Patricia Coonrad

June 01, 2020

IT FDN 100 A Sp 20: Foundations Of Programming: Python

Assignment 07

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

Exception Handling and Pickling in Python

# Introduction

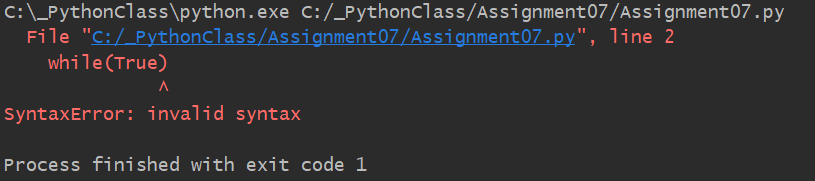
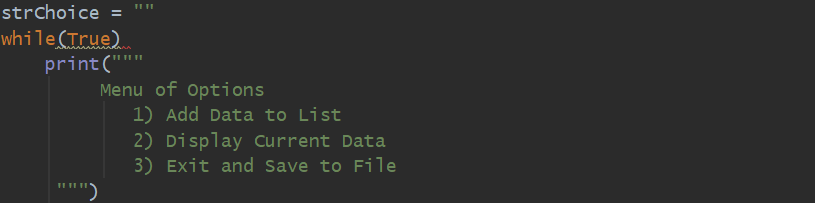
In this assignment I learned about Exception Handling and Pickling in Python. Besides watching the lecture video and reading the book chapter, I researched additional documentation and videos online about both topics. Using what I have learned, I will demonstrate how to use these Python features by creating two simple scripts.

# Errors and Exceptions

A Python program will stop processing and will display an error message when it encounters an error it cannot handle during its execution. These errors can be usually divided into two main categories: Syntax Errors, and Logical Errors or Exceptions.

A Syntax Error happens when the Python cannot understand a line of code. The majority of syntax errors are typos, incorrect indentation, or incorrect arguments.

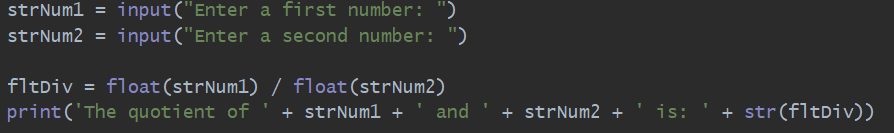
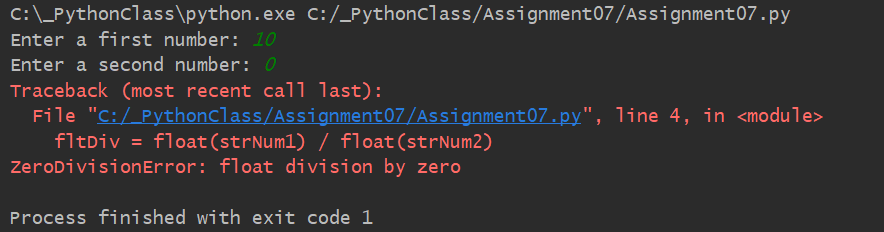
In below example, the while (True) syntax is missing the colon (':') at the end, so when the program was executed, a SyntaxError was raised (Figure 1).



***Figure 1. SyntaxError example***

A Logical Error or Exception happens when the syntax is correct but an error occurs when the code is executed. When this occurs, if the exception is not properly handled, Python will display a ‘Traceback’ message including details about the error such as where the exception happened, line of code, exception type and what caused it.

In the below example, the second argument of the division operation is zero, so when the program was executed, a ZeroDivisionError exception type was raised. This is one of the existing Python built-in exceptions (Figure 2).

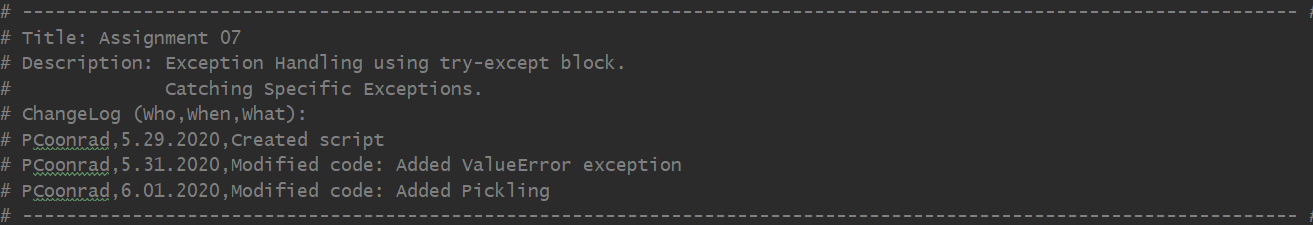
 

***Figure 2. ZeroDivisionError example***

# Demo Exception Handling Using Try-Except Block

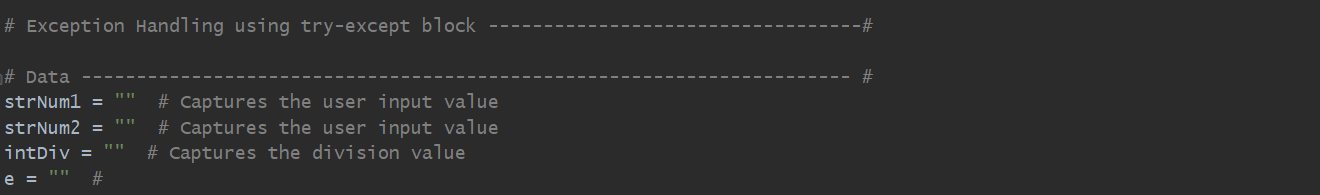
I have created the following script to demonstrate Python’s exception handling functionality which is used so that the program does not end abruptly or crashes. It is a way to handle errors more gracefully and provide more meaningful information to the user. I used try-except, which is a try statement with an except clause, and I added an else clause at the end. In this demo, the program requests the user to enter two numbers, and then it calculates the quotient of the two numbers.

I started by adding the script header at the beginning of the script which includes information about the script such as title, brief description of the program, and change log (Figure 3).



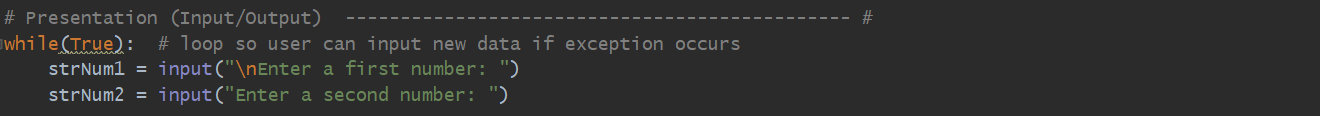
***Figure 3. Assignment07 Script header***

I have organized the code using ‘Separations of Concerns’. And under the Data layer, I defined the variables used when running the Exception Handling script (Figure 4).



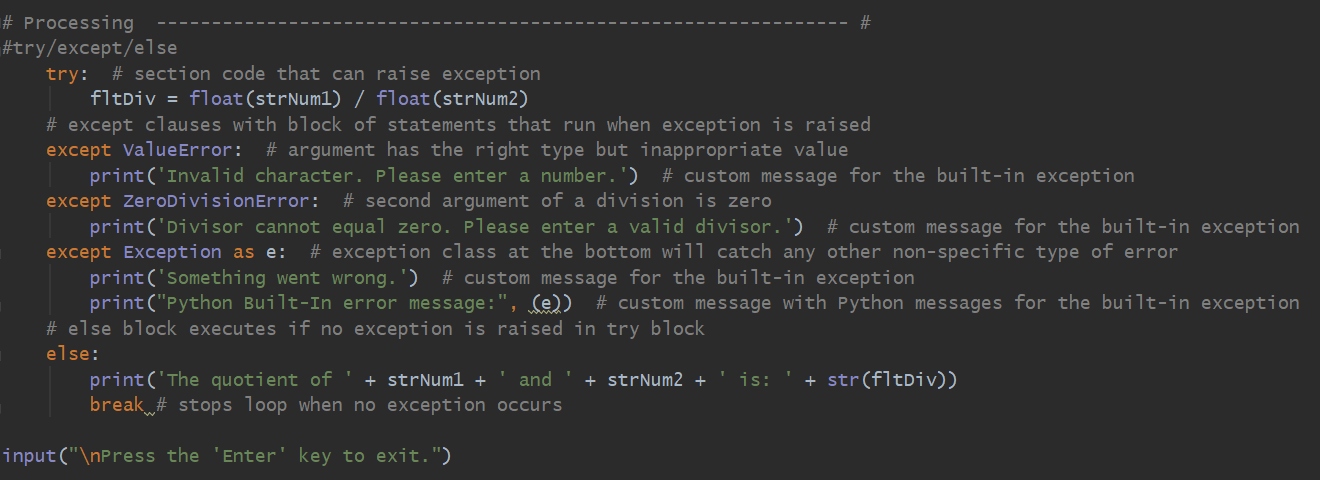
***Figure 4. Defining variables***

Under the Presentation (Input/Output) layer, I used the input() function to request information from the user. Note that I added a while loop so if the user hits an exception during execution, the program will loop back and prompt user for new input (Figure 5).



***Figure 5. Requesting user input***

Under the Processing layer, I used the try-except block to handle the exceptions that might happen in case the user inputs incorrect data. I listed the two specific exceptions first so that the program would execute those before executing the more generic exception that catches all errors. The exception type ValueError was raised if the user entered a character other than a number. The exception type ZeroDivisionError was raised if the user entered the number zero as the divisor. And the Exception type was raised for any additional exception that may occur. If any of these exceptions were raised, the system would display a custom error message for each exception type. Note that the message for the Exception type contains the custom message as well as some information from the built-in Python message. Finnaly, the input() function at the end prompts the user to press ‘Enter’ to exit the program (Figure 6).



***Figure 6. Try/except/else block***

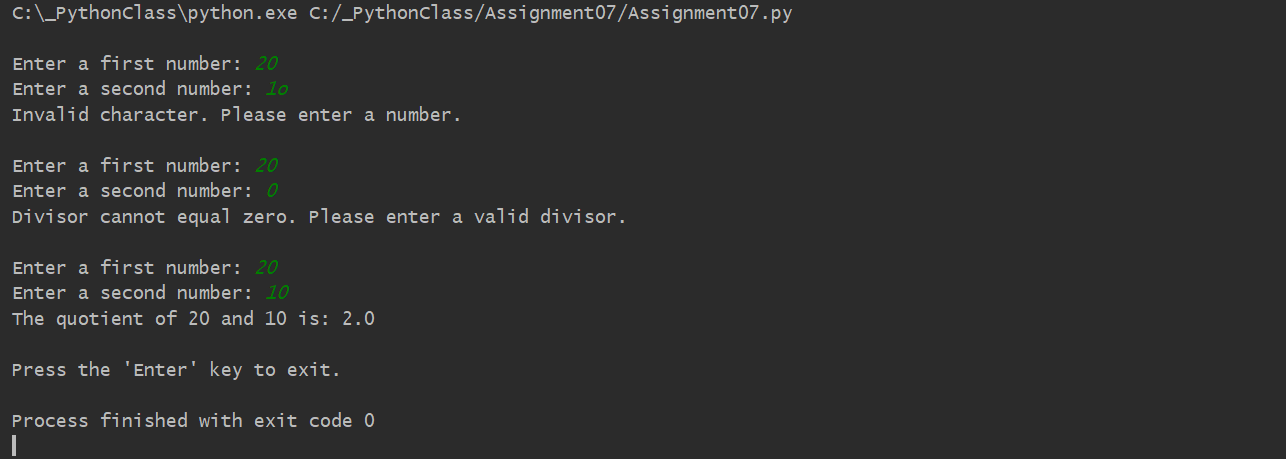
From the research I have done, the two below sites stood out regarding Exception Handling. The first one is a quick introductory video with simple demos of different exception types. The second site also provides an overview on the matter. It breaks down exception handling in language easy to understand and walks you through various examples. Both helped me better understand this subject.

<https://realpython.com/lessons/assertions-and-tryexcept/>

<https://www.freecodecamp.org/news/exception-handling-python/>

# Running Exception Handling Script Using PyCharm

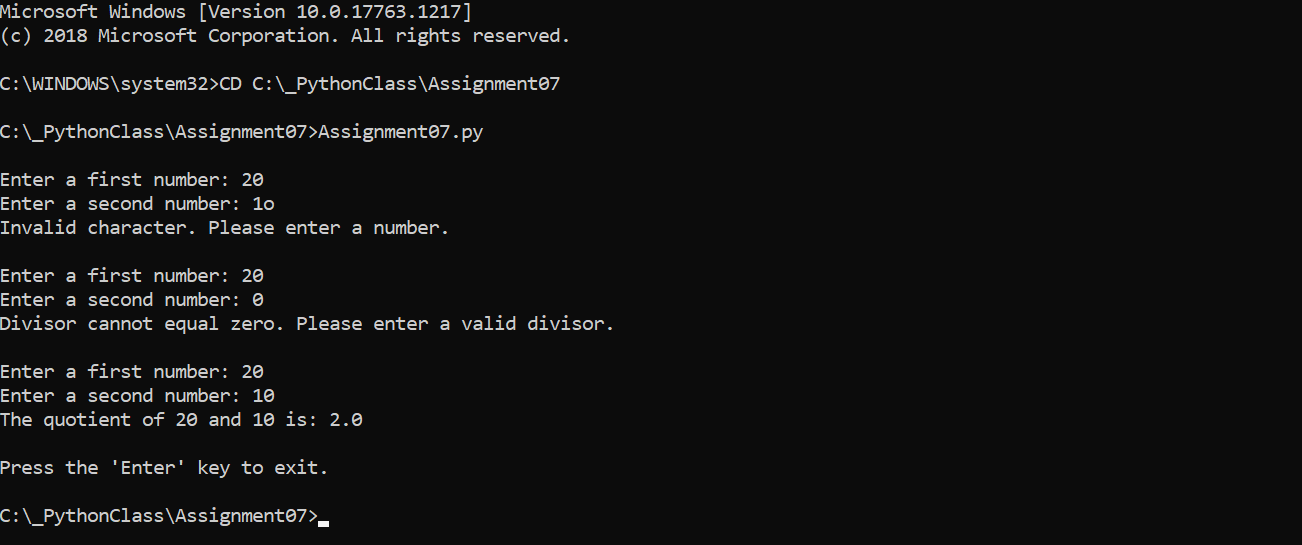
After the code was complete, I successfully ran the program using PyCharm (Figure 7).



***Figure 7. Screenshot of the script running in PyCharm***

# Running Exception Handling Script Using the Command Window

And I also successfully ran the program using the Command Window (Figure 8).



***Figure 8. Screenshot of the script running in the Command Window***

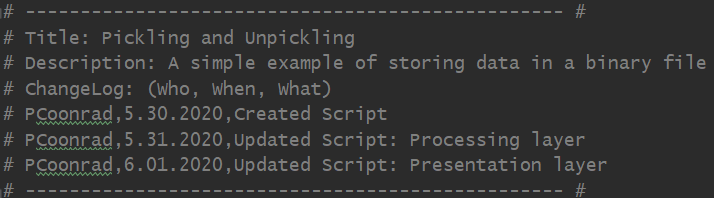
# Pickling

Binary files are another option for storing data in a file besides plain text. One of the advantages it that binary files are obscure (unreadable), and they can be smaller when you have a large data set. Pickling is the process of converting complex data into byte stream (0s and 1s). This process is also referred to as serialization. When we convert the byte stream back into a Python object, it is called Unpickling. Pickling is often used to store complex or large data. Pickle can store and reproduce list, dictionaries easily. The dump() method is used to save the object to the file, and the load() method loads the object in the file and returns the object.

# Demo Pickling

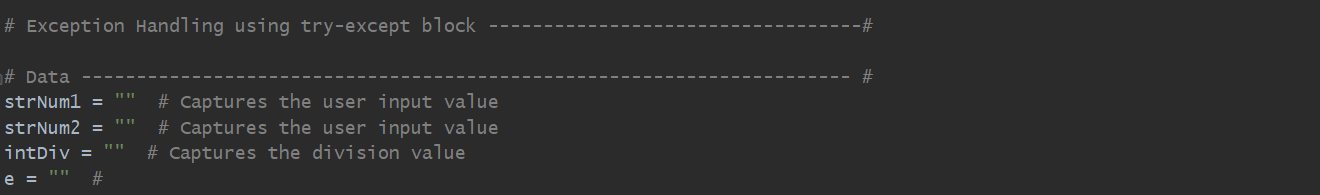
I have created the following script to demonstrate Python’s pickling functionality. In this demo, the program requests the user to enter an ID and a Name and stores the information in a list. The list is saved to a binary file, and then the data is later read from the file into a list object.

I started by adding the script header at the beginning of the script which includes information about the script such as title, brief description of the program, and change log (Figure 9).



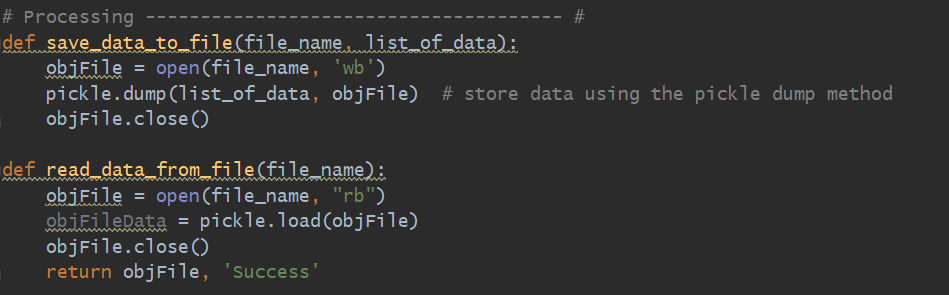
***Figure 9. Assignment07\_Pickling Script header***

I have organized the code using ‘Separations of Concerns’. And under the Data layer, I defined the variables used when running the Pickling script (Figure 10).



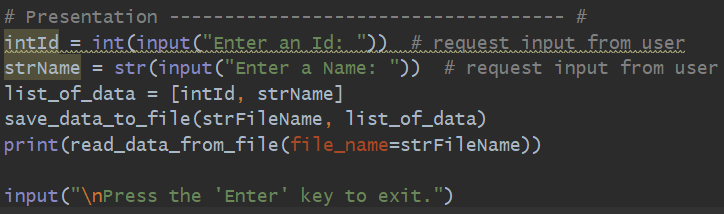
***Figure 10. Defining variables***

Under the Processing layer, saves the data to file and reads data from file using the data dump and load functions (Figure 11).



***Figure 11. Saving data to and reading from binary file***

Under the Presentation (Input/Output) layer, I used the input() function to request information from the user (Figure 12).



***Figure 12. Requesting user input and creating list***

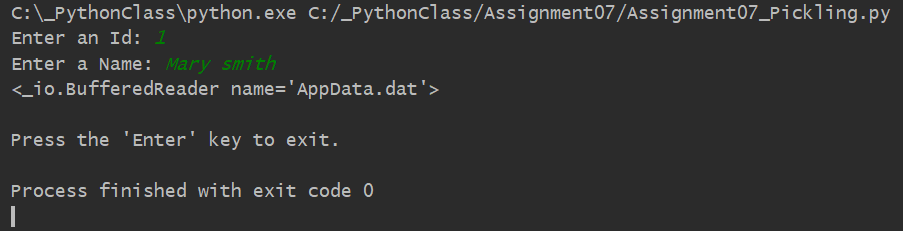
While researching about Pickling, I came across two sites that were very helpful. The video gives a quick overview and demo about Pickling and demonstrate the basic function of Pickling. The second site provides more in depth knowledge in an accessible language and approach. Both helped me better understand what Pickling is and how it works.

<https://www.youtube.com/watch?v=Pl4Hp8qwwes>

<https://www.pythoncentral.io/how-to-pickle-unpickle-tutorial/>

# Running Pickling Script Using PyCharm

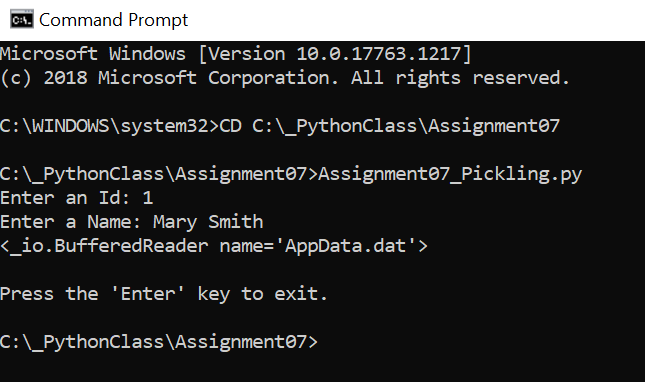
After the code was complete, I successfully ran the program using PyCharm (Figure 13).



***Figure 13. Screenshot of the script running in PyCharm***

# Running Pickling Using the Command Window

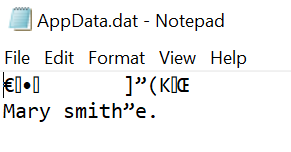
And I also successfully ran the program using the Command Window (Figure 14).



***Figure 14. Screenshot of the script running in the Command Window***

# Verify that it Worked

After I ran the program, I located the file and opened it to verify the file had data and it was correct (Figure 15).



***Figure 15. Verifying that the file has the correct data***

Summary

In the Module 07, I have researched and learned about Exception Handling and Pickling in Python. With the help of instructional videos, textbook and consulting additional resources, I have created two scripts demonstrating how to use Exception Handling and Pickling. I successfully ran both scripts using PyCharm and the Command Window.