Gabriela Tedeschi

August 22, 2021

IT FDN 110

Assignment 07

# Pickling & Structured Error Handling Demonstrations

## Introduction

In this document, I will describe the steps I took to write a script that demonstrates how pickling and structured error handling are used in the Python language. For pickling, I created one large demo that uses functions to separate concerns and that performs a practical task. For error handling, while I concerned taking the same approach, I chose to write several unconnected examples to demonstrate the different techniques we learned in a simple way.

## Pickling

I started this demo by simply declaring two variables I would need, one to store the file name and a list that would store the data. (While I’m not sure if it’s a best practice, I only declared the variables that exist outside of functions as this felt most helpful for planning purposes.) Then I created a Processing section, declaring two functions in the class Processor. For the writing function, I added a doc string, used the open function to open a file, used “ab”, which gives the command to write to the file in binary, and used the pickle.dump function to write the contents of the lst parameter to the file. At the end, a message to the user is returned.

For the reading function, I chose “rb” for the open function, which gives the command to read binary from the file. I then declared an empty list and use the pickle.load function to put the contents of the file into the list. At the end, the function returns the lst parameter.

Text

Description automatically generated

In the demo section, I printed some messages to the user to let them know what I was trying to demonstrate. Then, I assigned some items to the list, todo\_lst. I then passed this variable and the variable storing the file name as arguments into the writing function I created. This is within a print function, so it prints the function’s return message to the user, which lets them know they wrote to the file successfully.

I then emptied todo\_lst, called the reading function, and assigned the list the function returns to todo\_lst, so the contents of the file are stored in todo\_lst. Then, I printed todo\_lst. Because the list variable was emptied before the read function ran, the user knows that the contents of the file were read and assigned to the list successfully.

Text

Description automatically generated

## Structured Error Handling

The first two examples I created in Structured Error Handling Demos section simply use the general Exception class and provide both custom, easy-to-understand error messages that I wrote and Python’s built-in error info. The first example triggers a FileNotFoundError, while the second triggers a ZeroDivisionError.

Text

Description automatically generated

In the next subsection, I demonstrate how to use specific Exception classes – FileNotFoundError and ZeroDivisionError, the same two that I returned with my previous examples. To do this, I name these specific classes in my except statements and “e” stores the class information just as it does with the Exception class. I chose to create two examples with the same two except blocks and try blocks that are identical, except that the code comes in a different order. (The division by zero error comes first in one and the file not found error comes first in the other.) This allows the user to observe that only the first error will be caught and that this determines which class’s info will be returned.

Text

Description automatically generated

For the Custom Errors subsection, I demonstrate that you can raise custom errors by setting a custom condition as an if statement in the try block. I set a condition that the string must be 5 or fewer characters long. As my string is longer than this, the except block returns both the custom messages I wrote and Python’s info. Note that the message I provided in the raise Exception statement is stored in “e”.

A screenshot of a computer

Description automatically generated with medium confidence

In the final example, I demonstrate how to create custom Exception classes. I created two classes, added doc strings, and defined functions within them – one that gives a message about the max acceptable length of a string and one that gives a message about the minimum acceptable length of string.

I use these functions in the try block in a way that’s similar to the example above. Instead of using a raise Exception statement, I use the raise keyword with the names of the functions I created. Based on the conditional statements I used, one of these functions will be raised and its info will be stored in “e”. In this case, I used a string that is too short, so the information that prints is about the minimum string length error.

Text

Description automatically generated

## Running The Program

The images below show portions of the program running in PyCharm and the command shell:

**PyCharm**

Text

Description automatically generatedText

Description automatically generated

**Command Shell**

Graphical user interface, text

Description automatically generatedText

Description automatically generated

## Summary

In this document, I discussed the steps I took to create pickling and structured error handling demos. In the process, I gave an overview of how functions related to pickling and error handling work and the different options that these functions provide.