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 [1]:
```
import numpy as np
import pandas as pd
import requests
import json
import sys
sys.tracebacklimit = 0 # turn off the error tracebacks
```

In [2]:
```
headers = {"User-Agent": "DS6001", "From": "mdg7wj@virginia.edu"}
```

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)

Due to the structure of JSON files as dictionaries, the column headers must be associated with every data value individually (as a key), which means there is much more redundant memory usage than a CSV file, which only contains the column headers once. However, JSON files are generally more efficient with text, so a dataset that contains only one line of data and several lines of metadata comments would theoretically be smaller as a JSON than as a CSV.

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.

Because it is a web URL, requests.get() would work best for initially loading the data into Python. Then we will use json.loads() to actually convert the received text into a JSON object.

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

In [5]:
```
url = "https://data.nasa.gov/resource/y77d-th95.json"
nasa = requests.get(url, headers = headers)
nasa_json = json.loads(nasa.text)
```

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 [6]:
```
url = "http://www.reddit.com/r/popular/top.json"
reddit = requests.get(url, headers = headers)
reddit_json = json.loads(reddit.text)
reddit_df = pd.DataFrame(
    [u["data"]["subreddit"], u["data"]["title"], u["data"]["ups"], u["data"]
    for u in reddit_json["data"]["children"])
reddit_df.columns = ["subreddit", "title", "ups", "created_utc"]
reddit_df
```

Out[6]:

	subreddit	title	ups	created_utc
0	clevercomebacks	Zapped by Zappa. Good show!	49661	1.719264e+09
1	meirl	Meirl	49706	1.719279e+09
2	facepalm	RIP Sean Spicer\n	44665	1.719262e+09
3	facepalm	This is gold medal at the Olympics levels of a...	45115	1.719298e+09
4	Unexpected	When you see your gym crush	42505	1.719288e+09
5	meme	That's a series I would like to watch	38568	1.719315e+09
6	facepalm	They truly have no idea what they support.	35577	1.719271e+09
7	interestingasfuck	Stop.! Prevent Your Death' Sign At Florida Und...	35627	1.719275e+09
8	MadeMeSmile	All 15 of them surprised their grandparents wi...	38585	1.719325e+09
9	BeAmazed	Michael Jackson's voice with No background noi...	35329	1.719270e+09
10	nextfuckinglevel	Man runs into burning home to save his dog	35599	1.719320e+09
11	MadeMeSmile	She deserves a tip	33452	1.719279e+09
12	interestingasfuck	Saaya and Cleopatra have been courting since 4...	33171	1.719315e+09
13	Damnthatsinteresting	Franklin D. Roosevelt sent a list of countries...	32347	1.719311e+09
14	me_irl	me_irl	34037	1.719320e+09
15	Steam	i feel so stupid	29141	1.719298e+09
16	meirl	Meirl	32407	1.719317e+09
17	MurderedByWords	Comparing sex to assignments	27790	1.719295e+09
18	facepalm	What the fuck is he on about	26132	1.719267e+09
19	interestingasfuck	Tree Sprays Water After Having Branch Removed	25039	1.719322e+09
20	pics	Barack Obama’s Father’s Day post on insta	24114	1.719265e+09
21	MadeMeSmile	Truly a beautiful human inside and out...we ca...	30288	1.719330e+09
22	Damnthatsinteresting	17 year old 7'3 feet(2.20 meters) tall Chinese...	23409	1.719269e+09
23	Damnthatsinteresting	This airport in Japan is located in the middle...	26396	1.719321e+09

	subreddit	title	ups	created_utc
24	todayilearned	TIL on Nov. 6th, 2000. The DEA made the larges...	23155	1.719268e+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.

Part a

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

Part b

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

The path that leads to the team-by-team data is resultSets, followed by the 0th index, and then rowSet.

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)

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 [7]:
```
url = "https://stats.nba.com/js/data/sportvu/2015/shootingTeamData.json"
nba = requests.get(url, headers = headers)
nba_json = json.loads(nba.text)
```

In [8]:
```
nba_df = pd.json_normalize(nba_json, record_path = ["resultSets", "rowSet"])
nba_df
```

Out[8]:

	0	1	2	3	4	5	6	7	8	9	...
0	1610612744	Golden State	Warriors	GSW		82	48.7	114.9	14.9	0.498	...
1	1610612759	San Antonio	Spurs	SAS		82	48.3	103.5	14.8	0.481	...
2	1610612739	Cleveland	Cavaliers	CLE		82	48.7	104.3	16.9	0.481	...
3	1610612746	Los Angeles	Clippers	LAC		82	48.6	104.5	15.0	0.497	...
4	1610612760	Oklahoma City	Thunder	OKC		82	48.6	110.2	16.1	0.480	...
5	1610612737	Atlanta	Hawks	ATL		82	48.6	102.8	19.0	0.463	...
6	1610612745	Houston	Rockets	HOU		82	48.6	106.5	17.2	0.433	...
7	1610612757	Portland	Trail Blazers	POR		82	48.5	105.1	17.5	0.441	...
8	1610612758	Sacramento	Kings	SAC		81	48.4	106.7	18.7	0.452	...
9	1610612764	Washington	Wizards	WAS		82	48.5	104.1	15.4	0.480	...
10	1610612748	Miami	Heat	MIA		82	48.6	100.0	17.9	0.488	...
11	1610612761	Toronto	Raptors	TOR		81	48.5	102.7	23.0	0.462	...
12	1610612742	Dallas	Mavericks	DAL		82	49.0	102.3	18.2	0.473	...
13	1610612766	Charlotte	Hornets	CHA		82	48.6	103.4	16.8	0.459	...
14	1610612762	Utah	Jazz	UTA		82	49.0	97.7	18.1	0.445	...
15	1610612753	Orlando	Magic	ORL		81	48.7	102.0	18.0	0.456	...
16	1610612749	Milwaukee	Bucks	MIL		82	48.7	99.0	17.4	0.463	...
17	1610612740	New Orleans	Pelicans	NOP		82	48.5	102.7	19.9	0.458	...
18	1610612750	Minnesota	Timberwolves	MIN		82	48.6	102.4	15.1	0.464	...
19	1610612754	Indiana	Pacers	IND		82	48.8	102.2	13.7	0.453	...
20	1610612751	Brooklyn	Nets	BKN		82	48.4	98.6	14.4	0.457	...
21	1610612765	Detroit	Pistons	DET		82	48.7	102.0	17.5	0.464	...
22	1610612743	Denver	Nuggets	DEN		82	48.6	101.9	15.9	0.406	...
23	1610612738	Boston	Celtics	BOS		81	48.5	105.6	18.9	0.453	...
24	1610612741	Chicago	Bulls	CHI		82	48.9	101.6	18.1	0.458	...
25	1610612755	Philadelphia	76ers	PHI		82	48.6	97.4	19.7	0.445	...
26	1610612756	Phoenix	Suns	PHX		82	48.4	100.9	15.6	0.440	...
27	1610612752	New York	Knicks	NYK		82	48.5	98.4	10.4	0.447	...
28	1610612763	Memphis	Grizzlies	MEM		82	48.6	99.1	16.4	0.440	...

	0	1	2	3	4	5	6	7	8	9	...
29	1610612747	Los Angeles	Lakers	LAL		82	48.3	97.3	15.6	0.441	...

30 rows × 33 columns

In [9]:
```
nba_columns = nba_json["resultSets"][0]["headers"]
nba_df.columns = nba_columns
nba_df
```

Out[9]:

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

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 [10]:
```
nba_df.to_json("new_nba_json.json", orient="split")
```

In [ ]:
```

```