Ryan Fear

11/28/2021

IT FDN 110 A

Assignment 07

<https://github.com/RFear/IntroToProg-Python-Mod07>

Pickling and Structured Error Handling

# Introduction

This document will discuss the concept of pickling in python. Structured error handling and using exception classes will also be discussed. Each of these items are discussed in their respective sections. Each section defines the concept, provides a programming example, a description of what is occurring within the programming example, and additional resources that may be helpful. Understanding these core python principals are essential to a beginning python programmer.

# Pickling

In python pickling refers to serializing and de-serializing python objects to a file. Python objects which can be pickled include list, dictionaries, etc. One example of why this is useful is that a pickled python object can be open a reconstructed in another python script. For example, say one colleague is running an experiment and collecting a lot of data. They could “pickle” the data and send it to other colleagues who could run analyses on the data set.

## Programming Example

An example of pickling data to a file and opening a pickled file for data extraction is shown in Figure 1.

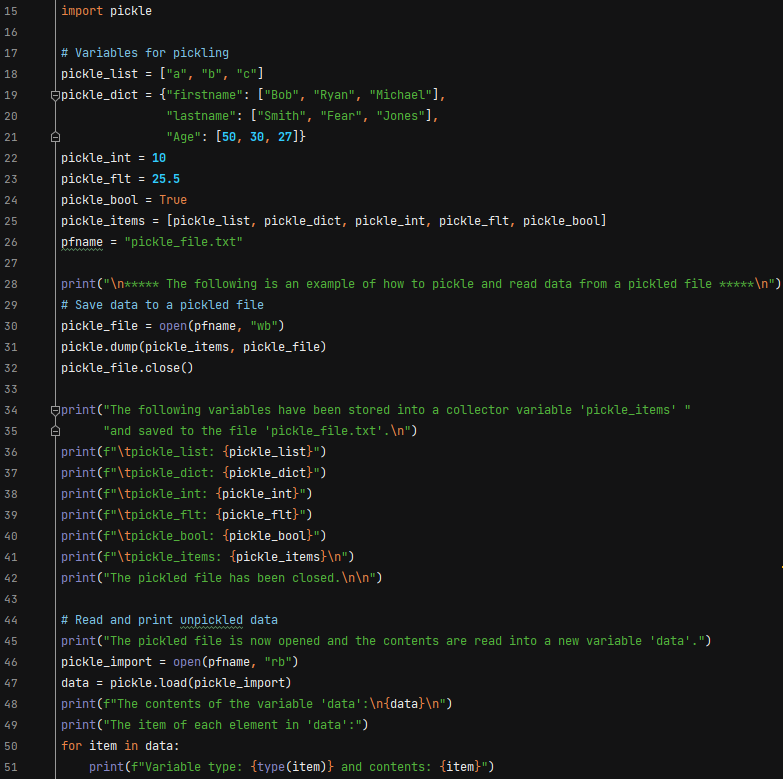


Figure : Pickling Example in Python

## Program Execution

The program execution in the PyCharm environment is shown in Figure 2.

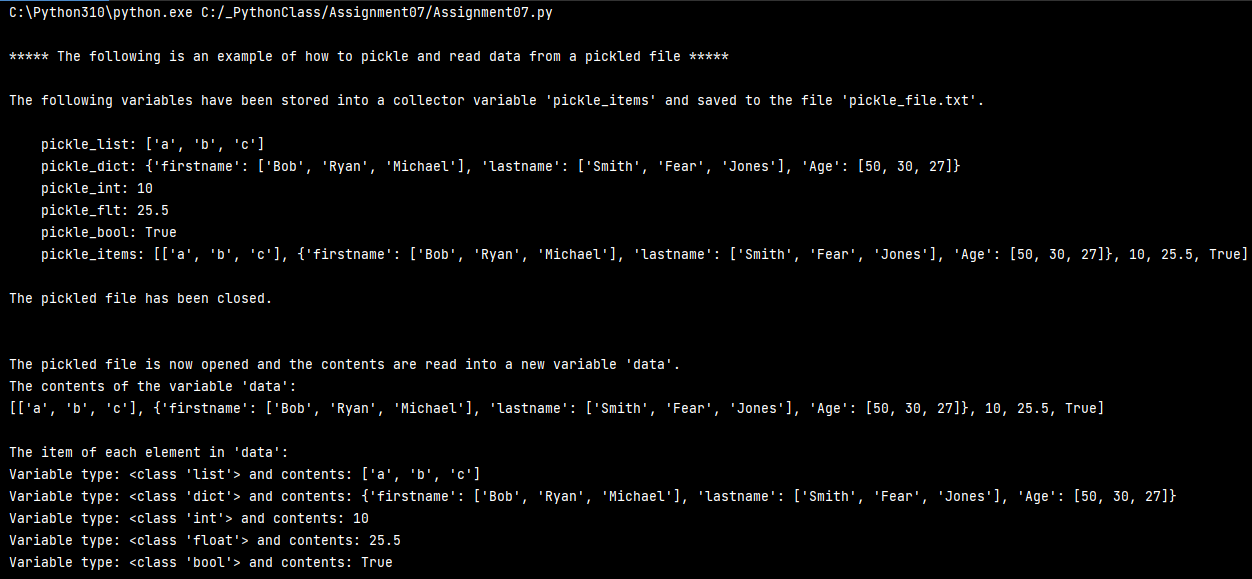


Figure : Program Execution of Pickling - PyCharm

The program execution in the Windows command line is shown in Figure 3.

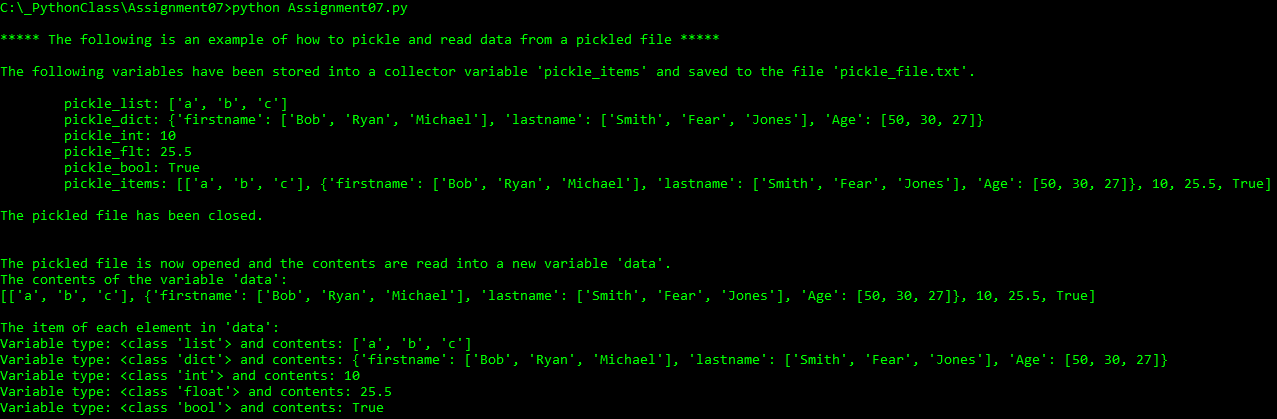


Figure : Program Execution of Pickling - Command Window

## Program Explanation

First, I import the pickle module on line 15. Next I define a list, dictionary, integer, floating point, and a Boolean then collect all of these items in a list called *pickle\_items*. I then save this data to a file called “pickle\_file.txt” with the code in lines 30-32. The contents of this file are shown in Figure 4, the data is written to the file in binary. Although humans may not be able to fully understand the content in this file, it is perfectly useable by the computer. In lines 34-42 I print to the screen what has been accomplished.

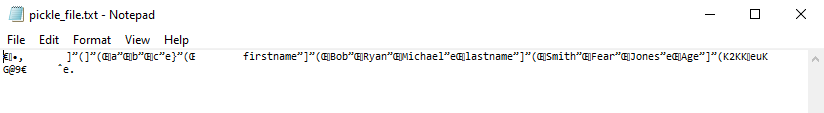


Figure : Content in pickle\_file.txt

The code in lines 46-47 is used to read in the data from “pickle\_file.txt”. Note, in this example the data exported was in a list, so when I read in the information, the variable *data* is a list with all of the contents from the variable *pickle\_items* defined on line 25. To prove this, I print out the data to the screen on lines 48-51.

## Additional Helpful Resources

A YouTube video and two web pages that discuss the topic of pickling in python further are included below.

<https://www.youtube.com/watch?v=2Tw39kZIbhs> [external site]

<https://www.datacamp.com/community/tutorials/pickle-python-tutorial> [external site]

<https://www.geeksforgeeks.org/understanding-python-pickling-example/> [external site]

# Structured Error Handling

It is common that when a program is interacting with a user errors will occur. When errors occur, and are not handled, the program could stop executing and the user is left frustrated that they cannot use your code. One way to deal with errors is structured error handling. This is commonly dealt with what is called a “try-exception” block of code. The logic for this block of code is shown in Figure 5. Note, there are additional ways to handle errors by including “else” and “finally” blocks of code and Figure 5 describes what happens if these are used.



Figure : Try-Except Code Block

## Programming Example

An example of structured error is shown in Figure 6.

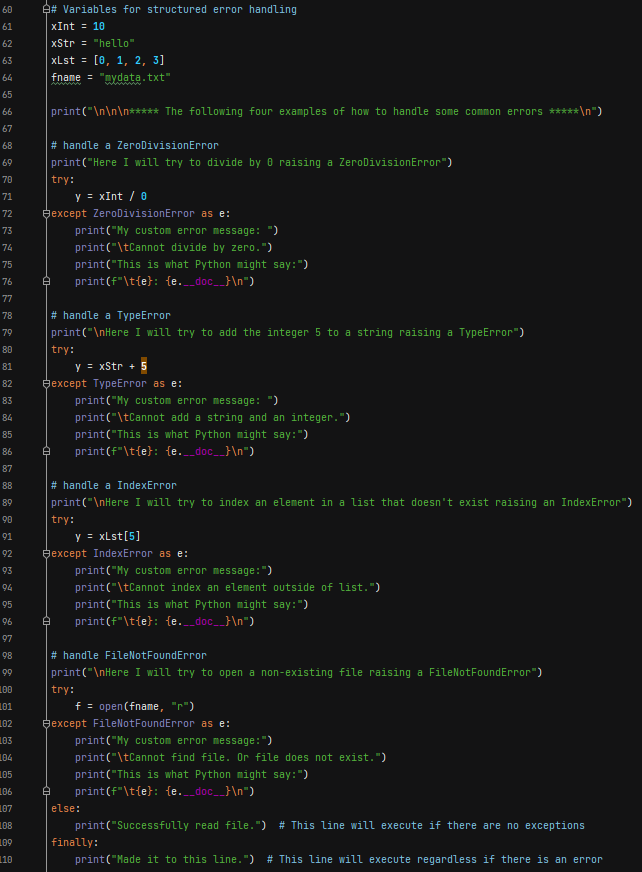


Figure : Structured Error Handling Example in Python

## Program Execution

The program execution in the PyCharm environment is shown in Figure 7.

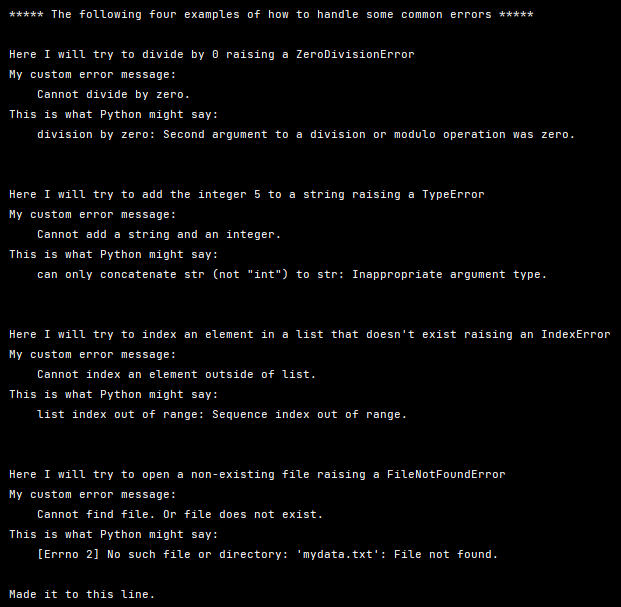


Figure : Program Execution of Structured Error Handling - PyCharm

The program execution in the Windows command line is shown in Figure 8.

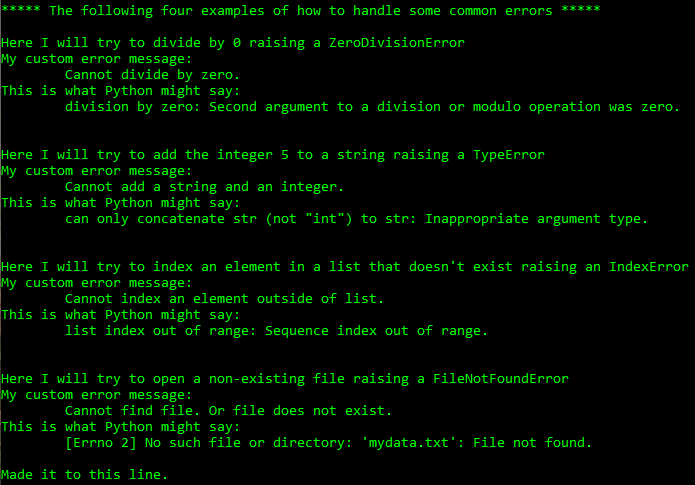


Figure : Program Execution of Structured Error Handling - Command Window

## Program Explanation

For the example I begin by defining some variables that will be used in lines 61-64. In the next four try-exception block I show common exceptions that are raised, a custom error message, and the message python may print out. The four exceptions are ZeroDivisionError, TypeError, IndexError, and FileNotFoundError.

In lines 70-76 I show an example of dividing by zero. If a program is performing arithmetic and this situation occurs this exception can be raised and another option could be entered. In lines 80-86 I show an example of a type error. This can occur when you are trying to perform arithmetic with two different variable types such as a string or an integer. In lines 89-96 I show an example of an IndexError, this can occur when trying to access an element of a list or dictionary that doesn’t exist. Finally, in lines 100-110 I show an example of trying to access a file that doesn’t exist.

Note, with the FileNotFoundError the “else” and “finally” blocks are used. As Figure 5 indicates, the “else” block executes when no exceptions are raised and the “finally” block will always execute. I will now create the file “mydata.txt”, notice the change in Figure 8 from Figure 7. Since the file “mydata.txt” now exist line 108 executes.

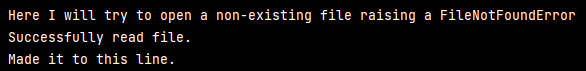


Figure : Else/Finally Execution

## Additional Helpful Resources

A YouTube video and two web pages that discuss the topic of structured error handling further are included below.

<https://www.youtube.com/watch?v=nlCKrKGHSSk> [external site]

<https://www.datacamp.com/community/tutorials/exception-handling-python> [external site]

<https://www.w3schools.com/python/python_try_except.asp> [external site]

# Summary

In this document the concepts of pickling and structured error handling in python were discussed. Programming example were provided along with demonstrations of their execution. Descriptions of program execution were also included. Additional resources to YouTube and other helpful web pages were also provided. With the concept of pickling and structured error handling a newer python programmer can now work with serialized data as well as gain greater control over how their code deals with errors due to external inputs.