Sarah Anderson

November 27, 2022

ITFDN 110

<https://github.com/sarahanderson94/ITFdn110_MOD07>

Assignment 07: Exception Handling and Pickling Demo

**Introduction**

This week I learned about exception handling and pickling. Exception handling is a process that allows the programmer to create special error messages for possible errors that they think a user might encounter. When Python finds an error, it spits out an error message, however, these messages are not always intuitive for a user. Exception handling is a way to make the error messages more useful to the user. Pickling is a way to save data in a binary format versus a plain text format. Some benefits of using pickling is that using binary format makes the information more obscure to human eyes and can reduce the size of the file. To demonstrate exception handling and pickling, I created a simple program that takes in user entered data and does simple division. The user entered data and quotient output are stored in a dictionary that is written to a binary file.

**Exception handling**

Exception handling is a tool in programming that allows the developer to control the error messages displayed to the user. The most basic format of exception handling is using a try statement with an except clause. With this format, some code that could potentially raise an error is sectioned into the try statement and the except clause is executed only if an error is triggered. The except clause can be specific or non-specific. A nonspecific except clause will execute if any error is encountered in the try code. Figure 1 shows an example of the most basic format with a non-specific except clause. In figure 1, the user entered a string of characters which could not be converted to a float value, so the Except clause was triggered and displayed the programmed error message. Figure 2 shows the error message built into Python for the same error of entering a string of characters that could not be converted to a float. This error message is much messier and may be difficult for the user to understand.

Text

Description automatically generated

Figure 1: Basic Try Statement with Except Clause

Text

Description automatically generated

Figure 2: Error Message without Try Statement and Except Clause

The Except clause could be specified. Instead of using the basic Except clause, I could call a specific Exception Type, such as ValueError. The exception would only be raised if the code in the try section raises a value error. If another error is encountered, such as a zero-division error, then the built in Python error message would be displayed.

It is possible to add multiple except clauses to one try statement. This allows the programmer to create different error messages depending on the error that is raised. Figure 3 shows an example of a try statement with multiple exception types. The code calls the divide() function, which asks for a user to input two numbers that are converted from strings to float values. Value 1 is then divided by value 2. The values are saved in a dictionary object. If the user enters a character that cannot be converted to a float the ValueError exception is raised and the print message from line 44 is displayed. If a user enters zero for value 2, then the message on line 46 is displayed. The last except clause is a catch all for any other errors raised. The last except clause captures and stores the Exception object when the error is raised. It saves it in “e”. By printing out information about “e”, the user gets more information on what the error was besides just seeing the message “There was a non specific error”.

Text

Description automatically generated

Figure 3: Handling Multiple Exceptions

Text

Description automatically generatedFigure 4: Raising a Custom Error

**Pickling**

Pickling is a process that converts python objects into a byte stream (binary file) and unpickling is the opposite, where a byte stream is converted into a python object. At the end of my demo program, I take the table of values imported from the text file and entered by the user and pickle them to a binary file. Figure 5 shows the code to pickle to a file in lines 79 through lines 82. The code is very similar to opening a text file, but you must import the pickle library first. Instead of using “w” to write to a file, you use “wb” which stands for “write bytes”. The dump() function will dumb the entire list object, “dataObject” to the new file object, “pickleFile”. Figure 5 also shows the code used to unpickle in lines 85 through 87. When opening the file to read data, you use “rb” which stands for “read bytes”. The load() function will load the entire list of dictionaries into “data”.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5: Pickling and Unpickling in Python

**Running the Demo**

Figure 6 and 7 shows the demo running in PyCharm and command console, respectively. Figure 8 shows the binary file created in the demo in the Notepad app. In the command console it cannot find my text file and does not work the same as when it runs in PyCharm. I will need to look into how the demo can function the same in Command Console.

**Text

Description automatically generated**

Figure 6: Demo running in PyCharm

Text

Description automatically generated

Figure 7: Demo running in Command Console

Graphical user interface, text, application

Description automatically generated

Figure 8: Binary File in Notepad

**Summary**

This assignment I learned how to use exception handling, pickling, and formatting a web page on github. It was challenging to create the demo on my own.