Kurt Mueller

November 24th, 2019

IT FDN 100 A -- Foundations of Programming (Python)

Assignment 07

# Pickling and Structured Error Handling in Python

Introduction

This paper will describe Pickling in Python, and include an example of Structured Error Handling. Pickling is the technique of saving data in a binary format, which can obscure the file content and may reduce the file size. Also known as flattening, serialization, or marshalling, using Pickle to read and write large amounts of data can be more efficient than reading and writing from or to plain text files. Structured Error Handling guards against potential (user-induced) errors and allows the programmer to customize how the program handles errors, instead of letting Python display default error messages that might be confusing to the user.

It is helpful to review the following links before starting to write the code:

Links:

<https://www.python-course.eu/python3_exception_handling.php>

The page is well organized and provides multiple examples of Exception Handling in Python. It was especially helpful for me because it embedded the try/except block within a while loop, and the example therefore fit nicely into my already-existing code.

<https://www.tutorialspoint.com/python/python_exceptions.htm>

Provides a list of standard Exceptions in Python and a short description of in which cases each is raised, i.e., a “Value Error” is raised when “the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified.”

<https://docs.python.org/3.7/library/pickle.html>

Comprehensive description of how pickle works and provides a list of functions (.dump, .load, etc.) to make the pickling process more convenient.

<https://www.pitt.edu/~naraehan/python3/pickling.html>

Short, well written example with a simple code example as well.

Work Instructions

We will write a simple script to read and write data to binary format, and include some simple structured error handling.

First, import pickle and create a file called ‘AppData.dat’ and a list called lstCustomer. Note that the file extension is .dat (binary).

import pickle

strFileName = 'AppData.dat'

lstCustomer = [ ]

Now we will create two new functions to save and read data to/from the binary file and utilize the .dump and .load functions within pickle. The first function will save the string data and write to the file. Note that .dump method is appended after pickle, and that “wb” (write binary) is used instead of “w”, as you would for a plain text file.

def save\_data\_to\_file(file\_name, list\_of\_data):

file = open(file\_name, "wb")

pickle.dump(list\_of\_data, file)

file.close()

The second function will read the data from the binary file. Note that .load method is appended after pickle, and that “rb” (read binary) is used instead of “r”, as you would for a plain text file.

def read\_data\_from\_file(file\_name):

file = open(file\_name, "rb")

list\_of\_data = pickle.load(file)

file.close()

return list\_of\_data

Now that the functions are written, this is a good opportunity to add some Structured Error Handling. A user ID should be only numbers, so anticipating that the user may attempt to add a string character instead of an integer as their user ID, construct a while loop with try/except to handle the error:

while True:

try:

intID = int(input("Enter an ID: "))

intID = int(intID)

break

except ValueError:

print("Not an integer! Please try again...")

If the user enters a string value, a custom error will be produced as shown below in Figure 1:

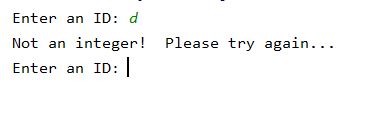


Figure 1: Custom Error message using try/except

This is a more user-friendly error that simply prompts the user to enter an integer versus a string for the User ID. It is a good alternative to the Python generated ValueError message, as shown in Figure 2.



Figure 2: Python ValueError message

Add more code to prompt the user to enter their name, store the entries into a list object, and print the entry back to the user:

strName = str(input("Enter a Name: ")) # get Name from user

lstCustomer = [intID, strName] # store the entries into a list object

print(lstCustomer) #print entry back to the user

Lastly, add code to call the functions declared previously to save and read the data to/from the file as shown below:

save\_data\_to\_file(strFileName, lstCustomer)

print(read\_data\_from\_file(strFileName))

The code should output similar to Figure 3:

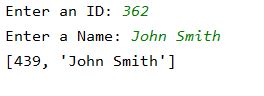
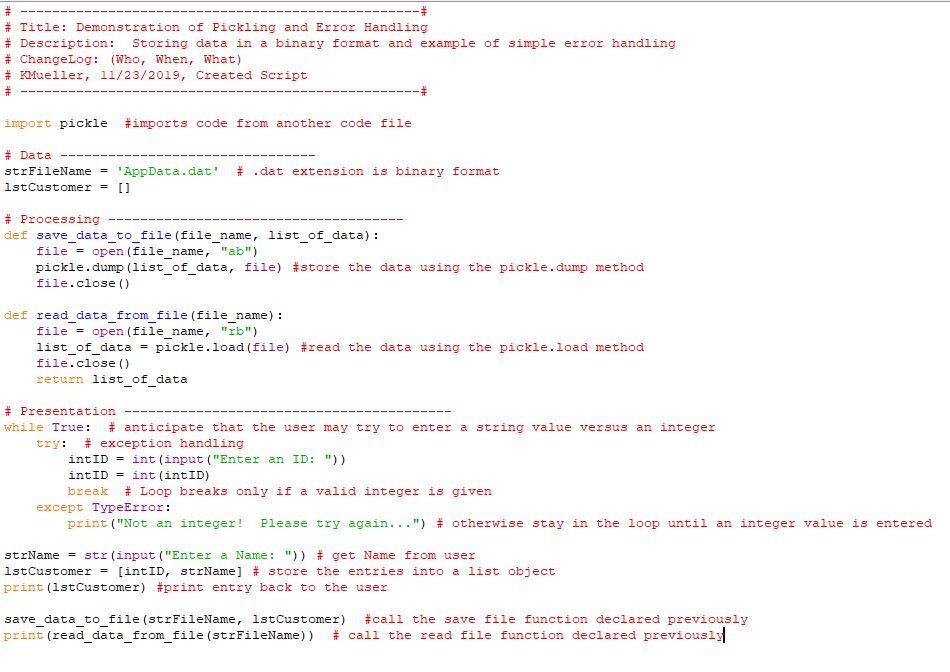


Figure 3: Code output in PyCharm

Note that data should also be present in the ‘AppData.dat’ file, although parts of it will be obscured and unreadable (to a human).

Figure 4 shows the entire script with comments:



Summary

This paper has described how to use python pickle to read and write data to a binary file and has also demonstrated simple Structured Error Handling using try/except. This code could be improved by adding additional Structured Error Handling for the User Name input (for example, add some error handling to require the user to enter both a first and last name.