Hannah Chung

May 30, 2020

IT FDN 100 Foundations of Programming: Python

Assignment 07

https://github.com/hannahinissaquah/IntroToProg-Python-Mod07

# **Assignment07: Files and Exceptions**

Introduction

This week, we took a closer look at reading and writing data to text files, learned how to work with binary files using Python’s pickle module, and were introduced to structured error handling (try-except). In this document, I will provide an overview of Python’s Pickling and Exception Handling features and share web sites that provided useful information for me on these topics. To demonstrate how *pickling* and *structured error handling* work, I created a script that will allow my 9-year old son to track his reading minutes. I will explain the steps I took to create the script that will allow him to display his reading log, add new entries to his reading log, create a new log or overwrite the log, and to exit the program.

Pickling in Python

The pickle module is used to convert Python objects into something that can be written into a file and then easily read back out. So far we have been reading and writing data to plain text files. While text files are convenient, they are limited to storing only characters that humans can read. Pickling allows us to store complex data, including Booleans, integers, floats, complex numbers, strings, tuples, lists, sets, and dictionaries, in binary format. Storing data in a binary format can obscure the file’s content making it difficult for humans to read; however, it is not encrypted and you should never try to unpickle data from an untrusted source.

Pickle uses a process known as “serialization” to convert an object in memory to a byte stream that can be stored on disk or sent over a network. Subsequently, this stream can then be retrieved and de-serialized back to a Python object.

To use pickle, first import the pickle module as shown in Figure 1.

****

**Figure 1 Screenshot of import pickle**

Next, since pickled objects can only be stored in a binary file, use the open() function to open a binary file (.dat extension) for writing and pass “wb” as the file access mode. If the file exists, its contents will be overwritten. If the file doesn’t exist, it is created. You can also use “ab” to append a binary file; if the file exists, new data is appended to it and if the file doesn’t exist, it’s created. Once the file is opened for writing, you can use pickle.dump(), which takes two arguments: the object you want to pickle and the file in which to store it. Lastly, close the file using the close() function. To retrieve the pickled file back into Python, use the open() function again, but this time with “rb” as the file access mode. Next, use pickle.load(), which takes one argument: the file from which to load the pickled object.

Part of this week’s assignment was to search the web for examples of how to use Python’s Pickling features. A couple of websites that helped me better understand Python’s Pickling features include:

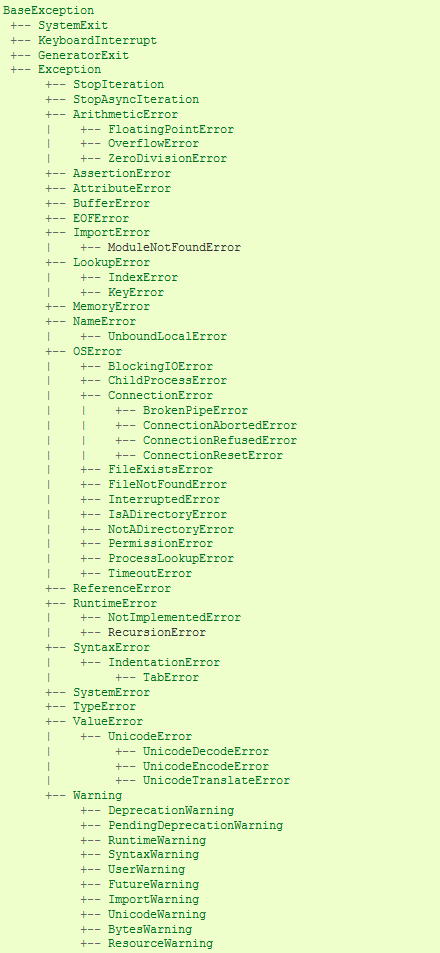
* Journal Dev - Python Pickle Example <https://www.journaldev.com/15638/python-pickle-example> (external site)
* Data Camp - Pickle in Python: Object Serialization <https://www.datacamp.com/community/tutorials/pickle-python-tutorial> (external site)

There were others that I reviewed including Python’s Standard Library (<https://docs.python.org/3/library/pickle.html> - external site) but found the two referenced above were concisely written and easiest to read, which helped me to better grasp this topic and not get confused by other topics that we have not yet covered. In addition to being easy to read, I liked the embedded short video by Journal Dev showing execution of the pickling features.

Exception Handling in Python

When we write programs, we should think about exceptions that may be introduced when other people use the programs, and handle the exceptions that are raised to guard against potential failures causing the program to exit abruptly. For example, if the user entered a string object while you were expecting an integer object as input, we can use the try statement and except clause to catch the exception and inform the user that something went wrong by specifying the error in a user-friendly way.

Exception is a built-in python class that can catch any type of error, but specific errors can be caught using more specific exception classes. Figure 2 shows a list of Python’s Built-In Exceptions. A simple way to handle exceptions is to use the try statement with an except clause. The try statement will run a block of code that could potentially raise an exception. The except clause will execute a block of statements only if an exception is raised.



**Figure 2 Python's Built-In Exceptions (Source: https://docs.python.org/3/library/exceptions.html#exception-hierarchy)**

A few of websites that helped me better understand Python’s Exception Handling features include:

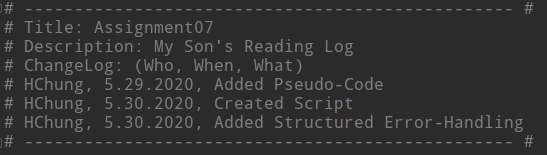
* Python’s Standard Library - Built-in Exceptions https://docs.python.org/3/library/exceptions.html#
* Journal Dev - Python Exception Handling – Python try except <https://www.journaldev.com/14444/python-exception-handling-try-except> (external site)
* Data Camp - Exception and Error Handling in Python <https://www.datacamp.com/community/tutorials/exception-handling-python> (external site)

Python’s Standard Library document provided a lot of detailed information regarding the BaseException and Exception classes and helped me to have a clearer understanding of the exception hierarchy. Again, I found the Journal Dev and Data Camp tutorials well-written and easy to understand, and they both introduced me to new features I did not previously know about, including using else and finally. I found the Exception Handling example on Journal Dev’s site easy to follow especially with the programmer’s comments included in the example script.

My Demo – My Son’s Reading Log

**Getting started**

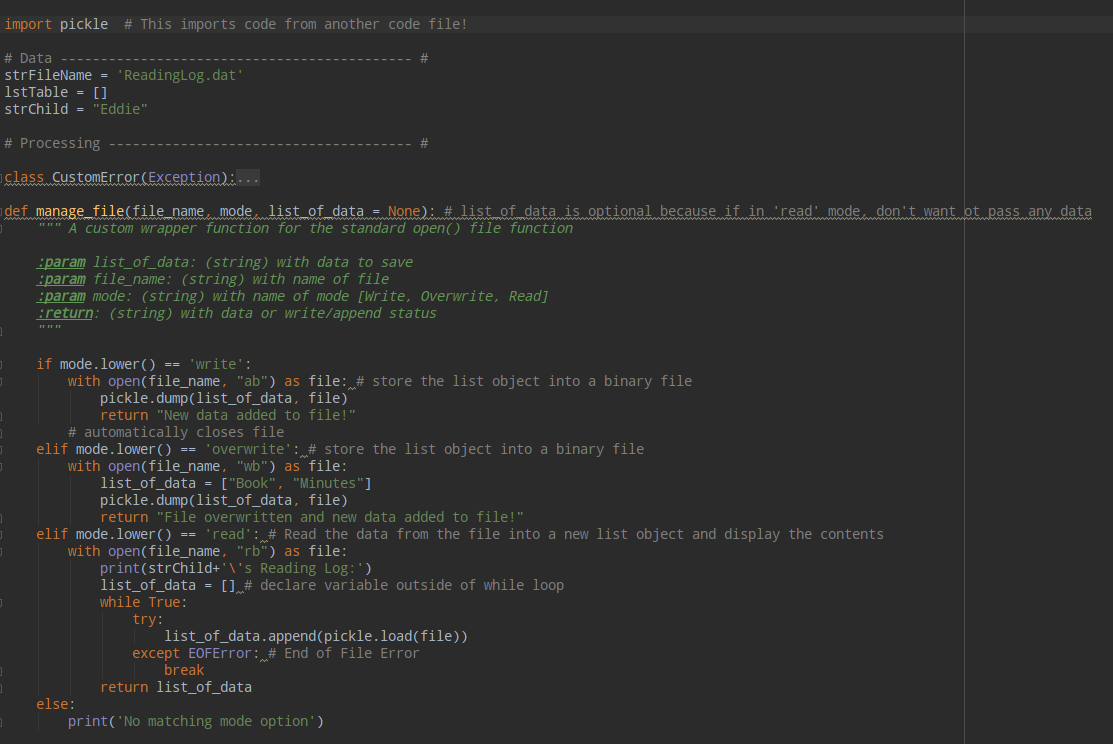
First, I created a new PyCharm project that uses the \_PythonClass\Assignment07 folder as its location. Within the project, I created a new Python file called “Assignment07.py”. Figure 3 shows a screenshot of my script header.



**Figure 3 Screenshot of Assignment07.py – Script Header**

**Using the Pickle Module**

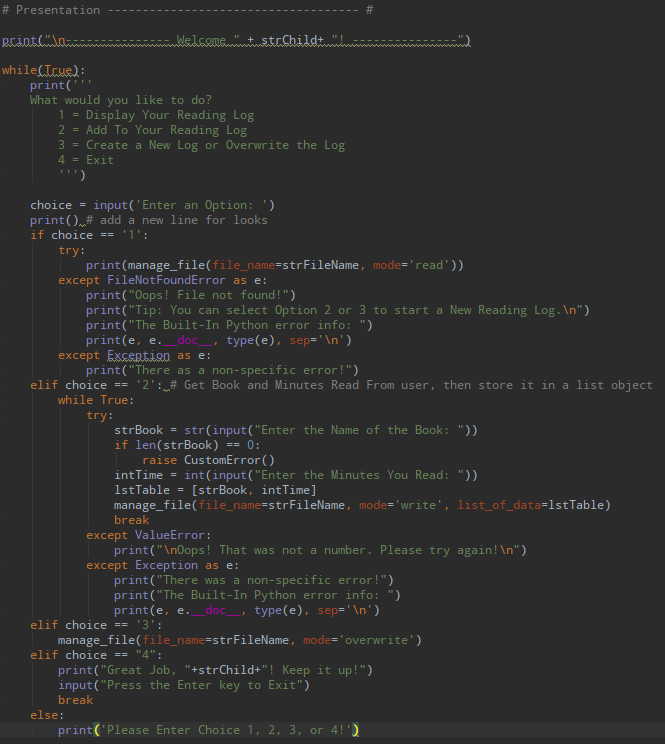
Figure 4 shows a screenshot of my script’s pickling features. At the beginning of the script, you will see that I imported the pickle module. I then created a custom wrapper function similar to the one shown in Listing 8 of Module 07 and defined the function containing the block of statements that perform reading and writing to a binary file as manage\_file. In addition, I used the with construct, which automatically closes the file when the code reaches the end of the with block. If the file is opened in write access mode using “ab” or “wb”, the list object will be stored into the binary file ReadingLog.dat using pickle.dump(). If the file is opened in “rb” or read-mode, the data is read from the file into a list object using pickle.load(). I created a while loop with try-except to tell the program to load all rows of data until the end of the file EOFError exception is raised. When the exception is raised, I break out of the loop.



**Figure 4 Screenshot of Script's Pickling Features**

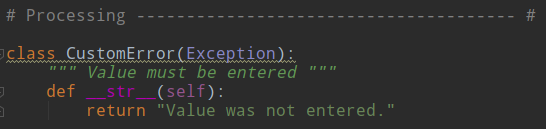
**Exception Handling**

Figure 5 shows a screenshot of the script’s Presentation section, which includes more examples of Exception handling. If the user chooses Option 1, this will open a file for reading. If no file exists, Python will stop running and display an error message, which is called FileNotFoundError. To handle this exception, I used the try-except. I tell the program to try opening the file in read-mode. If it raises the FileNotFoundError, the except block code will be executed, which will inform the user of the error so the user can take corrective action. In addition to printing my own custom error messages, I am also showing how to capture the Exception’s argument in a variable e and to display the built-in python error. Below the specific exception FileNotFoundError, I also included the non-specific error, which should always come after the specific exception, if it is going to be used. It is better to catch specific exception classes, as using the generalized exception class is not very useful for handling.

****

**Figure 5 Screenshot of Presentation Section with Exception Handling**

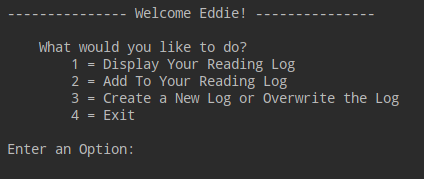
If the user chooses Option 2 (add data to the reading log), the user may not enter a value or may enter an incorrect value type (string vs integer). If the user does not enter a value for the book name, a custom error will be raised. In addition to raising a custom error, I have created a custom exception class, which derives its code from Python’s Exception class. Figure 6 shows a screenshot of the custom exception class that was created. Since the variable intTime is expecting an integer value, if the user enters a string or non-integer value, the ValueError exception is raised and the except block is executed. The while loop will continue until the try block is successfully executed.



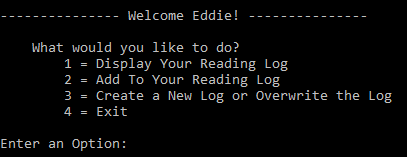
**Figure 6 Screenshot of Custom Exception Class**

**Running the Script**

When the program starts, a menu of options will appear. Figure 7 shows the script running in PyCharm and Figure 8 shows the script running in the Command Window.

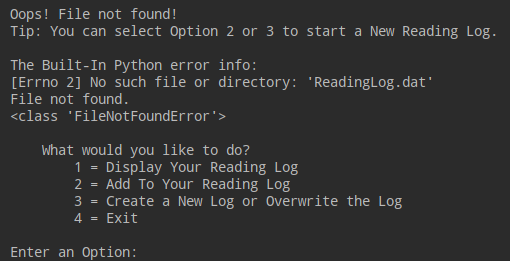


**Figure 7 PyCharm - Menu of Options**

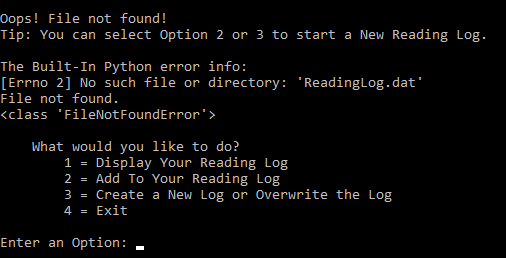
****

**Figure 8 Command Window - Menu of Options**

If user selects Option 1 and no file is found, an error message will appear and return to the main menu. Figure 9 shows the exception in PyCharm and Figure 10 shows the exception in the Command Window.

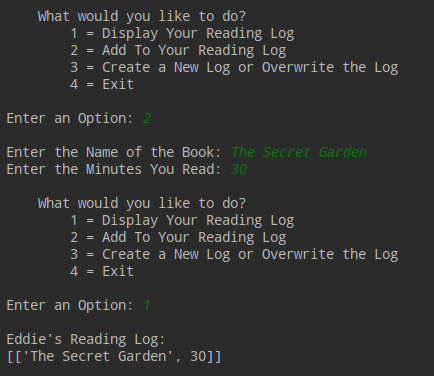


**Figure 9 PyCharm – FileNotFoundError**

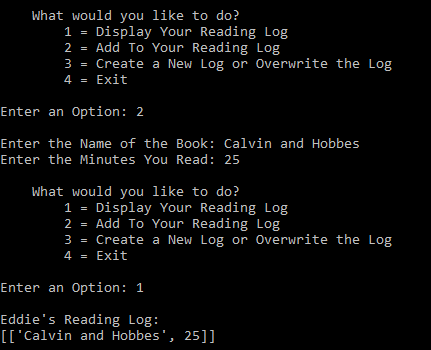


**Figure 10 Command Window – FileNotFoundError**

Figure 11 (PyCharm) and Figure 12 (Command Window) show an example of the user selecting Option 2, which will create a new binary file (in append mode) and allow the user to add new data using pickle.dump(). After the new data is entered, if Option 1 is selected, the program will retrieve data from the binary file by using pickle.load().

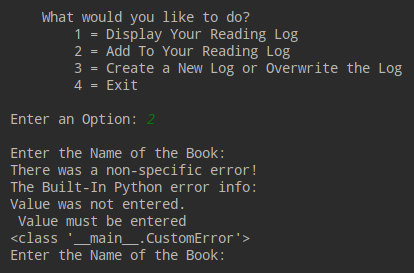


**Figure 11 PyCharm - Option 2 and Executing Pickling**

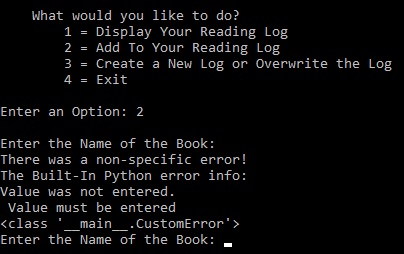
****

**Figure 12 Command Window - Option 2 and Executing Pickling**

If the user selects Option 2 but does not enter a value for the book name, a custom error message is displayed. Figure 13 shows the custom error message in PyCharm and Figure 14 shows the custom error message displayed in the Command Window.

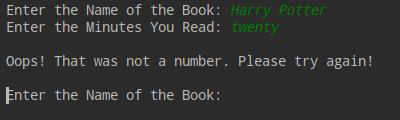


**Figure 13 PyCharm - Option 2 and Custom Exception Class**

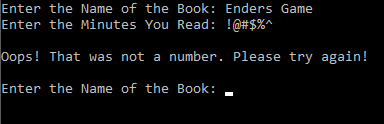


**Figure 14 Command Window - Option 2 and Custom Exception Class**

If the number of minutes is not an integer, a custom error message is displayed. Figure 15 shows the custom error message in PyCharm and Figure 16 shows the custom error message displayed in the Command Window.

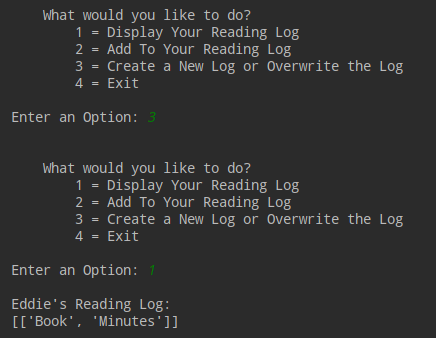
****

**Figure 15 PyCharm - Option 2 and ValueError Exception Displaying Custom Error Message**

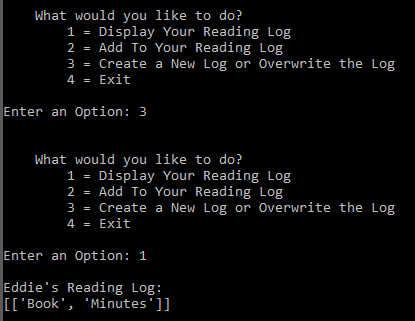
******

**Figure 16 Command Window - Option 2 and ValueError Exception Displaying Custom Error Message**

Selecting Option 3 will either create a new file or overwrite an existing file. Figure 17 (PyCharm) and Figure 18 (Command Window) show Option 3 being selected and then Option 1 is selected to display the reading log, which now shows only the header row containing the values “Book” and “Minutes”.

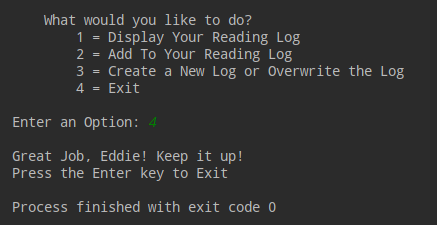
****

**Figure 17 PyCharm - Option 3 then Option 1**

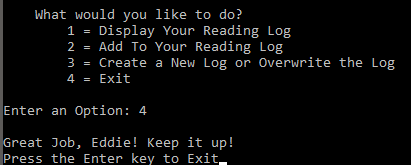
****

**Figure 18 Command Window - Option 3 then Option 1**

If Option 4 is selected, a message appears and instructs the user to press the Enter key to exit. Figure 19 shows Option 4 being selected in PyCharm and Figure 20 shows Option 4 being selected in the Command Window.

******

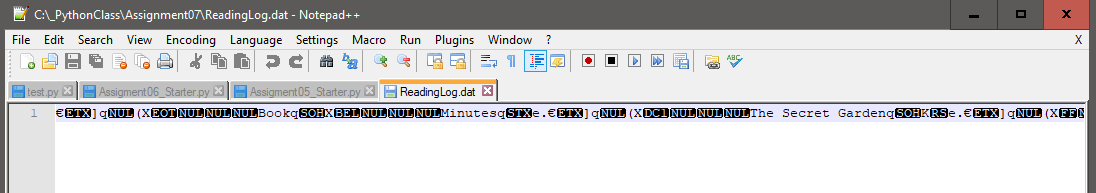
**Figure 19 PyCharm - Option 4**

******

***Figure 20 Command Window - Option 4***

**Verifying the Script Worked**

I verified that the script worked by opening the ReadingLog.dat file in a text editor located in the \_PythonClass\Assignment07 folder. Figure 21 shows a screenshot of its contents.

******

**Figure 21 Screenshot of ReadingLog.dat**

Summary

This week, we took a closer look at reading and writing data to text files, learned how to work with binary files using Python’s pickle module, and were introduced to structured error handling (try-except). For this assignment, I created a script that will allow my 9-year old son to track his reading minutes. If given more time, there are a few things I would like to improve such as how the retrieved data is displayed to the user as well as using a more efficient option to store and manage the data in memory rather than opening and closing the file each time the user chooses Option 1, 2, or 3. In addition, if Option 3 (overwrite) option is selected, I would ask the user to verify that choice as a safeguard before proceeding. Moreover, I would ask my 9-year old son to test the program to see if the custom error messages can be easily understood and corrective action taken by him as well as to help identify additional potential exceptions that should be trapped.