really need that due to length, but it’s still good practice (Figure 1).

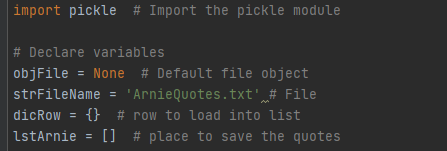


Figure 1. Declaration of variables

The next section of the script was my example using the pickling function to save data in a binary format. For the example to work, we need to load in some data so I chose to load it in from a \*.txt file so I could compare with the \*.dat file output later. I used similar code to what we have used in the past to load in lines of data from a file, split it into categories, save it as a dictionary row, and then append to a list for reference (Figure 2). As mentioned previously for some reason I decided to load in cheesy one liner quotes from Arnold Schwarzenegger movies.

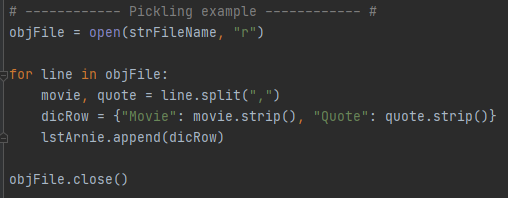


Figure 2. Load in movie quotes from \*.txt file

Once the data was loaded into the script, it was a natural next step to display the contents to the user. The script pauses for the user to press enter to move to the next stage of the script, which is pickling the data just loaded in (Figure 3)

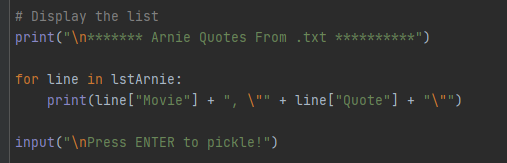


Figure 3. Display data that was loaded in from the file and pause for user input

After the user hits enter, the script continues and the data is pickled. The first step in the pickling process is to create a file object we will want to save to. I used the “wb” option to open a file in the binary format and ensure that if data was already present in the file it would be replaced by the new content. Data is then saved to the file using the dump method of the pickle module. The file is then closed to prevent data leaks and the user is notified the operation was successful. The script pauses for the user to press enter to move to the next stage of the script, which is unpickling the data (Figure 4). The data saved into the \*.dat file format is not as easily human readable (Figure 5).

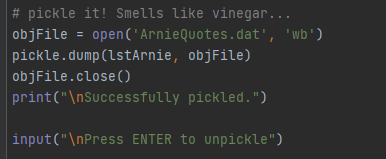


Figure 4. Pickle data into \*.dat binary file format

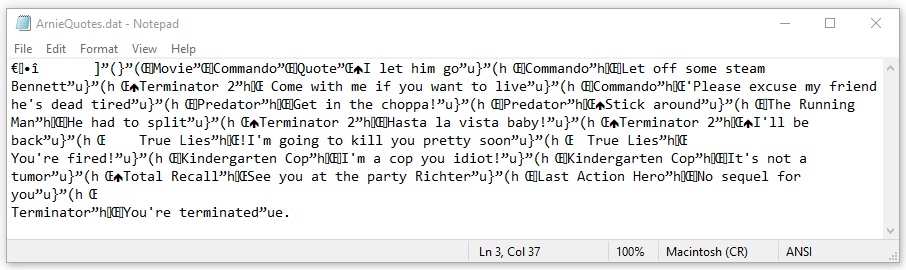


Figure 5. Data saved into \*.dat binary file format

After the user hits enter, the script continues and the data is pickled. To unpickle the data the binary file is opened with the “rb” option for read binary. Data is then saved to a new list using the load method of the pickle module. The file is then closed to prevent data leaks and the unplicked list is displayed to the user to confirm it’s in a proper readable format. The user is then notified the data was successfully unpickled (Figure 6).

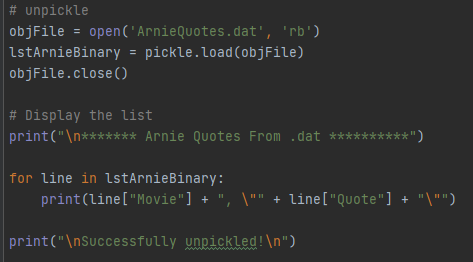


Figure 6. Unpickle data and display list to the user

# Writing the Exception Handling Portion of the Script

This section of the script was my example using exception handling options to tell the user what went wrong, and make sure the script does not just crash out. The first error handling example was designed to use a custom exception if the user did not enter the answer I wanted. The user is asked if the silly quotes made them laugh. If the user enters anything except “y” an exception is raised to let them know this is not possible. After the exception is raised the “except” portion of the try block displays the custom error message as well as a reprimand the user needs to watch more cheesy movies (Figure 7).

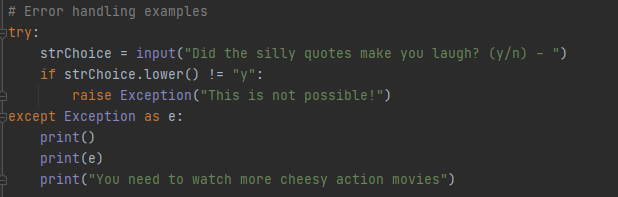


Figure 7. First error handling example with a custom exception

For my last exception handling example I built this into a while loop so the user would get multiple opportunities to get an input that would be useful. The user is asked how they would rate the program on a scale form 1 – 10. It should be obvious to anyone the program is exceptional (although delivered late) and the user is not allowed to enter anything less. If the user enters a value less than or equal to 0, this is clearly not the type of response that was asked for and the user gets a sarcastic quote in return. If the user enters a value that is a string or a float a ValueError is raised and the context of the error is printed as well as a message letting the user know we are not going to investigate why they entered something that was not asked for. Finally, if the user enters an integer between 1-10 like asked, the program is not satisfied and bumps the value up to 11 (Figure 8).

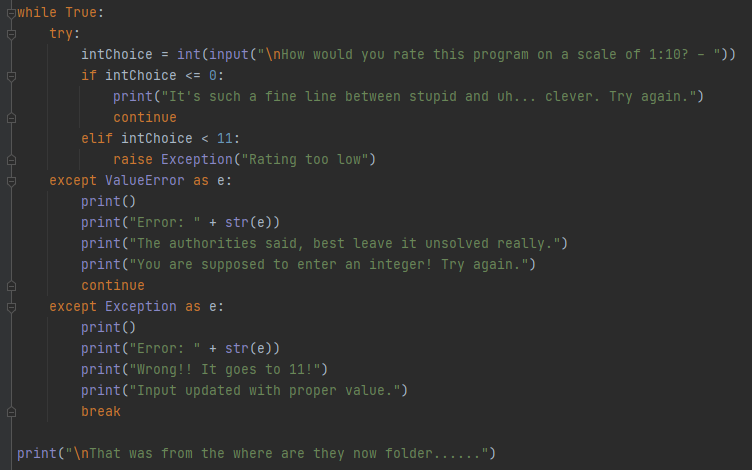


Figure 8. While loop with exceptions

# Summary

This module was intended to be an introduction to reading in files in various formats (\*.txt and \*.dat) as well as expose us to exceptions. The part of the module I was the most interested in was the exception handling, as experience tells me the user will always do something crazy and crash out the program. Making sure you have a path to handle the problem and letting the user know why a substitution was made or a portion of the analysis was skipped is critical to making sure they can fix the issue. I am sure I will use this exception handling a lot in the future!