Lab Assignment 3: How to Load, Convert, and Write JSON Files in Python

DS 6001: Practice and Application of Data Science

Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

Problem 0

Import the following libraries:

In [28]:
```
import numpy as np
import pandas as pd
import requests
import json
import os
import io
import sys
sys.tracebacklimit = 0 # turn off the error tracebacks
```

Problem 1

JSON and CSV are both text-based formats for the storage of data. It's possible to open either one in a plain text editor. Given this similarity, why does a CSV file usually take less memory than a JSON formatted file for the same data? Under what conditions could a JSON file be smaller in memory than a CSV file for the same data? (2 points)

Problem 1 answer

Because CSV files contain only data and delimiters, only the essential characters are present to take up memory. JSON files require organizational hierarchy represented by characters like {} braces. The combined additional characters might explain why CSV files are typically smaller in memory than JSON files.

A JSON file may, in fact, be smaller in memory for a CSV file for the same data if the data has frequent null values. Because JSON represents only present values, the memory required for the JSON file may be smaller than the memory required for a CSV file of the same data. This is because the CSV file relies on placeholder characters for the null values to ensure they remain represented as null.

In [29]:
```
##################################
```

Problem 2

NASA has a dataset of all meteorites that have fallen to Earth between the years A.D. 860 and 2013. The data contain the name of each meteorite, along with the coordinates of the place where the meteorite hit, the mass of the meteorite, and the date of the collison. The data is stored as a JSON here: https://data.nasa.gov/resource/y77d-th95.json

Look at the data in your web-browser and explain which strategy for loading the JSON into Python makes the most sense and why.

Then write and run the code that will work for loading the data into Python. (2 points)

Problem 2 answer

The strategy for loading this JSON into Python that makes the most sense is the most simple.

Using requests.get() to perform the http call, loading the text of the results of the call into an object, and using pd.json_normalize to transform the JSON into a pandas dataframe.

This is the most sensible because this JSON file is not complex, in that the data we are interested in is not nested beyond the first layer of the JSON file.

In [30]:
```
url = 'https://data.nasa.gov/resource/y77d-th95.json'
r = requests.get(url)
r
```

Out[30]:
```
<Response [200]>
```

In [31]:
```
nasa_json = json.loads(r.text)
nasa_df = pd.json_normalize(nasa_json)
nasa_df
```

Out[31]:

	name	id	nametype	recclass	mass	fall	year	reclat	reclong	geolocation.type
0	Aachen	1	Valid	L5	21	Fell	1880-01-01T00:00:00.000	50.775000	6.083330	Point
1	Aarhus	2	Valid	H6	720	Fell	1951-01-01T00:00:00.000	56.183330	10.233330	Point
2	Abee	6	Valid	EH4	107000	Fell	1952-01-01T00:00:00.000	54.216670	-113.000000	Point
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	1976-01-01T00:00:00.000	16.883330	-99.900000	Point
4	Achiras	370	Valid	L6	780	Fell	1902-01-01T00:00:00.000	-33.166670	-64.950000	Point
...	...	...	...	...	...	...	...	...	...	...
995	Tirupati	24009	Valid	H6	230	Fell	1934-01-01T00:00:00.000	13.633330	79.416670	Point
996	Tissint	54823	Valid	Martian (shergottite)	7000	Fell	2011-01-01T00:00:00.000	29.481950	-7.611230	Point
997	Tjabe	24011	Valid	H6	20000	Fell	1869-01-01T00:00:00.000	-7.083330	111.533330	Point
998	Tjerebon	24012	Valid	L5	16500	Fell	1922-01-01T00:00:00.000	-6.666670	106.583330	Point
999	Tomakovka	24019	Valid	LL6	600	Fell	1905-01-01T00:00:00.000	47.850000	34.766670	Point

1000 rows × 13 columns

Problem 3

The textbook chapter for this module shows, as an example, how to pull data in JSON format from Reddit's top 25 posts on /r/popular. The steps outlined there pull all of the features in the data into the dataframe, resulting in a dataframe with 172 columns.

If we only wanted a few features, then looping across elements of the JSON list itself and extracting only the data we want may be a more efficient approach.

Use looping - and not `pd.read_json()` or `pd.json_normalize()` - to create a dataframe with 25 rows (one for each of the top 25 posts), and only columns for `subreddit`, `title`, `ups`, and `created_utc`. The JSON file exists at

http://www.reddit.com/r/popular/top.json, and don't forget to specify `headers = {'User-agent': 'DS6001'}` within `requests.get()`. (3 points)

In [32]:
```
url3 = 'https://www.reddit.com/r/popular/top.json'
r3 = requests.get(url3, headers = {'User-agent': 'DS6001'})
reddit_json = json.loads(r3.text)
type(reddit_json)
```

Out[32]: dict

In [33]:
```
reddit_df = pd.DataFrame(
    [b['data']['subreddit'], b['data']['title'],
     b['data']['ups'], b['data']['created_utc']] for b in reddit_json['data']['children'])

reddit_df.columns = ['subreddit', 'title','ups','created_utc']
reddit_df
```

Out[33]:

	subreddit	title	ups	created_utc
0	interestingasfuck	The Chinese Tianlong-3 Rocket Accidentally Lau...	55385	1.719747e+09
1	Damnthatsinteresting	Mosquito coil holder made using a 3D printing ...	55055	1.719775e+09
2	mildlyinfuriating	To the guy who is mildly infuriated by their n...	50322	1.719724e+09
3	MadeMeSmile	The hug.... wow	44352	1.719737e+09
4	meirl	Meirl	42315	1.719750e+09
5	interestingasfuck	This 9 year old girl dodges and manages to esc...	42254	1.719749e+09
6	clevercomebacks	Absolutely unreal 🙃👇	36977	1.719717e+09
7	mildlyinfuriating	There are no assigned parking spaces at my par...	35416	1.719748e+09
8	mildlyinteresting	My city has recently been filling small pothol...	35813	1.719754e+09
9	MadeMeSmile	Now that's a good life	34548	1.719752e+09
10	Steam	"Reality is often disappointing"	33555	1.719742e+09
11	BritishSuccess	Taylor Swift has donated enough money to cover...	32840	1.719741e+09
12	facepalm	What was she thinking	33454	1.719756e+09
13	inthenews	Trump's jet sits next to Russian government pl...	30824	1.719747e+09
14	pics	Ex governor of New Jersey Chris Christie, pour...	30179	1.719731e+09
15	facepalm	How can humanity disappoint so much	30110	1.719742e+09
16	interestingasfuck	Ukraine handed over all their nuclear weapons ...	29039	1.719742e+09
17	facepalm	I would just collapse if I was the boss	29209	1.719746e+09
18	meirl	Meirl	27491	1.719729e+09
19	nextfuckinglevel	Predator visits the office in Japan	26847	1.719743e+09
20	Unexpected	Brutal move. Is this legal?	27484	1.719774e+09
21	todayilearned	TIL Stephen Hawking completed a final multiver...	26503	1.719755e+09
22	TheBoys	Its layers to this	26306	1.719718e+09
23	MadeMeSmile	She doesn't know a sign was put up, she just k...	26353	1.719718e+09
24	BlackPeopleTwitter	So vote between a guy with a cold or a lying f...	25902	1.719758e+09

Problem 4

The NBA has saved data on all 30 teams' shooting statistics for the 2014-2015 season here: https://stats.nba.com/js/data/sportvu/2015/shootingTeamData.json. Take a moment and look at this JSON file in your web browser. The structure of this particular JSON is complicated, but see if you can find the team-by-team data. In this problem

our goal is to use `pd.json_normalize()` to get the data into a dataframe. The following questions will guide you towards this goal.

Prob 4 Part a

Download the raw text of the NBA JSON file and register it as JSON formatted data in Python's memory. (2 points)

In [34]:
```
with open('shootingTeamData.txt', 'r') as s:
    nba_txt = s.read()
nba_json = json.loads(nba_txt)
```

Prob 4 Part b

Describe, in words, the path that leads to the team-by-team data. (2 points)

Inside the json file, just beyond the initial metadata is a list called resultSets.

That list contains two things:

- a list of strings representing headers for the row data,
- the team-by-team data, which is stored as a list of lists.

Prob 4 Part c

Use the `pd.json_normalize()` function to pull the team-by-team data into a dataframe. This is going to be tricky. You will need to use indexing on the JSON data as well as the `record_path` parameter.

If you are successful, you will have a dataframe with 30 rows and 33 columns. The first row will refer to the Golden State Warriors, the second row will refer to the San Antonio Spurs, and the third row will refer to the Cleveland Cavaliers. The columns will only be named 0, 1, 2, ... at this point. (4 points)

In [35]:
```
nba_df = pd.json_normalize(
    nba_json, record_path = ['resultSets', 'rowSet'])
nba_df
```

Out[35]:

	0	1	2	3	4	5	6	7	8	9	...	23	24	25	26	27	28	
0	1610612744	Golden State	Warriors	GSW		82	48.7	114.9	14.9	0.498	...	0.478	21.2	42.5	0.497	2.3	6.3	0
1	1610612759	San Antonio	Spurs	SAS		82	48.3	103.5	14.8	0.481	...	0.506	18.3	39.8	0.460	0.9	2.6	0
2	1610612739	Cleveland	Cavaliers	CLE		82	48.7	104.3	16.9	0.481	...	0.473	18.2	40.7	0.447	1.7	5.7	0
3	1610612746	Los Angeles	Clippers	LAC		82	48.6	104.5	15.0	0.497	...	0.480	18.9	42.0	0.450	2.0	6.0	0
4	1610612760	Oklahoma City	Thunder	OKC		82	48.6	110.2	16.1	0.480	...	0.497	17.5	38.7	0.451	1.6	5.1	0
5	1610612737	Atlanta	Hawks	ATL		82	48.6	102.8	19.0	0.463	...	0.483	19.4	44.6	0.435	1.0	3.1	0
6	1610612745	Houston	Rockets	HOU		82	48.6	106.5	17.2	0.433	...	0.472	15.5	36.4	0.426	2.3	7.4	0
7	1610612757	Portland	Trail Blazers	POR		82	48.5	105.1	17.5	0.441	...	0.447	18.0	39.8	0.453	1.7	5.9	0
8	1610612758	Sacramento	Kings	SAC		81	48.4	106.7	18.7	0.452	...	0.473	18.1	39.7	0.454	0.9	3.1	0
9	1610612764	Washington	Wizards	WAS		82	48.5	104.1	15.4	0.480	...	0.483	19.5	44.3	0.439	0.7	2.7	0
10	1610612748	Miami	Heat	MIA		82	48.6	100.0	17.9	0.488	...	0.490	15.7	35.2	0.445	0.8	2.9	0
11	1610612761	Toronto	Raptors	TOR		81	48.5	102.7	23.0	0.462	...	0.461	14.1	32.4	0.436	1.8	5.6	0
12	1610612742	Dallas	Mavericks	DAL		82	49.0	102.3	18.2	0.473	...	0.464	17.5	41.4	0.423	1.4	5.3	0
13	1610612766	Charlotte	Hornets	CHA		82	48.6	103.4	16.8	0.459	...	0.449	17.0	39.8	0.427	1.8	6.0	0
14	1610612762	Utah	Jazz	UTA		82	49.0	97.7	18.1	0.445	...	0.468	15.9	37.2	0.426	1.4	4.3	0
15	1610612753	Orlando	Magic	ORL		81	48.7	102.0	18.0	0.456	...	0.475	18.5	42.6	0.435	0.7	2.7	0
16	1610612749	Milwaukee	Bucks	MIL		82	48.7	99.0	17.4	0.463	...	0.477	13.2	29.4	0.448	1.1	4.0	0
17	1610612740	New Orleans	Pelicans	NOP		82	48.5	102.7	19.9	0.458	...	0.460	17.9	41.1	0.434	0.6	2.6	0
18	1610612750	Minnesota	Timberwolves	MIN		82	48.6	102.4	15.1	0.464	...	0.471	16.1	35.4	0.455	0.7	2.6	0
19	1610612754	Indiana	Pacers	IND		82	48.8	102.2	13.7	0.453	...	0.465	16.4	38.1	0.431	1.7	5.7	0
20	1610612751	Brooklyn	Nets	BKN		82	48.4	98.6	14.4	0.457	...	0.464	15.8	36.1	0.438	1.0	3.3	0
21	1610612765	Detroit	Pistons	DET		82	48.7	102.0	17.5	0.464	...	0.452	15.7	37.2	0.422	0.9	4.0	0
22	1610612743	Denver	Nuggets	DEN		82	48.6	101.9	15.9	0.406	...	0.448	16.4	37.8	0.434	1.1	4.3	0
23	1610612738	Boston	Celtics	BOS		81	48.5	105.6	18.9	0.453	...	0.451	16.9	39.9	0.424	1.6	5.7	0
24	1610612741	Chicago	Bulls	CHI		82	48.9	101.6	18.1	0.458	...	0.442	17.0	38.5	0.441	1.3	3.9	0
25	1610612755	Philadelphia	76ers	PHI		82	48.6	97.4	19.7	0.445	...	0.449	15.3	37.4	0.409	1.6	5.7	0
26	1610612756	Phoenix	Suns	PHX		82	48.4	100.9	15.6	0.440	...	0.447	16.6	39.5	0.421	1.4	5.0	0
27	1610612752	New York	Knicks	NYK		82	48.5	98.4	10.4	0.447	...	0.439	15.9	36.4	0.438	1.5	4.9	0
28	1610612763	Memphis	Grizzlies	MEM		82	48.6	99.1	16.4	0.440	...	0.459	16.1	38.5	0.418	0.7	2.5	0
29	1610612747	Los Angeles	Lakers	LAL		82	48.3	97.3	15.6	0.441	...	0.420	14.0	34.5	0.406	2.2	7.9	0

30 rows × 33 columns

Prob 4 Part d

Find the path that leads to the headers (the column names), and extract these names as a list. Then set the `.columns` attribute of the dataframe you created in part c equal to this list. The result should be that the dataframe now has the correct column names. (3 points)

In [36]:
```
headers_df = pd.json_normalize(
    nba_json, record_path=['resultSets', 'headers'])
```

```
headers_df
```

Out[36]:

	0
0	TEAM_ID
1	TEAM_CITY
2	TEAM_NAME
3	TEAM_ABBREVIATION
4	TEAM_CODE
5	GP
6	MIN
7	PTS
8	PTS_DRIVE
9	FGP_DRIVE
10	PTS_CLOSE
11	FGP_CLOSE
12	PTS_CATCH_SHOOT
13	FGP_CATCH_SHOOT
14	PTS_PULL_UP
15	FGP_PULL_UP
16	FGA_DRIVE
17	FGA_CLOSE
18	FGA_CATCH_SHOOT
19	FGA_PULL_UP
20	EFG_PCT
21	CFGM
22	CFGA
23	CFGP
24	UFGM
25	UFGA
26	UFGP
27	CFG3M
28	CFG3A
29	CFG3P
30	UFG3M
31	UFG3A
32	UFG3P

In [37]:

```
nba_df.columns = headers_df
nba_df
```

Out[37]:

	(TEAM_ID,)	(TEAM_CITY,)	(TEAM_NAME,)	(TEAM_ABBREVIATION,)	(TEAM_CODE,)	(GP,)	(MIN,)	(PTS,)	(PTS_DRIV
0	1610612744	Golden State	Warriors	GSW		82	48.7	114.9	1
1	1610612759	San Antonio	Spurs	SAS		82	48.3	103.5	1
2	1610612739	Cleveland	Cavaliers	CLE		82	48.7	104.3	1
3	1610612746	Los Angeles	Clippers	LAC		82	48.6	104.5	1
4	1610612760	Oklahoma City	Thunder	OKC		82	48.6	110.2	1
5	1610612737	Atlanta	Hawks	ATL		82	48.6	102.8	1
6	1610612745	Houston	Rockets	HOU		82	48.6	106.5	1
7	1610612757	Portland	Trail Blazers	POR		82	48.5	105.1	1
8	1610612758	Sacramento	Kings	SAC		81	48.4	106.7	1
9	1610612764	Washington	Wizards	WAS		82	48.5	104.1	1
10	1610612748	Miami	Heat	MIA		82	48.6	100.0	1
11	1610612761	Toronto	Raptors	TOR		81	48.5	102.7	2
12	1610612742	Dallas	Mavericks	DAL		82	49.0	102.3	1
13	1610612766	Charlotte	Hornets	CHA		82	48.6	103.4	1
14	1610612762	Utah	Jazz	UTA		82	49.0	97.7	1
15	1610612753	Orlando	Magic	ORL		81	48.7	102.0	1
16	1610612749	Milwaukee	Bucks	MIL		82	48.7	99.0	1
17	1610612740	New Orleans	Pelicans	NOP		82	48.5	102.7	1
18	1610612750	Minnesota	Timberwolves	MIN		82	48.6	102.4	1
19	1610612754	Indiana	Pacers	IND		82	48.8	102.2	1
20	1610612751	Brooklyn	Nets	BKN		82	48.4	98.6	1
21	1610612765	Detroit	Pistons	DET		82	48.7	102.0	1
22	1610612743	Denver	Nuggets	DEN		82	48.6	101.9	1
23	1610612738	Boston	Celtics	BOS		81	48.5	105.6	1
24	1610612741	Chicago	Bulls	CHI		82	48.9	101.6	1
25	1610612755	Philadelphia	76ers	PHI		82	48.6	97.4	1
26	1610612756	Phoenix	Suns	PHX		82	48.4	100.9	1
27	1610612752	New York	Knicks	NYK		82	48.5	98.4	1
28	1610612763	Memphis	Grizzlies	MEM		82	48.6	99.1	1
29	1610612747	Los Angeles	Lakers	LAL		82	48.3	97.3	1

30 rows × 33 columns

Problem 5

Save the NBA dataframe you extracted in problem 4 as a JSON-formatted text file on your local machine. Format the JSON so that it is organized as dictionary with three lists: `columns` lists the column names, `index` lists the row names, and `data` is a list-of-lists of data points, one list for each row. (Hint: this is possible with one line of code) (2 points)

In [38]:
```
nba_5json = nba_df.to_json(orient="split")
```

In [39]:
```
os.chdir("/Users/kianadane/Documents/GitHub/ds6001/Mod_3")
```

In [40]:
```
final_file = open('lab3_prob5.txt', 'w')
final_file.write(str(nba_5json))
final_file.close()
```

In [ ]:
```

```