*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.

In [1]:
```
import numpy as np
import pandas as pd
```

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 (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.

计算过程

Energy

In [2]:
```
# 读取文件时:
# 1) Exclude the footer and header information from the datafile.
energy = pd.read_excel('Energy Indicators.xls', header = None, skipfooter = 1)
energy.head(2)
```

Out[2]:

	0	1	2	3	4	5
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	Environmental Indicators: Energy	NaN	NaN	NaN

In [3]:
```
# 2) 数据内容从第 19 行开始，到第 245 行结束。因为程序 index 从 0 开始，所以是 19-1=18 开始。
energy = energy.iloc[18:245, :]
energy.head()
```

Out[3]:

	0	1	2	3	4	5
18	NaN	Afghanistan	Afghanistan	321	10	78.6693
19	NaN	Albania	Albania	102	35	100
20	NaN	Algeria	Algeria	1959	51	0.55101
21	NaN	American Samoa	American Samoa	...	...	0.641026
22	NaN	Andorra	Andorra	9	121	88.6957

In [4]:
```
# 3) Get rid of the first two unnecessary columns
energy = energy.iloc[:,2:]
energy.head(2)
```

Out[4]:

	2	3	4	5
18	Afghanistan	321	10	78.6693
19	Albania	102	35	100

In [5]:
```
# 4) change the column labels to: ['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']
energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']
energy.head(2)
```

Out[5]:

	Country	Energy Supply	Energy Supply per Capita	% Renewable
18	Afghanistan	321	10	78.6693
19	Albania	102	35	100

In [6]:
```
# 5) Use np.NaN to replace the missing data "..." for all countries (必须在 step 6 之前完成，不然 '...' 会复制 1M 遍)
energy.replace('...', np.nan, inplace = True)
energy.head()
```

Out[6]:

	Country	Energy Supply	Energy Supply per Capita	% Renewable
18	Afghanistan	321.0	10.0	78.669280
19	Albania	102.0	35.0	100.000000
20	Algeria	1959.0	51.0	0.551010
21	American Samoa	NaN	NaN	0.641026
22	Andorra	9.0	121.0	88.695650

In [7]:
```
# 6) Convert Energy Supply to gigajoules (there are 1,000,000 gigajoules in a petajoule)
energy['Energy Supply'] = energy['Energy Supply'] * 1000000
energy.head()
```

Out[7]:

	Country	Energy Supply	Energy Supply per Capita	% Renewable
18	Afghanistan	3.210000e+08	10.0	78.669280
19	Albania	1.020000e+08	35.0	100.000000
20	Algeria	1.959000e+09	51.0	0.551010
21	American Samoa	NaN	NaN	0.641026
22	Andorra	9.000000e+06	121.0	88.695650

`str.find('xx')` 如果没有找到 'xx'，则会返回 index = -1。如果找到，则返回正数 index。

In [8]:
```
#for data in energy['Country']:
    # 7) Remove numbers in the countries name -滤遍每个字节，only collect no
t digit char and concate them finally
    #ctry1 = ''.join([i for i in data if not i.isdigit()])
    #print(ctry1)
    # 8) Remove parenthesis in the countries name - 去掉从第一个 '(' 之后的所
有内容
    #if ctry1.find('(') > -1:
        #ctry1 = ctry1[:ctry1.find('(')]
    #print(ctry1)
    # 9) Remove leading and trailing whitespaces
    #ctry3 = ctry1.strip()
    #print(ctry3)
```

In [9]:
```
# 将上述计算写成 Function
def remove_num_par(data):
    # 7) Remove numbers in the countries name -滤遍每个字节，only collect no
t digit char and concate them finally
    ctry = ''.join([i for i in data if not i.isdigit()])
    # 8) Remove parenthesis in the countries name - 去掉从第一个 '(' 之后的所
有内容
    if ctry.find('(') > -1:
        ctry = ctry[:ctry.find('(')]
    # 9) Remove leading and trailing whitespaces
    return ctry.strip()
```

In [10]:
```
energy['Country'] = energy['Country'].apply(remove_num_par)
energy['Country']
```

Out[10]:
```
18                    Afghanistan
19                        Albania
20                        Algeria
21                 American Samoa
22                        Andorra
                   ...
240                       Viet Nam
241    Wallis and Futuna Islands
242                         Yemen
243                        Zambia
244                      Zimbabwe
Name: Country, Length: 227, dtype: object
```

In [11]:
```
# 10) 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"
rename_list = {'Republic of Korea': 'South Korea',
               'United States of America': 'United States',
               'United Kingdom of Great Britain and Northern Ireland': 'Uni
ted Kingdom',
               'China, Hong Kong Special Administrative Region': 'Hong Kon
g'}

energy.replace({'Country': rename_list}, inplace = True)
energy.loc[61, 'Country']
```

Out[11]: 'Hong Kong'

World Bank

In [12]:
```
# 1) 从第 5 行开始，读取数据。因为程序从 index 0 开始，所以 skiprows = 4
GDP = pd.read_csv('world_bank.csv', skiprows = 4)
GDP.head(2)
```

Out[12]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	
0	Aruba	ABW	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	...	
1	Andorra	AND	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	...	4.01819

2 rows × 60 columns

In [13]:
```
# 2) Rename the column "Country Name"
GDP.rename(columns = {'Country Name': 'Country'}, inplace = True)
GDP.head(2)
```

Out[13]:

	Country	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	
0	Aruba	ABW	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	...	
1	Andorra	AND	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	...	4.01819

2 rows × 60 columns

In [14]:
```
# 3) rename the following list of countries:
# "Korea, Rep.": "South Korea",
# "Iran, Islamic Rep.": "Iran",
# "Hong Kong SAR, China": "Hong Kong"
rename_list = {"Korea, Rep.": "South Korea",
               "Iran, Islamic Rep.": "Iran",
               "Hong Kong SAR, China": "Hong Kong"}

GDP.replace({'Country': rename_list}, inplace = True)
```

Sciamgo Journal

In [15]:
```
# load the Sciamgo Journal and Country Rank data for Energy Engineering and
Power Technology,
# it ranks countries based on their journal contributions in the aforementi
oned area.
# Call this DataFrame ScimEn.
ScimEn = pd.read_excel('scimagojr-3.xlsx')
ScimEn.head(2)
```

Out[15]:

	Rank	Country	Documents	Citable documents	Citations	Self-citations	Citations per document	H index
0	1	China	127050	126767	597237	411683	4.7	138
1	2	United States	96661	94747	792274	265436	8.2	230

Join the three datasets: GDP, Energy, and ScimEn into a new dataset

In [16]:
```
# 1) Join the three datasets: GDP, Energy, and ScimEn into a new dataset
GDP_Energy = pd.merge(GDP, energy, on = 'Country')
df = pd.merge(GDP_Energy, ScimEn, on = 'Country')
df.head(2)
```

Out[16]:

	Country	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	
0	Andorra	AND	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	...	9
1	Afghanistan	AFG	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	...	321

2 rows × 70 columns

In [17]:
```
# 2) The index of this DataFrame should be the name of the country
df.set_index('Country', inplace = True)
df.head(2)
```

Out[17]:

	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	1966	...
Country											
Andorra	AND	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Afghanistan	AFG	GDP at market prices (constant 2010 US$)	NY.GDP.MKTP.KD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

2 rows × 69 columns

In [18]:
```
# 3) Rebuild the columns
col = ['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']

df = df[col]
df.head(2)
```

Out[18]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	E S p C
Andorra	168	2	2	13	0	6.5	1	9000000.0	
Afghanistan	163	3	3	0	0	0.0	0	321000000.0	

In [19]:
```
# for Question 2
df_len = len(df)
```

In [20]:
```
# 4) only the top 15 countries by Scimagojr 'Rank'
df = df[df['Rank'].isin([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])]
df.sort_values(by=['Rank'], inplace=True)
df
```

Out[20]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	En Su pe Ca
China	1	127050	126767	597237	411683	4.70	138	1.271910e+11	
United States	2	96661	94747	792274	265436	8.20	230	9.083800e+10	2
Japan	3	30504	30287	223024	61554	7.31	134	1.898400e+10	1
United Kingdom	4	20944	20357	206091	37874	9.84	139	7.920000e+09	1
Russian Federation	5	18534	18301	34266	12422	1.85	57	3.070900e+10	2
Canada	6	17899	17620	215003	40930	12.01	149	1.043100e+10	2
Germany	7	17027	16831	140566	27426	8.26	126	1.326100e+10	1
India	8	15005	14841	128763	37209	8.58	115	3.319500e+10	
France	9	13153	12973	130632	28601	9.93	114	1.059700e+10	1
South Korea	10	11983	11923	114675	22595	9.57	104	1.100700e+10	2
Italy	11	10964	10794	111850	26661	10.20	106	6.530000e+09	1
Spain	12	9428	9330	123336	23964	13.08	115	4.923000e+09	1
Iran	13	8896	8819	57470	19125	6.46	72	9.172000e+09	
Australia	14	8831	8725	90765	15606	10.28	107	5.386000e+09	2
Brazil	15	8668	8596	60702	14396	7.00	86	1.214900e+10	

YC 答案

In [21]:
```
def answer_one():
    # ============Energy Begin==============
    # 1) Exclude the footer and header information from the datafile.
    energy = pd.read_excel('Energy Indicators.xls', header = None, skipfoot
er = 1)
    # 2) 数据内容从第 19 行开始，到第 245 行结束。因为程序 index 从 0 开始，所以
是 19-1=18 开始。
    energy = energy.iloc[18:245, :]
    # 3) Get rid of the first two unnecessary columns
    energy = energy.iloc[:,2:]
    # 4) change the column labels to: ['Country', 'Energy Supply', 'Energy
 Supply per Capita', '% Renewable']
    energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capit
a', '% Renewable']
    # 5) Use np.NaN to replace the missing data "..." for all countries (必
须在 step 6 之前完成，不然 '...' 会复制 1M 遍)
    energy.replace('...', np.nan, inplace = True)
    # 6) Convert Energy Supply to gigajoules (there are 1,000,000 gigajoule
s in a petajoule)
    energy['Energy Supply'] = energy['Energy Supply'] * 1000000

    def remove_num_par(data):
        # 7) Remove numbers in the countries name -滤遍每个字节，only collec
t not digit char and concate them finally
        ctry = ''.join([i for i in data if not i.isdigit()])
        # 8) Remove parenthesis in the countries name - 去掉从第一个 '(' 之后
的所有内容
        if ctry.find('(') > -1:
            ctry = ctry[:ctry.find('(')]
        # 9) Remove leading and trailing whitespaces
        return ctry.strip()

    energy['Country'] = energy['Country'].apply(remove_num_par)
    # 10) Rename the following list of countries (for use in later question
s):
    # "Republic of Korea": "South Korea",
    # "United States of America": "United States",
    # "United Kingdom of Great Britain and Northern Ireland": "United Kingd
om",
    # "China, Hong Kong Special Administrative Region": "Hong Kong"
    rename_list = {'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': rename_list}, inplace = True)
    # ============Energy End==============

    # ============World Bank Begin=============
    # 1) 从第 5 行开始，读取数据。因为程序从 index 0 开始，所以 skiprows = 4
    GDP = pd.read_csv('world_bank.csv', skiprows = 4)
    # 2) Rename the column "Country Name"
    GDP.rename(columns = {'Country Name': 'Country'}, inplace = True)
    # 3) rename the following list of countries:
    # "Korea, Rep.": "South Korea",
```

```
    # "Iran, Islamic Rep.": "Iran",
    # "Hong Kong SAR, China": "Hong Kong"
    rename_list = {"Korea, Rep.": "South Korea",
                   "Iran, Islamic Rep.": "Iran",
                   "Hong Kong SAR, China": "Hong Kong"}

    GDP.replace({'Country': rename_list}, inplace = True)
    # ============World Bank End=============

    # ============Sciamgo Journal Begin=============
    # load the Sciamgo Journal and Country Rank data for Energy Engineering
and Power Technology,
    # it ranks countries based on their journal contributions in the aforem
entioned area.
    # Call this DataFrame ScimEn.
    ScimEn = pd.read_excel('scimagojr-3.xlsx')
    # ============Sciamgo Journal End=============

    # ====Join the three datasets: GDP, Energy, and ScimEn Begin====
    # 1) Join the three datasets: GDP, Energy, and ScimEn into a new datase
t
    GDP_Energy = pd.merge(GDP, energy, on = 'Country')
    df = pd.merge(GDP_Energy, ScimEn, on = 'Country')
    # 2) The index of this DataFrame should be the name of the country
    df.set_index('Country', inplace = True)
    # 3) Rebuild the columns
    col = ['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']

    df = df[col]
    # 4) only the top 15 countries by Scimagojr 'Rank'
    df = df[df['Rank'].isin([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])]
    df.sort_values(by=['Rank'], inplace=True)
    # ====Join the three datasets: GDP, Energy, and ScimEn End====

    return df
```

标准答案

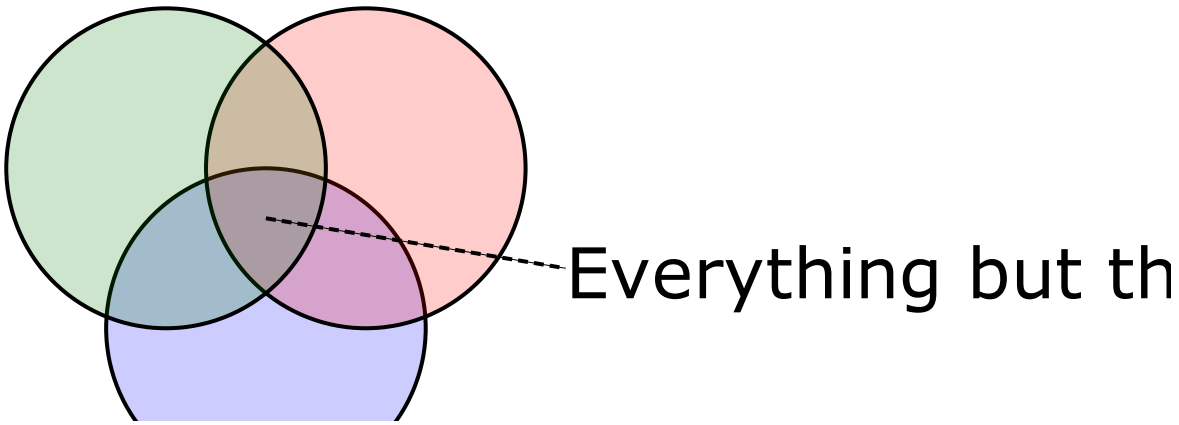
In [23]:
```
#def answer_one():
#    energy = pd.read_excel('Energy Indicators.xls')
#    energy = energy[16:243]
#    energy = energy.drop(energy.columns[[0, 1]], axis=1)
#    energy.rename(columns={'Environmental Indicators: Energy': 'Countr
y','Unnamed: 3':'Energy Supply','Unnamed: 4':'Energy Supply per Capita','Un
named: 5':'% Renewable'}, inplace=True)
#    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)
#    di = {"Republic of Korea": "South Korea",
#    "United States of America": "United States",
#    "United Kingdom of Great Britain and Northern Ireland": "United Kingdo
m",
#    "China, Hong Kong Special Administrative Region": "Hong Kong"}
#    energy.replace({"Country": di},inplace = True)
#    #energy


#    # In[275]:

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


#    # In[113]:

#    ScimEn = pd.read_excel('scimagojr-3.xlsx')
#    df = pd.merge(pd.merge(energy, GDP, on='Country'), ScimEn, on='Countr
y')

#    # We only need 2006-2015 data
#    df.set_index('Country',inplace=True)
#    df = df[['Rank', 'Documents', 'Citable documents', 'Citations', 'Self-
citations', 'Citations per document', 'H index', 'Energy Supply', 'Energy S
upply per Capita', '% Renewable', '2006', '2007', '2008', '2009', '2010',
 '2011', '2012', '2013', '2014', '2015']]
#    df = (df.loc[df['Rank'].isin([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])])
#    df.sort('Rank',inplace=True)


#    return df
```

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 [24]:
```
%%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 bu
t this!</text>
</svg>
```



计算过程

|A ∪ B ∪ C| = |A| + |B| + |C| - |A ∩ B| - |A ∩ C| - |B ∩ C| + |A ∩ B ∩ C|

In [25]:
```
GDP_Scim = pd.merge(GDP, ScimEn, on = 'Country')
```

In [26]:
```
Energy_Scim = pd.merge(energy, ScimEn, on = 'Country')
```

In [27]:
```
# 合并后剩下行数
df_len
```

Out[27]: 162

In [28]:
```
# 3 组 DataFrame 不重复的情况下，总行数
len(energy) + len(GDP) + len(ScimEn) - len(GDP_Energy) - len(Energy_Scim) -
len(GDP_Scim) + len(Energy_Scim)
```

Out[28]: 330

In [29]:
```
# 消失的行数
318 - 162
```

Out[29]: 156

YC 答案

In [30]:
```
def answer_two():
    return 156
```

标准答案

In [31]:
```
def answer_two():
    return 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.

计算过程

In [32]:
```
df = answer_one()
df.head(1)
```

Out[32]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capit
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93

In [33]:
```
# 计算 10 年平均 GDP
last10yr = ['2006','2007','2008','2009','2010','2011','2012','2013','2014',
'2015']
# 默认的 mean() 计算列的平均值
avgGDP = df[last10yr].mean(axis=1)
avgGDP.head(2)
```

Out[33]:
```
Country
China            6.348609e+12
United States    1.536434e+13
dtype: float64
```

In [34]:
```
# average GDP sorted in descending order
avgGDP.sort_values(ascending = False, inplace = True)
avgGDP.columns = ['avgGDP']
avgGDP.head(2)
```

Out[34]:
```
Country
United States    1.536434e+13
China            6.348609e+12
dtype: float64
```

YC 答案

In [35]:
```
def answer_three():
    df = answer_one()
    # 计算 10 年平均 GDP
    last10yr = ['2006','2007','2008','2009','2010','2011','2012','2013','20
14','2015']
    # 默认的 mean() 计算列的平均值
    avgGDP = df[last10yr].mean(axis=1)
    # average GDP sorted in descending order
    avgGDP.sort_values(ascending = False, inplace = True)
    # return a Series named avgGDP
    avgGDP.columns = ['avgGDP']
    return avgGDP
```

标准答案

In [36]:
```
def answer_three():
    Top15 = answer_one()
    avgGDP = Top15[['2006','2007','2008','2009','2010','2011','2012','2013'
,'2014',
                    '2015']].mean(axis=1).sort_values(ascending = False)
    avgGDP.rename('avgGDP',inplace=True)
    return 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.

计算过程

In [37]:
```
df = answer_one()
avgGDP = answer_three()
# the country with the 6th largest average GDP
country_6th = avgGDP.index[5]
df.loc[country_6th, '2015'] - df.loc[country_6th, '2006']
```

Out[37]:
```
246702696075.3999
```

YC 答案

In [38]:
```
def answer_four():
    df = answer_one()
    avgGDP = answer_three()
    # the country with the 6th largest average GDP
    country_6th = avgGDP.index[5]
    return df.loc[country_6th, '2015'] - df.loc[country_6th, '2006']
```

标准答案

In [39]:
```
def answer_four():
    Top15 = answer_one()
    avgGDP = answer_three()
    avgGDP_6th = avgGDP.index[5]

    return Top15.loc[avgGDP_6th,'2015'] - Top15.loc[avgGDP_6th,'2006']
```

Question 5 (6.6%)

What is the mean `Energy Supply per Capita` ?

This function should return a single number.

计算过程

In [40]:
```
df = answer_one()
df.head(2)
```

Out[40]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

In [41]:
```
df['Energy Supply per Capita'].mean()
```

Out[41]: 157.6

YC 答案

In [42]:
```
def answer_five():
    df = answer_one()
    return df['Energy Supply per Capita'].mean()
```

标准答案

In [43]:
```
def answer_five():
    Top15 = answer_one()
    return Top15['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.

计算过程

In [44]:
```
df = answer_one()
df.head(2)
```

Out[44]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

In [45]:
```
max_renew = (df['% Renewable'].idxmax(), df['% Renewable'].max())
max_renew
```

Out[45]: ('Brazil', 69.64803)

YC 答案

In [46]:
```
def answer_six():
    df = answer_one()
    return (df['% Renewable'].idxmax(), df['% Renewable'].max())
```

标准答案

In [47]:
```
def answer_six():
    Top15 = answer_one()
    return (Top15['% Renewable'].idxmax(1),Top15.loc[Top15['% Renewable'].i
dxmax(1),'% Renewable'])
```

In [48]:
```
answer_six()
```

Out[48]: ('Brazil', 69.64803)

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.

计算过程

In [49]:
```
df = answer_one()
df.head(2)
```

Out[49]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

In [50]:
```
# Create a new column that is the ratio of Self-Citations to Total Citation
s
df['Citations_ratio'] = df['Self-citations'] / df['Citations']
df.head(2)
```

Out[50]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

2 rows × 21 columns

In [51]:
```
# The maximum value for the new column, and what country has the highest ra
tio
max_citations_rt = (df['Citations_ratio'].idxmax(), df['Citations_ratio'].m
ax())
max_citations_rt
```

Out[51]:
```
('China', 0.6893126179389422)
```

YC 答案

In [52]:
```
def answer_seven():
    df = answer_one()
    # Create a new column that is the ratio of Self-Citations to Total Cita
tions
    df['Citations_ratio'] = df['Self-citations'] / df['Citations']
    return (df['Citations_ratio'].idxmax(), df['Citations_ratio'].max())
```

标准答案

In [53]:
```
def answer_seven():
    Top15 = answer_one()
    Top15['Citations_ratio'] = Top15['Self-citations'] / Top15['Citations']
    return (Top15['Citations_ratio'].idxmax(1),
            Top15.loc[Top15['Citations_ratio'].idxmax(1),'Citations_ratio'
])
```

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.

计算过程

In [54]:
```
df = answer_one()
df.head(2)
```

Out[54]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

In [55]:
```
# Create a column that estimates the population using Energy Supply and Ene
rgy Supply per capita
df['Est Population'] = df['Energy Supply'] / df['Energy Supply per Capita']
pop = df['Est Population'].sort_values(ascending = False)
pop.head()
```

Out[55]:
```
Country
China                 1.367645e+09
India                 1.276731e+09
United States         3.176154e+08
Brazil                2.059153e+08
Russian Federation    1.435000e+08
Name: Est Population, dtype: float64
```

In [56]:
```
# The third most populous country
pop.index[2]
```

Out[56]:
```
'United States'
```

YC 答案

In [57]:
```
def answer_eight():
    df = answer_one()
    # Create a column that estimates the population using Energy Supply and
Energy Supply per capita
    df['Est Population'] = df['Energy Supply'] / df['Energy Supply per Capi
ta']
    pop = df['Est Population'].sort_values(ascending = False)
    # The third most populous country
    return pop.index[2]
```

In [58]:
```
def answer_eight():
    Top15 = answer_one()
    Top15['Est Population'] = Top15['Energy Supply'] / Top15['Energy Supply
per Capita']
    return Top15['Est Population'].nlargest(3).index[2]
```

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)

计算过程

In [59]:
```
df = answer_one()
df.head(2)
```

Out[59]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

In [60]:
```
# Estimate the population
df['Est Population'] = df['Energy Supply'] / df['Energy Supply per Capita']
# Create a column that estimates the number of citable documents per person
df['Citable doc per Person'] = df['Citable documents'] / df['Est Populatio
n']
df.head(2)
```

Out[60]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

2 rows × 22 columns

In [61]:
```
# the correlation between the number of citable documents per capita and th
e energy supply per capita
# Use the .corr() method, (Pearson's correlation)
df[['Citable doc per Person', 'Energy Supply per Capita']].corr(method = 'p
earson').iloc[0,1]
```

Out[61]: 0.7940010435442943

YC 答案

In [62]:
```
def answer_nine():
    df = answer_one()
    # Estimate the population
    df['Est Population'] = df['Energy Supply'] / df['Energy Supply per Capi
ta']
    # Create a column that estimates the number of citable documents per pe
rson
    df['Citable doc per Person'] = df['Citable documents'] / df['Est Popula
tion']
    # the correlation between the number of citable documents per capita an
d the energy supply per capita
    # Use the .corr() method, (Pearson's correlation)
    return df[['Citable doc per Person', 'Energy Supply per Capita']].corr(
method = 'pearson').iloc[0,1]
```

In [63]:
```
def answer_nine():
    Top15 = answer_one()
    Top15['Est Population'] = Top15['Energy Supply'] / Top15['Energy Supply
per Capita']
    Top15['Citable Doc per Person'] = Top15['Citable documents'] / Top15['E
st Population']
    return Top15[['Citable Doc per Person','Energy Supply per Capita']].cor
r(method='pearson').iloc[0,1]
```

In [64]:
```
def plot9():
    import matplotlib as plt
    %matplotlib inline

    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Cap
ita']
    Top15['Citable docs per Capita'] = Top15['Citable documents'] / Top15[
'PopEst']
    Top15.plot(x='Citable docs per Capita', y='Energy Supply per Capita', k
ind='scatter', xlim=[0, 0.0006])
```

In [65]:
```
#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.*

计算过程

In [66]:
```
df = answer_one()
df.head(2)
```

Out[66]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

In [67]:
```
# the median of % Renewable for all countries in the top 15
renewable_med = df['% Renewable'].mean()
```

In [68]:
```
# 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.
df['HighRenew'] = (df['% Renewable'] >= renewable_med) * 1
df['HighRenew']
```

Out[68]:
```
Country
China                  0
United States          0
Japan                  0
United Kingdom         0
Russian Federation     0
Canada                 1
Germany                0
India                  0
France                 0
South Korea            0
Italy                  1
Spain                  1
Iran                   0
Australia              0
Brazil                 1
Name: HighRenew, dtype: int32
```

YC 答案

In [69]:
```
def answer_ten():
    df = answer_one()
    # the median of % Renewable for all countries in the top 15
    renewable_med = df['% Renewable'].mean()
    # 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.
    return (df['% Renewable'] >= renewable_med) * 1
```

标准答案

In [70]:
```
def answer_ten():
    Top15 = answer_one()
    renew_median = Top15['% Renewable'].median()
    Top15['HighRenew'] = (Top15['% Renewable']>=renew_median)
    return Top15['HighRenew'].rename('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']`

计算过程

In [71]:
```
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'}

df = answer_one()
df['Continent'] = None
df.head(2)
```

Out[71]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

2 rows × 21 columns

In [72]:
```
for country in df.index:
    df.loc[country, 'Continent'] = ContinentDict[country]
df['Continent'].head(2)
```

Out[72]:
```
Country
China                         Asia
United States    North America
Name: Continent, dtype: object
```

In [73]:
```
# the estimated population of each country
df['Esti Population'] = df['Energy Supply'] / df['Energy Supply per Capita']
df.head(2)
```

Out[73]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

2 rows × 22 columns

Use `groupby` :

In [74]:
```
# Use groupby
df.groupby('Continent')['Esti Population'].agg({'size': np.size,
                                                'sum': np.sum, 'mean': np.mean,
                                                'std': np.std})
```

```
C:\Users\XZV838\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: using a dict on a Series for aggregation
is deprecated and will be removed in a future version. Use
named aggregation instead.

    >>> grouper.agg(name_1=func_1, name_2=func_2)

  after removing the cwd from sys.path.
```

Out[74]:

Continent	size	sum	mean	std
Asia	5.0	2.898666e+09	5.797333e+08	6.790979e+08
Australia	1.0	2.331602e+07	2.331602e+07	NaN
Europe	6.0	4.579297e+08	7.632161e+07	3.464767e+07
North America	2.0	3.528552e+08	1.764276e+08	1.996696e+08
South America	1.0	2.059153e+08	2.059153e+08	NaN

Use `pivot_table` :

In [75]:
```
df.pivot_table(index = 'Continent', values = ['Esti Population'],
               aggfunc = {np.size, np.sum, np.mean, np.std})
```

Out[75]:

	Esti Population			
	mean	size	std	sum
Continent				
Asia	5.797333e+08	5.0	6.790979e+08	2.898666e+09
Australia	2.331602e+07	1.0	NaN	2.331602e+07
Europe	7.632161e+07	6.0	3.464767e+07	4.579297e+08
North America	1.764276e+08	2.0	1.996696e+08	3.528552e+08
South America	2.059153e+08	1.0	NaN	2.059153e+08

YC 答案

In [76]:
```
def answer_eleven():
    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'}

    df = answer_one()
    df['Continent'] = None

    for country in df.index:
        df.loc[country, 'Continent'] = ContinentDict[country]

    # the estimated population of each country
    df['Esti Population'] = df['Energy Supply'] / df['Energy Supply per Cap
ita']

    # Use groupby
    return df.groupby('Continent')['Esti Population'].agg({'size': np.size,
                                                            'sum': np.sum,
'mean': np.mean,
                                                            'std': np.std})
```

标准答案

In [77]:
```
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['Continent'] = None
    for country in Top15.index:
        Top15.loc[country,'Continent'] = ContinentDict[country]
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Cap
ita']
    #Top15.reset_index(inplace=True)
    #Top15.set_index('Continent',inplace=True)

    return Top15.groupby('Continent')['PopEst'].aggregate({'size':np.size,
'sum':np.sum,'mean':np.mean,'std':np.std})
```

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.*

计算过程

In [78]:
```
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'}

df = answer_one()
df['Continent'] = None
df.head(2)
```

Out[78]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

2 rows × 21 columns

In [79]:
```
for country in df.index:
    df.loc[country, 'Continent'] = ContinentDict[country]
```

In [80]:
```
df.reset_index(inplace=True)
df.set_index('Continent',inplace=True)
df['Bin Renewable'] = pd.cut(df['% Renewable'],5)
```

In [81]:
```
df.groupby(['Continent','Bin Renewable']).size()
```

Out[81]:
```
Continent      Bin Renewable
Asia           (2.212, 15.753]     4
               (15.753, 29.227]    1
Australia      (2.212, 15.753]     1
Europe         (2.212, 15.753]     1
               (15.753, 29.227]    3
               (29.227, 42.701]    2
North America  (2.212, 15.753]     1
               (56.174, 69.648]    1
South America  (56.174, 69.648]    1
dtype: int64
```

YC 答案

In [82]:
```
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'}

    df = answer_one()
    df['Continent'] = None

    for country in df.index:
        df.loc[country, 'Continent'] = ContinentDict[country]
```

标准答案

In [83]:
```
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'}

    Top15['Continent'] = None
    for country in Top15.index:
        Top15.loc[country,'Continent'] = ContinentDict[country]
    Top15.reset_index(inplace=True)
    Top15.set_index('Continent',inplace=True)
    Top15['Bin Renewable'] = pd.cut(Top15['% Renewable'],5)

    return Top15.groupby(['Continent','Bin Renewable']).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.

计算过程

In [84]:
```
df = answer_one()
df['PopEst'] = df['Energy Supply'] / df['Energy Supply per Capita']
df.head(2)
```

Out[84]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Ener Supp per Capi
China	1	127050	126767	597237	411683	4.7	138	1.271910e+11	93
United States	2	96661	94747	792274	265436	8.2	230	9.083800e+10	286

2 rows × 21 columns

In [85]:
```
df['PopEst'].apply(lambda x: '{:,}'.format(x))
```

Out[85]:
```
Country
China                1,367,645,161.2903225
United States          317,615,384.61538464
Japan                  127,409,395.97315437
United Kingdom          63,870,967.741935484
Russian Federation             143,500,000.0
Canada                  35,239,864.86486486
Germany                 80,369,696.96969697
India                1,276,730,769.2307692
France                  63,837,349.39759036
South Korea             49,805,429.864253394
Italy                   59,908,256.880733944
Spain                     46,443,396.2264151
Iran                    77,075,630.25210084
Australia               23,316,017.316017315
Brazil                  205,915,254.23728815
Name: PopEst, dtype: object
```

YC 答案

In [86]:
```
def answer_thirteen():
    df = answer_one()
    df['PopEst'] = df['Energy Supply'] / df['Energy Supply per Capita']
    return df['PopEst'].apply(lambda x: '{:,}'.format(x))
```

标准答案

In [87]:
```
def answer_thirteen():
    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Cap
ita']

    return Top15['PopEst'].apply(lambda x: '{:,}'.format(x))
```

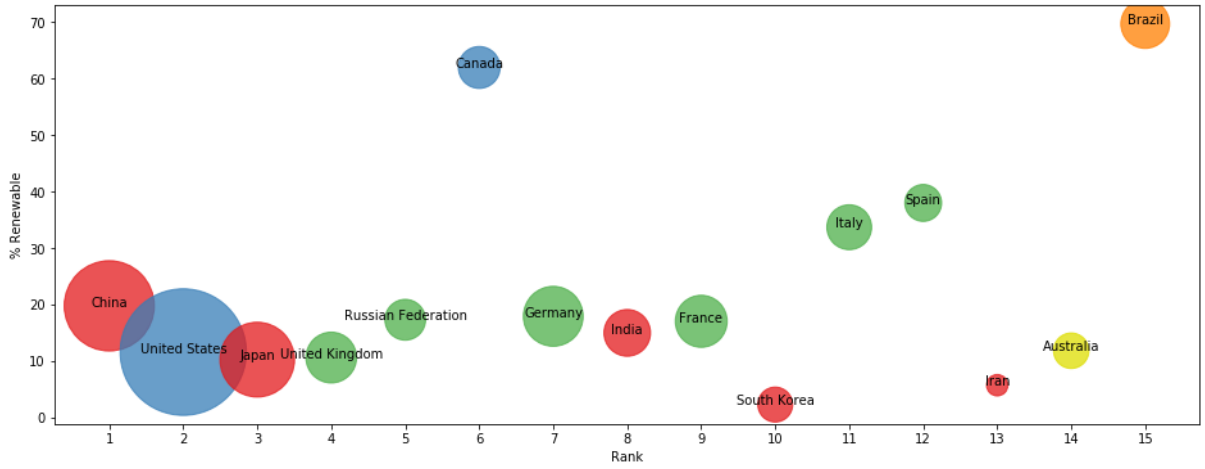
Optional

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

In [88]:
```
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='c
enter')

    print("This is an example of a visualization that can be created to hel
p 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 [90]:
```
#plot_optional() # Be sure to comment out plot_optional() before submitting the assignment!
```

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 [ ]: