Henry Chisolm

September 5, 2021

Document Write-up:

Assignment 07

**Write a Script that demonstrates Pickling and Exception Handling.**

**Introduction:**

The purpose of this assignment is to create a new script that demonstrates the techniques of pickling and exception handling in a python. Pickling is the process where a data from a file is converted and stored in a binary way which obscures the data and condenses it. Exception handling is the process when writing scripts where one identifies errors and can custom code a message (or rather error message) so it is easier for another user or collaborator to understand where the program may be faltering. To do this I created a script that stored some data for a gradebook and then later loaded the data. Additionally, we were asked to post this in a GitHub page.

**Scope of the Assignment:**

The scope of this assignment involves creating a script, which displays the concepts of pickling and exception handling. So to analyze this assignment we will look at the (1) initial code to set up the assignment, (2) code for pickling, and (3) code to do error handling.

*(1) Initial Script:*

This assignment asked us to make a program that highlighted pickling and exception handling without being provided with initial code to adapt and modify. Additionally, we were asked to research the topics of pickling and error handling to highlight and explain different ways it can be used and utilized in a program. So to highlight this I created a grade book program, with two functions storeGrades and loadGrades, to be the beginning code for a gradebook program.

*storeGrades:*

I created a function with two local variable (it took me a while to realize the local v. global variable in how I was programming my script and the ramnifications), History and ComputerScience followed by a {} or a dictionary entry with name keys and grades. The purpose of this function was to have a hard coded beginning for further adapting and entering data.

Text

Description automatically generated with medium confidence

(fig. 1 Illustrating function store grades)

This was followed by creating a variable Grades DB which acted as directory for the data that we had just scripted, allowing both the courses of History and ComputerScience (our variables) to be pulled up.

Graphical user interface, text

Description automatically generated with medium confidence(fig. 2 illustrating dictionary storage)

Finally in this function we pickle the information, which will be discussed later. To do this through a variable strFile which we have open and create a file “PicklingExamples.txt” in an “ab”, which will append the file (create or open and add information but won’t erase initial information in a binary form)

*loadGrades*:

Next up was creating a function that loaded the data from the function storeGrades in the prior step. In this case we did a try command with an except that dealt with exception handling, that will be discussed later in this paper. We opened the strFile which we created earlier on “rb” which is read mode but also in binary, we further have to unpickle it which is discussed later but the outcome is the file is loaded into our program.

Graphical user interface, text, application

Description automatically generated(fig. 3 function loadGrades)

A for loop was used bringing up the dictionary keys displaying what was initially in the program. What is displayed isn’t the most elegant but it does convey the information.

*Presentation and Variable Handling:*

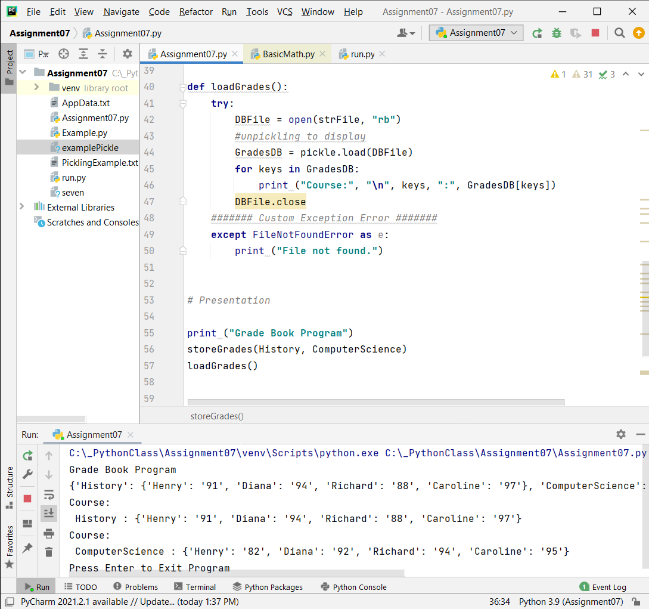
Finally, the program runs the two functions that were scripted storeGrades and loadGrades. A few points were added to get the script to pause and or display data for screen capture. A print and input key were added to just pause the program before our functions were called.

*Text

Description automatically generated with low confidence(fig. 4 Adding Breaks)*

This is followed by our functions and finally printing or GradesDB which was the dictionry we set up in the storeGrades function. Although in this case I had created a local variable in the function as opposed to a global variable so, I added the variable from the storeGrades function outside of the function to have information to be drawn upon.

Final results of our program running:

 Text

Description automatically generated

(fig5 pycharm running of our program) (fig 6 OS Command shell running of the program)

*Pickling*:

We were asked to highlight the technique of pickling in this assignment. This technique is converting our file, in this case the information from the storeGrades function and storing it in a binary manner in a strFile that we created to open called “PicklingExample.txt”. To accomplish this we start with the command *import pickle*, which imports from python the information to do this conversion process. We do it at the beginning of the script so the subsequent script can access it if it is needed.

Graphical user interface, text

Description automatically generated(fig. 7 Our Pickling)

In our storeGrades function we create a variable DBFile which opens our strFile (which inturns creates or is linked to our “PicklingExamples.txt file) in “ab” which appends the information. We ad the “b” so it is opening and looking for the information in binary mode, which is how pickled data is stored. We further use the commands pickle.dump to take the variable we created GradesDB and DBFile (or rather the information we are asking to be stored and the place for it to be stored) to finish the pickling process. This results in the information being stored as a binary file which obscures our data.



(fig. 8 Showing our pickled data)

The process is unpickled to display our data in the loadGrades function. In this function we load our “PicklingExample.txt” from a binary file back to information we can see and utilize using the command pickle.load, sourcing our .txt file.

(fig 9 getting our pickled data back)

The final result of this is that we were able to pickle our information, converting it and storing it binary and then able to use is in an unpickled manner further down the line.

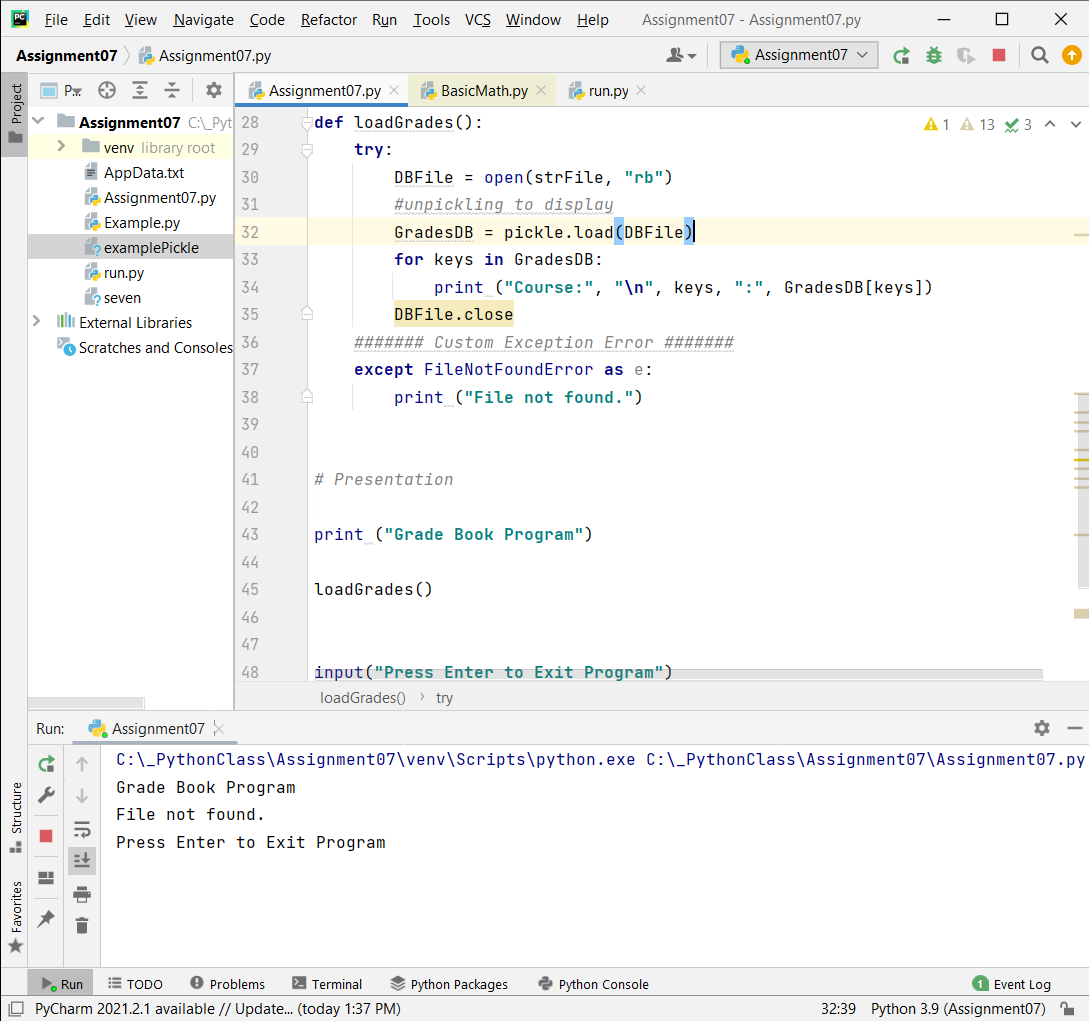
*Exception Handling*:

Exception Handling is a technique that is used to identify where errors could happen, by a user or another developer who is altering or running our program, and then create a custom message so that the user can identify what is happening and where they can go back and run the program as intended. In this program a Exception Hnadling was utilized in the loadGrades function. Specifically, a “FileNotFoundError in the e directory was used.

A picture containing text, indoor, orange, dark

Description automatically generated(fig, 10 our exception handling)

So instead of returning an error message that would be hard for a novice or beginning user to decipher the program instead prints out “File not found”. This was accomplished using a try and except loop. Where is our loadGrades function we have the script loading our “PicklingExample.txt” in “ab” and then providing what to do in an except mode which is what we coded: print “File not Found”.

(fig. 12 Tripping the exception handling)

To trip this error I had to go back in the script and delete the function storeGrades and the call for a different .txt file which did not exist to trip this error message. We do get the outcome which is desired in that a message is printed indicate what happens, which allows for troubleshooting far easier for the end user or someone adapting the script.

**Summary:**

In this assignment we created a script that highlighted Pickling and exception handling. Creating a script was easier than adapting someone else’s script for me, but I still ran into quite a few obstacles that some more practice and experience will help. I did have to deal with quite a bit in some unconventional manners in coding both global and local variables. Additionally having to delete information to trip an error message and adding code to get an OS Command shell to display information seeing as my initial program only loaded and stored data that was hardcoded into the program.