Assignment #07

Introduction

This assignment is focusing on pickling and utilizing try-exception blocks of code to not allow the user to input anything that may break our program or cause any unintended functionalities. I created a simple program that asks users to enter names or characters, and then simply pickles the entries to a file and unpickles them upon user request.

Pickling in Python

I began this assignment by creating some functions that will pickle, that is, serialize the entries given by the user, as well as append additional entries to our name list.

Saving the pickle file, using append binary mode: Text

Description automatically generated

Reading from pickle file using read binary and printing the result from the load:

Text

Description automatically generated

Then, I needed a function that will append the user entries to a list to be recalled later:

Text

Description automatically generated with medium confidence

I quickly realized that when reading, or loading from a pickle file, that it will only grab the first entry given. I wanted a function that would allow continuous pickle loads until reaching the end of the file. To do this, we use a try-except block within a while loop that will continue loading until it receives an end of file error:

Graphical user interface, text, application

Description automatically generated

Try-Except Handling

Now that we had established the functions that will pickle user entries, save them to files, and allow them to be unpickled/recalled, I wanted to create some try-except blocks to prevent the user from entering things into my program that would either be pointless, or break my program.

Because my menu only has three options, I didn’t want to allow the user to enter anything that was not listed. To do this, I created a try-except block that will raise a value error in the event a user enters a character rather than one of the integers in the menu:

Graphical user interface, text, application, email

Description automatically generated

Then, I realized that still allowed the user to enter any integer value that wasn’t listed on my menu. Now, this doesn’t break my program or anything, but it’s a pointless function and doesn’t look very nice. So, I created a couple if statements that would not return the user entry unless it was either a 1, 2, or 3:

Text

Description automatically generated with medium confidence

If the user selects 1, the program will ask the user to enter a list of characters. I specifically did not want to allow integers, but I was having some trouble making this happen, because the value error exception was not working for the inverse of my previous usage. To remedy that, I imported the regular expression module and created an if statement that only allowed characters that match the library’s A-Z characters:

Graphical user interface, text, application, email

Description automatically generated

Then once the user selects 1, and the user’s entry was acceptable, the user\_entry function will call my add and save to pickle functions I created earlier.

A picture containing graphical user interface

Description automatically generated

Finally, if the user’s entry was 2, our program simply calls load\_all\_pickle to loop through the saved pickle file and keep loading until it reaches the end of file error. And if the user chooses entry 3 the program will simply break and close:

Graphical user interface

Description automatically generated with low confidence

Summary

This assignment was pretty fun and a good learning experience. Working with pickle files was pretty enlightening and confusing at times, I ran into some strange Unicode issues that was a good learning experience to hash out. I really enjoyed creating the try-except blocks, especially with the load\_all\_pickle function that utilized the EOFError, as trying to find ways to load all the entries was driving me a bit crazy. The importing of modules is really interesting, too, feels like that’s a huge part of programming in Python that we’re just getting into.