Matthew Peetz

Regis University

MSDS 621 Data Wrangling

Week 1 Lab: Converting File Between Different Formats

Assignment Requirements

Assignment Part 1

In this week's Lecture notebook, we examined CSV and JSON file formats. We wrote code to manually convert a specific CSV file to a specific JSON in the process.

```
import json
import csv

# Read CSV file. This wouldn't work well for very large files
with open('data_wk1/scientists.csv') as f:
    reader = csv.DictReader(f)
    rows = list(reader)

# Write JSON file to disk
with open('data_wk1/scientists.json', 'w') as f:
    json.dump(rows, f)
```

The comment above the CSV section makes an assumption and says it wouldn't work for large files.
(The fourth line above reads "# Read CSV file. This wouldn't work well for very large files")

Use the following articles to understand the terms **eager evaluation** and **lazy evaluation**:

- https://en.wikipedia.org/wiki/Eager_evaluation
- https://en.wikipedia.org/wiki/Lazy_evaluation

Now answer the following questions:

1. In light of the two definitions above, what is the assumption made by that comment?
2. Explain why that assumption is right or wrong.
3. Is it safe to use the code above on large files?

Eager Evalution - In Eager Evaluation a functions entire argumnent is evaluated before it is applied.

Lazy Evalulation - In Lazy Evalulation a functions argument is not evaluated until it needs to be applied. This leads to faster computing and does not require parts of the function that are not needed to be called.

1) The assumption made by the comment is that for a large file and Eager Evalution the agrument would need to be called for the entire file, which is very large, and would require computing time that scales with the size of the file. In addition each row is being read into a list and then processed, which takes more tie.

2) According to the article, "Working With Large CSV Files in Python", it can take a lot of memory to process large csv files. That is becuase the file needs to be held in the machines memory, which may not be large enough.

3) It would not be ideal to run the above code on very large files. There are multiple alternatives that can be used and were listed in the article, including processing the file in chunks or using an alternative library like dask.

https://www.geeksforgeeks.org/working-with-large-csv-files-in-python/

Assignment Part 2

Programmers hate code written for a specific case, "I don't care if it can solve one special case, I want it to solve *all* cases." This generalization process is called **"abstraction"**.

1. Generalize the CSV->JSON code above into a function that can work for any CSV file and any JSON file (within reason).

In [1]:
```
# Code to open the csv file and convert it to a json dile
import csv
import json

def con_csv_json(csv_file, json_file):
    with open(csv_file, 'r',  encoding="utf8") as f:
        reader = csv.DictReader(f)
        rows = list(reader)

    with open(json_file, 'w') as f:
        json.dump(rows, f)

con_csv_json('data/scientists.csv', 'data/newsci.json')
```

Showing that the JSON file was indeed created with the new name

In [2]:
```
# Code taken from week 1 lesson
def printplus(obj):
    """
    Pretty-prints the object passed in.

    """
    # Dict
    if isinstance(obj, dict):
        for k, v in sorted(obj.items()):
            print (u'{0}: {1}'.format(k, v))

    # List or tuple
    elif isinstance(obj, list) or isinstance(obj, tuple):
        for x in obj:
            print (x)

    # Other
    else:
        print (obj)

# showing that the json file was made
input_filename = 'data/newsci.json'

with open(input_filename) as f:
    data_2 = json.load(f)
printplus(data_2)
```

```
{'Name': 'Rosaline Franklin', 'Born': '1920-07-25', 'Died': '1958-04-16', 'Age': '3
7', 'Occupation': 'Chemist'}
{'Name': 'William Gosset', 'Born': '1876-06-13', 'Died': '1937-10-16', 'Age': '61',
'Occupation': 'Statistician'}
{'Name': 'Florence Nightingale', 'Born': '1820-05-12', 'Died': '1910-08-13', 'Age':
'90', 'Occupation': 'Nurse'}
{'Name': 'Marie Curie', 'Born': '1867-11-07', 'Died': '1934-07-04', 'Age': '66', 'O
ccupation': 'Chemist'}
{'Name': 'Rachel Carson', 'Born': '1907-05-27', 'Died': '1964-04-14', 'Age': '56',
'Occupation': 'Biologist'}
{'Name': 'John Snow', 'Born': '1813-03-15', 'Died': '1858-06-16', 'Age': '45', 'Occ
upation': 'Physician'}
{'Name': 'Alan Turing', 'Born': '1912-06-23', 'Died': '1954-06-07', 'Age': '41', 'O
ccupation': 'Computer Scientist'}
{'Name': 'Johann Gauss', 'Born': '1777-04-30', 'Died': '1855-02-23', 'Age': '77', '
Occupation': 'Mathematician'}
```

Being able to convert in only one direction is only helpful half the time. Specious math aside,

1. Write a generalized JSON->CSV converter function.

In [3]:
```
# JSON -> CSV Converter

import pandas as pd # loading in the pandas library

def con_json_csv(json_file, csv_file):
    df = pd.read_json(json_file,  encoding="utf8")
    df.to_csv(csv_file)

con_json_csv('data/scientists.json', 'data/newsci.csv')
```

Showing that the file was converted from JSON to CSV

In [4]:
```
# Loading pprint
from pprint import pprint
```

In [5]:
```
# showing that the json file was made
with open('data/newsci.csv', 'r') as infile:
    reader = csv.DictReader(infile)
    for line in reader:
        pprint(dict(line))
```

```
{'': '0',
 'Age': '37',
 'Born': '1920-07-25',
 'Died': '1958-04-16',
 'Name': 'Rosaline Franklin',
 'Occupation': 'Chemist'}
{'': '1',
 'Age': '61',
 'Born': '1876-06-13',
 'Died': '1937-10-16',
 'Name': 'William Gosset',
 'Occupation': 'Statistician'}
{'': '2',
 'Age': '90',
 'Born': '1820-05-12',
 'Died': '1910-08-13',
 'Name': 'Florence Nightingale',
 'Occupation': 'Nurse'}
{'': '3',
 'Age': '66',
 'Born': '1867-11-07',
 'Died': '1934-07-04',
 'Name': 'Marie Curie',
 'Occupation': 'Chemist'}
{'': '4',
 'Age': '56',
 'Born': '1907-05-27',
 'Died': '1964-04-14',
 'Name': 'Rachel Carson',
 'Occupation': 'Biologist'}
{'': '5',
 'Age': '45',
 'Born': '1813-03-15',
 'Died': '1858-06-16',
 'Name': 'John Snow',
 'Occupation': 'Physician'}
{'': '6',
 'Age': '41',
 'Born': '1912-06-23',
 'Died': '1954-06-07',
 'Name': 'Alan Turing',
 'Occupation': 'Computer Scientist'}
{'': '7',
 'Age': '77',
 'Born': '1777-04-30',
 'Died': '1855-02-23',
 'Name': 'Johann Gauss',
 'Occupation': 'Mathematician'}
```


1. Use the two function you just created, demonstrate a "round-trip" (CSV->JSON->CSV or JSON->CSV->JSON)for the following dataset.

> **Dataset:** Consumer Complaint Database data found at https://catalog.data.gov/dataset/consumer-complaint-database#topic=consumer_navigation

Round trip conversion JSON -> CSV -> JSON

In [6]:
```
# converting from json to CSV
con_json_csv('data/complaints.json', 'data/newcomplaints.csv')
```

In [7]:
```
# converting from csv to json
con_csv_json('data/newcomplaints.csv', 'data/finalcomplaints.json')
```

1. Now compare the original file and the round-trip files. These should be reasonably identical.

> What do I mean by reasonably identical? You might run into some instances where the various packages encapsulate text in quotes. Minor issues like this are okay, but you need to explain why they are specifically occurring, if you decide not to clean them up.
>
> If you wish to resolve this issue, You could:
> *Look at the various parameters to resolve this issue*
> Run the round trip twice. Then you would be make your output file from the first run your input file for the second run. Not ideal, but it works.

In [8]:
```
# reading the two files in as uni-code and comparing them
file_1 = (pd.read_json('data/complaints.json', encoding="utf8" ))
```

In [9]:
```
file_2 = (pd.read_json('data/finalcomplaints.json', encoding="utf8" ))
```

In [10]:
```
# Looking at the original file
file_1.head(5)
```

Out[10]:

	date_received	product	sub_product	issue	sub_issue	complaint_what_happened	com
0	2023-05-13	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else	This is my Follow-up request that I have been ...	Com t
1	2023-06-15	Credit reporting, credit repair services, or o...	Credit reporting	Problem with a credit reporting company's inve...	Their investigation did not fix an error on yo...		
2	2023-03-02	Payday loan, title loan, or personal loan	Personal line of credit	Problem when making payments			Com t
3	2023-03-01	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else		Com t
4	2023-05-12	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else		Com t

In [11]:
```
# Looking at the new file
file_2.head(5)
```

Out[11]:

		date_received	product	sub_product	issue	sub_issue	complaint_what_happened	c
0	0	2023-05-13	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else	This is my Follow-up request that I have been ...	C
1	1	2023-06-15	Credit reporting, credit repair services, or o...	Credit reporting	Problem with a credit reporting company's inve...	Their investigation did not fix an error on yo...		
2	2	2023-03-02	Payday loan, title loan, or personal loan	Personal line of credit	Problem when making payments			C
3	3	2023-03-01	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else		C
4	4	2023-05-12	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else		C

Hint: Mac and Linux have a command line tool called 'diff' that will show differences between two files. Windows users can use the 'fc' command on the command line. See this answer on StackOverflow for alternatives: https://stackoverflow.com/questions/6877238/what-is-the-windows-equivalent-of-the-diff-command

In [12]:
```
file_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3774954 entries, 0 to 3774953
Data columns (total 18 columns):
 #   Column                     Dtype
---  ------                     -----
 0   date_received              object
 1   product                    object
 2   sub_product                object
 3   issue                      object
 4   sub_issue                  object
 5   complaint_what_happened    object
 6   company_public_response    object
 7   company                    object
 8   state                      object
 9   zip_code                   object
 10  tags                       object
 11  consumer_consent_provided  object
 12  submitted_via              object
 13  date_sent_to_company       object
 14  company_response           object
 15  timely                     object
 16  consumer_disputed          object
 17  complaint_id               int64
dtypes: int64(1), object(17)
memory usage: 518.4+ MB
```

In [13]:
```
file_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3774954 entries, 0 to 3774953
Data columns (total 19 columns):
 #   Column                     Dtype
---  ------                     -----
 0                              int64
 1   date_received              object
 2   product                    object
 3   sub_product                object
 4   issue                      object
 5   sub_issue                  object
 6   complaint_what_happened    object
 7   company_public_response    object
 8   company                    object
 9   state                      object
 10  zip_code                   object
 11  tags                       object
 12  consumer_consent_provided  object
 13  submitted_via              object
 14  date_sent_to_company       object
 15  company_response           object
 16  timely                     object
 17  consumer_disputed          object
 18  complaint_id               int64
dtypes: int64(2), object(17)
memory usage: 547.2+ MB
```

There were some changes made to the file during the round trip conversion, namely a new column was added that appears to be a duplication of the index column. It was likely done when the CSV file was created. One way around this could be to tell the CSV reader to not create an index column when it is reading in the file.

It also helped that the file was read in as uni-code. I noticed that there were issues when I did not do this the first time caused by certain characters not being read correctly.

The file also changed in size (due to the column addition). With that in mind it appears that the data itself was preserved correctly.

Conclusion

Going back to the initial question I found a library on GeeksforGeeks that allows for easy timing of a code block, this is what it would look like to read in the rather large csv file that was created from the json file. It takes about 90 seconds. The code for this is used below. That is compared with 269 seconds to read it in using the JSON code that was provided. So in retrospect it was faster to read it in using the original code.

https://www.geeksforgeeks.org/how-to-check-the-execution-time-of-python-script/

In [14]:
```
# This is the code that states it will take a long time with a large file
# Read CSV file. This wouldn't work well for very large files
import time

start = time.time()

with open('data/newcomplaints.csv', encoding="utf8") as f:
    reader = csv.DictReader(f)
    rows = list(reader)

end = time.time()

print(end-start)
```

```
82.15476274490356
```

In [15]:
```
# Same but with a JSON file to compare
# Write JSON file to disk
start = time.time()

with open('data/complaints.json', 'w') as f:
    json.dump(rows, f)

end = time.time()

print(end-start)
```

```
279.3206205368042
```

After building the code to convert from JSON to CSV and CSV to JSON it was found that issues would arise with the data if the "encoding='utf8'" was not used. The characters were being changed and the program was unable to contiue running with the unidentifiable characters. Reading everything in as UTF8 encoding fixed this issue. The conversion is very very slow, and takes several minutes to run on my machine. It was also found that an additional column was being added when converting between formats. This column could easily be dropped once the final data frame was created.