*You are currently looking at **version 1.5** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ (https://www.coursera.org/learn/python-data-analysis/resources/0dhYG) course resource.*

Assignment 3 - More Pandas

This assignment requires more individual learning then the last one did - you are encouraged to check out the pandas documentation (http://pandas.pydata.org/pandas-docs/stable/) to find functions or methods you might not have used yet, or ask questions on Stack Overflow (http://stackoverflow.com/) and tag them as pandas and python related. And of course, the discussion forums are open for interaction with your peers and the course staff.

Question 1 (20%)

Load the energy data from the file `Energy Indicators.xls`, which is a list of indicators of <u>energy supply and renewable electricity production</u> (<u>Energy%20Indicators.xls</u>) from the <u>United Nations (http://unstats.un.org/unsd/environment/excel_file_tables/2013/Energy%20Indicators.xls</u>) for the year 2013, and should be put into a DataFrame with the variable name of **energy**.

Keep in mind that this is an Excel file, and not a comma separated values file. Also, make sure to exclude the footer and header information from the datafile. The first two columns are unneccessary, so you should get rid of them, and you should change the column labels so that the columns are:

`['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']`

Convert `Energy Supply` to gigajoules (there are 1,000,000 gigajoules in a petajoule). For all countries which have missing data (e.g. data with "...") make sure this is reflected as `np.NaN` values.

Rename the following list of countries (for use in later questions):

`"Republic of Korea": "South Korea",`
`"United States of America": "United States",`
`"United Kingdom of Great Britain and Northern Ireland": "United Kingdom",`
`"China, Hong Kong Special Administrative Region": "Hong Kong"`

There are also several countries with numbers and/or parenthesis in their name. Be sure to remove these,

e.g.

`'Bolivia (Plurinational State of)'` should be `'Bolivia'`,

`'Switzerland17'` should be `'Switzerland'`.

Next, load the GDP data from the file `world_bank.csv`, which is a csv containing countries' GDP from 1960 to 2015 from <u>World Bank</u> (<u>http://data.worldbank.org/indicator/NY.GDP.MKTP.CD</u>). Call this DataFrame **GDP**.

Make sure to skip the header, and rename the following list of countries:

```
    "Korea, Rep.": "South Korea",
    "Iran, Islamic Rep.": "Iran",
    "Hong Kong SAR, China": "Hong Kong"
```

Finally, load the Sciamgo Journal and Country Rank data for Energy Engineering and Power Technology (http://www.scimagojr.com/countryrank.php?category=2102) from the file `scimagojr-3.xlsx`, which ranks countries based on their journal contributions in the aforementioned area. Call this DataFrame **ScimEn**.

Join the three datasets: GDP, Energy, and ScimEn into a new dataset (using the intersection of country names). Use only the last 10 years (2006-2015) of GDP data and only the top 15 countries by Scimagojr 'Rank' (Rank 1 through 15).

The index of this DataFrame should be the name of the country, and the columns should be ['Rank', 'Documents', 'Citable documents', 'Citations', 'Self-citations', 'Citations per document', 'H index', 'Energy Supply', 'Energy Supply per Capita', '% Renewable', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015'].

This function should return a DataFrame with 20 columns and 15 entries.

In [116]:
```python
import pandas as pd
import numpy as np
energy = pd.read_excel('Energy Indicators.xls')
energy = energy[16:243]
energy = energy.drop(energy.columns[[0, 1]], axis=1)
energy.rename(columns={'Environmental Indicators: Energy': 'Country','Unnamed: 3':'Energy Supply','Unnamed: 4':'Energy
Supply per Capita','Unnamed: 5':'% Renewable'}, inplace=True) # renaming
energy.replace('...', np.nan,inplace = True)
energy['Energy Supply'] *= 1000000

def remove_digit(data):
    newData = ''.join([i for i in data if not i.isdigit()])
    i = newData.find('(')
    if i>-1: newData = newData[:i]
    return newData.strip()
energy['Country'] = energy['Country'].apply(remove_digit)
renamess = {"Republic of Korea": "South Korea",
"United States of America": "United States",
"United Kingdom of Great Britain and Northern Ireland": "United Kingdom",
"China, Hong Kong Special Administrative Region": "Hong Kong"}
energy.replace({"Country": renamess},inplace = True)

GDP = pd.read_csv('world_bank.csv', skiprows=4)
GDP.rename(columns={'Country Name': 'Country'}, inplace=True)
renamess2 = {"Korea, Rep.": "South Korea",
"Iran, Islamic Rep.": "Iran",
"Hong Kong SAR, China": "Hong Kong"}
GDP.replace({"Country": renamess2},inplace = True)

ScimEn = pd.read_excel('scimagojr-3.xlsx')
data = pd.merge(pd.merge(energy, GDP, on='Country'), ScimEn, on='Country')
data.set_index('Country',inplace=True)

data = data[['Rank', 'Documents', 'Citable documents', 'Citations', 'Self-citations', 'Citations per document', 'H ind
ex', 'Energy Supply', 'Energy Supply per Capita', '% Renewable', '2006', '2007', '2008', '2009', '2010', '2011', '201
2', '2013', '2014', '2015']]
data = data[(data['Rank']>=1) & (data['Rank']<=15)]
data.sort('Rank',inplace=True)
def answer_one():

    return data
```
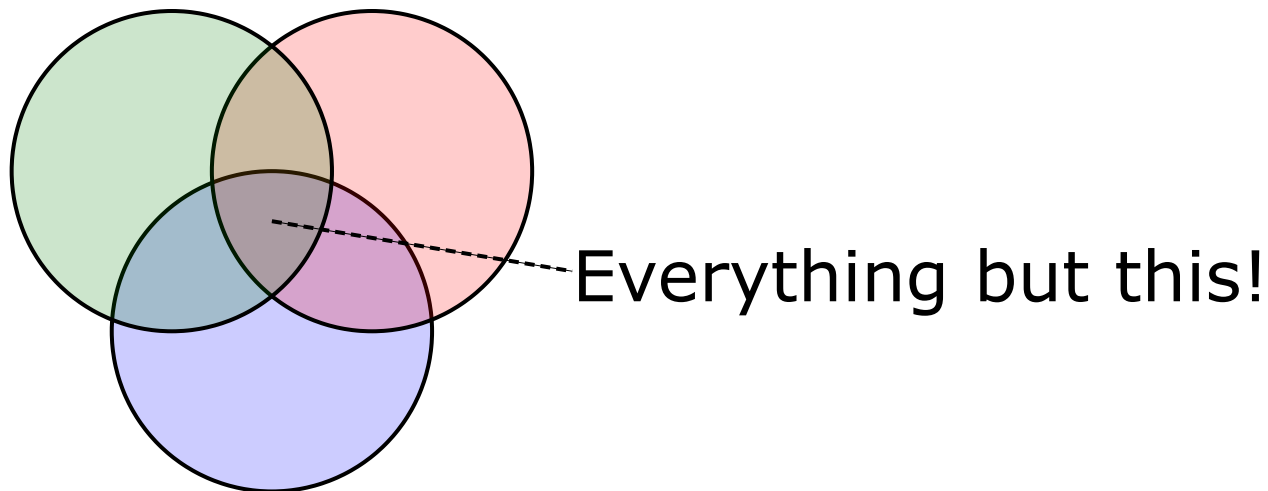
Question 2 (6.6%)

The previous question joined three datasets then reduced this to just the top 15 entries. When you joined the datasets, but before you reduced this to the top 15 items, how many entries did you lose?

This function should return a single number.

In [29]:
```
%%HTML
<svg width="800" height="300">
  <circle cx="150" cy="180" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="blue" />
  <circle cx="200" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="red" />
  <circle cx="100" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="green" />
  <line x1="150" y1="125" x2="300" y2="150" stroke="black" stroke-width="2" fill="black" stroke-dasharray="5,3"/>
  <text  x="300" y="165" font-family="Verdana" font-size="35">Everything but this!</text>
</svg>
```

```
In [117]: def answer_two():
              lev1=pd.merge(pd.merge(energy,GDP,how='outer',on='Country'),ScimEn,how='outer',on='Country')
              s1=len(lev1)
              lev2=pd.merge(pd.merge(energy,GDP,on='Country'),ScimEn,on='Country')
              s2=len(lev2)
              return s1-s2
```

Answer the following questions in the context of only the top 15 countries by Scimagojr Rank (aka the DataFrame returned by answer_one())

Question 3 (6.6%)

What is the average GDP over the last 10 years for each country? (exclude missing values from this calculation.)

This function should return a Series named avgGDP with 15 countries and their average GDP sorted in descending order.

```
In [118]: def answer_three():
              Top15 = answer_one()
              first=Top15.copy()
              years=['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']


              return first[years].mean(axis=1).sort_values(ascending=False).rename('avgGDP')
```

Question 4 (6.6%)

By how much had the GDP changed over the 10 year span for the country with the 6th largest average GDP?

This function should return a single number.

```
In [119]: def answer_four():
              Top15 = answer_one()
              second=Top15.copy()
              years=['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']
              t=second[years].mean(axis=1).sort_values(ascending=False).rename('avgGDP')
              this=t.reset_index()
              p=this.loc[5]['Country']
              tt=(second.loc[p]['2015']-second.loc[p]['2006'])



              return tt
```

Question 5 (6.6%)

What is the mean Energy Supply per Capita?

This function should return a single number.

```
In [120]: def answer_five():
              Top15 = answer_one()
              third=Top15.copy()

              return third['Energy Supply per Capita'].mean()
```

Question 6 (6.6%)

What country has the maximum % Renewable and what is the percentage?

This function should return a tuple with the name of the country and the percentage.

```
In [121]: def answer_six():
              Top15 = answer_one()
              fourth=Top15.copy()
              country=fourth['% Renewable'].argmax()
              value=fourth.loc[country]['% Renewable']
              return country,value
```

Question 7 (6.6%)

Create a new column that is the ratio of Self-Citations to Total Citations. What is the maximum value for this new column, and what country has the highest ratio?

This function should return a tuple with the name of the country and the ratio.

```
In [129]: def answer_seven():
              Top15 = answer_one()
              a=Top15.copy()
              a['Ratio']=a['Self-citations']/a['Citations']
              country=a['Ratio'].argmax()
              value=a.loc[country]['Ratio']

              return country,value
```

Question 8 (6.6%)

Create a column that estimates the population using Energy Supply and Energy Supply per capita. What is the third most populous country according to this estimate?

This function should return a single string value.

```
In [123]: def answer_eight():
              Top15 = answer_one()
              b=Top15.copy()
              b['Population']=b['Energy Supply']/b['Energy Supply per Capita']
              country=b.sort_values(by='Population',ascending=False)


              return country.iloc[2].name
```

Question 9 (6.6%)

Create a column that estimates the number of citable documents per person. What is the correlation between the number of citable documents per capita and the energy supply per capita? Use the `.corr()` method, (Pearson's correlation).

This function should return a single number.

(Optional: Use the built-in function `plot9()` to visualize the relationship between Energy Supply per Capita vs. Citable docs per Capita)

```
In [124]: def answer_nine():
              Top15 = answer_one()
              c=Top15.copy()
              c['EstimatePop'] = c['Energy Supply'] / c['Energy Supply per Capita']
              c['CitDocPerPerson'] = c['Citable documents'] / c['EstimatePop']
              return c[['Energy Supply per Capita', 'CitDocPerPerson']].corr().ix['Energy Supply per Capita', 'CitDocPerPerson']
```

```
In [102]: # def plot9():
          #     import matplotlib as plt
          #     %matplotlib inline

          #     Top15 = answer_one()
          #     Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
          #     Top15['Citable docs per Capita'] = Top15['Citable documents'] / Top15['PopEst']
          #     Top15.plot(x='Citable docs per Capita', y='Energy Supply per Capita', kind='scatter', xlim=[0, 0.0006])
          # # plot9()
```

```
In [104]: #plot9() # Be sure to comment out plot9() before submitting the assignment!
```

Question 10 (6.6%)

Create a new column with a 1 if the country's % Renewable value is at or above the median for all countries in the top 15, and a 0 if the country's % Renewable value is below the median.

This function should return a series named `HighRenew` *whose index is the country name sorted in ascending order of rank.*

```
In [125]: def answer_ten():
              Top15 = answer_one()
              d=Top15.copy()

              avgg=d['% Renewable'].median()
              d['HighRenew']=d['% Renewable']
              d['HighRenew']=d['HighRenew'].apply(lambda x:1 if x>=avgg else 0)
              d.sort_values(by='Rank',ascending=True)

              return d['HighRenew']
```

Question 11 (6.6%)

Use the following dictionary to group the Countries by Continent, then create a dateframe that displays the sample size (the number of countries in each continent bin), and the sum, mean, and std deviation for the estimated population of each country.

```
ContinentDict  = {'China':'Asia',
                  'United States':'North America',
                  'Japan':'Asia',
                  'United Kingdom':'Europe',
                  'Russian Federation':'Europe',
                  'Canada':'North America',
                  'Germany':'Europe',
                  'India':'Asia',
                  'France':'Europe',
                  'South Korea':'Asia',
                  'Italy':'Europe',
                  'Spain':'Europe',
                  'Iran':'Asia',
                  'Australia':'Australia',
                  'Brazil':'South America'}
```

This function should return a DataFrame with index named Continent ['Asia', 'Australia', 'Europe', 'North America', 'South America'] and columns ['size', 'sum', 'mean', 'std']

```
In [110]:  def answer_eleven():
               Top15 = answer_one()
               ContinentDict  = {'China':'Asia',
                                 'United States':'North America',
                                 'Japan':'Asia',
                                 'United Kingdom':'Europe',
                                 'Russian Federation':'Europe',
                                 'Canada':'North America',
                                 'Germany':'Europe',
                                 'India':'Asia',
                                 'France':'Europe',
                                 'South Korea':'Asia',
                                 'Italy':'Europe',
                                 'Spain':'Europe',
                                 'Iran':'Asia',
                                 'Australia':'Australia',
                                 'Brazil':'South America'}
               groups = pd.DataFrame(columns = ['size', 'sum', 'mean', 'std'])
               Top15['Estimate Population'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
               for group, frame in Top15.groupby(ContinentDict):
                   groups.loc[group] = [len(frame), frame['Estimate Population'].sum(),frame['Estimate Population'].mean(),frame[
           'Estimate Population'].std()]
               return groups
```

Question 12 (6.6%)

Cut % Renewable into 5 bins. Group Top15 by the Continent, as well as these new % Renewable bins. How many countries are in each of these groups?

*This function should return a **Series** with a MultiIndex of* `Continent`*, then the bins for* `% Renewable`*. Do not include groups with no countries.*

```
In [126]: def answer_twelve():
              ContinentDict  = {'China':'Asia',
                                'United States':'North America',
                                'Japan':'Asia',
                                'United Kingdom':'Europe',
                                'Russian Federation':'Europe',
                                'Canada':'North America',
                                'Germany':'Europe',
                                'India':'Asia',
                                'France':'Europe',
                                'South Korea':'Asia',
                                'Italy':'Europe',
                                'Spain':'Europe',
                                'Iran':'Asia',
                                'Australia':'Australia',
                                'Brazil':'South America'}
              Top15 = answer_one()
              Top15=Top15.reset_index()
              Top15['Continent']=[ContinentDict[country] for country in Top15['Country']]
              Top15['Bins']=pd.cut(Top15['% Renewable'],5)

              return Top15.groupby(['Continent','Bins']).size()
```

Question 13 (6.6%)

Convert the Population Estimate series to a string with thousands separator (using commas). Do not round the results.

e.g. 317615384.61538464 -> 317,615,384.61538464

This function should return a Series PopEst whose index is the country name and whose values are the population estimate string.

In [127]:
```python
def answer_thirteen():
    Top15 = answer_one()
    Top15['EstimatePop'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15['EstimatePop']=Top15['EstimatePop'].astype(float)
    p=[]
    for i in Top15['EstimatePop']:
        i='{0:,}'.format(i)
        p.append(i)
    Top15['EstimatePop']=p
    return Top15['EstimatePop']
```

Optional

Use the built in function `plot_optional()` to see an example visualization.

In [ ]:
```python
def plot_optional():
    import matplotlib as plt
    %matplotlib inline
    Top15 = answer_one()
    ax = Top15.plot(x='Rank', y='% Renewable', kind='scatter',
                    c=['#e41a1c','#377eb8','#e41a1c','#4daf4a','#4daf4a','#377eb8','#4daf4a','#e41a1c',
                       '#4daf4a','#e41a1c','#4daf4a','#4daf4a','#e41a1c','#dede00','#ff7f00'],
                    xticks=range(1,16), s=6*Top15['2014']/10**10, alpha=.75, figsize=[16,6]);

    for i, txt in enumerate(Top15.index):
        ax.annotate(txt, [Top15['Rank'][i], Top15['% Renewable'][i]], ha='center')

    print("This is an example of a visualization that can be created to help understand the data. \
This is a bubble chart showing % Renewable vs. Rank. The size of the bubble corresponds to the countries' \
2014 GDP, and the color corresponds to the continent.")
```

In [114]:
```python
p=answer_one()
```

In [115]:
```
p
```

Out[115]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Energy Supply per Capita	% Renewable	...	2011
China	1	127050	126767	597237	411683	4.70	138	1.271910e+11	93.0	19.754910	...	6.612490e+12
United States	2	96661	94747	792274	265436	8.20	230	9.083800e+10	286.0	11.570980	...	1.520402e+13
Japan	3	30504	30287	223024	61554	7.31	134	1.898400e+10	149.0	10.232820	...	5.473738e+12
United Kingdom	4	20944	20357	206091	37874	9.84	139	7.920000e+09	124.0	10.600470	...	2.450911e+12
Russian Federation	5	18534	18301	34266	12422	1.85	57	3.070900e+10	214.0	17.288680	...	1.589943e+12
Canada	6	17899	17620	215003	40930	12.01	149	1.043100e+10	296.0	61.945430	...	1.664087e+12
Germany	7	17027	16831	140566	27426	8.26	126	1.326100e+10	165.0	17.901530	...	3.542371e+12
India	8	15005	14841	128763	37209	8.58	115	3.319500e+10	26.0	14.969080	...	1.821872e+12
France	9	13153	12973	130632	28601	9.93	114	1.059700e+10	166.0	17.020280	...	2.702032e+12
South Korea	10	11983	11923	114675	22595	9.57	104	1.100700e+10	221.0	2.279353	...	1.134796e+12
Italy	11	10964	10794	111850	26661	10.20	106	6.530000e+09	109.0	33.667230	...	2.137439e+12
Spain	12	9428	9330	123336	23964	13.08	115	4.923000e+09	106.0	37.968590	...	1.417355e+12
Iran	13	8896	8819	57470	19125	6.46	72	9.172000e+09	119.0	5.707721	...	4.853309e+11
Australia	14	8831	8725	90765	15606	10.28	107	5.386000e+09	231.0	11.810810	...	1.169431e+12
Brazil	15	8668	8596	60702	14396	7.00	86	1.214900e+10	59.0	69.648030	...	2.295245e+12

15 rows × 25 columns

◄ ▬▬▬▬▬▬▬▬▬▬▬▬▬▬ ►

In [ ]: