lab03

February 2, 2025

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

1.1 DS 6001: Practice and Application of Data Science

1.1.1 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.

1.2 Problem 0

Import the following libraries:

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

1.3 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)

- JSON formatted files usually take more memory since they are able to store extra information about data such as metadata as well as accommodating nested data structures. Because of this extra information, a JSON formatted file will normally take up more memory than a CSV, which only supports tabular data structures and is not ideal for storing metadata.
- A JSON formatted file may be smaller in size if some variables don't have any values. In this case, the key and value may be omitted entirely whereas a CSV would require some space for the comma.

1.4 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)

- I'll use `requests.get().text` then `json.loads()` to load the JSON into python. This makes the most sense to me since `requests.get()` will download the data, the `.text` method will return the raw text of the JSON as a string, then finally `json.loads()` will take that string and convert it to a python list.

[9]:
```
meteor_url = 'https://data.nasa.gov/resource/y77d-th95.json'
meteor = json.loads(requests.get(meteor_url).text)
meteor[1]
```

[9]:
```
{'name': 'Aarhus',
 'id': '2',
 'nametype': 'Valid',
 'recclass': 'H6',
 'mass': '720',
 'fall': 'Fell',
 'year': '1951-01-01T00:00:00.000',
 'reclat': '56.183330',
 'reclong': '10.233330',
 'geolocation': {'type': 'Point', 'coordinates': [10.23333, 56.18333]}}
```

1.5 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)

[30]:
```
url = 'http://www.reddit.com/r/popular/top.json'
rdt = json.loads(requests.get(url,headers={'User-Agent': 'DS6001'}).text)
```

[31]:
```
top = pd.DataFrame()
for post in rdt['data']['children']:
```

```
    newpost = pd.DataFrame({'sub':[post['data']['subreddit']],
                  'title':[post['data']['title']],
                  'ups':[post['data']['ups']],
                  'created_utc':[post['data']['created_utc']]
                  })
    top = pd.concat([top,newpost],ignore_index=True)

top
```

[31]:

	sub	title	\
0	mildlyinteresting	Found a random guy who looks like me waiting f…	
1	comics	Burn	
2	meirl	meirl	
3	nextfuckinglevel	Little kid had 25 seconds to sink four balls. …	
4	holdmycatnip	Flight highlight	
5	BeAmazed	Fingal's Cave is a geological formation locate…	
6	BlackPeopleTwitter	Textbook racism	
7	pics	Photo related to HIV awareness by the CDC Hist…	
8	interestingasfuck	Atheism in a nutshell	
9	me_irl	me_irl	
10	BikiniBottomTwitter	It’s been too much already	
11	SipsTea	lmao	
12	MadeMeSmile	The vice principal and a student who share the…	
13	oddlyspecific	Only the cat heard	
14	meirl	meirl	
15	me_irl	me_irl	
16	cats	My 15 year old sheep	
17	WhitePeopleTwitter	Oh my god	
18	interestingasfuck	An Argali mountain sheep killed by its own horns.	
19	BeAmazed	Birds nest hidden in a leaf	
20	BlackPeopleTwitter	True Patriots hate Racist Takes	
21	mildlyinfuriating	My brother put light brown sugar into the same…	
22	pics	Deadly Palisades and Eaton fires in Los Angele…	
23	Unexpected	Dentists in America	
24	technology	Trump admin emails air traffic controllers to …	

	ups	created_utc
0	138871	1.738427e+09
1	109244	1.738414e+09
2	88608	1.738383e+09
3	68603	1.738402e+09
4	68065	1.738385e+09
5	59120	1.738430e+09
6	57986	1.738379e+09
7	57620	1.738411e+09
8	57640	1.738421e+09

```
9    55660  1.738417e+09
10   54348  1.738379e+09
11   49735  1.738384e+09
12   49853  1.738412e+09
13   49465  1.738420e+09
14   46665  1.738422e+09
15   46133  1.738443e+09
16   42547  1.738381e+09
17   42596  1.738432e+09
18   43059  1.738434e+09
19   41893  1.738409e+09
20   41901  1.738422e+09
21   40217  1.738414e+09
22   39272  1.738382e+09
23   38491  1.738390e+09
24   38963  1.738435e+09
```

1.6 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.

1.6.1 Part a

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

[34]:
```
nba_url = 'https://stats.nba.com/js/data/sportvu/2015/shootingTeamData.json'
nba = json.loads(requests.get(nba_url).text)
```

1.6.2 Part b

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

- The path to the team-by-team data is resultSets -> 0 -> rowSet, the team-by-team data is contained under each integer number within the `rowSet` key. The headers for each piece of team data is contained in resultSets -> 0 -> headers.

1.6.3 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)

[51]:
```
nba_df = pd.json_normalize(nba['resultSets'][0],record_path=['rowSet'])
nba_df
```

[51]:

	0	1	2	3	4	5	6	7	8	\
0	1610612744	Golden State	Warriors	GSW		82	48.7	114.9	14.9	
1	1610612759	San Antonio	Spurs	SAS		82	48.3	103.5	14.8	
2	1610612739	Cleveland	Cavaliers	CLE		82	48.7	104.3	16.9	
3	1610612746	Los Angeles	Clippers	LAC		82	48.6	104.5	15.0	
4	1610612760	Oklahoma City	Thunder	OKC		82	48.6	110.2	16.1	
5	1610612737	Atlanta	Hawks	ATL		82	48.6	102.8	19.0	
6	1610612745	Houston	Rockets	HOU		82	48.6	106.5	17.2	
7	1610612757	Portland	Trail Blazers	POR		82	48.5	105.1	17.5	
8	1610612758	Sacramento	Kings	SAC		81	48.4	106.7	18.7	
9	1610612764	Washington	Wizards	WAS		82	48.5	104.1	15.4	
10	1610612748	Miami	Heat	MIA		82	48.6	100.0	17.9	
11	1610612761	Toronto	Raptors	TOR		81	48.5	102.7	23.0	
12	1610612742	Dallas	Mavericks	DAL		82	49.0	102.3	18.2	
13	1610612766	Charlotte	Hornets	CHA		82	48.6	103.4	16.8	
14	1610612762	Utah	Jazz	UTA		82	49.0	97.7	18.1	
15	1610612753	Orlando	Magic	ORL		81	48.7	102.0	18.0	
16	1610612749	Milwaukee	Bucks	MIL		82	48.7	99.0	17.4	
17	1610612740	New Orleans	Pelicans	NOP		82	48.5	102.7	19.9	
18	1610612750	Minnesota	Timberwolves	MIN		82	48.6	102.4	15.1	
19	1610612754	Indiana	Pacers	IND		82	48.8	102.2	13.7	
20	1610612751	Brooklyn	Nets	BKN		82	48.4	98.6	14.4	
21	1610612765	Detroit	Pistons	DET		82	48.7	102.0	17.5	
22	1610612743	Denver	Nuggets	DEN		82	48.6	101.9	15.9	
23	1610612738	Boston	Celtics	BOS		81	48.5	105.6	18.9	
24	1610612741	Chicago	Bulls	CHI		82	48.9	101.6	18.1	
25	1610612755	Philadelphia	76ers	PHI		82	48.6	97.4	19.7	
26	1610612756	Phoenix	Suns	PHX		82	48.4	100.9	15.6	
27	1610612752	New York	Knicks	NYK		82	48.5	98.4	10.4	
28	1610612763	Memphis	Grizzlies	MEM		82	48.6	99.1	16.4	
29	1610612747	Los Angeles	Lakers	LAL		82	48.3	97.3	15.6	

	9	…	23	24	25	26	27	28	29	30	31	32
0	0.498	…	0.478	21.2	42.5	0.497	2.3	6.3	0.363	10.8	25.3	0.429
1	0.481	…	0.506	18.3	39.8	0.460	0.9	2.6	0.341	6.1	15.9	0.381
2	0.481	…	0.473	18.2	40.7	0.447	1.7	5.7	0.299	9.0	23.9	0.378
3	0.497	…	0.480	18.9	42.0	0.450	2.0	6.0	0.334	7.7	20.8	0.373
4	0.480	…	0.497	17.5	38.7	0.451	1.6	5.1	0.321	6.6	18.6	0.356
5	0.463	…	0.483	19.4	44.6	0.435	1.0	3.1	0.311	9.0	25.3	0.355
6	0.433	…	0.472	15.5	36.4	0.426	2.3	7.4	0.318	8.4	23.5	0.355
7	0.441	…	0.447	18.0	39.8	0.453	1.7	5.9	0.295	8.8	22.6	0.389

```
8   0.452  …  0.473  18.1  39.7  0.454  0.9  3.1  0.276   7.2  19.4  0.372
9   0.480  …  0.483  19.5  44.3  0.439  0.7  2.7  0.254   8.0  21.5  0.371
10  0.488  …  0.490  15.7  35.2  0.445  0.8  2.9  0.282   5.3  15.1  0.347
11  0.462  …  0.461  14.1  32.4  0.436  1.8  5.6  0.327   6.8  17.7  0.384
12  0.473  …  0.464  17.5  41.4  0.423  1.4  5.3  0.273   8.4  23.3  0.360
13  0.459  …  0.449  17.0  39.8  0.427  1.8  6.0  0.297   8.9  23.4  0.379
14  0.445  …  0.468  15.9  37.2  0.426  1.4  4.3  0.318   7.1  19.5  0.363
15  0.456  …  0.475  18.5  42.6  0.435  0.7  2.7  0.249   7.1  19.5  0.363
16  0.463  …  0.477  13.2  29.4  0.448  1.1  4.0  0.270   4.3  11.6  0.370
17  0.458  …  0.460  17.9  41.1  0.434  0.6  2.6  0.247   7.9  21.2  0.374
18  0.464  …  0.471  16.1  35.4  0.455  0.7  2.6  0.272   4.8  13.8  0.350
19  0.453  …  0.465  16.4  38.1  0.431  1.7  5.7  0.299   6.4  17.4  0.368
20  0.457  …  0.464  15.8  36.1  0.438  1.0  3.3  0.303   5.5  15.1  0.363
21  0.464  …  0.452  15.7  37.2  0.422  0.9  4.0  0.227   8.1  22.2  0.366
22  0.406  …  0.448  16.4  37.8  0.434  1.1  4.3  0.264   6.9  19.5  0.354
23  0.453  …  0.451  16.9  39.9  0.424  1.6  5.7  0.274   7.1  20.3  0.350
24  0.458  …  0.442  17.0  38.5  0.441  1.3  3.9  0.332   6.6  17.5  0.380
25  0.445  …  0.449  15.3  37.4  0.409  1.6  5.7  0.281   7.7  21.8  0.354
26  0.440  …  0.447  16.6  39.5  0.421  1.4  5.0  0.288   7.6  20.8  0.363
27  0.447  …  0.439  15.9  36.4  0.438  1.5  4.9  0.305   5.9  16.6  0.358
28  0.440  …  0.459  16.1  38.5  0.418  0.7  2.5  0.278   5.4  16.0  0.340
29  0.441  …  0.420  14.0  34.5  0.406  2.2  7.9  0.278   5.6  16.7  0.335

[30 rows x 33 columns]
```

1.6.4 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)

[55]:
```
nba_cols = nba['resultSets'][0]['headers']
nba_df.columns = nba_cols
nba_df.head(3)
```

[55]:
```
      TEAM_ID     TEAM_CITY  TEAM_NAME TEAM_ABBREVIATION TEAM_CODE  GP   MIN  \
0  1610612744  Golden State   Warriors               GSW           82  48.7
1  1610612759   San Antonio      Spurs               SAS           82  48.3
2  1610612739     Cleveland  Cavaliers               CLE           82  48.7

     PTS  PTS_DRIVE  FGP_DRIVE  …   CFGP  UFGM  UFGA   UFGP  CFG3M  CFG3A  \
0  114.9       14.9      0.498  …  0.478  21.2  42.5  0.497    2.3    6.3
1  103.5       14.8      0.481  …  0.506  18.3  39.8  0.460    0.9    2.6
2  104.3       16.9      0.481  …  0.473  18.2  40.7  0.447    1.7    5.7

   CFG3P  UFG3M  UFG3A  UFG3P
0  0.363   10.8   25.3  0.429
1  0.341    6.1   15.9  0.381
```

```
2  0.299    9.0   23.9  0.378

[3 rows x 33 columns]
```

1.7 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)

[57]:
```
new_nba = nba_df.to_json('nba.json',orient='split')
```