Connor McNiff

June 1, 2020

Foundations of Programming: Python

Assignment07

<https://github.com/cmcniff/-IntroToProg-Python-Mod07/upload>

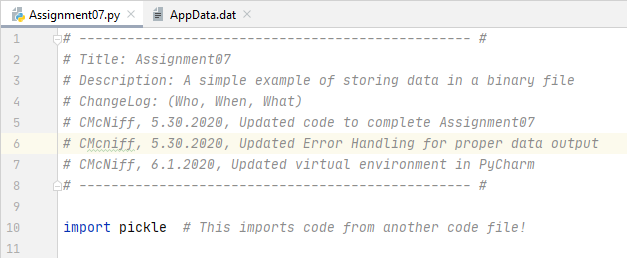
**Pickling and Error Handling**

Introduction

In this week’s assignment we were tasked with writing a code using pickling and error handling. As a quick overview, Error handling is the use in code to protect against potential failures that would cause the code to exit while pickling converts data to make the file smaller.

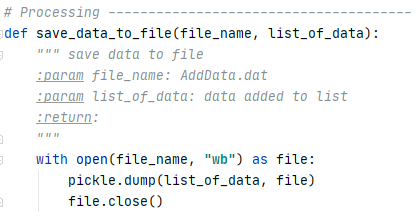
Pickling

Building off the brief explanation of pickling above, pickling can be described as serializing and de-serializing Python object structure. In previous week’s assignments we saved data in just “plain” text form where pickling saves data in a binary format; referring to the serializing and de-serializing process. The website <https://www.datacamp.com/community/tutorials/pickle-python-tutorial#what> gives a step by step explanation of what pickling is, how to use pickling and all parameters of pickling in Python. Serializing the object in Python converts it to character stream and reconstructs it to save space. The first step of writing this script was to identify that the data being read and wrote from the “AppData.dat” file would be converted to a byte stream as shown in Figure 1. After initially setting up the “import pickle” function, two methods of pickling were used in this assignment, dump and load.



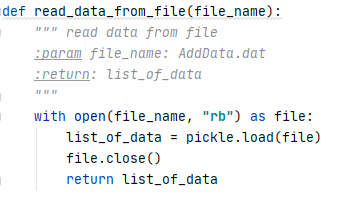
**Figure 1 – Import Pickle**

The first processing definition that was defined in our script was “save\_data\_to\_file()” where we identified the file name (where we would be transferring to pickle data to and from), the data being read and written, and the format in which the file would be utilized (“wb” for written in binary form). One the file is opened using the “open()” function the “pickle.dump()” function was used to identify the data that will be pickled and the file to which that data will be saved. When both arguments are defined, the file is instructed to close. All arguments are defined within the “save\_data\_to\_file() definition in the “Processing” section of our script which is shown in Figure 2 below:



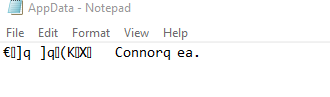
**Figure 2 – Pickle Dump**

The second processing definition that was defined in our script was “read\_data\_from\_file()” where we loaded our pickled data back into Python. Using the “open()” function we used the same file identifier except with the “rb” argument (for read in binary from) and used the “pickle.load()” function to load data into Python that is identified as “list\_of\_data”. The contents of the “file” that were pickled are now not in binary form and assigned to this new “list\_of\_data” variable. Once all of that is defined the file is then closed as shown in Figure 3 below:



**Figure 3 – Pickle Load**

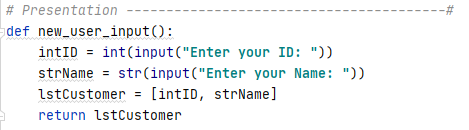
Figure 4 below shows the file we used in our processing definitions “AppData.dat” and how it stores our data in binary from:



**Figure 4 – AppData.dat Binary Data**

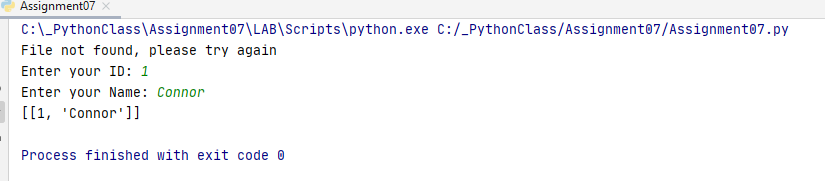
Presentation

Instead of defining our “list\_of\_data” variable at the beginning of the script, we used input() functions to obtain user data to be used in our pickling that was explained above. This script called for the user to enter “ID” and “Name” data using the data variables “intID” and “strName” and putting all data together in one variable titled “lstCustomer” in which our “new\_user\_input()” definition will return the combination of user input data variable “lstCustomer” as shown in Figure 5 below:

****

**Figure 5 – New User Data**

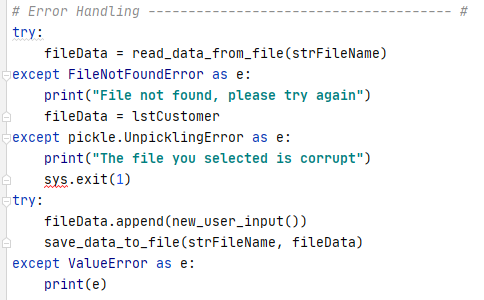
Figure 6 below shows the processing functions running and displaying the binary data from Figure 4 in PyCharm successfully:



**Figure 6 – Running in PyCharm**

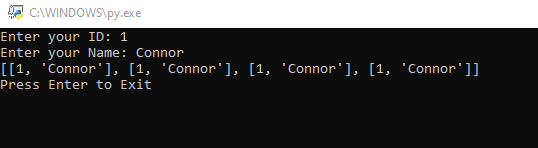
Error Handling

As stated in our introduction, error handling is built into our script to protect against failures and errors. In our script we used the “try – except” method block of code where we can trap user errors. It was recommended in this week’s module notes that we add a try-except block when human interaction is anticipated. The try-except block used in the script explains that if the file identified in our C drive as “AppData.dat” is not found then present the “lstCustomer” data for the user to enter data and write that data into the defined “AppData.dat” file from our script path. Another try-except block used was for pickling data that was corrupt where we exit the corrupt file and create a new file from user input data. Figure 7 below shows the two exceptions applied and what should be “tried” to fix those errors:



**Figure 7 – Try Except Block**

Figure 8 below shows the program running in the command prompt:



**Figure 8 – Running in Command Prompt**

Summary

From this week’s script we built off of utilizing functions and utilized pickling to save data size and error handling to prevent user errors. Overall, we developed more complex scripts to simplify and streamline our programs.