Miguel Quintana

May 31, 2023

IT FDN 100 A

Assignment 07

**This week Video lecture, Book, and paper.**

This week we learned about pickling modules and exception handling.

When we import the built-in function pickling, it allows us to execute a series of pre-made functions with simple commands, One example is that if we want to read binary code using pickling, we only need to apply the "rb" code and it will automatically read the specific file that is requested. The benefit of having a built-in function in Python is that it will simplify the code and make it much more readable.

When we code, we tend to make mistakes. Those mistakes can create error messages that are sometimes difficult to understand. However, when we notice a mistake that occurs to us, we can customize those errors with our own words by using structured error handling, What we mean by that is that using the built-in function Try-Exception, we can create our own error messages that are much easier to understand.

One of the tools of the exception class is that you can catch specific errors using more specific exception classes, and you can "raise" errors based on custom conditions, as we will demonstrate in the week 07 assignment. The best moment to use this class is when you want to document a specific error with your own words.

As we mentioned before, the "Exception" class is a built-in class in Python, and its main function is to hold information about errors. Python automatically creates an exception object when an error occurs, and the exception object automatically fills with information about the error.

This week, we also learned about binary files. The difference between binary files and text files is the form of the data in them. Binary files are a more difficult way to read for humans but can be much easier and faster for computers. Binary format can obscure the file’s content and may reduce the file’s size. When we are dealing with small data, that might be irrelevant, but with a large amount of data, it can be easier to handle.

Another important topic this week is making our GitHub pages more professional, useful, and easy to read. One way to do this is with markdown commands that help format my documents. From what I notice, markdowns are very similar to html and use different characters and words to create specific tasks; for example, "#" indicates a header, two "#" indicate an introduction, and three "#" indicate a subtopic.

**Website reviews:**

***Pickling:*** <https://www.geeksforgeeks.org/understanding-python-pickling-example/>

This website explains in detail the main function of pickling, demonstrating how to save data on a file, how to initialize data to be stored in db, and also the advantages of using the pickling module, which are recursive objects, object sharing, and user-defined classes and their instances.

***Exception Handling in Python:***<https://www.geeksforgeeks.org/python-exception-handling/>

This website first explains the different types of exceptions in Python, such as SyntaxError, TypeError, IndexError, among others. Also explains the difference between syntax errors and exceptions. Also go through the Try-Exception statements and catching exceptions.

This website makes a specific effort to explain the advantages of Exception handling, like improved program readability, simplified error handling, cleaner code, and easy debugging.

**Assignment 07:**

The first part of this assignment is to import pickle. When we import pickle we can use a number of built-in functions that will convert regular data into binary data as well as convert binary data into regular data.

After we do this, we proceed with the data that we will use in this assignment. We will create two variables that will collect data from the user, in this case the variables country and strName. The first will ask for the country of origin of the person, and the second will ask for the name of the person. We will also create a list that will hold both inputs, and we will put this data on a file that will convert this data to binary code. To do so, we need to create an objFile variable that will open and dump the data in the "PeopleData.dat" file.

After we save the data, we can create more code that will present the current data in the file, very similar to what we did in the previous step, but instead of using the append function, we will use a reading function.

Finally, we will print the object file data "objFileData".

import pickle

# Pickling

country = str(input("Country of Origin: "))

strName = str(input("Enter a Name: "))

lstPeople = [country, strName]

print(lstPeople)

# Now we store the data with the pickle.dump method

objFile = open("PeopleData.dat", "ab")

pickle.dump(lstPeople, objFile)

objFile.close()

# And, we read the data back with the pickle.load method

objFile = open("PeopleData.dat", "rb")

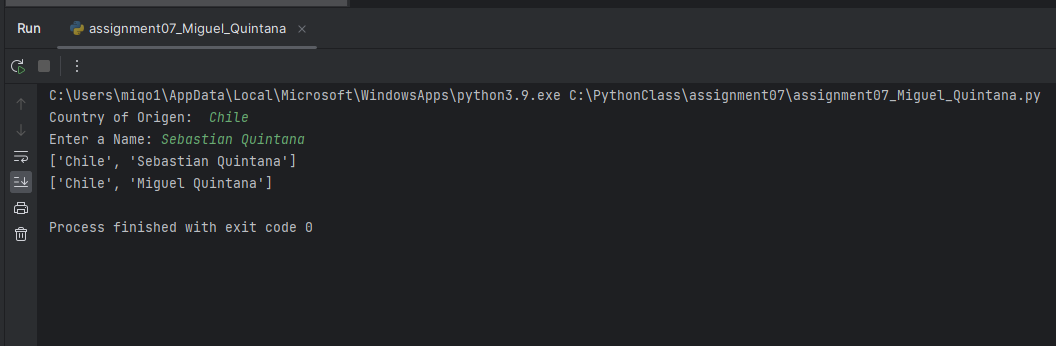
objFileData = pickle.load(objFile) #load() only loads one row of data.

objFile.close()

print(objFileData)

Listing 1

And this is the result on the screen:



Listing 2

The second part of the assignment is to create some structured error handling (Try-Except). To do that, I used our previous pickle assignment. The first and second error handling codes that I wrote for this section have to do with the specific data that we need to collect. We are looking for countries and names, so no numbers should ever be present as input. If the user puts a number as input, using the built-in function isnumeric() will detect the presence of a number and will send a Try-Except message saying that do not use numbers for the file.

The third one will detect if blank spaces are detected as inputs and also show a message telling the user that is not allowed. To get that result, we will use the built-in function isspace().

try:

country = str(input("Country of Origen: "))

if country.isnumeric():

raise Exception('Do not use numbers for the file\'s name')

except Exception as e:

print("There was a non-specific error!")

print("Built-In Python error info: ")

print(e, e.\_\_doc\_\_, type(e), sep='\n')

try:

strName = str(input("Enter a Name: "))

if strName.isnumeric():

raise Exception('Do not use numbers for the file\'s name')

except Exception as e:

print("There was a non-specific error!")

print("Built-In Python error info: ")

print(e, e.\_\_doc\_\_, type(e), sep='\n')

try:

strName = str(input("Enter a Name: "))

if strName.isspace():

raise Exception('Do not use space for the file\'s name')

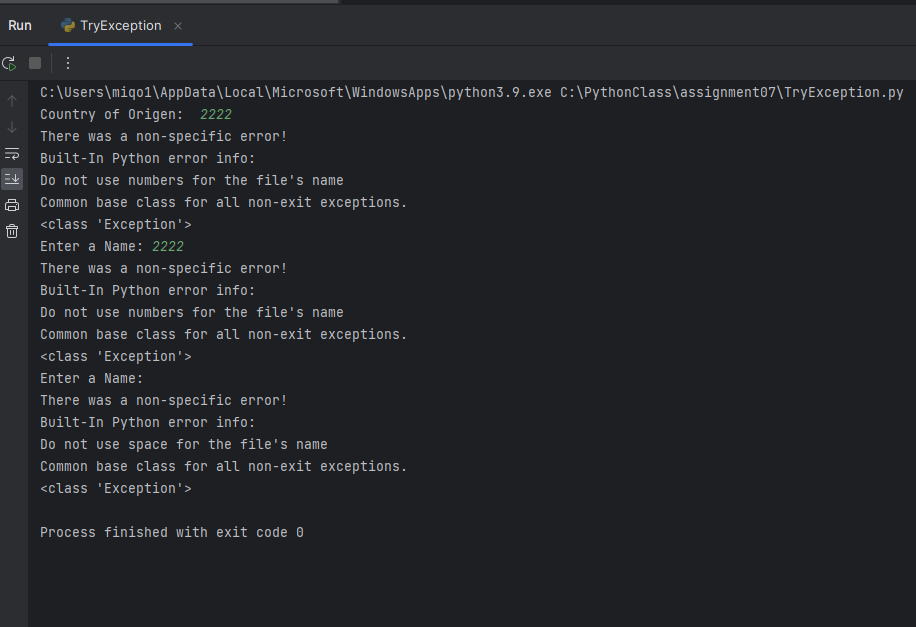
except Exception as e:

print("There was a non-specific error!")

print("Built-In Python error info: ")

print(e, e.\_\_doc\_\_, type(e), sep='\n')

Listing 3



Listing 4