The objective of this assignment is to get familiar with working with packages for dealing with JSON.

1. Get familiar with packages for dealing with JSON
2. Study examples with JSON strings and files
3. Work on exercise to be completed and submitted

Resource/Data for assignment is located.

* reference: <http://pandas.pydata.org/pandas-docs/stable/io.html#io-json-reader>
* data source: <http://jsonstudio.com/resources/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

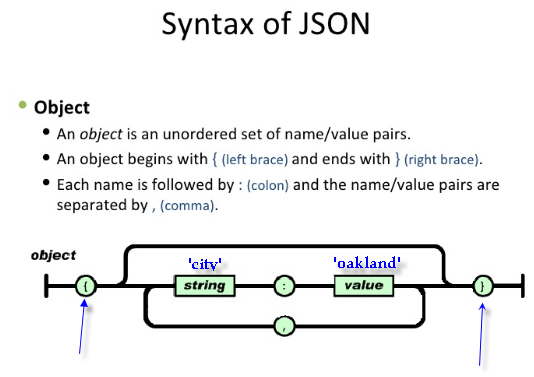
Basic JSON

What is JSON first? JSON: JavaScript Object Notation. JSON is syntax for storing and exchanging data. JSON is text, written with JavaScript object notation

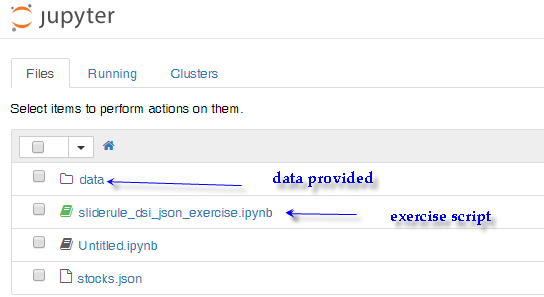
Why is it used? When exchanging data between a browser and a server, the data can only be text. JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server. We can also convert any JSON received from the server into JavaScript objects.

What is the benefit of this approach?

This way we can work with the data as JavaScript objects, with no complicated parsing and translations



The Jupyter notebook was launched using the Anaconda Prompt from window location of where the data was stored.



Part I: Launched the exercise script.

1. First command is to import the pandas



1. Next is to import json. The command that follows is a pandas utility function called json\_normalize that flattens semi-structured JSON objects.



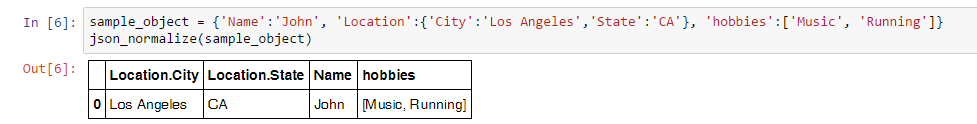
1. The next is an example of an object.



If you look at the sample object it is not exactly like a typical JSON syntax.

1. You have string “location” with two string and value(called nested)
2. You have a string “hobbies” with a list for its value.

Here is output information:



We can see for the nested “Location” string we created two columns to accommodate that sort of data and for string “hobbies” we created list for value and only one column since there is only one string.

Observation

*We can make the observation that numbers of “strings” equals the number of columns at least when using the json\_normalize. I don’t know the script content of json\_normalize.*

Part II: JSON example with string

JSON example, with string

* demonstrates creation of normalized data frames (tables) from nested json string
* source: <http://pandas.pydata.org/pandas-docs/stable/io.html#normalization>

In this case we are going to take a nest json string and make it a normalized data frame (table).

1. Go to the source of the format. This is important location on web.

<http://pandas.pydata.org/pandas-docs/stable/io.html#normalization>

This section is called normalization.

 I know normalization has something to do with re-scaling but not clear in this context.

From the website:

“Pandas provide a utility function to take a dict or list of dicts and normalize this semi-structured data into a flat table”

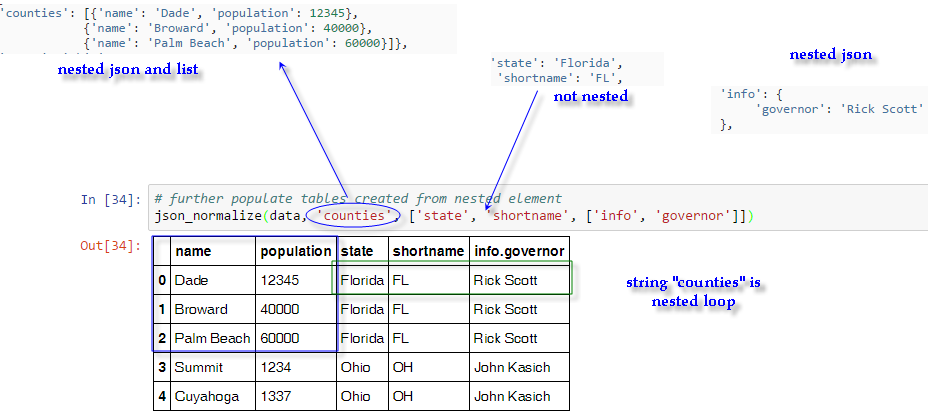


The above took the info in data using the json\_normalize(data, ‘counties’, [‘state’, ‘shortname’, [‘info, ‘governor’]])

What is relation between strings and columns? The number of string equals the number of columns. Column info.governor is slightly different also I would expect name and population have counties.name or counties.pop… be used. I am not sure why not.

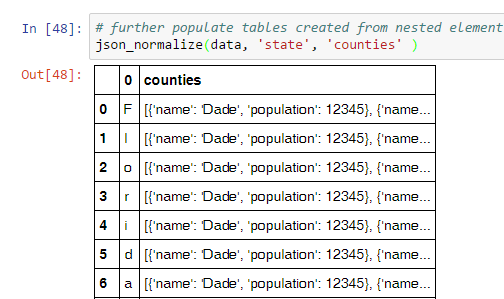
json\_normalize utility format requires the columns to be placed in certain order.

When you are using the json\_normalize function there is a unique way to specify the command..



What is unique about data? You can only have one state and one governor but many counties. So, when presenting the data the order in the above equation matters in that the beginning columns (keys) should be for information changing.

For example,



Part III: JSON example with file

## Objectives:

* demonstrates reading in a json file as a string and as a table
* uses small sample file containing data about projects funded by the World Bank
* data source: <http://jsonstudio.com/resources/>

In this exercise, I am going to take read in a json file.



The command to load is json.load((open(‘data/world\_bank\_projects\_less.json’)))

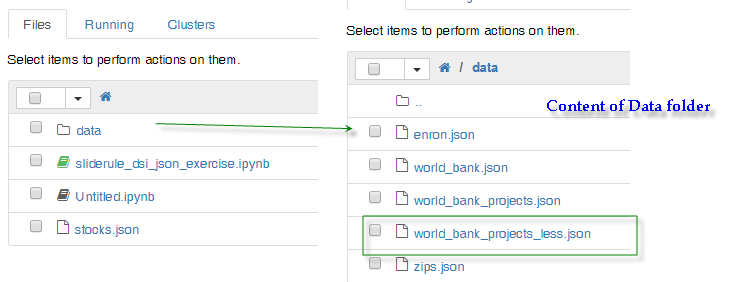
Observation 2: (know your working folder)

Now to open the file called *world\_bank\_projects\_less.json* or the command to work- the file has to be in the working directory. You can check your Jupyter Notebook working directory by using the following command and output will give the path.



Observation 3: (starting Jupyter book from data folder- place where data you want use)

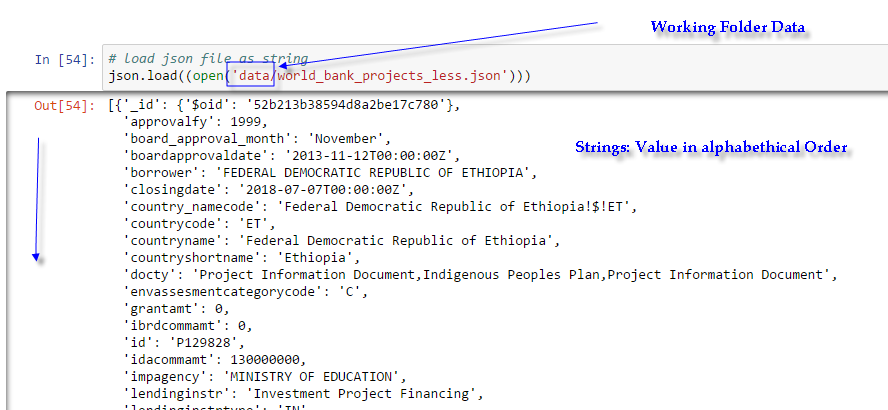
My approach has been to start Jupyter notebook in the folder where the data or files are located. I had to also make sure to run the Anaconda prompt “as an administrator” otherwise I will have write issues.



Run the command: json load command has two forms one file(json.load) and another for string(json.loads). The s is for string.

We will be using one for file.





I see a lot of strings in what looks to be alphabetical order with value.

Format is ‘string’:’value’ which is typical json format

However, it is not presentable in this form- so what can we do?

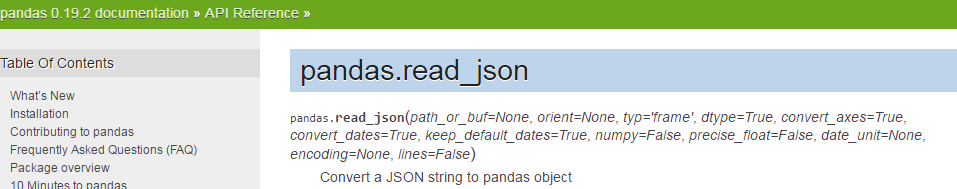
In this format, the data is not visually useful for me to make informed decision or observation.

We have to convert it to Pandas Data Frame. Luckily there are utilities to do this.



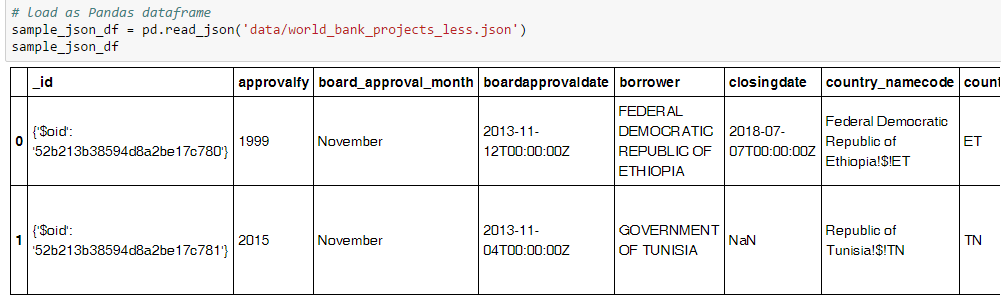
This takes a json file and converts it to Pandas data frame. I would like to understand the content of pd.read\_json

Pandas documentation has a section in pandas.read\_json



It would be nice see the code of this utility that is able to change the format from json to pandas object.

Here is output:



We use nice header on first row and it sorted in alphabetical order via string name. It basically has become a nice table which easy to read. The World Bank project file was trimmed to only have two rows for presentation purpose.

What tools and utilities have been used so far to convert json file to data frame?

Step one: Importation to python Jupyter notebook

* 1. Imported pandas package (import pandas as pd)
  2. Imported json (import json)
  3. Imported a pandas utility for json called json\_normalize

(from pandas.io.json import json\_normalize)

Step two:

1. Created a json string and used json\_normalize to convert to Pandas dataframe 
2. Loaded or opened a json file by using json.load utility (json.load((open(data\filename)))
3. Finally we took the json file and converted to pandas dataframe using the following command:

wbpDF = pd.read\_json(‘data/world\_bank\_projects.json’)

Part IV: json exercises

Objective

Using data in file 'data/world\_bank\_projects.json' and the techniques demonstrated above,

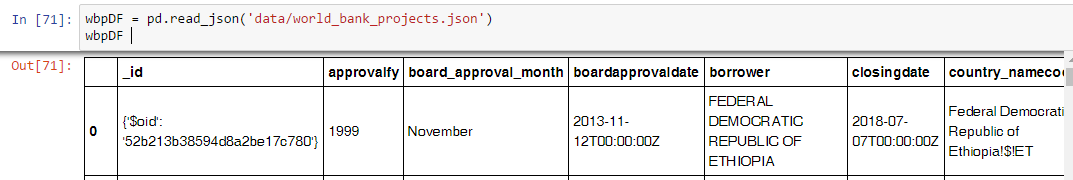
1. Find the 10 countries with most projects
2. Find the top 10 major project themes (using column 'mjtheme\_namecode')
3. In 2. above you will notice that some entries have only the code and the name is missing. Create a dataframe with the missing names filled in.

My Thought Process

My first step would be to find out as much as possible about the data or information.

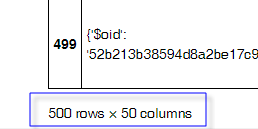
In this case after running the code

wbpDF = pd.read\_json(‘data/world\_bank\_projects.json’)

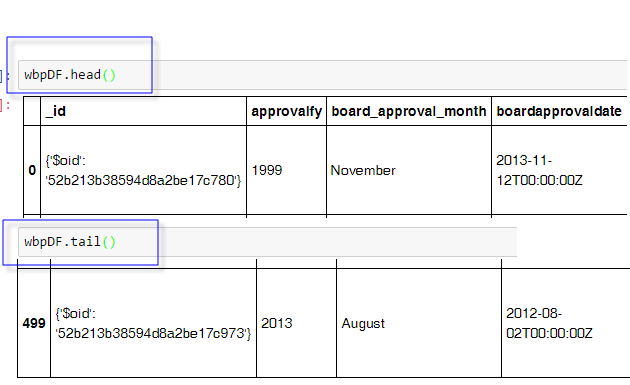


1. How many columns or rows do we have?

I can scroll down to find how many rows/columns.



Or just for practice I can use the head() and tail() on DataFrame- head() first five rows and tail() last five rows.



We have 500 rows and 50 columns in this table.

1. I have used excel for many years and I am comfortable with it.

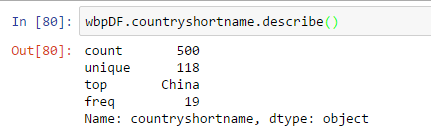


This allowed me to see the file in excel and it easier to see than inline response in jupyter notebook.

Question1: Find the 10 countries with most projects

There is a column for countries (countryshortname) and this is a matter of finding the countries with high occurrence and then sorting by occurrence to find the top 10?

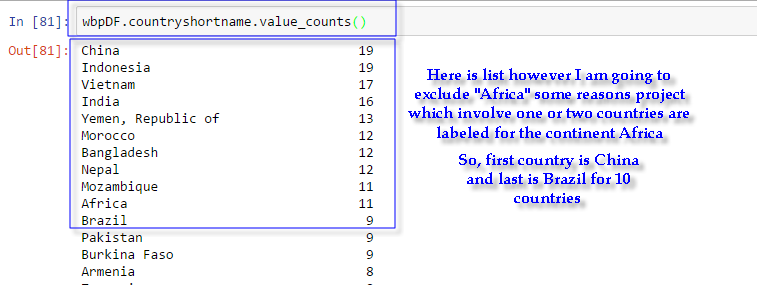
We run a simple test code to just get an overview of the countryshortname column.



By country short name we have 500 non-null and 118 unique (countries) and top country by number of projects is China and there are 19 World Bank projects for China. I was able to verify this in the excel file.

In the pandas documentation there is usage of something called value\_counts()

In this case, we want to know for column (country) which countries have highest number of project and information to be presented from highest to lowest.

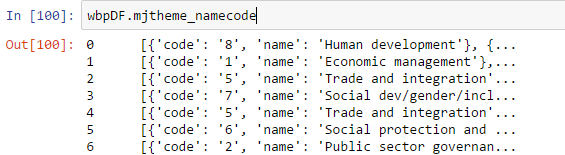


The data above is asking by country but it shows “Africa” as country. I looked at all those projects and some were country specific and some region specific but not continental.

Question2: Find the top 10 major project themes (using column 'mjtheme\_namecode')

mjtheme\_namecode

Let’s just get list of by using code wbpDF.mjtheme\_namecode



the content of column for example:

[{'code': '9', 'name': 'Urban development'}, {'code': '7', 'name': 'Social dev/gender/inclusion'}]

I say List because there are two brackets [….]

Inside the List there is json format information- string: value

[{string:value, string:value}, {string:value, string:value}]

The question is how do I partition this information?

String1 :‘code’ value: int

string2: ‘name’ value: string

incomplete.

I tried different approaches. I have DataFrame = wbpDF and column called ‘mjtheme\_namecode’ the content of that column is not series which is just one-dimensional or just a value. In this case, the content is List with JSON formatted data.

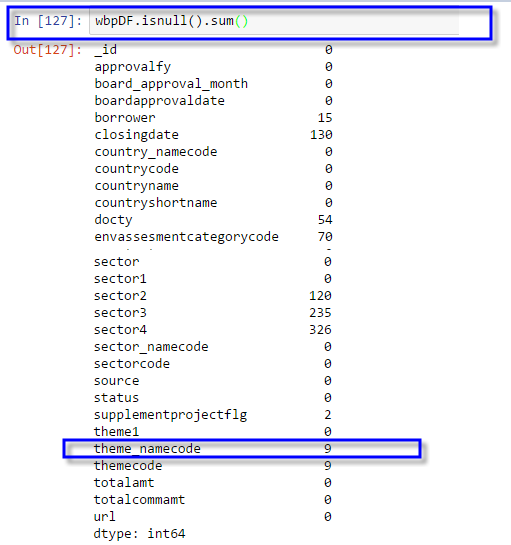
My feeling is the best approach is to use the json\_normalize which pandas utility to take json formatted data into Data Frame namely row/column with scale values.

I have attempted different format of json\_normalize but they haven’t worked. I am missing a basic format solution.

Question3:

In 2. above you will notice that some entries have only the code and the name is missing. Create a dataframe with the missing names filled in.

I ran the code. It seems to indicate nine lines.



There several tutorials about dealing with missing data I need to look at…