JSON and XML - Lab

Introduction

In this lab, we'll continue investigating new formats for datasets. Specifically, we'll investigate two of the most popular data formats for the web: JSON and XML including strenghts and weaknesses.

Objectives

You will be able to:

- Effectively use the JSON module to load and parse JSON documents
- Read and access data stored in JSON and XML
- Compare and contrast the JSON and XML as data interchange types

XML

In [1]:
```
import xml.etree.ElementTree as ET
```

Create an XML tree and retrieve the root tag.

In [2]:
```
#Your code here
tree = ET.parse('nyc_2001_campaign_finance.xml')
root = tree.getroot()
```

How many direct descendents does the root tag have?

In [3]:
```
#Answer: 1
count = 0
for child in root:
    count += 1
print(count)
```

```
1
```

How many different types of tags are there within the entire XML file?

In [4]:
```
# Your code here
tags = []
for element in root.iter():
    tags.append(element.tag)
print(len(set(tags)))
```

```
13
```

Create a DataFrame listing the number of each type of tag.

Sort the DataFrame in descending order by the tag count. The first entry should demonstrate there are 286 row tags in the XML file.
(Your DataFrame will be a single column, so could also be thought of as a Series.)

In [5]:
```
import pandas as pd
```

In [6]:
```
#Your code here
tags = {}
for element in root.iter():
    tags[element.tag] = tags.get(element.tag, 0) + 1
df = pd.DataFrame.from_dict(tags, orient='index')
df.columns = ['count']
df = df.sort_values(by='count', ascending=False)
df.head()
```

Out[6]:

	count
row	286

candid	285
candname	285
canclass	285
election	284

JSON

Open the same dataset from json

```
In [7]:  #Your code here
         import json
         f = open('nyc_2001_campaign_finance.json')
         data = json.load(f)
```

What is the root data type of the json file?

```
In [8]:  ### Your code here
         type(data)

Out[8]:  dict
```

Navigate to the 'data' key of your loaded json object. What data type is this?

```
In [9]:  #Your code here
         type(data['data'])

Out[9]:  list
```

Preview the first entry from the value returned by the 'data' key above.

```
In [10]:  #Your code here
          data['data'][0]

Out[10]:  [1,
           'E3E9CC9F-7443-43F6-94AF-B5A0F802DBA1',
           1,
           1315925633,
           '392904',
           1315925633,
           '392904',
           '{\n  "invalidCells" : {\n    "1519001" : "TOTALPAY",\n    "1518998" : "PRIMARYPAY",\n    "151900
          0" : "RUNOFFPAY",\n    "1518999" : "GENERALPAY",\n    "1518994" : "OFFICECD",\n    "1518996" : "OF
          FICEDIST",\n    "1518991" : "ELECTION"\n  }\n}',
           None,
           'CANDID',
           'CANDNAME',
           None,
           'OFFICEBORO',
           None,
           'CANCLASS',
           None,
           None,
           None,
           None]
```

Preview the Entry under meta -> view -> columns (the keys of three successively nested dictionaries)

```
In [11]:  data['meta']['view']['columns']
```

```
           {'dataTypeName': 'meta_data',
            'fieldName': ':position',
 'flags' : ['hidden'],
'format': {}
```

Create a DataFrame from your json data

The previous two questions previewed one entry from the data object within the json file, as well as the column details associated with that data from the meta entry within the json file. Both should have 19 entries. Create a DataFrame of the data. Be sure to use the information from the meta entry to add appropriate column names to your DataFrame.

In [12]:
```python
#Your code here
df = pd.DataFrame(data['data'])
cols = [i['name'] for i in data['meta']['view']['columns']]
df.columns = cols
df.head()
```

Out[12]:

	sid	id	position	created_at	created_meta	updated_at	updated_meta	meta	ELECTION	CANDID
0	1	E3E9CC9F-7443-43F6-94AF-B5A0F802DBA1	1	1315925633	392904	1315925633	392904	{\n "invalidCells" : {\n "1519001" : "TOTA...	None	CANDID
1	2	9D257416-581A-4C42-85CC-B6EAD9DED97F	2	1315925633	392904	1315925633	392904	{\n}	2001	B4
2	3	B80D7891-93CF-49E8-86E8-182B618E68F2	3	1315925633	392904	1315925633	392904	{\n}	2001	445
3	4	BB012003-78F5-406D-8A87-7FF8A425EE3F	4	1315925633	392904	1315925633	392904	{\n}	2001	HF
4	5	945825F9-2F5D-47C2-A16B-75B93E61E1AD	5	1315925633	392904	1315925633	392904	{\n}	2001	IR

What's wrong with the first row of the DataFrame?

In [13]:
```python
df.meta.iloc[0]
```

Out[13]:
```
'{\n  "invalidCells" : {\n    "1519001" : "TOTALPAY",\n    "1518998" : "PRIMARYPAY",\n    "151900
0" : "RUNOFFPAY",\n    "1518999" : "GENERALPAY",\n    "1518994" : "OFFICECD",\n    "1518996" : "OF
FICEDIST",\n    "1518991" : "ELECTION"\n  }\n}'
```

#Your answer here The first row has invalidCells according to the meta data, and appears to have column names in other entries. It appears to be a faulty record of sorts.

Summary

Congratulations! You've started exploring some more complicated data structures used for the web and got to practice data munging and exploring!