Jon Larson

08-23-2022

Foundations of Programming: Python

Assignment 07

https://github.com/jtlarson/IntroToProg-Python-Mod07

Demonstrating Exception Handling and Pickling

# Introduction

The goal of this assignment is to research and integrate custom ‘exception handling’ and ‘pickling’ into the program’s operation. I will discuss some of what I learned about these topics, and also demonstrate the use of these concepts in an original program which I created for this purpose. I will demonstrate how this program uses custom exception handling to provide key feedback elements to the user, and even help control the flow of the program. I will also show how pickling can be used—particularly the ability to store entire objects (such as a list) and not just the values within.

# Research

To begin, I researched exception handling on the web. I first consulted the official Python docs on this subject (Python Errors and Exceptions, <https://docs.python.org/3/tutorial/errors.html#>, 2022, (External Site)) because I prefer to start with the most ‘authoritative’ source available. The documentation points out the distinction between syntax errors, such as a missing “:”, and “exceptions” which arise when python reaches an ‘impasse’ as it tries to complete the programmed instructions. An example of this might be if the program asks a user to input a divisor, and the user input “0”. Since dividing by 0 is mathematically impossible, python cannot continue *without* *further instructions*, so by default it will quit the program with a “ZeroDivisionError” exception. I emphasize ‘*without* *further instructions’* because this assignment is all about showing how we can provide extra instructions for python to execute if something fails. We can even provide multiple ‘instruction sets’ inside individual “exception clauses” that cover different possible errors. Exception handling is done using a “try… except” structure, where code (that might be expected to fail) can be placed within a ‘try’ block, followed by one or more ‘except’ statements that fit different exceptions that might be expected. Conceptually, this exception handling is very similar to the ‘if… elif’ conditionals that we already use—it can even end with an optional ‘else’ clause that covers any exceptions not already handled.

The second topic we needed to research is ‘pickling’, which is “the process whereby a Python object hierarchy is converted into a byte stream” (Pickle – Python Object Serialization, <https://docs.python.org/3.10/library/pickle.html?highlight=pickle#> , 2022 (External site)). In plain language, this means that an object, such as a list, is encoded into a stream of bytes that can be sent outside of Python—over the network or into a file. The pickle module does this in binary format so external programs (such as a network transmission protocol or file system) don’t need to ‘understand’ anything about the bytes that are being sent—they just need to send them exactly as they are to the destination. Of course, Python also has the ability to read, or ‘un-pickle’ a byte stream and re-create the object that was previously serialized. In the assignment this week, I use this feature to store and retrieve a list object in a file. This is helpful because I don’t have to worry about iterating through all the values and storing them in an appropriate format—such as CSV. Instead I simply ‘pickle’ the whole list object and write it to the file. When I retrieve it, I can use it as-is, without adding the values to another list.

# Code Explanation

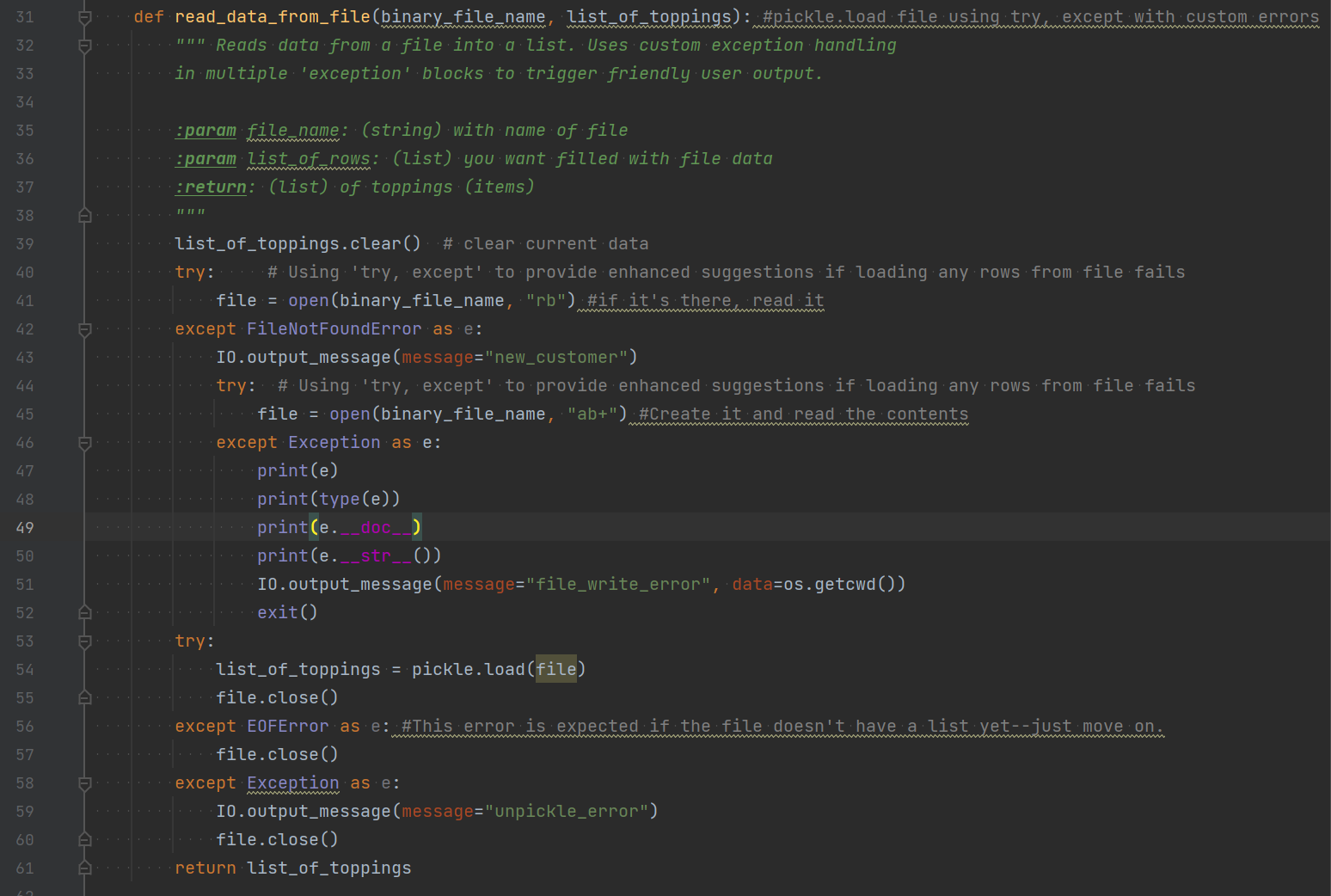
The assignment this week does not provide any ‘starting’ code or structural elements, so it is up to me to create a program that demonstrates pickling and exception handling while conforming to recommended practices like ‘separation of concerns’ and clear arrangement of code and comments. I chose to use a program that is similar to some of the previous assignments, but gives me the freedom to incorporate the required elements. I call my program “Bracket’s Sandwich Shop” but for the purpose of this assignment, the file is simply “A07-jtlarson.py”.

My sandwich shop is named after the “[]” that are used to indicate a python list, and can also be imagined as slices of bread that would form a sandwich when filled. The fact that a list is designed to hold arbitrary values gives me a place to add and store sandwich ‘toppings’ that can be ‘eaten’, saved/retrieved from a ‘ToGo bag’ (file), or lost (more on that later). In order to accomplish these tasks, I created the following program structure:

* Header
* Module imports
* Data – global variables
* “Processing” class – data processing functions
* “IO” class – human interaction elements
* “Main Body” code

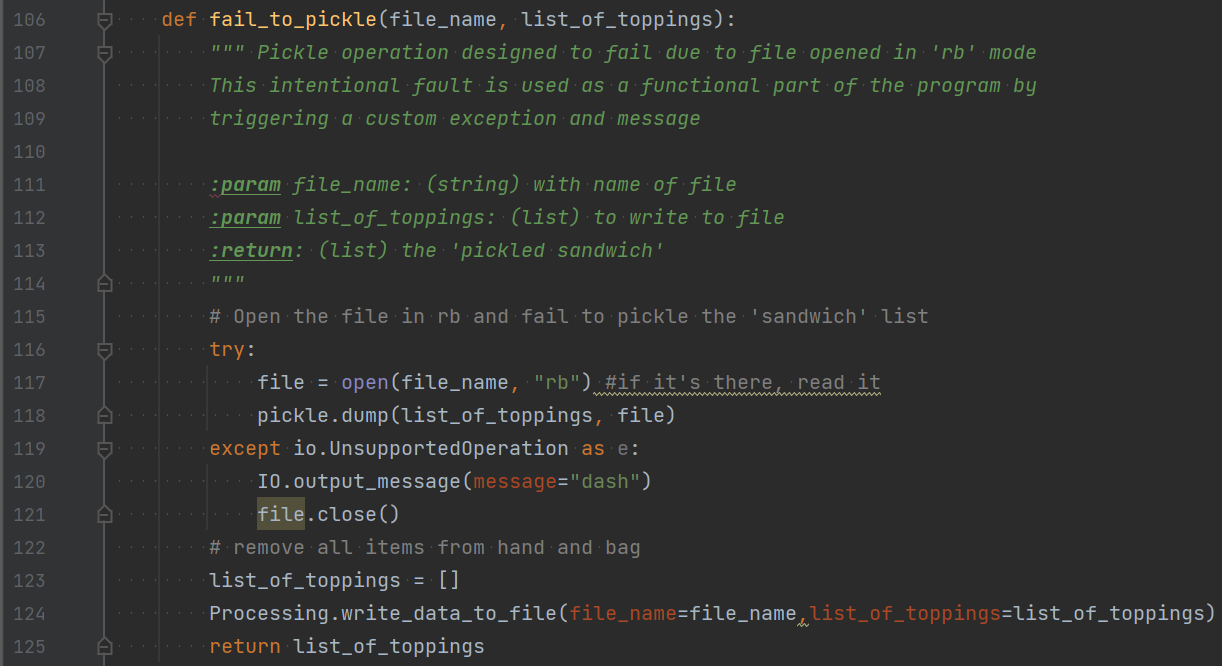
The program requires the ‘pickle’ module of course, but I also added the ‘os’ and ‘io’ modules to support my exception handling functions. The ‘data’ section contains four global variables that are required by the “Main Body” code. I used UPPERCASE to denote the file name string should be considered ‘static.’

The “Processing.read\_data\_from\_file” function uses the pickle module to try reading data from the save file. This function is an excellent place to combine pickle and a “try… except” structure, since we can’t be sure that the file we are looking for will be present and contain data in the format we expect. In my function, I used the ‘rb’ mode (which errors if a file isn’t present) and leverage the ‘FileNotFoundError’ error that occurs on first run to trigger a ‘new customer’ message to the user. I also nest a second ‘try… except’ block that makes a second attempt to open the file—this time with ‘ab’ write mode, which will create the file if it isn’t found. If that also fails, then a general “Exception” class will print out info about the error and prompt the user to verify write access. This is followed by a third “try… except” block that handles the ‘pickle.load’ process—suppressing an expected error if a list isn’t found, and giving the user a warning (but not ending the program) for other errors (Figure 1).



**Figure 1 - "read\_data\_from\_file" function demonstrates use of pickle and exception handling**

Another example of the use of error handling can be found in the “Processing.fail\_to\_pickle” function shown in Figure 2. In this function, I intentionally open the file ‘read-only’ so that the subsequent pickle.dump function will fail. I then catch that failure (this is why I needed to import the “io” class) and use it to trigger a message to the user indicating that their sandwich has been ‘lost’.



**Figure 2 - "fail\_to\_pickle" function - a second example of pickle and error handling**

In designing the flow of the code, I gave careful attention to the separation of concerns—that is, separating the code into a logical order with discrete sections and functions that each have a clear task. This is reflected in the class functions I designed for this program, and the collection of user messages in a dedicated function (“IO.output\_mesage”).

# Running the code

You can see a screenshot of the program running in Powershell in Figure 3 below:

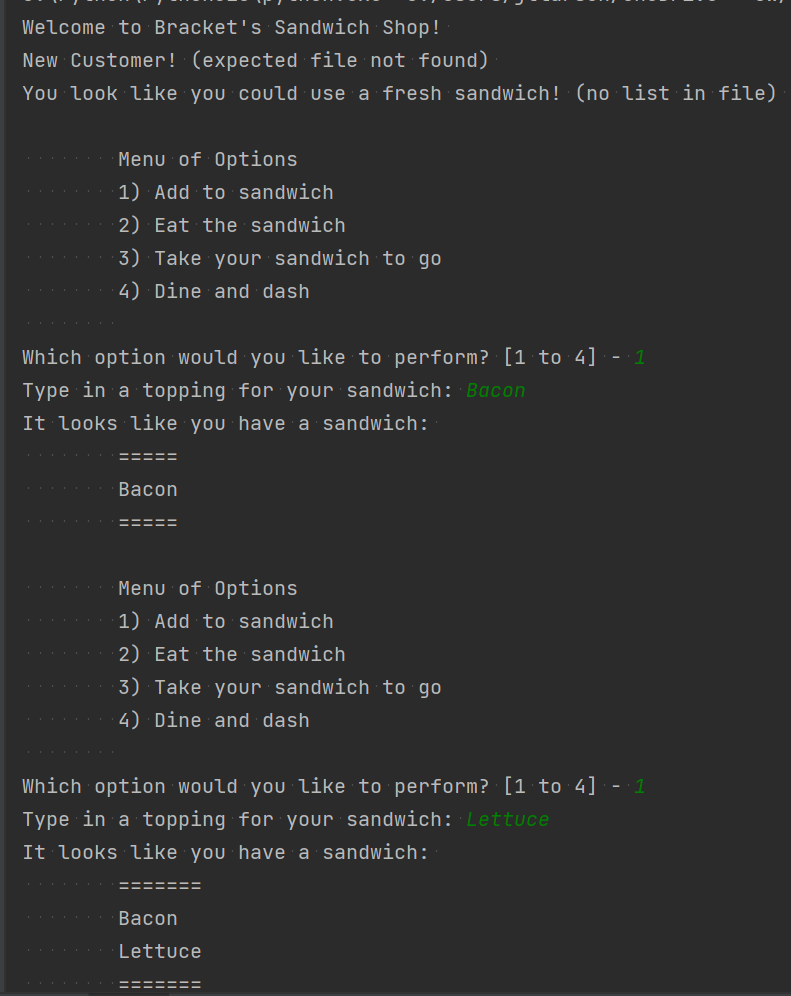
Text

Description automatically generated

**Figure 3 – “Bracket’s Sandwich Shop” running in PowerShell**

## Running in PyCharm

Below (Figure 4) is another screenshot of the program in operation—this time in PyCharm. This time I decided to delete the ‘ToGo’ file to demonstrate how the exception for a missing file is used to prompt a friendly greeting (“New Customer…”):



**Figure 4 - Program operating in PyCharm. What topping should I add next..?**

# Summary

In this document I described a couple research topics that we had to learn about to complete this week’s project. I also explained how I used error handling in a functional way in my program, and included screenshots that show how I used pickling to store/retrieve a list object from a binary file. structure of the pre-existing code and how it reflects the “separation of concerns.” I described some changes I chose to make (and the reasoning behind those changes) to the template code. I also demonstrated the program operation in PowerShell and PyCharm.