Dania Mahmood

March 2, 2021

Foundations of Programming, Python

Assignment 7: Demonstrating Error Handling, Pickling and Unpickling

GitHub Link: <https://github.com/daniamah/Intro_To_Python_Mod7>

# Exception Handling and Pickling in Python

Introduction

This week for our assignment we were asked to research on how Python handles errors and how to script your own customized errors. After our research we had to create our own demo script to demonstrate using Error Handling. We also learned about pickling and unpickling and were asked to include that in our demo script as well. For my demo script, I created a program that lets a user enter a name and a phone number to create a list of their contacts. I then pickle and unpickle the contacts file.

Error Handling

Python offers many built in exceptions that are raised when a program encounters an something unexpected or wrong. When a program hits an exception, it stops running until the exception is handled.

One way of handling errors is the Try Except method. The critical operation which can raise an exception is coded inside the try statement and the code that handles the exceptions is written in except statement. If no exception occurs, the code in the except statement is skipped and normal flow continues. Every exception in Python inherits from the base Exception class. Programmers can manually raise exceptions by using the raise keyword. My research found that a try statement can have an optional finally clause that is executed no matter what. It is generally used to close any open resources.

PicklingWhen users work with complex data Python offers a way to serialize it. This process in **Python** is called pickling and works using the **dump** function. In order to use pickling users need to import Python’s pickle module. In order to use the dump function users have to open the file with ‘wb’ attribute (wb stands for write binary).

Unpickling  
It is the inverse of Pickling process. While the process of retrieving original Python objects from the stored string representation is called unpickling, we use **load** function to do this. In order to use the load function, users have to open the file with attribute ‘rb’ (rb stands for read binary).

First of all users have to import pickle module in order to use the two main methods, **dump** and **load**.

**import pickle**

**Pickling Example**

**import** pickle  
  
a = [1, 2, 3]  
file = open(**"picklefile.txt"**,**"wb"**)  
pickle.**dump**(a,file)  
file.close()

**Unpickling Example**

**import** pickle   
file = open(**"picklefile.txt"**, **"rb”)**   
a = pickle.**load**(file)  
print(a)

# Script

**import** pickle  
  
*#defining variables*lstTable =[]  
choice = **""***# Load current data***try**:  
 objFile = open(**"Contacts.txt"**, **"rb"**)  
 FileData = pickle.load(objFile)  
 **for** i **in** FileData:  
 print(i[**"Name"**] + **" | "** + i[**"Phone Number"**])  
 objFile.close()  
 print(**"Successfully loaded file"**)  
**except** FileNotFoundError **as** e:  
 print(**"File not found"**)  
   
*#demonstarting error handling***while** choice != **'x'**:  
 **try**:  
 strname = str(input(**'Enter Name of contact: '**))  
 **if** strname.isnumeric() == **True**:  
 **raise** Exception(**"Only use alphabets for name \n"**)  
  
 phone = str(input(**'Enter phone number: '**))  
 **if** phone.isalpha() == **True**:  
 **raise** Exception(**"Only use digits for phone \n"**)  
  
 dicRow = {**"Name"**: strname, **"Phone Number"**: phone}  
 lstTable.append(dicRow)  
 **except** Exception **as** e:  
 print(e)  
  
 **try**:  
 choice = input(**"Type 'x' to exit or 'c' to continue "**).lower()  
 **if** choice **not in** [**'x'**, **'c'**]:  
 **raise** Exception(**"Only use 'x' to exit or 'c' to continue \n"**)  
 **except** Exception **as** e:  
 print( e, e.\_\_doc\_\_)  
  
print(**"Here are your tasks pre-pickled:"**)  
**for** i **in** lstTable:  
 print(i[**"Name"**] + **" | "** + i[**"Phone Number"**])  
  
print(**"Pickling tasks..."**)  
objFile = open(**"Contacts.txt"**, **"wb"**)  
pickle.dump(lstTable, objFile)  
objFile.close()  
  
print(**"Data Unpickled! contacts below: "**)  
objFile = open(**"Contacts.txt"**, **"rb"**)  
FileData = pickle.load(objFile)  
**for** i **in** FileData:  
 print(i[**"Name"**] + **" | "** + i[**"Phone Number"**])  
objFile.close()

# Summary

For my research, I was able to find great resources on the internet. Particularly for my research I used [PythonProgramming](https://pythonprogramming.net/python-pickle-module-save-objects-serialization/) and [programiz](https://www.programiz.com/python-programming/exception-handling). I then created a demo script to demonstrate using Try Except Error Handling and Pickling. I learned that Error Handling is very useful to catch errors and provide users useful feedback. Programmers can customize their error messages to provide more useful information as sometimes using the default errors that Python provides may be less descriptive.

I also learned Pickling Data which converts data to a byte stream that can be unpickled back to more readable data when needed.