Yuting Lin

Dec. 8, 2020

Introduction to Python Scripts

Assignment07

<https://github.com/filizlin/IntroToProg-Python-Mod07.git>

<https://filizlin.github.io/IntroToProg-Python-Mod07/>.

# Pickling and Error Handling

## Introduction

### Pickling

The idea of Pickling means preserving a complex data set (from a pre-existing database) into a binary file (.dat). Once the data is stored in the Pickling list, the user can randomly access the part of data that they would like to retrieve. In the binary file, the data is harder to interpret with the added symbols (it is not secured). A step-by-step walk through of the set up can be found in the Youtube video, Python Pickle Module for saving data (serialization).

### Structured Error Handling

Structured Error Handling allows the user to receive the error message in plain text. It works by having the developer “predicting” the possible types of error and calling them out using the try-except comment.

## Python Script Example

### Pickling

The following is a script designated to showcase the idea of Pickling. A set of data is pre-loaded int the program as a pickle list, and the user can call out the specific set of data by typing in a commend. The header section is eliminated for clarity:

import pickle  
  
*#Declaring the list to pickle#*ToDoList = [**"Homework"**,**"Lecture"**,**"Lab"**]  
Priority = [**"Low"**, **"High"**, **"Low"**]  
Deadline = [**"Soon"**, **"1 week"**, **"1 month"**]  
TimeEstimate = [**"5 hours"**, **"20 hours"**, **"10 hours"**]  
  
*#Storing the data as pickle#*DataFile = open(**"ToDoList"**, **"wb"**)  
pickle.dump(ToDoList, DataFile)  
pickle.dump(Priority, DataFile)  
pickle.dump(Deadline, DataFile)  
pickle.dump(TimeEstimate, DataFile)  
DataFile.close()  
  
*#opening the pickle file#*DataFile = open(**"ToDoList"**, **"rb"**)  
ToDoList = pickle.load(DataFile)  
Priority = pickle.load(DataFile)  
Deadline = pickle.load(DataFile)  
TimeEstimate = pickle.load(DataFile)  
  
*#Unpickling the Data#*while(True):  
 print(**'''Here is the list of data in the ToDoList.data:   
 1) To Do List  
 2) Priority  
 3) Deadline  
 4) Time Estimate for the To Do List item  
 5) Exit the Program  
 '''**)  
 strChoice = input(**"Please select an item to retrieve from the File [1/2/3/4]: "**)  
 if str(strChoice).strip() == **'1'**:  
 print(ToDoList)  
 continue  
  
 elif str(strChoice).strip() == **'2'**:  
 print(Priority)  
 continue  
  
 elif str(strChoice).strip() == **'3'**:  
 print(Deadline)  
 continue  
  
 elif str(strChoice).strip() == **'4'**:  
 print(TimeEstimate)  
 continue  
  
 elif str(strChoice).strip() == **'5'**:  
 break  
  
 else:  
 print(**"Invalid entry. Please enter a number between 1-5."**)  
 continue

First, the pickle module is imported into the script. Then a set of data (ToDoList, Deadline, Priority, and TImeEstimate) is loaded into the script. A binary file is then called out with “wb” indicating that it is writing to a binary file (and will create a binary file if it doesn’t exist). The set of data is “dumped” into the binary file and loaded. Then the user has a choice to select which data they would like to call out.

Graphical user interface, text, application, email

Description automatically generated

**Figure 1.** Screenshot of Running the Pickling.py script using Pycharm

Graphical user interface, text, application

Description automatically generated

**Figure 2.** Screenshot of Running the Pickling.py script using Terminal

Graphical user interface, text, email

Description automatically generated

**Figure 3.** ScreenShot of the ToDoList.dat file that is created with Pickling.

### Error Handling

The following script showcases structured error handling. The script requires the user to enter a number based on the menu options. If the user input the number out of range, a custom error will show telling the user to enter a number within range. If the number entered is not numerical, the user will also be notified that. Any error outside of these will be indicated as non-specified error:

*# Data-------------------------------------------------------------------- #  
# Declare variables and constants #*strchoice = None  
  
  
*# Processing------------------------------------------------------------- #*class RangeError(Exception):  
 *"""The number input is out of range"""* def \_\_str\_\_(self):  
 return **"Please enter a number between 1 and 5"***# Presentation (Input/Output) ------------------------------------------ #*while(True):  
 print(**'''  
 Please choose on of the following options:  
 1) open file  
 2) add data to file  
 3) remove data from file  
 4) save file  
 5) Exit the program  
 '''**)  
 try:  
 strChoice = input(**"Please enter your choice: "**)  
 if **'1'** <= strChoice <= **'5'**:  
 print(**"proceeding with your request"**)  
 elif strChoice.isnumeric() == False:  
 raise Exception(**"Please enter a numeric value [9 instead of nine]"**)  
 elif **'1'** >= strChoice or strChoice >= **'5'**:  
 raise RangeError()  
  
  
 except Exception as e:  
 print(**"An error has occur"**)  
 print(**"Built-In Python error info: "**)  
 print(e, e.\_\_doc\_\_, sep=**'**\n**'**)  
  
In the code, the data is first declared (as global). Then the specific error class (Range Error) is indicated as an Exception showing that the number is out of range. The docstring information will show if the function is defined as def \_\_doc\_\_(self), but in this case the error message indicated is shown with the \_\_str\_\_(self). A While(true) loop is then used asking the user to input an option. If the input is within the range, the script will proceed with the request. If the input is out of the rang, the range error will be raised. If the input is not numeric, an Exception will be raised asking the user to input a number (the two methods are interchangeable). And with all the other exceptions (in the except comment), the message will show as an random error and indicating the python error.

Text

Description automatically generated

**Figure 4.** Screenshot of Running the ErroHandling.py script using Pycharm

A picture containing text

Description automatically generated

**Figure 5.** Screenshot of Running the ErroHandling.py script using Terminal

## Summary

Structured error handling can help the user better understand the error that was occurring at the user input (instead of showing the pre-built python error, which can be hard to interpret sometimes). The pickling allow the user to store mass data in a data base and call out specific ones,

## Citation

The resources used to complete this document includes the IT FDN 110 A Module 7 course material provided by Prof. Randle Root and the textbook: Python Programming for the Absolute Beginner, Third Edition, By Michael Dawson.