Jessica Trinh

Assignment 07

IT FDN 100A

Assignment 07 Documentation

Github Link: <https://github.com/trinh-j/IntroToProg-Python-Mod07>

Binary Files, Pickling, and Error Handling

# Content Outline

Introduction …………………………………………………………………………………………………………………………………. 1

Pickle Module ………………………………………………………………………………………………………………………………. 1

File types (Binary vs. Text) …………………………………………………………………………………………………………. 2-3

Writing/Appending

Reading

Excepting Handling ………………………………………………………………………………………………………………………. 4

# Introduction

In this assignment, we were tasked with researching the titular topics, writing a basic script to demonstrate their usage, and furthermore describing each topic in documentation similar to Professor Randal Root’s modules and again in a GitHub webpage.

# Pickle

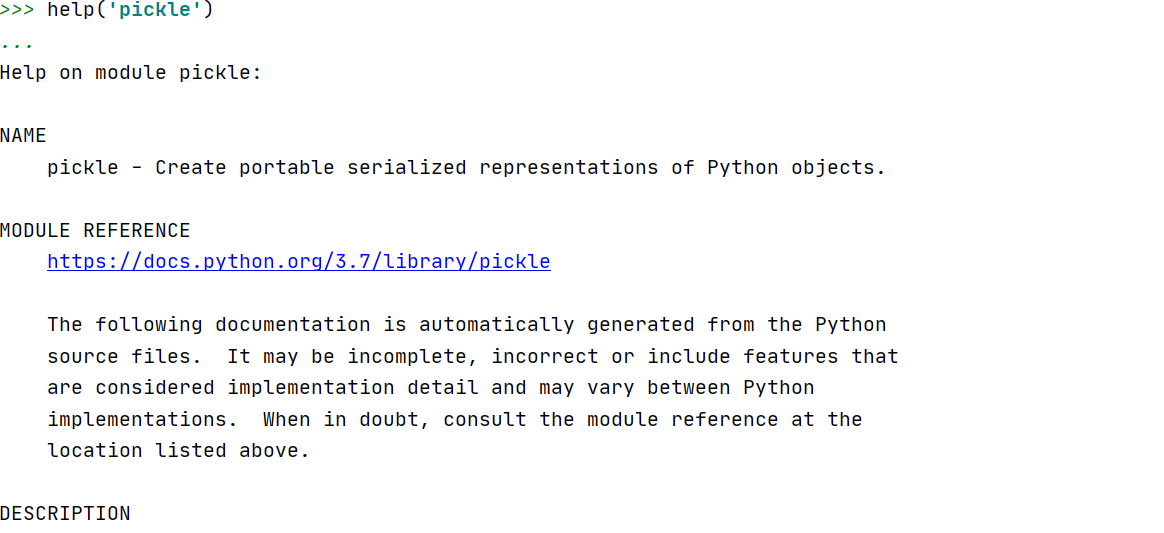
What is **Pickle**? You’re not wrong if your answer was the tangy, zesty refreshing preserved cucumber usually served as a side to a delicious sandwich. In the context of coding, specifically in Python, **Pickle** is a module included in the Python package, that is used to convert object types to a serialized or binary form, to be later referenced when needed. This is especially useful with very large objects that require a lot of processing and/or are used frequently in a program; *without pickling* it may take a while for the object to load into memory every time it needs to be used [[1](1.%09https:/pythonprogramming.net/python-pickle-module-save-objects-serialization/)]. *With pickling*, you are “preserving” the object by processing it into serialized bytes; when the object is needed, it is referenced by its unique serial identity and can be loaded with ease, and without taking up too much of your computer’s memory.

In order to pickle the object, you need to import the module first:



**Figure 1. Import** pickle **module in python script (prior to accessing functions).**

Once imported, you can now access classes and functions associated with the pickle module. To understand the module in greater detail and see available classes and functions, you can type in help(‘pickle’) in your console window just as I have done below and/or visit the link [[2](https://docs.python.org/3.7/library/pickle.html)] listed under MODULE REFERECE (Figure 2).



**Figure 2. Partial output of** help(‘pickle’)**in IDE python console.**

# File Object Types

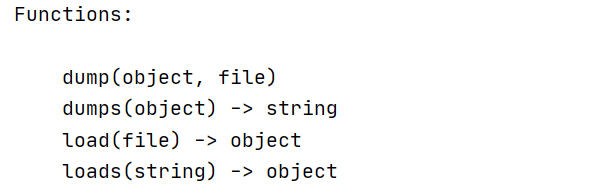
## Text (.txt)

In all of our assignments leading up to this one, we have been working with text files, from which we read and to which we write unprocessed data. Data in a text file is stored as a sequence of characters [[3](https://thepythonguru.com/python-how-to-read-and-write-files/)]. Common access/editing modes accompanied with opening a file are “a” for append, “r” for read, and “w” for write. Text files use the .txt extension.

## Binary (.dat)

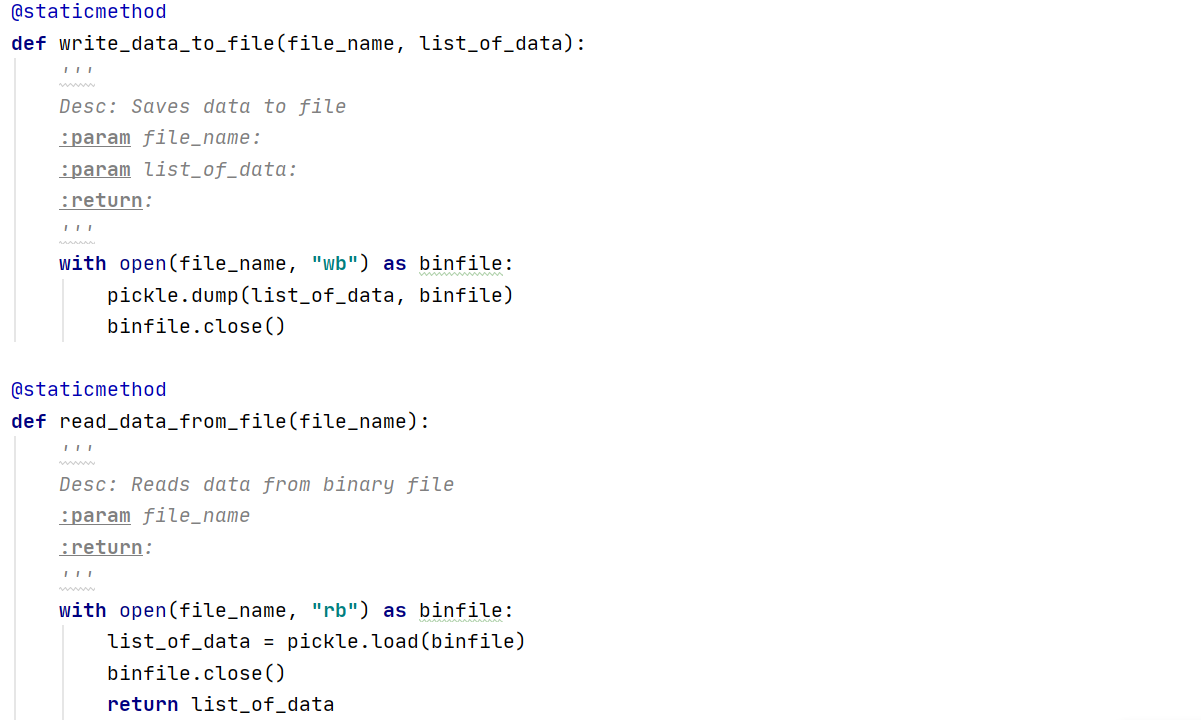
When data is pickled and written to a binary file, it is written in a format comparable to that of computer memory [3]. Similar to creating a text file, creating a file object for pickled data uses the .open() function. However, the most notable difference is that the comparable access/editing modes for binary files, instead, are —“ab” for append, “rb” for read, and “wb” for write; the “b” standing for *byte*. Binary files use the .dat extension.

Using the Pickle module, you can write to and read data from a binary file. There are four functions associated with the Pickle module—load(), loads(), dump() and dumps()—each with pre-specified parameters (Figure 3).



**Figure 3. Functions in** pickle **module.**

To demonstrate how data is written to and read from a file, I will use the dump() and load() functions.



**Figure 4. Two Pickle functions,** dump() and load()**, used in developer-defined functions to write and read binary data.**

For the first function, write\_data\_to\_file, a file object is created using the .open() function, which takes a file, file\_name, and access mode, ‘wb’, as arguments. Next we reference the pickle module and call out the function dump, with pickle.dump(). This function’s parameters are an object—*what you want to dump*, and a file—*where you want to dump it* (see Figure 3); the object passed in this function will be serialized and stored to the given file. Lastly, for this function, we close the file.

Again, for the next function read\_data\_from\_file()we initiate the file object using .open(). Then we use the load function from the pickle module, pickle.load(), to grab and convert the binary data that is stored in the file back to an object. Finally, we close the file and return the object saved to variable list\_of\_data.

# Exception/Error Handling

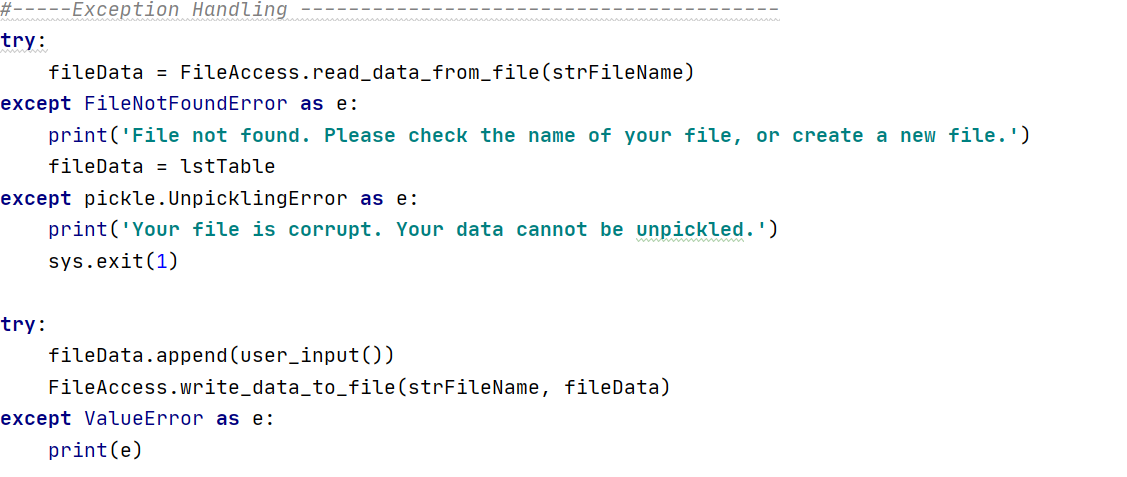
When a program runs into a problem it raises an *exception*, or an error message that something has happened which cannot be processed by the program [5]. Often, these error messages can be lengthy and daunting; although they are useful to the developer, the message may not be appealing or readable to the user. Additionally, it is typical that the program crashes and ends immediately upon raising an exception. We can avert this by trapping the problem using *exception handling*. The *try-except* method is a basic way of handling exceptions/errors.

A reason why the developer may want to use exception handling is to display the error message more concisely, rather than parsing through a lengthy default error message. A concise, readable message is also useful for the end-user, should they run into any issues using a program.

Errors fall into different levels, from generic to specific. The format of a try-except block is organized such that specific errors are captured first.

Similar to the pickle module, before we can use any functions associated with exception handling we need to import the sys module using **import** sys [[4](https://www.programiz.com/python-programming/exception-handling)].

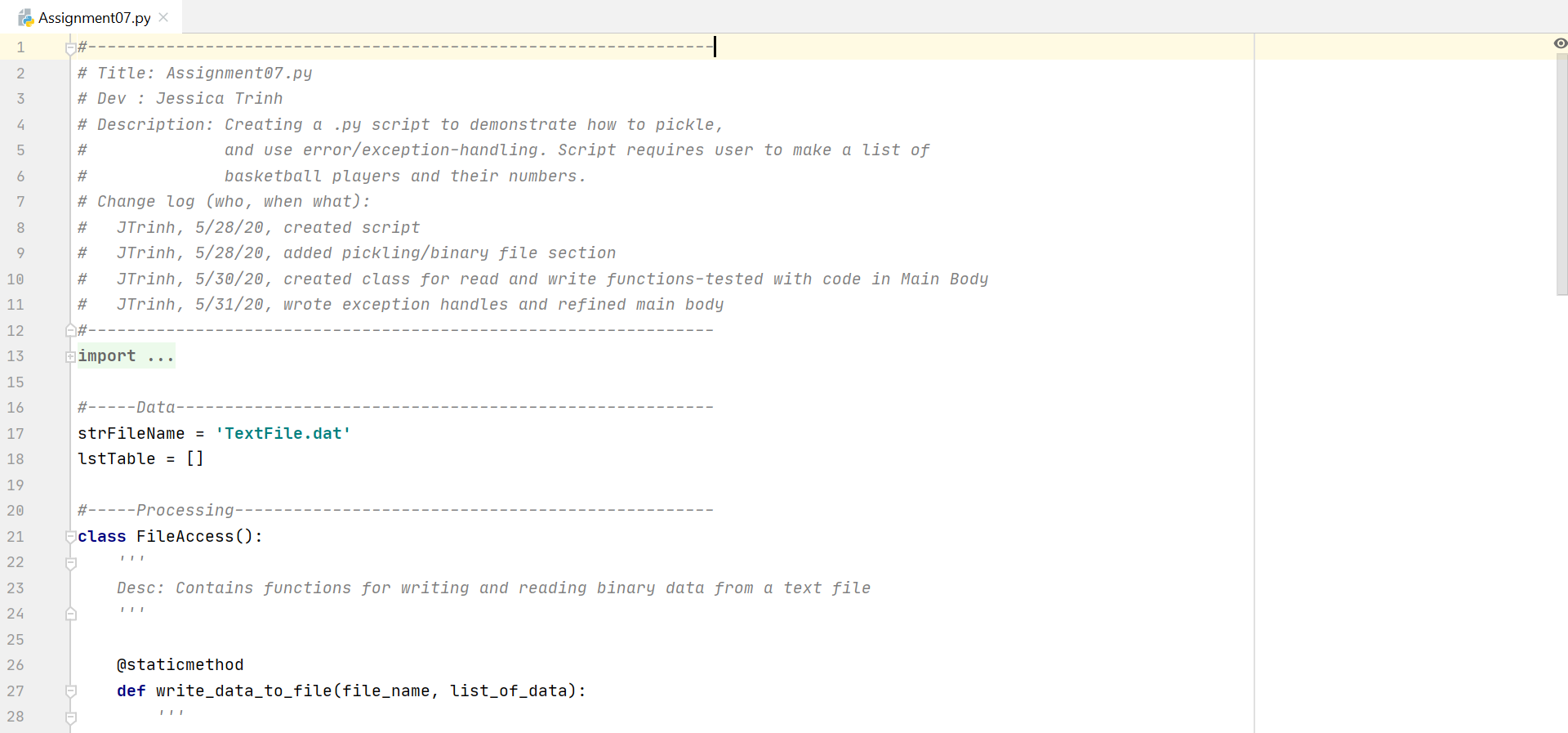
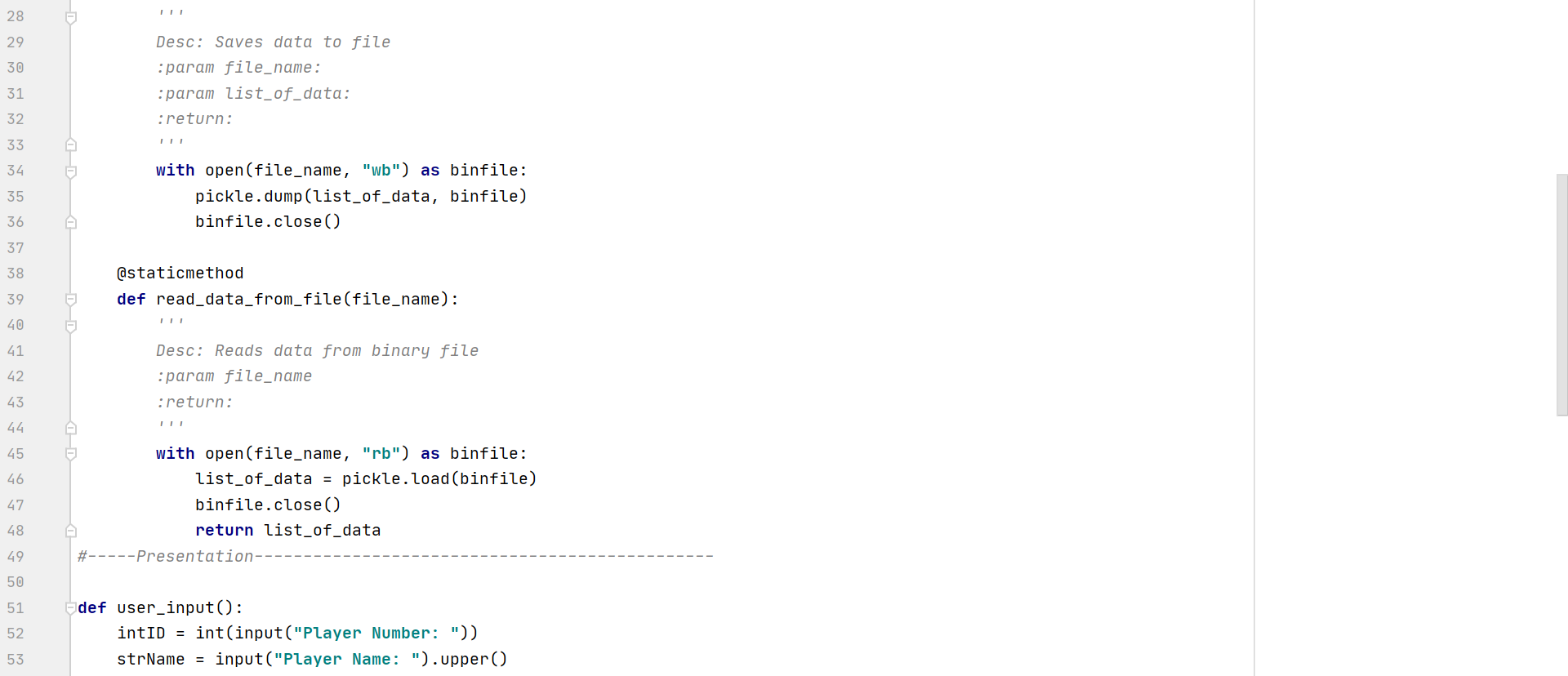
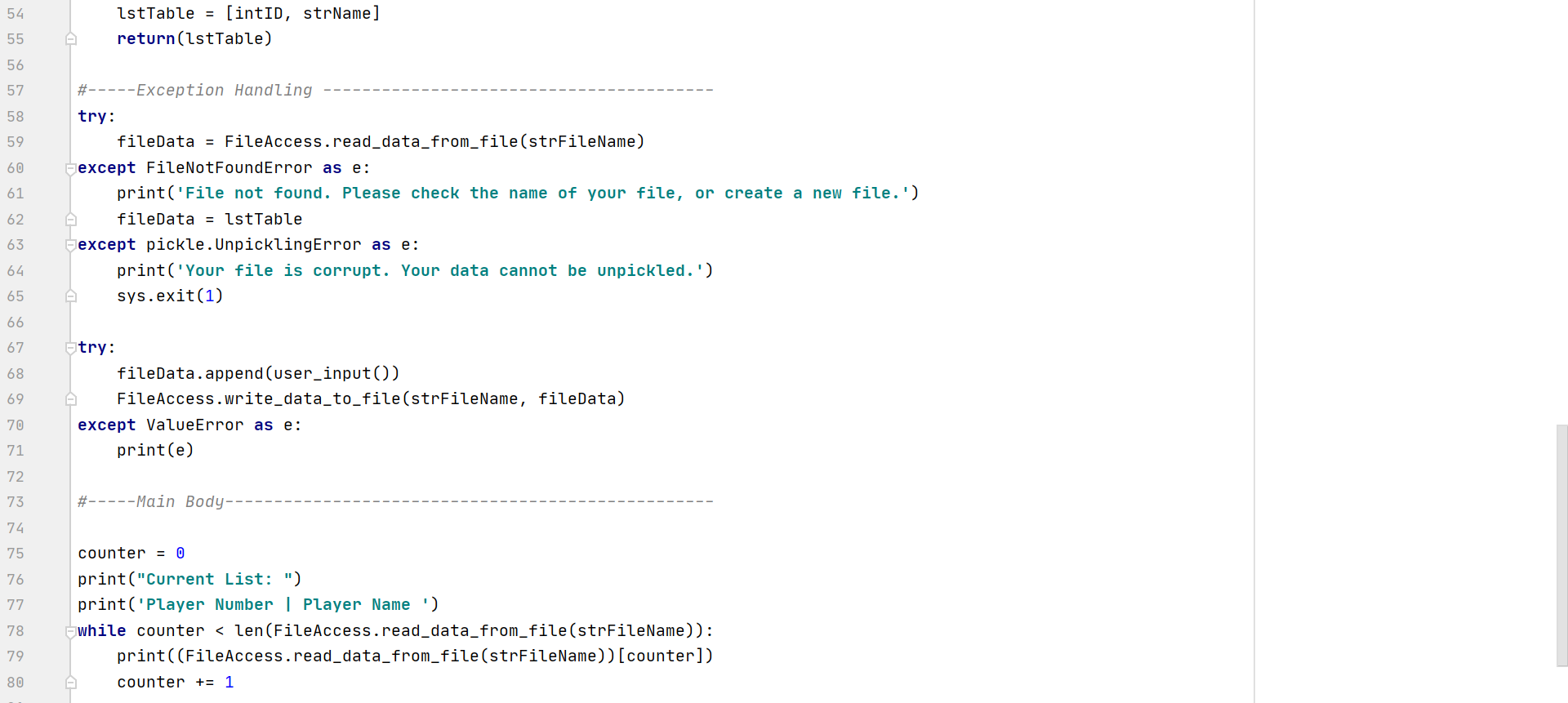
Below (Figure 5) try-except blocks are used to sift through errors, from most to least specific. The first try-except block finds errors with file-handling, while the second try-error block looks for errors regarding saving/writing data to the binary file.



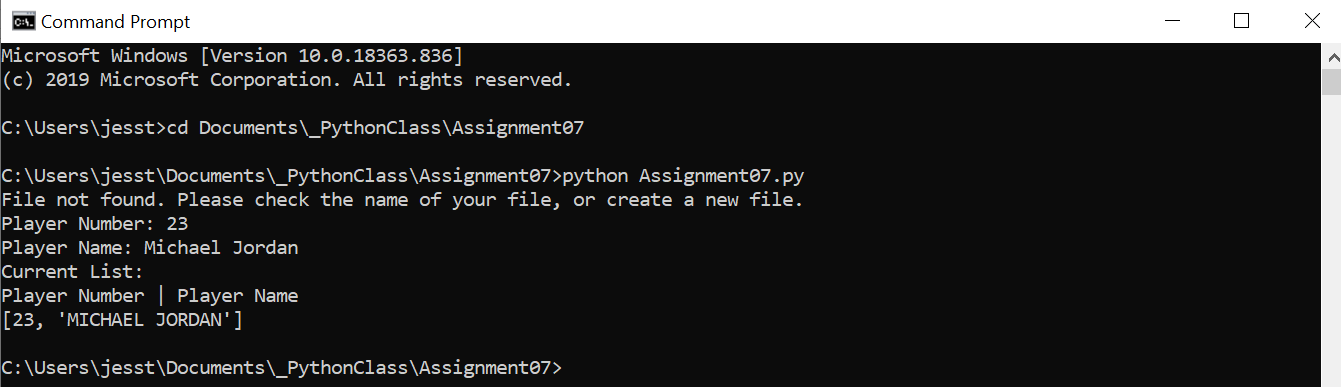
**Figure 5. Try and except blocks to catch errors in the program and display error messages in a user-friendly manner.**

Using pickling and exception handling, we can piece together a script. For this assignment, I have created a script (Figure 6) to ask the end-user for (Basketball) player numbers and names.

# The Script

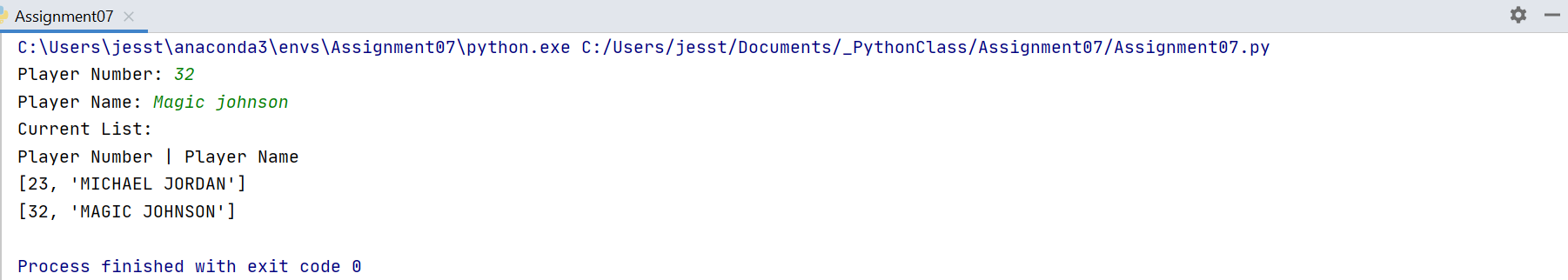
   **Figure 6. Full script with appropriate arguments/parameters in place.**

To confirm that the script functions in not only the IDE console, I tested it in the command line (Figure 7). As you can see, the binary file did not exist, so a message was produced from the first try-except block in figure 5. Once bypassed, the program creates the file itself and proceeds to ask the user for information to store into the newly created binary file.



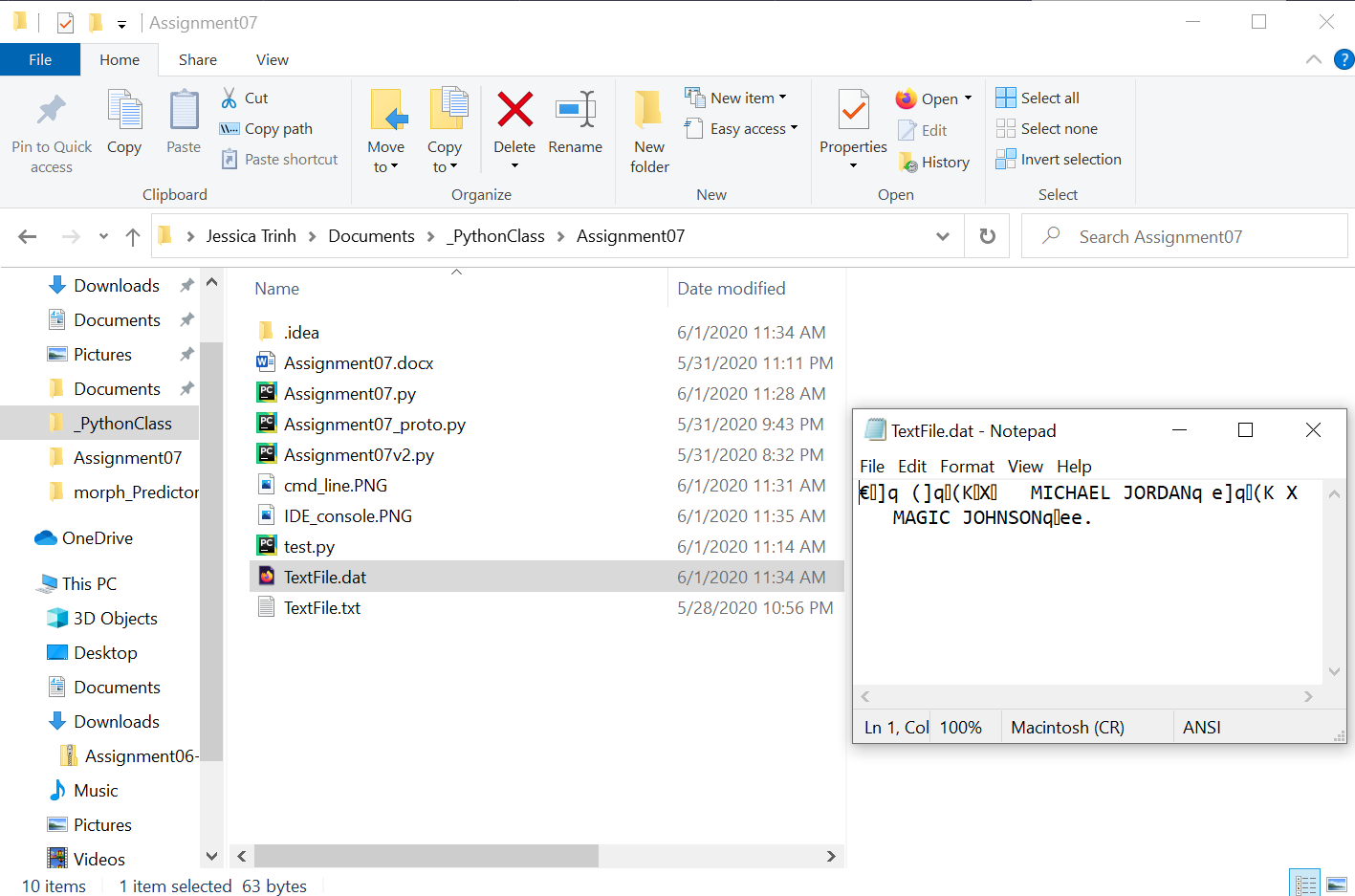
**Figure 7. Script executed in the command line.**

Below (Figure 8), the program is ran for the second time, this time in the IDE console. Since the binary file has already been created, the program is able to locate the file and use it to append more end-user data. Here, we have added Player 23, Magic Johnson, and appended this list to our existing data from the binary file.



**Figure 8. Script executed in the IDE PyCharm console.**

To confirm that the end-user’s data is saved to a binary file, we can go to the file directory and open the binary file in a text editor (Figure 9). We can see that upon opening the .dat file in a notepad, that the data in it is encripted.



**Figure 9. Binary file TextFile.dat located in program directory. File is opened in a notepad (right side of figure) with encrypted data.**

# Summary/Discussion

For this assignment, we were challenged with doing our own research on the titular topics and creating a scrip to demonstrate how to use these topics properly. Although this assignment was supposed to be “fun and easy” I found it quite frustrating, mainly due to lack of direction. There were things discussed in the recorded lectures that were not necessary for the completion of this assignment (i.e. readline()), but I did learn some. I had a lot of text scripts/different version of this assignment to test (and fail) many times, and I plan to continue to do so as I believe that I still have room to improve my understanding of pickling and exception-handling. Nonetheless, this assignment felt like a change of pace from the last six we have completed, and even though for the first several days I was thrown for a loop, it was a good challenge.

# Resources

1. <https://pythonprogramming.net/python-pickle-module-save-objects-serialization/>
2. <https://docs.python.org/3.7/library/pickle.html>

1. <https://thepythonguru.com/python-how-to-read-and-write-files/>
2. <https://www.programiz.com/python-programming/exception-handling>
3. (Chapter 7 in the course textbook)