# Annie Ng

# Mar. 6, 2020

# Foundations of Programming: Python

# Assignment07

**Creating Scripts With Pickling and Error Handling**

# **Introduction**

In this assignment, I will show the steps on creating a script with the pickle module and error handling in Python. This assignment introduces another way to save data to a binary file instead of a text file. We are building more skills using various functions and handling unexpected errors from the user’s input.

## **Pickle Module (Pickling)**

Pickle module is a module in Python that allows you to save and store complex information (python objects) in a format that is not a text file, but in a binary file. You can pickle numbers, strings, tuples, list, and dictionaries (Python Programming for Absolute Beginner, 3rd Edition, Michael Dawson, Cengage Learning 2010, p. 200-202). A binary file will contain code that has symbols to represent the code and is not human-readable, but is computer-readable (<https://www.webopedia.com/TERM/B/binary_file.html>). The advantage of using pickling is that you can save a python object to a file and retrieve the object to make changes. The link below shows how pickling is used:

<https://www.youtube.com/watch?v=Pl4Hp8qwwes> (External Site).

<https://pythonbasics.org/pickle/> (External Site). This website explains how to use pickling to save/serialize objects. Unpickling means to deserialize the object/load the object from a disk.

Although pickling sounds useful, there are some things you must be careful when the pickling. Pickle files can be hacked, so do not open a pickle file from an untrusted source (<https://wiki.python.org/moin/UsingPickle> , External Site). Also, you cannot pickle classes, functions, and methods. So just have to be aware when using the pickle module.



***Figure 1. Script Using Pickle Module to Write and Read Binary File***

******

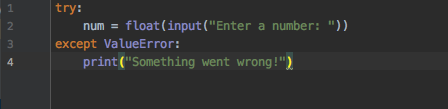
***Figure 2. Data Saved in a Binary File***

## **Exception/Error Handling**

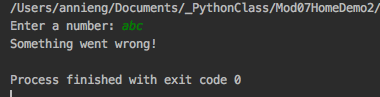
Usually when a program runs into an error in Python, the program stops and an error message appears, which means an exception is raised and something exceptional has occurred (Python Programming for Absolute Beginner, 3rd Edition, Michael Dawson, Cengage Learning 2010, p. 205). We can anticipate and catch these exceptions/errors using the “Try-Except” block. The try statement is to “section off some code that could potentially raise an exception” and the except clause is the statement shown only if the exception is raised (Python Programming for Absolute Beginner, 3rd Edition, Michael Dawson, Cengage Learning 2010, p.206). If no exceptions are raised, the program skips the except clause and continues to the rest of the script/code.

In Python, there are built-in exceptions types such as IOError, IndexError, KeyError, NameError, SyntaxError, TypeError, ValueError, and ZeroDivisionError (Python Programming for Absolute Beginner, 3rd Edition, Michael Dawson, Cengage Learning 2010, p. 205-207). You can include the specific exception in your code in the except clause to catch specific errors. This is a good practice rather than using the general except clause because it could be any error.

It is possible to create custom-made exceptions, which can be more friendly to the user and the language used is less technical compared to the messages in the built-in exceptions in Python. Additionally, there can be multiple exceptions in the “Try-Except” block, catching more specific types of exceptions (<https://www.python-course.eu/python3_exception_handling.php>, External Site).



***Figure 3. Demonstration of the Try-Except Block for Exception/Error Handling***

******

***Figure 4. Output of the Exception/Error***

## **Pseudo-Code**

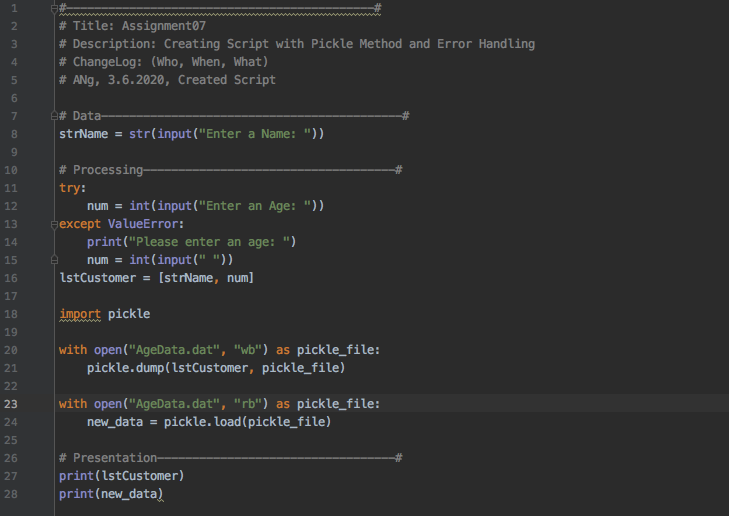
The pseudo-code we have been practicing in this class is to organize our script into three sections: data, processing, and presentation with a script header that includes title, description, and change log.

## **#Data, Processing, and Presentation**

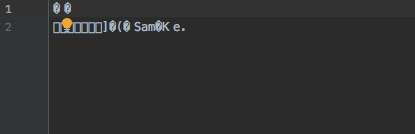
In my script, there is the pseudo-code of the data, processing, and presentation sections. In the data section I have global variable where the code prompts the user to enter a name. In the processing section, it prompts the user to enter an age in the try-except block as a local variable. If the user does not enter the age, the exception is raised and here I put in a ValueError to indicate that the input should be a value type. Again, if the exception is raised, then the message “please enter an age” will prompt the user to enter a number. Here, the exception/error is caught and redirects the user to enter information that is being asked for.

Once the information is entered, I pickled the list (name and age) into a binary file, using the import pickle function. Using “with” in the pickle function open and closes the file automatically without having to write the open and close separately on each line of the code. I created a binary file called AgeData.dat and opened the binary file in write mode and specify the object file as pickle\_file. Next, I saved the data, lstCustomer with pickle.dump to the file object pickle\_file. To read the data we would open the connection of AgeData.dat in read mode and create a new variable, new\_data for the data that is loaded from the pickle\_file.

Lastly, the code prints the name and age in a list in the presentation section and unpickles the file printing the new\_data from the pickle\_file.



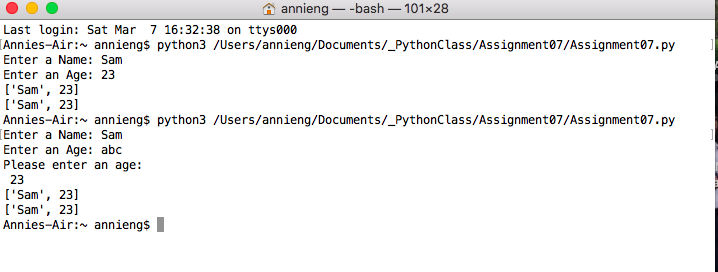
***Figure 5. Script with Pickling and Error Handling***



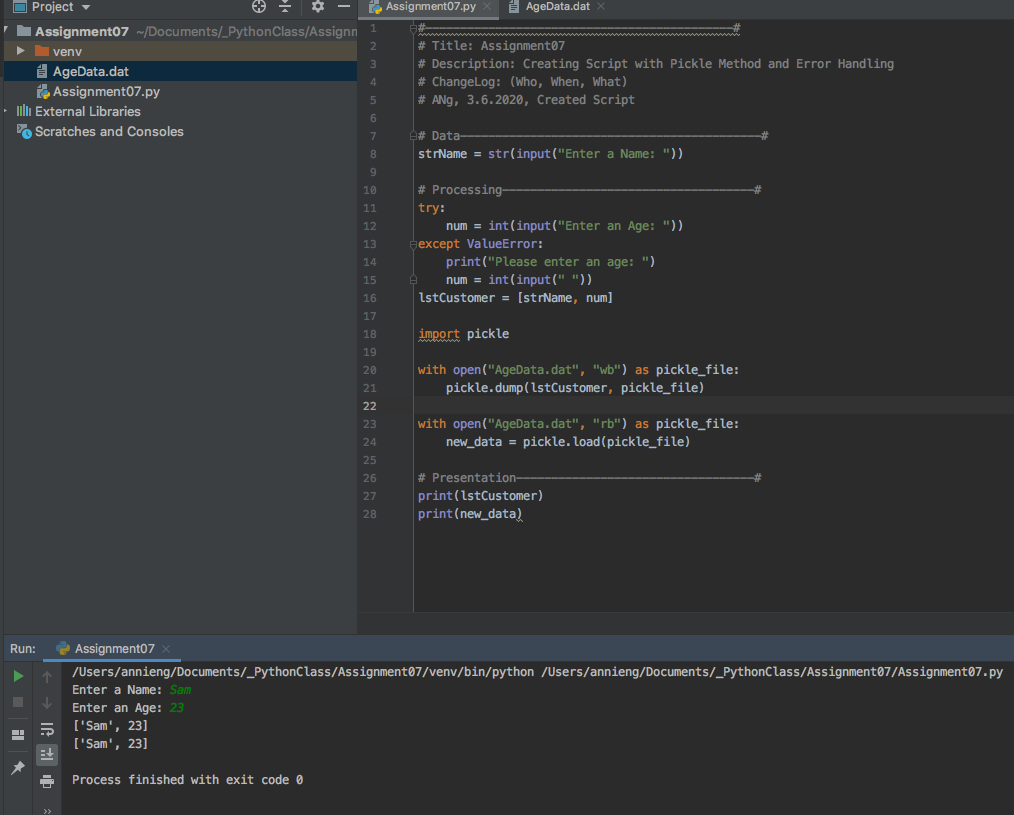
***Figure 6.******Screen shot of Data Saved in a Binary File***

**Summary**

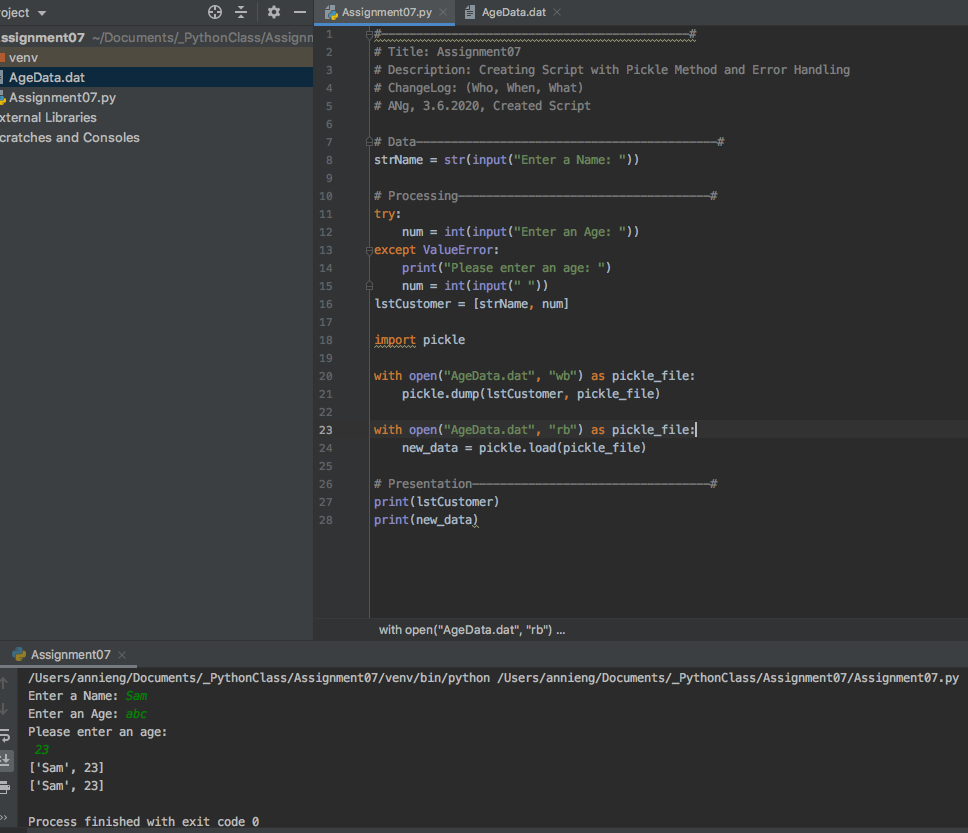
In this assignment we are creating a script with pickling and error handling. Pickling involves saving and reading from a binary file. Error handling, also known as exception handling, allows us to catch errors during a program’s execution.



***Screen shot of Script in Terminal Mac OS***

******

***Screen Shot of Script Running in PyCharm with Pickling***

******

***Screen Shot of Script Running in PyCharm with Try-Except Block Catching Exception***