

Chloe Wei  
May 31, 2020  
IT FDN 100 A  
Assignment 07

[GitHub webpage](#)

## Introduction to Pickle module & Error Handling

Now that I have had opportunities to explore the basics of coding with Python on PyCharm, I will explore and explain my use of Python's "pickle" module and use of error handling within my code. This will be a simple code based on this week's Lab07-1 script to take a Customer and ID.

### My Approach to the Pickle & Exceptions

#### Problem Solving by Organizing Your Thoughts

There are always days when having unobstructed coding time is not an option. Therefore, I benefit from writing pseudocode and an accompanying flowchart. With this, I can break-up my code to manageable chunks, make changes to blocks, and staying on track when I do have to set a project aside for another day. I adjust the flow chart & pseudocode first when debugging, then make the corrections to my code. It may be an extra step, but it keeps me organized when debugging my code.

#### Create a Pseudocode & Flowchart

Since my ultimate goal would be to add new customers to my database, I worked out the following pseudocode to add a new customer, create a data file (unless it already exists) and be able to see my list of customers:

```
Run Assignment07.py file
Load DB Menu
  If 'Add Customer' selected
    Ask for Customer First and Last Name
    Ask for 3 dig ID
    Try/Except: unpickle dat file
      Except IOError: dat file not found
      Else: Unpickle data
  If 'Save List' selected
    Pickle data to 'AppData.dat'
  If 'View List' selected
    Display CustomerID db
  If Exit, dbl check
    if 'y' to
      save before exit
    else:
```

Return to menu  
End session  
Close Terminal/Window

I then created the following flow chart based on the pseudocode, which helps me sort my functions and statements as I start adding meat to my script:

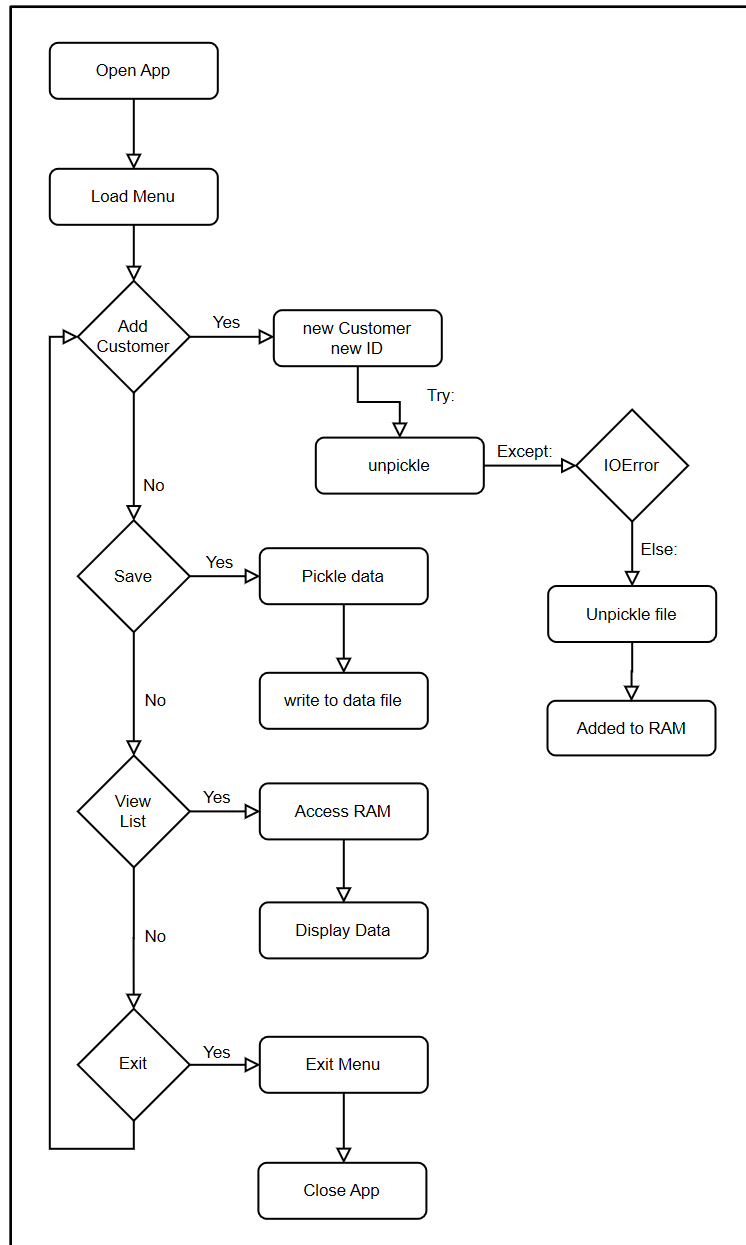


Figure 1: Map of Customer\_ID db

As I test each function, I follow along with my flowchart then get a better idea of where I need to focus my attention to when debugging in PyCharms. This came in handy when I was creating the view function (how to approach showing a list from a file that doesn't exist) and the later

stages when cleaning up my revised code. The above pseudocode and flowchart are the 5<sup>th</sup> and 4<sup>th</sup> versions for the CustomerID db.

## Error Handling

Now that the core of my script is created (functions to handle the processing and input/output, I/O), I can now address how I could handle possible errors in my code.

### Try for Exception `IOError`

Error Handling is an especially useful assertion for any system that takes instructions. It only returns an error (i.e. wrong data types, syntactic errors) when the exception is true otherwise stays quiet in the script. Although the interpreter will catch and return information about these errors, Try/Except statements will keep the application running while returning to the UI why an error occurred and how not to trigger it again.

I will show how it works for 'Add Customer' block of code that checks if a file exists. Before I add the new customer to memory, the try statement will attempt to read the data file, 'AppData.dat'. If this is the first instance of our application, the exception will catch this process and return an `IOError` then display "Remember to save your session."

```
try:
    lstCustomer = Processes.read_data_from_file(strFileName)
except IOError:      # except Error: if file not found, continue with ram data
    print("\nRemember to save your session.")
    print()
else:
    print("Customer added to database")
    print()
```

Figure 2: Check for data file before loading data

This happens because the data file has not yet been created or is not located in the same folder as the Assignment07.py file. Because we used the Try/Except statement, I can keep the application running but am alerted that data does not exist in any form (data file or in RAM). The exception allows me to continue my session and add data & create the data file. As you can see with the figure 3, the UI will display an empty list if nothing was added by either UI inputs or from a file (since a file has yet to be generated).

```
New database will be created

*****
ID | Customer Name
-----
*****
```

Figure 3: No data file found & no data in RAM

I can now enter my new customer and create the data file when I save my data. I just made up a name and ID to test my code. The below example (figure 4) displays to the UI a reminder to save the session before exiting the app after it adds the customer, Happy Cat, to the customer list.

**\*\* Note \*\*** For every instance a new customer is added, and the list was not backed up to the file 'AppData.dat', this reminder will be returned to the display.

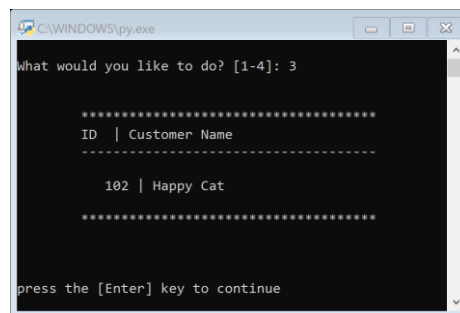
```
Enter Customer First & Last name: Happy Cat
Enter 3 digit number for their ID [ex 123]: 102

Remember to save your session.

You have enter customer id for : Happy Cat
```

Figure 4: PyCharm UI after adding a Customer to db

Creating a return in the except stated can also be added to the else statement. By including an else statement to the Try/Except, I can now control what will be displayed to the UI if the error was not triggered. Therefore, regardless if the data was from a file or from RAM, any customers (& the respective ID) will still be displayed if it exists in the customer list.



```
C:\WINDOWS\py.exe
What would you like to do? [1-4]: 3

*****
ID | Customer Name
-----
102 | Happy Cat
*****

press the [Enter] key to continue
```

Figure 5: Command UI displaying only customer, Happy Cat

If I were to select the 'View List' menu option, any value in the variable, `lstCustomer`, will now be filled regardless if it was backed up or not.

### Pickle: Load and Dump

Speaking of reading data from the data file, I will share how I used the pickle module for my simple code.

```
import pickle # This imports code from another code file!
```

Figure 6: Important to import the pickle module

Importing the pickle module is just like import a library for other languages. By including it in the first line of code after the script header, I can now save my database as binary data in the 'AppData.dat' file. Reading (and writing) a collection of data from (or to) a text file using pickle is easier and requires less code than the conventional `write()` function in python.

```
objFile = open(file_name, "wb")
pickle.dump(list_of_data, objFile)
objFile.close()
```

Figure 7: saving list to dat file

To write data to the data file, just took these three lines of code within my `save_data()` function. And as you will see in figure 8 (below), only three lines for reading my data as well!

```
in_file = open(file_name, "rb")
list_of_data = pickle.load(in_file)
in_file.close()
```

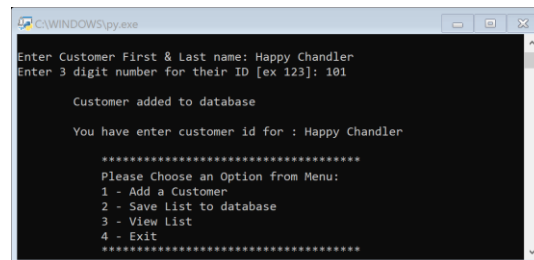
Figure 8: deserializing list from dat file

By pickling the data, I can access and manipulate any compilation of data without using loops to write to (or from) data files. After I retrieve and convert my binary data back to a list table, `lstCustomer`, I just use a two line for loop to display the data now saved in RAM (figure 9 below).

```
for i in list_of_rows:
    print("\t\t\t" + i["ID"] + " | " + i["Customer"])
```

Figure 9: returning Customers found in `lstCustomer`

Now that I have tested my code a few times more, saved my data file created, saved as binary, then read back as my original list, I tried adding another 'Happy' customer to list of customers.



```
C:\WINDOWS\py.exe
Enter Customer First & Last name: Happy Chandler
Enter 3 digit number for their ID [ex 123]: 101

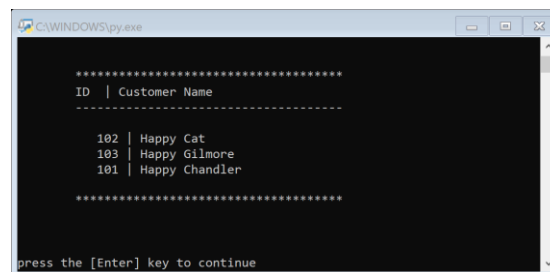
Customer added to database

You have enter customer id for : Happy Chandler

*****
Please Choose an Option from Menu:
1 - Add a Customer
2 - Save List to database
3 - View List
4 - Exit
*****
```

Figure 10: Display new return post data file created

Now the UI will return 'Customer added to database' because the 'Appdata.dat' file was finally generated.



```
C:\WINDOWS\py.exe

*****
ID | Customer Name
-----
102 | Happy Cat
103 | Happy Gilmore
101 | Happy Chandler
*****

press the [Enter] key to continue
```

Figure 11: New list in memory

**Note** Although my new 'Happy' customer presents in the list, we will still need to save to the newly created data file or lose it when ending my session.

Figure 12: double check before exit

As a precaution, I included a double check to return prior to ending the session. It's a little insurance to not lose any data added after my recent save. If the user of the app did save, it won't hurt the data file, but will just overwrite with the data stored in memory.

### What is the Purpose of Pickling?

Although pickling our data does not make it secure or invulnerable to cyber-attacks and may not be compatible with other programming languages, pickle offers convenience of shared data to be sent and retrieved between *safe channels* by marshalling (serializing) python objects (in this example, lists of dictionaries).

As demonstrated in my included code, Assignment07.py, very few lines of code to read and write from files, hardly any complicated while or for loops to produce a streamlined UI experience. Additional information on [python.org](https://python.org) & *Real Python* gives more uses and examples of how to use pickle, advantages, functions, and other modules (similar to pickle) to use alongside or in lieu of the pickle module (ex, shelve, dill, and creating zip files for larger files).

### Summary

Python is really a great steppingstone for new coders as well as those with an immense coding repertoire. Utilizing the pickle module and incorporate error handling to my python script will just help me create a better UI experience. *Data camp* helped me with the correct syntax for error handling and I really loved the clarity of *Real Python*'s explanation of pickling. In the book "Python for the absolute beginner" Pp.201-4, I was also introduced to the inclusion of the shelve module. However, for my sample script, shelve, zip and data security was not needed. Other sites that helped me understand the pickle module are listed below. Hope this has been a helpful sample of error handling & pickle for python.

Sites:

Error handling:

(data camp) <https://www.datacamp.com/community/tutorials/exception-handling-python>

(tutorials point) [https://www.tutorialspoint.com/python/python\\_exceptions.htm](https://www.tutorialspoint.com/python/python_exceptions.htm)

Pickle:

(Real Python) <https://realpython.com/python-pickle-module/>

(Python.org) <https://docs.python.org/3/library/pickle.html>

(Python Basics) <https://pythonbasics.org/pickle/>

(data camp) <https://www.datacamp.com/community/tutorials/pickle-python-tutorial>

(geeks for geeks) <https://www.geeksforgeeks.org/pickle-python-object-serialization/>