Jordan Nussbaum

05/30/2022

Foundations of Programming: Python

Assignment 07

Github Link - https://github.com/Jnussb/IntroToProg-Python-Mod07-

Error Handling and Pickling

Introduction

Before starting the 7th module assignment, Professor Root told us that our task would be to implement the concept of error handling and pickling data. I've learned to incorporate them into some of my codes that I made prior to this week. Throughout this paper, I will show you how I was able to utilize these concepts to better grasp the basic understanding of error handling and pickling within Python.

Error Handling

Python has many built-in exceptions that are raised when your program encounters an error. When these exceptions occur, the Python interpreter stops the current process and passes it to the calling process until it is handled. If not handled, the program will crash. There are two types of error that occur in Python. One being syntax errors and the other is logical errors also known as exception errors which occur after passing the syntax test. We can handle errors with many different handling exceptions. I used raised exceptions for a predefined condition. In other words, when I want to code for the limitation of certain conditions then we can raise an exception.
https://www.geeksforgeeks.org/errors-and-exceptions-in-python/

Pickling

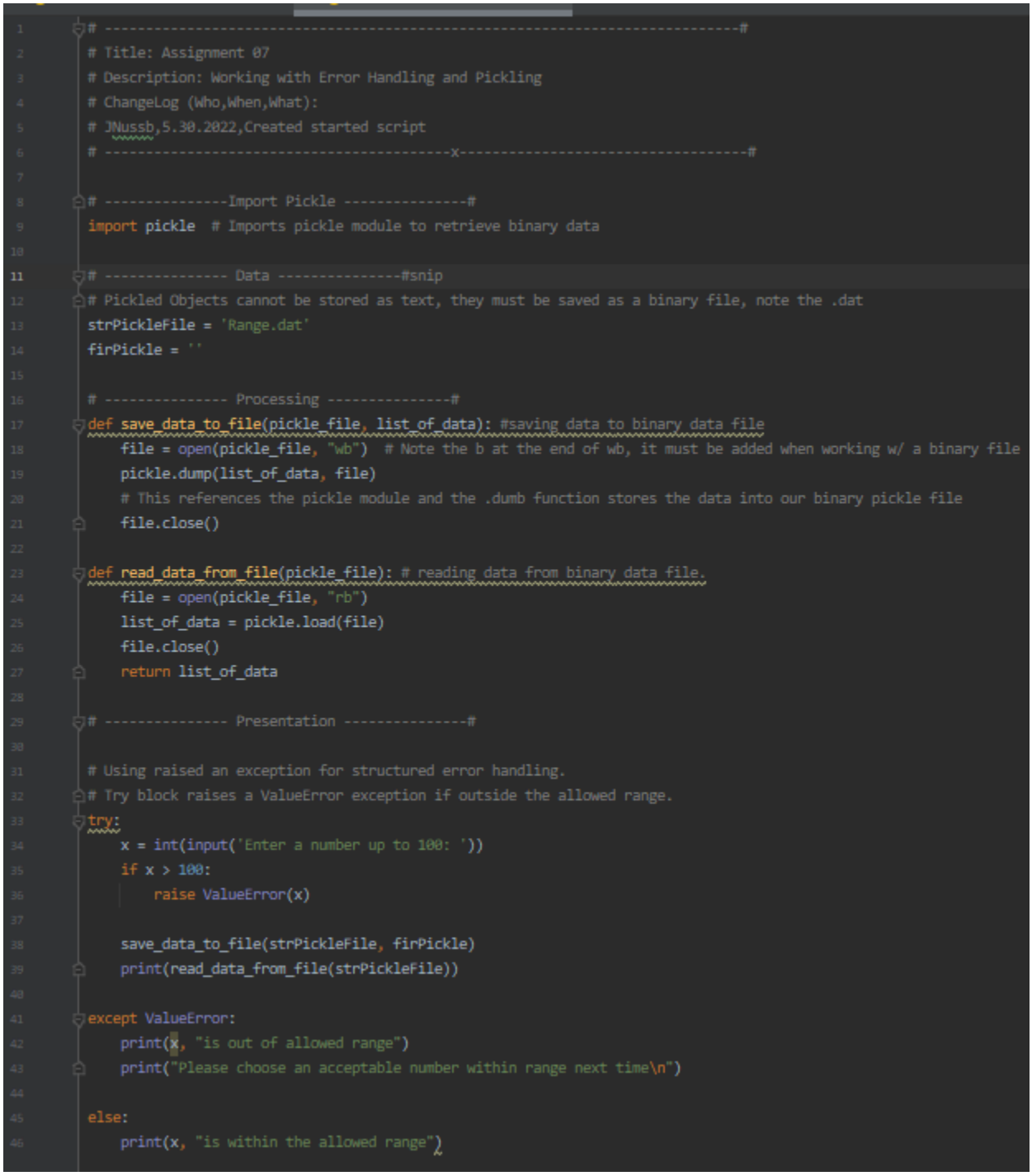
Python pickle module is used for serializing and de-serializing python object structures. This type of process used to convert any kind of python object whether it be a list or a dictionary into byte streams (0s and 1s) is called pickling. Also, pickling is alternatively known as serialization or flattening or marshalling. Whereas unpickling is the inverse operation, in which a byte stream (from a binary file or bytes like object) is converted back into an object hierarchy. How common is the use of pickling and unpickling? It's very common to use them since there are many advantages. One being that objects that are serialized have certain references to themselves. Another is object sharing which ensures all references point back to the master copy. That is pertinent for mutable objects to be shared.

https://www.datacamp.com/tutorial/pickle-python-tutorial

The Script

This script has both error handling and pickling. I start off the script by using the import pickle module. If you notice the ‘dat’ is there instead of the usual ‘txt’, the reason is pickled data cannot be stored into a text file. Pickled data must be stored into a binary file which is the file used to store raw bytes that are not readable to humans. It has been changed to ‘dat’ to change the file type to binary file. To pickle the list_of_data, you need to specify the name of the file you will write it to. In order to open the file for writing, you have to use the open () function. The ‘w’ in ‘wb’ is for writing to the file and the ‘b’ refers to binary mode which will be written in the form of bytes objects. Once the file is opened for writing, you use pickle.dump () function which takes two arguments, data that is being used to pickle & the file. When reading from the file, you use the .load () function to get the program to load out the pickled data. When it comes to error handling, I used the raised exception to prompt a user to pick a number up to 100. If the user chooses a number outside that range then the raise exception is called out to tell the user to pick a reasonable number next time.

Figure 1: Next Page



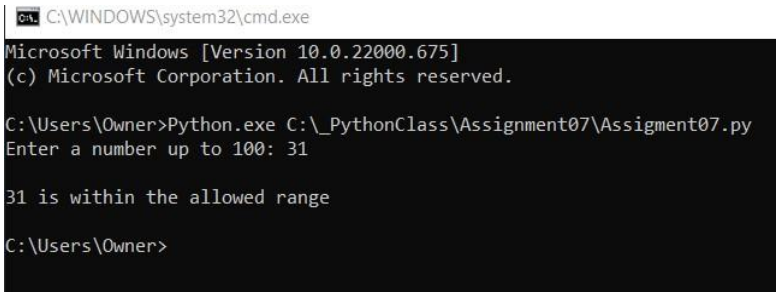
```python
# ------------------------------------------------------------------------#
# Title: Assignment 07
# Description: Working with Error Handling and Pickling
# ChangeLog (Who,When,What):
# JNussb,5.30.2022,Created started script
# ---------------------------------------x--------------------------------#

# ---------------Import Pickle ---------------#
import pickle  # Imports pickle module to retrieve binary data

# --------------- Data ---------------#snip
# Pickled Objects cannot be stored as text, they must be saved as a binary file, note the .dat
strPickleFile = 'Range.dat'
firPickle = ''

# --------------- Processing ---------------#
def save_data_to_file(pickle_file, list_of_data): #saving data to binary data file
    file = open(pickle_file, "wb")  # Note the b at the end of wb, it must be added when working w/ a binary file
    pickle.dump(list_of_data, file)
    # This references the pickle module and the .dumb function stores the data into our binary pickle file
    file.close()

def read_data_from_file(pickle_file): # reading data from binary data file.
    file = open(pickle_file, "rb")
    list_of_data = pickle.load(file)
    file.close()
    return list_of_data

# --------------- Presentation ---------------#

# Using raised an exception for structured error handling.
# Try block raises a ValueError exception if outside the allowed range.
try:
    x = int(input('Enter a number up to 100: '))
    if x > 100:
        raise ValueError(x)

    save_data_to_file(strPickleFile, firPickle)
    print(read_data_from_file(strPickleFile))

except ValueError:
    print(x, "is out of allowed range")
    print("Please choose an acceptable number within range next time\n")

else:
    print(x, "is within the allowed range")
```

Summary

Throughout this assignment, I've learned to implement concepts of error handling and pickling. I was able to utilize them into a raised exception error handling code. I've learned a lot over built-in error handling exceptions. Now, I'm looking forward to using more of these built-in user-friendly error handling exceptions.

Figure 2 & 3: Command Shell

(no error)



(error)

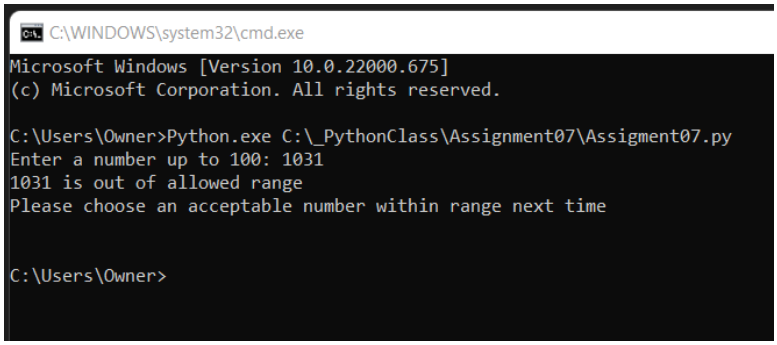
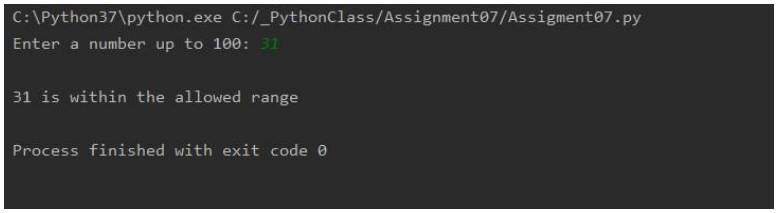


Figure 4 & 5: Pycharm

(no error)



(error)

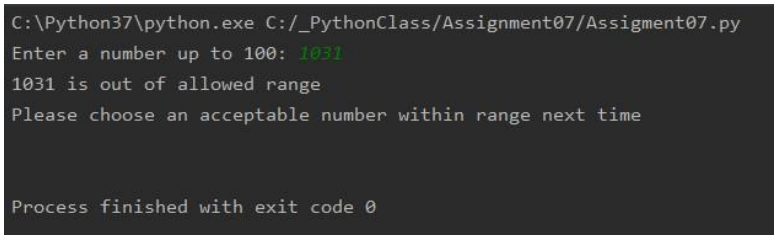


Figure 6: Pickled File