Chani Orr

8/23/21

Foundation of Programming: Python

Assignment07

<https://github.com/GitWreckedChani/IntroToProg_Python_Mod07>

<https://github.com/GitWreckedChani/IntroToProg_Python_Mod07/blob/main/index.md>

Files and Exceptions

Purpose of The Lab

In this lab we explored how to save to and read from binary files, and looked more in depth into how to use try/except calls to handle exception objects. For application, we documented this knowledge in a GitHub webpage using GitHub flavored Markdown.

We also answered the following questions: What are the benefits of putting built-in Python *commands* into *functions*?; What are the benefits of using *structured error handling*?; What are the differences between a *text file* and a *binary file*?; How is the *Exception class* used?; How do you *"derive"* a new class from the *Exception class*?; When might you create a class derived from the *Exception class*?; What is the *Markdown language*?; How do you use *Markdown* on a *GitHub webpage*?

Getting Started

In this module we were given a directive on how the program should run and a starting script to build from. The starting script laid out a plan for the script, hence I forewent the pseudocode.

The Difference Between Text Files and Binary Files  
This project used similar functions to what we had used previously, but with some key differences. In this unit, I created two functions that would interact with a binary file: one to read data from a binary file, and one to write data into a binary file. Notice that in the function to save the file, we are able to shove the data into a text file whole-sale but using the ‘dump’ method, as

| (Image 1.0 - Save Data To A File Function) | opposed to having to pack it from a list into a dictionary into a tuple and thread it into the file. Similarly to previous functions, the first parameter is the file where we want the data saved to. The second parameter, however, is “wb” - meaning “write-binary” - designating the way this data is to be saved. |
| --- | --- |

| Notice that in the text file, the text is difficult to read...but not impossible. This information is a binary stream type - the benefit of which being that it uses less memory - but it is not encrypted in the way that we think of secure information. | (Image 2.0 - Notepad, “AppData.dat”) |
| --- | --- |

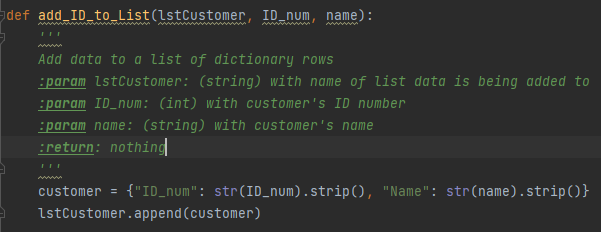
| (Image 3.0 - Read Data From A File Function) | The read function differs from previous read functions in that the second parameter it requires is ***“rb”***, meaning “read-binary”. Also of note is the ***load*** method (not to be confused with ***loads*** method), which takes the file path and deserializes it. It returns the object as a dictionary, which is then applied to the variable “cust” (for customer), then the file is closed. I set the return to also print the variable returned, and while it does not seem to |
| --- | --- |

have impacted the code any more than just having ***print(cust)*** without the ***return*** - nor has returning nothing and calling to ***print(cust)*** elsewhere in the program, I wonder at what the practical difference is and under what circumstances this might matter.

Buddy Functions

Add I.D. To List Function

I needed a few other helpful functions to make the code run. Since I was given a list, I wanted a way to add things to that list. Once I had a customer’s identification number and name, I could append this to the Master List of customers. Notice that these are the three parameters for the values called by the function: list name, ID number, and customer name. I did not set it to return anything because the list being appended is not technically a new product so I did not think it was needed, although I am uncertain of this action.



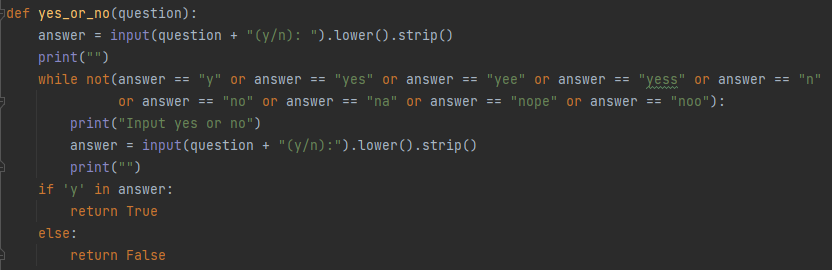
(Image 4.0 - Add A Customer To Our List Function)

Input Id Function

| The input I.D. function is where I get input from the user in the form of an identification number and a name and save these to respective variables. | (Image 5.0 - Input Customer Identification and Name Function) |
| --- | --- |

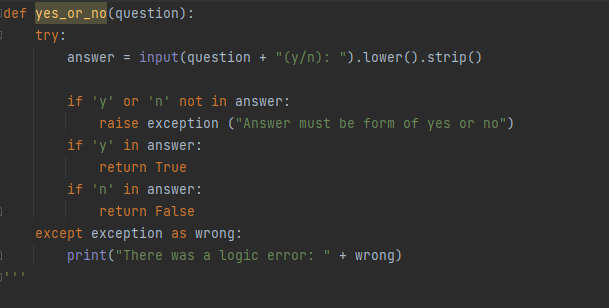
Two Yes-Or-No Functions

Initially I wrote a function that looked like this: the while loop allows for a few pre-determined variations of yes or no, then returned anything recognized as a yes in answer as True and anything else (catch all) as False. If the user answered with anything not approved, the loop would repeat the question.



(Image 6.0 - The First Yes\_Or\_No Function I wrote)

When I finished and tested the program, I went back and wrote the second yes/no function as a try/except function that would check for ‘y’ or ‘n’ in answer.



(Image 6.1 - The Second Yes\_Or\_NO Function)

It was during the first test that my roommate and critic sat down next to me on the couch to do some coding of his own. After side-eyeing my code for some time, he questioned me on it.

| Collin: | “Why did you do it as a catch?” |
| --- | --- |
| Me: | “A catch?” |
| Collin: | \*gestures\* |
| Me: | “We learned about try/except the past two modules.” |
| Collin: | “But why there?” |
| Me: | (o.o) |
| Collin: | “Did you put it there because you wanted to demonstrate it and didn’t know where else to do it?” |
| Me: | \*embarrassment intensifies\* |
| Collin: | “Okay, explain to me what a try/except does.” |
| Me: | \*Gives vague, unsatisfactory answer\* |
| Collin: | “Alright. There are better ways to do this.” \*scrolls up and sees my previous yes\_or\_no function commented out\* “See, I like that one much better. Swap those. You want to try/except the parts that are more likely to crash the program. Where are you actually using these functions? Yeah, the main body. You should use a catch to wrap that.” |
| Me: | \*stares incredulously\* |
| Collin: | “ ‘Wrap’. It’s exactly what it sounds like.” |
| Me: | “You can do that?...Ok if you say so.” \*stabs at the code like a baby chewing on a text book.\* |
| Collin: | \*looks up again\* “What are you setting the except to? Don’t look at me, this is a Python thing; you’re on your own.” |

I attempted to explain the except clauses I could remember and form a justification for each one based on what the code contains. We ended up researching possible excepts together to find which was most applicable. In the end he suggested a catch-all except with a customized message. I commented out the second function, but left it in the code to remind myself to put into words my posibile initial misconception of the purpose of the try/except function.

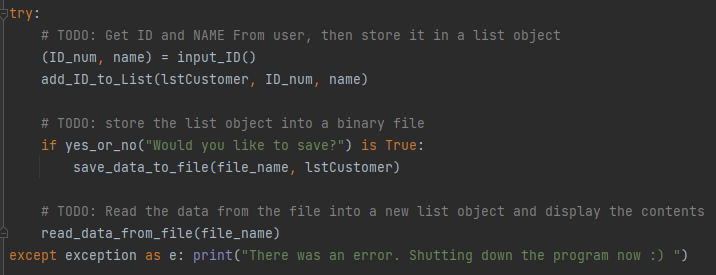
I think I was thinking something like:  
Dev: I asked them for ‘y’ or ‘n’...  
Code: They’re prolly gonna type “yeet fam” or some \*\*\*\*  
Dev: Hmm...okay, if you don’t see ‘y’, try ‘yeet fam’ and accept that as a yes  
Code: \*rolls eyes\* whatever

Dev: Oh! Also try ‘yee’! And ‘ye haw’!  
Code: Fiiiiiine uhg

This brings us at last to the main body of the code.

Main Body

Once the program finally interfaces with the user, it requests an I.D. and name. It then saves those and appends them to a list. Next it asks if the user wants to save said list to a file. If yes, the code saves that list to a binary file. Then it reads that file back to the user. And if any of that fails, it alerts the user that there was an error and that the program is closing.



(Image 7.0 - The Main Body)

Summery

To summarize, pickling and unpickling data to a file is different from packing and unpacking it in that pickling saves it as a bytestream. Over the course of a particularly large code, this could mean using significantly less memory. Try/except is useful in crash-proofing your code and can even be used to make code more user friendly.

| What are the benefits of putting built-in Python commands into functions?: | They can be used across multiple scripts, as well as made customizable. They are clean and orderly and can help cut back on confusion and redundancy. |
| --- | --- |
| What are the benefits of using structured error handling?: | You can fix your bugs immediately and make sure the code runs smoothly. |
| What are the differences between a text file and a binary file?: | Storing data in a binary format can obscure the file's content and may reduce the file's size. |
| How is the Exception class used?: | "Exception" is a built-in python class used to hold information about an error. Python automatically creates an Exception object when an error occurs. |
| How do you "derive" a new class from the Exception class?: | Derived classes "inherit" data and functions from the base class that can be replaced and customized. |
| When might you create a class derived from the Exception class?: | When you want to communicate the specifics of an error to someone not familiar with exception jargon. |
| What is the Markdown language?: | Markdown is a plain text formatting syntax for writers. It allows you to quickly write structured content for the web, and have it seamlessly converted to HTML |
| How do you use Markdown on a GitHub webpage?: | Use can use it to make text for webpages |