*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 energy supply and renewable electricity production (https://hub.coursera-notebooks.org/user/yracpdmrdlfafpmzrbykhd/notebooks/Energy%20Indicators.xls) from the United Nations (http://unstats.un.org/unsd/environment/excel_file_tables/2013/Energy%20Indicators.xls) 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 World Bank (http://data.worldbank.org/indicator/NY.GDP.MKTP.CD). 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 [9]:
```
import pandas as pd
import numpy as np

energy = pd.read_excel('Energy Indicators.xls', sheetname ='Energy')
energy = energy.drop(energy.index[0:16])
energy = energy.drop(energy.index[227:266])
energy = energy.rename(columns={'Unnamed: 0':'del', 'Unnamed: 1':'del2', 'Environ
       'Unnamed: 3':'Energy Supply', 'Unnamed: 4':'Energy Supply per Capita', 'Un
energy = energy.drop('del', axis=1)
energy = energy.drop('del2', axis=1)
energy = energy.replace('...', np.NaN)
energy['Country'] = energy['Country'].str.replace(r"\d+$","")
energy.replace(to_replace=r'^\.+', value=np.nan, inplace=True, regex=True)
energy.replace(to_replace='Republic of Korea', value='South Korea', inplace=True)
###energy.replace(to_replace='Iran ', value='Iran', inplace=True)
energy.replace(to_replace='United States of America', value='United States', inpl
energy.replace(to_replace='China, Hong Kong Special Administrative Region', value
energy.replace(to_replace='United Kingdom of Great Britain and Northern Ireland',
energy['Energy Supply'] = energy['Energy Supply'] * 1000000
energy['Country'] = energy['Country'].str.replace(r"\(.*\)","")
energy['Country'] = energy['Country'].str.strip()
energy = energy.set_index(['Country'])

GDP = pd.read_csv('world_bank.csv', header =4)
GDP.replace(to_replace='Korea, Rep.', value='South Korea', inplace=True)
GDP.replace(to_replace='Iran, Islamic Rep.', value='Iran', inplace=True)
GDP.replace(to_replace='Hong Kong SAR, China', value='Hong Kong', inplace=True)
GDP = GDP.rename(columns={'Country Name':'Country'})
GDP = GDP.set_index(['Country'])

ScimEn = pd.read_excel('scimagojr-3.xlsx')
ScimEn = ScimEn.set_index(['Country'])

def answer_one():
    merge_inner = pd.merge(pd.merge(energy, GDP, how='inner', left_index = True,
    merge_inner = merge_inner.drop(merge_inner.columns[3:52], axis=1)
    merge_inner = merge_inner.ix[merge_inner['Rank']<=15]

    return merge_inner

answer_one()
```

Out[9]:

	Energy Supply	Energy Supply per Capita	% Renewable	2006	2007	2008
Country						
China	1.271910e+11	93.0	19.754910	3.992331e+12	4.559041e+12	4.997775e+12
United States	9.083800e+10	286.0	11.570980	1.479230e+13	1.505540e+13	1.501149e+13

	Energy Supply	Energy Supply per Capita	% Renewable	2006	2007	2008
Country						
Japan	1.898400e+10	149.0	10.232820	5.496542e+12	5.617036e+12	5.558527e+12
United Kingdom	7.920000e+09	124.0	10.600470	2.419631e+12	2.482203e+12	2.470614e+12
Russian Federation	3.070900e+10	214.0	17.288680	1.385793e+12	1.504071e+12	1.583004e+12
Canada	1.043100e+10	296.0	61.945430	1.564469e+12	1.596740e+12	1.612713e+12
Germany	1.326100e+10	165.0	17.901530	3.332891e+12	3.441561e+12	3.478809e+12
India	3.319500e+10	26.0	14.969080	1.265894e+12	1.374865e+12	1.428361e+12
France	1.059700e+10	166.0	17.020280	2.607840e+12	2.669424e+12	2.674637e+12
South Korea	1.100700e+10	221.0	2.279353	9.410199e+11	9.924316e+11	1.020510e+12
Italy	6.530000e+09	109.0	33.667230	2.202170e+12	2.234627e+12	2.211154e+12
Spain	4.923000e+09	106.0	37.968590	1.414823e+12	1.468146e+12	1.484530e+12
Iran	9.172000e+09	119.0	5.707721	3.895523e+11	4.250646e+11	4.289909e+11
Australia	5.386000e+09	231.0	11.810810	1.021939e+12	1.060340e+12	1.099644e+12
Brazil	1.214900e+10	59.0	69.648030	1.845080e+12	1.957118e+12	2.056809e+12

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 [1]:
```
%%HTML
<svg width="800" height="300">
  <circle cx="150" cy="180" r="80" fill-opacity="0.2" stroke="black" stroke-width
  <circle cx="200" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width
  <circle cx="100" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width
  <line x1="150" y1="125" x2="300" y2="150" stroke="black" stroke-width="2" fill=
  <text  x="300" y="165" font-family="Verdana" font-size="35">Everything but this
</svg>
```

Everything but this!

In [3]:
```
def answer_two():
    merge_inner = pd.merge(pd.merge(energy, GDP, how='inner', left_index = True,
    merge_outer = pd.merge(pd.merge(energy, GDP, how='outer', left_index = True,
    len_i = len(merge_inner.index)
    len_o = len(merge_outer.index)
    return len_o - len_i
answer_two()
```

Out[3]: 156

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 [4]:
```
def answer_three():
    Top15 = answer_one()
    avgGDP = Top15[['2006','2007','2008','2009','2010','2011','2012','2013','2014
    avgGDP.sort()
    return avgGDP
answer_three()
```

Out[4]:
```
Country
Iran                 4.441558e+11
South Korea          1.106715e+12
Australia            1.164043e+12
Spain                1.418078e+12
Russian Federation   1.565459e+12
Canada               1.660647e+12
India                1.769297e+12
Italy                2.120175e+12
Brazil               2.189794e+12
United Kingdom       2.487907e+12
France               2.681725e+12
Germany              3.493025e+12
Japan                5.542208e+12
China                6.348609e+12
United States        1.536434e+13
dtype: float64
```

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 [5]:
```
def answer_four():
    Top15 = answer_one()
    avgGDP = answer_three()
    min_GDP = Top15.loc[:,['2006','2007','2008','2009','2010','2011','2012','2013
    max_GDP = Top15.loc[:,['2006','2007','2008','2009','2010','2011','2012','2013
    min_GDP_six = min_GDP.loc['Canada']
    max_GDP_six = max_GDP.loc['Canada']
    a_4 = max_GDP_six - min_GDP_six
    return a_4
answer_four()
```

Out[5]:
```
228139911279.36011
```

Question 5 (6.6%)

What is the mean Energy Supply per Capita?

This function should return a single number.

In [7]:
```
def answer_five():
    Top15 = answer_one()
    a_5 = Top15['Energy Supply per Capita'].mean()
    #Per_capita = Top15.loc[:,['Energy Supply per Capita']].mean()
    #Per_capita = Per_capita.round(decimals =4)
    #a_5 = Per_capita['Energy Supply per Capita']
    #a_5 = round(a_5, 1)
    return print(a_5) #Per_capita.iloc[0] #a_5 #Per_capita.iloc[0]
answer_five()
```

```
157.6
```

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 [410]:
```
def answer_six():
    Top15 = answer_one()
    re_per = Top15.loc[:,['% Renewable']]
    re_per = re_per.reset_index()
    re_sort = re_per.sort(columns ='% Renewable')
    re_sort = re_sort.iloc[[14]]
    tupl = tuple(re_sort.loc[14])
    return tupl
answer_six()
```

Out[410]:
```
('Brazil', 69.648030000000006)
```

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 [10]:
```
def answer_seven():
    Top15 = answer_one()
    Top15['ratio'] = Top15['Self-citations']/Top15['Citations']
    ratio = Top15.loc[:,['ratio']]
    ratio = ratio.reset_index()
    ratio = ratio.sort(columns= 'ratio', ascending = 0 )
    tupp = tuple(ratio.loc[0])
    return tupp
answer_seven()
```

Out[10]:
```
('China', 0.68931261793894216)
```

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 [8]:
```
def answer_eight():
    Top15 = answer_one()
    Top15['popu'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
    popu = Top15.loc[:,['popu']]
    popu = popu.reset_index()
    popu = popu.sort(columns= 'popu')
    popu = popu.sort(columns= 'popu', ascending = 0)
    popu = popu.iloc[2]['Country']
    return popu
answer_eight()
```

Out[8]:
```
'United States'
```

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 [456]:
```
def answer_nine():
    Top15 = answer_one()
    Top15['popu'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
    Top15['Citable documents per person'] = Top15['Citable documents']/Top15['pop
    corr = Top15.loc[:,['Citable documents per person', 'Energy Supply per Capita
    corr = corr.corr()
    return corr.iloc[0]['Energy Supply per Capita']
answer_nine()
```

Out[456]: 0.79400104354429435

In [457]:
```
#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['PopEs
    # Top15.plot(x='Citable docs per Capita', y='Energy Supply per Capita', kind=
```

In [458]:
```
#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 [510]:
```
def answer_ten():
    Top15 = answer_one()
    re_median = Top15.loc[:,['% Renewable']].median()
    Top15['HighRenew'] = np.where(Top15['% Renewable'] >= 17.02028, '1', '0')
    HighRenew = Top15.iloc[0:15]['HighRenew']
    return HighRenew
answer_ten()
```

Out[510]:
```
Country
China                1
United States        0
Japan                0
United Kingdom       0
Russian Federation   1
Canada               1
Germany              1
India                0
France               1
South Korea          0
Italy                1
Spain                1
Iran                 0
Australia            0
Brazil               1
Name: HighRenew, dtype: object
```

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 [487]:
```
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'}
    Top15['popu'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
    Continent = Top15.groupby(ContinentDict).popu.agg(['size', 'sum', 'mean', 'st
    return Continent
answer_eleven()
```

Out[487]:

	size	sum	mean	std
Asia	5	2.898666e+09	5.797333e+08	6.790979e+08
Australia	1	2.331602e+07	2.331602e+07	NaN
Europe	6	4.579297e+08	7.632161e+07	3.464767e+07
North America	2	3.528552e+08	1.764276e+08	1.996696e+08
South America	1	2.059153e+08	2.059153e+08	NaN

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 [4]:
```
def answer_twelve():
    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'}
    out, bins = pd.cut(Top15['% Renewable'].values, bins = 5, retbins = True)
    con = Top15.index.to_series().map(ContinentDict).values
    Top15.reset_index(inplace=True)
    Top15.index = pd.MultiIndex.from_arrays([con, out])
    bins_12 = Top15.iloc[1:15]['Country']
    Top15 = Top15.reset_index()
    Top15 = Top15[['level_0','level_1','Country']]
    Top15 = Top15.rename(columns={'level_0': 'Continents', 'level_1': '% Renewabl
    Top15_count = Top15.groupby(['Continents','% Renewable'] ).Country.count()
    Top15_count =Top15_count.reset_index() \
                    .sort_values(
                    ['Continents', 'Country'],
                    ascending=[True, False]
                    ).set_index(['Continents', '% Renewable']).Country
    #Top15 = Top15.set_index(['level_0', 'level_1'])    #remove this maybe
    #Top15 = Top15.groupby(by=['Continents', '% Renewable']).sum()  #setting inde
    #Top15.sort_values('Country')
    #Top15 = Top15.groupby('level_1').sum()
    #Top15['Continent'] = Top15.Country.replace(ContinentDict)
    #Top15.groupby(['Continent', '% Renewable']).apply(lambda x: x.Country.tolist
    ##Top15['times'] = ['1','4','1','3','1','2','1','1','1',] ------------ w.e
    ##bins12 = Top15['times']
    return Top15_count #bins_
answer_twelve()
```

Out[4]:
```
Continents     % Renewable
Asia           (2.212, 15.753]     4
               (15.753, 29.227]    1
Australia      (2.212, 15.753]     1
Europe         (15.753, 29.227]    3
               (29.227, 42.701]    2
               (2.212, 15.753]     1
North America  (2.212, 15.753]     1
               (56.174, 69.648]    1
South America  (56.174, 69.648]    1
Name: Country, dtype: int64
```


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 [10]:
```
def answer_thirteen():
    Top15 = answer_one()
    Top15['Population Estimate'] = Top15['Energy Supply']/Top15['Energy Supply pe
    #Top15['Population Estimate'] = Top15['Population Estimate'].map('{:,.4f}'.fo
    Top15['Population Estimate'] = Top15['Population Estimate'].map(lambda x: "{:
    PopEst = Top15.iloc[0:15]['Population Estimate']
    PopEst = PopEst.sort_index()
    return PopEst
answer_thirteen()
```

Out[10]:
```
Country
Australia              23,316,017.316017315
Brazil                 205,915,254.23728815
Canada                  35,239,864.86486486
China                1,367,645,161.2903225
France                  63,837,349.39759036
Germany                 80,369,696.96969697
India                1,276,730,769.2307692
Iran                    77,075,630.25210084
Italy                  59,908,256.880733944
Japan                  127,409,395.97315437
Russian Federation            143,500,000.0
South Korea            49,805,429.864253394
Spain                    46,443,396.2264151
United Kingdom         63,870,967.741935484
United States          317,615,384.61538464
Name: Population Estimate, dtype: object
```

Optional

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

In [ ]:
```
#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','#377eb
                       #'#4daf4a','#e41a1c','#4daf4a','#4daf4a','#e41a1c','#dede0
                    #xticks=range(1,16), s=6*Top15['2014']/10**10, alpha=.75, fig

    #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 und
#This is a bubble chart showing % Renewable vs. Rank. The size of the bubble corr
#2014 GDP, and the color corresponds to the continent.")
```

In [ ]:
```
#plot_optional() # Be sure to comment out plot_optional() before submitting the a
```