Name: Charles Lin
Date: February 22, 2023
Course: IT FDN 110 A
GitHubURL: https://github.com/charleslin89/IntroToProg-Python-Mod07

Assignment 07 Files and Exceptions

Introduction

I learned about working with text and binary files, getting to the specific row, ways to build error exception codes, using pickle function to read binary file, and building GitHub web page.

Reading and Writing Text Files

We can open(), write(), and close() text files. After we open a file, we can also pick out specific rows to put in a list by looping through rows and looking up the particular value match. When writing, we have the option to append to existing rows or overwrite the whole file. Codes relating to file treatments can be grouped into a function.

In addition to plain text files, there are binary files that are obscured but not encrypted. The content is harder to read in a binary file. In Python we can dump the content and use pickle.load method to read the content.

Pickling Feature

Pickling is used to store python objects. This means things like lists, dictionaries, class objects, and more. If we have a large dataset, and we're loading that massive data set into memory every time we run the program, it makes a lot of sense to pickle it, and then load that instead, because it will be far faster than a csv text file.

The Python pickle module -
https://realpython.com/python-pickle-module/
The process of serialization to send complex object hierarchies over a network or save the internal state of your objects to a database. Serialization converts a data structure into a linear form that can be stored or transmitted over a network. Python serializes objects in a binary format, which means the result is not human readable. Though it's faster, and it works with many more Python types right out of the box. Python team use terms pickling and unpickling to refer to serializing and deserializing. We can dump() or load() data in pickle module.

Python Numerical Methods' Pickle Files
https://pythonnumericalmethods.berkeley.edu/notebooks/chapter11.03-Pickle-Files.html

The term “pickle” comes from saving dictionaries, lists, etc to share with others, like pickling vegetables. The link was published a few years back, but I find the pickle concept and examples easy to follow.

Error Handling

Try-Except lets us communicate error messages that are more meaningful. We can customize the error message by

Script 5: Error handling -
https://www.linkedin.com/learning/search?keywords=python%20error%20handling&u=2091572
Notes: Building in Try-Except gives us the option to let other portions of the job to run, instead of failing all together.

All About Exceptions -
https://www.learnpython.dev/03-intermediate-python/40-exceptions/10-all-about-exceptions/
Notes: The navigation of the webpage is curious, but the content layout is similar to Professor Root’s lecture material.

Assignment with Pickling and Structured Error Handling

I want my script to do the following:

1. Initialize and seed a dictionary with 2 customers
2. Write to a binary file
3. Read content from the binary file
4. Write to a text file Observe how data looks on binary and text files
5. Turn reading binary file into a function
6. Add try/except to read text file
7. Place try/except in the read function to treat scenario when text file is not found
8. Show data to user and ask if a user wants to write to a binary or text file
9. Write data to what the user selected

On Pickling and Structured Error Handling (Try/Except)

Pickling

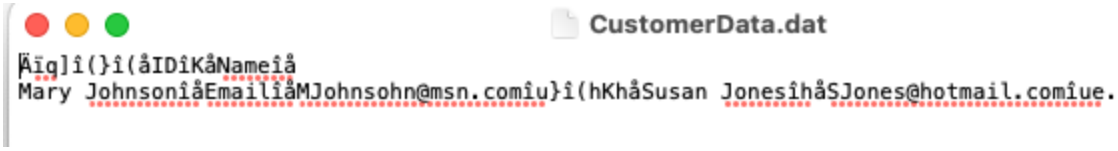
The code writes 2 rows of customer data from a list of rows to a binary file, then deserializes and reads the binary file’s content to a dictionary.

```python
# Processing ------------------------------------------------------------
# Pickle Example

import pickle  # This imports code from another code file

# initializing customer data to be stored in dictionary
dicRow1 = {"ID": 1, "Name": "Mary Johnson", "Email":"MJohnsohn@msn.com"}
dicRow2 = {'ID': 2, "Name": "Susan Jones", "Email":"SJones@hotmail.com"}
lstCustomer = [dicRow1, dicRow2]

# Now store data with pickle.dump
objFile = open("CustomerData.dat", "ab")
pickle.dump(lstCustomer, objFile)
objFile.close()

# Read data back with pickle.load
objFile = open("CustomerData.dat", "rb")
objFileData = pickle.load(objFile)
objFile.close()

# Display data by the dictionary keys and format with pipe
for objRow in objFileData:
    print(str(objRow["ID"]) + '|' + objRow["Name"] + '|' + objRow['Email'])
```

Listing 1 Try/Except to give meaningful message

```
Assignment07
/Users/charles.lin/PycharmProjects/Assignment07/venv/bir
1|Mary Johnson|MJohnsohn@msn.com
2|Susan Jones|SJones@hotmail.com

Process finished with exit code 0
```

Figure 1 The result of List 1



Listing 2 Binary file opened in TextEdit reader

I wondered if the size of binary file helps with faster processing speed. I wrote the list from Listing 1 to a txt file to observe the size difference. Txt file was smaller than binary file. So I think the faster processing comes from an easier processing mechanism, and not from the file size.

```python
# Display data by the dictionary keys and format with pipe
for objRow in objFileData:
    print(str(objRow["ID"]) + '|' + objRow["Name"] + '|' + objRow['Email'])

# Observe file size difference if written to a text file
objFile = open("CustomerData.txt", "w")
for objRow in lstCustomer:
    objFile.write(str(objRow["ID"]) + ',' + str(objRow["Name"]) + ',' + str(objRow["Email"]) + '\n')
objFile.close()
print()
print("Text file also has been written.")
```

Listing 3 Write to a txt file

Figure 2 Txt file from Listing 3

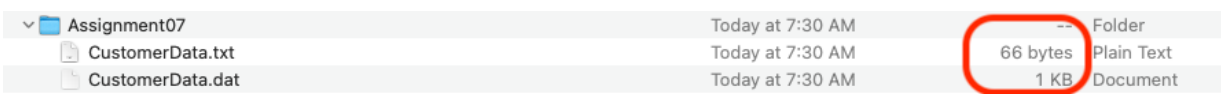


Figure 3 Size comparison between text and binary files

Try/Except

When I read a file, if a file doesn't exist and I didn't include any try/except, I would get a FileNotFoundError, and the job stops. In this example, I missed the last "a" in CustomerData. I have "CustomerDat.txt" in the script, and the job stops there.

```python
# Try/Except Example
objFile = open("CustomerDat.txt", "r")
for objRow in objFile:
    lstRow = objRow.split(",")
    dicRow = {"ID": lstRow[0].strip(), "Name": lstRow[1].strip(), "Email": lstRow[2].strip()}
    lstTable.append(dicRow)
print("Try Except Example of reading a text file:")
print("------------------------------------------------------------")
for objRow in lstTable:
    print(str(objRow["ID"]) + '|' + objRow["Name"] + '|' + objRow['Email'])
print("------------------------------------------------------------")
objFile.close()

print()
print("This is the summary.")
print()
```

Listing 4 Reading a file with wrong file name that doesn't exist

```
This portion comes from calling a function.
------------------------------------------------------------
1|Mary Johnson|MJohnsohn@msn.com
2|Susan Jones|SJones@hotmail.com
------------------------------------------------------------

Traceback (most recent call last):
  File "/Users/charles.lin/PycharmProjects/Assignment07/Assignment07.py", line 74, in <module>
    objFile = open("CustomerDat.txt", "r")
FileNotFoundError: [Errno 2] No such file or directory: 'CustomerDat.txt'

Process finished with exit code 1
```

Figure 4 Error message with no try/except

If I include a try/except, I can customize the error message and the job can continue.

```python
# Try/Except Example
# ----------------------------------------------------------------------
filePath = "CustomerDat.txt"
try:
    objFile = open(filePath, "r")
except FileNotFoundError as e:
    print("Built-In Python error Info: ")
    print(e)
    print("No data would reutrn.")
    print()
print("Try Except Example of reading a text file:")
print("-------------------------------------------------------------")
for objRow in lstTable:
    print(str(objRow["ID"]) + '|' + objRow["Name"] + '|' + objRow['Email'])
print("-------------------------------------------------------------")
objFile.close()

print()
print("This is the next section.")
print()
```

Listing 5 With try/except with built-in FileNotFoundError

```
Built-In Python error Info:
[Errno 2] No such file or directory: 'CustomerDat.txt'
No data would reutrn.

Try Except Example of reading a text file:
-------------------------------------------------------------
-------------------------------------------------------------

This is the next section.


Process finished with exit code 0
```

Figure 5 Result from Listing 5 that shows the error and the job continues to the next step

I intentionally set variable filePath to be “CustomerDat.txt” to illustrate the exception error. I customized the error to show Python’s Built-in exception and to communicate nothing will show.

```python
# Declare variables
binary_file_name = "CustomerData.dat"
text_file_name = "CustomerData.txt"
filePath = "CustomerDat.txt"
lstTable=[]
list_of_rows=[]
strChoice = ""
```

Listing 6 Variable “filePath” that points to a file that doesn’t exist so cannot be read from

```
Try Except Example when the file it tries to read is not available.
Change 'filePath' Variable to 'CustomerData.txt' to proceed.
-------------------------------------------------------------
The Built-In Python error Info shows -
[Errno 2] No such file or directory: 'CustomerDat.txt'
No data would return.
```

Figure 6 Built-in exception object and customized message when “filePath” from Listing 6 cannot be found

The output in Terminal when the file cannot be found (variable filePath = “CustomerDat.txt”)

```
Assignment07 — -zsh — 80×28
(base) √ ~/PycharmProjects/Assignment07 ➜ python Assignment07_CharlesLin.py
Binary file has been written.
------------------------------------------------------------
1|Mary Johnson|MJohnsohn@msn.com
2|Susan Jones|SJones@hotmail.com
------------------------------------------------------------

The dictionary is also written to a text file to observe sizes.
------------------------------------------------------------

Try Except Example when the file it tries to read is not available.
Change 'filePath' Variable to 'CustomerData.txt' to proceed.
------------------------------------------------------------
The Built-In Python error Info shows -
[Errno 2] No such file or directory: 'CustomerDat.txt'
No data would return.

Read the result from the function to write records below to a binary or text fil
e decideed by user:
------------------------------------------------------------
The file it tries to read is not available.
[Errno 2] No such file or directory: 'CustomerDat.txt'
<class 'FileNotFoundError'>

Cannot read any data to write from.
Press Enter to Exit
(base) √ ~/PycharmProjects/Assignment07 ➜
```

Listing 7 When file is not found, message shows built-in exception and communicates the file is not available

When variable filePath is set to “CustomerData.txt”, then the data can be read, and this is the output in Terminal when the user chooses to write to a binary file. The user can also choose to write to a text file.

```
Assignment07 — -zsh — 79×31
(base) √ ~/PycharmProjects/Assignment07 ➜ python Assignment07_CharlesLin.py
Binary file has been written.
-------------------------------------------------------------
1|Mary Johnson|MJohnsohn@msn.com
2|Susan Jones|SJones@hotmail.com
-------------------------------------------------------------

The dictionary is also written to a text file to observe sizes.
-------------------------------------------------------------

Try Except Example of reading a text file, and the file is available:
-------------------------------------------------------------
1|Mary Johnson|MJohnsohn@msn.com
2|Susan Jones|SJones@hotmail.com
-------------------------------------------------------------

Read the result from the function to write records below to a binary or text fi
le decideed by user:
-------------------------------------------------------------
This is the list of customers
-------------------------------------------------------------
1|Mary Johnson|MJohnsohn@msn.com
2|Susan Jones|SJones@hotmail.com
-------------------------------------------------------------
Choose [b] to write to a binary file or [t] to write to a text file, or [Exit]
to finish: b
Data written to a binary file.

Press Enter to Exit
(base) √ ~/PycharmProjects/Assignment07 ➜                charles.lin@SNS004807
```

Listing 8 When the file is available, it shows the records that will be written and tells the user the it is writing to a binary file.

The selected file type then is written in the working folder.

```
CustomerDataFromFunction.dat
Áïu]î(}î(åIDîå1îåNameîå
Mary JohnsonîåEmailîåMJohnsohn@msn.comîu}î(hå2îhåSusan JonesîhåSJones@hotmail.comîue.
```

Figure 8 Binary file written by Listing 8

Figure 9 Text file written by Listing 8

Summary

When the assignment request is very high-level, I need to think of a code design that is feasible with what I know, and fulfills the requirements. I have a good sense of error handling (try/except), and I'm becoming familiar with publishing GitHub's page.