Assignment 3

June 21, 2020

---

*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 course resource.*

---

1 Assignment 3 - More Pandas

This assignment requires more individual learning then the last one did - you are encouraged to check out the pandas documentation to find functions or methods you might not have used yet, or ask questions on Stack Overflow 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.

1.0.1 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 from the United Nations 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. 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 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 [2]: def answer_one():
            import pandas as pd
            import numpy as np
            energy = pd.read_excel('Energy Indicators.xls', skip_footer=38, skiprows=17, parse_c
            col_names = ['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']
            energy.columns = col_names
            energy.loc[energy['Energy Supply'] == '...'] = np.NaN
            energy[['Energy Supply', 'Energy Supply per Capita']] = energy[['Energy Supply', 'En
            energy['Energy Supply'] = energy['Energy Supply']*10**6
            energy['Country'] = energy['Country'].str.replace(r" \(.*\)","")
            energy['Country'] = energy['Country'].str.replace(r"([0-9]+)$","")
            replace_dict={"Republic of Korea": "South Korea",
                          "United States of America": "United States",
                          "United Kingdom of Great Britain and Northern Ireland": "United Kingdo
                          "China, Hong Kong Special Administrative Region": "Hong Kong"}
            energy['Country'].replace(to_replace=replace_dict, inplace=True)
            energy.reset_index()
            energy = energy.set_index('Country')

            GDP = pd.read_csv('world_bank.csv', skiprows=4)
            replace_dict = {"Korea, Rep.": "South Korea",
                            "Iran, Islamic Rep.": "Iran",
                            "Hong Kong SAR, China": "Hong Kong"
                           }
            GDP['Country Name'].replace(to_replace=replace_dict, inplace=True)
            years_to_keep = np.arange(2006, 2016).astype(str)
            GDP = GDP[np.append(['Country Name'],years_to_keep)]
            GDP.reset_index()
            GDP = GDP.rename(columns={'Country Name': 'Country'})
            GDP = GDP.set_index('Country')

            ScimEn = pd.read_excel('scimagojr-3.xlsx', header=0)
            ScimEn.reset_index()
```

```
    ScimEn = ScimEn.set_index('Country')
    first_merge = pd.merge(energy, GDP, how='outer', left_index=True, right_index=True)
    result = pd.merge(ScimEn, first_merge, how='outer', left_index=True, right_index=Tru
    result = result.reset_index().dropna(thresh=result.shape[1]-10).set_index('Country')
    result = result.loc[result['Rank']<=15]
    return result

answer_one()
```

Out[2]:

Country	Rank	Documents	Citable documents	Citations
Australia	14.0	8831.0	8725.0	90765.0
Brazil	15.0	8668.0	8596.0	60702.0
Canada	6.0	17899.0	17620.0	215003.0
China	1.0	127050.0	126767.0	597237.0
France	9.0	13153.0	12973.0	130632.0
Germany	7.0	17027.0	16831.0	140566.0
India	8.0	15005.0	14841.0	128763.0
Iran	13.0	8896.0	8819.0	57470.0
Italy	11.0	10964.0	10794.0	111850.0
Japan	3.0	30504.0	30287.0	223024.0
Russian Federation	5.0	18534.0	18301.0	34266.0
South Korea	10.0	11983.0	11923.0	114675.0
Spain	12.0	9428.0	9330.0	123336.0
United Kingdom	4.0	20944.0	20357.0	206091.0
United States	2.0	96661.0	94747.0	792274.0

Country	Self-citations	Citations per document	H index
Australia	15606.0	10.28	107.0
Brazil	14396.0	7.00	86.0
Canada	40930.0	12.01	149.0
China	411683.0	4.70	138.0
France	28601.0	9.93	114.0
Germany	27426.0	8.26	126.0
India	37209.0	8.58	115.0
Iran	19125.0	6.46	72.0
Italy	26661.0	10.20	106.0
Japan	61554.0	7.31	134.0
Russian Federation	12422.0	1.85	57.0
South Korea	22595.0	9.57	104.0
Spain	23964.0	13.08	115.0
United Kingdom	37874.0	9.84	139.0
United States	265436.0	8.20	230.0

Country	Energy Supply	Energy Supply per Capita	% Renewable
Australia	5.386000e+09	231.0	11.810810

Brazil	1.214900e+10	59.0	69.648030
Canada	1.043100e+10	296.0	61.945430
China	1.271910e+11	93.0	19.754910
France	1.059700e+10	166.0	17.020280
Germany	1.326100e+10	165.0	17.901530
India	3.319500e+10	26.0	14.969080
Iran	9.172000e+09	119.0	5.707721
Italy	6.530000e+09	109.0	33.667230
Japan	1.898400e+10	149.0	10.232820
Russian Federation	3.070900e+10	214.0	17.288680
South Korea	1.100700e+10	221.0	2.279353
Spain	4.923000e+09	106.0	37.968590
United Kingdom	7.920000e+09	124.0	10.600470
United States	9.083800e+10	286.0	11.570980

	2006	2007	2008	2009	\
Country					
Australia	1.021939e+12	1.060340e+12	1.099644e+12	1.119654e+12	
Brazil	1.845080e+12	1.957118e+12	2.056809e+12	2.054215e+12	
Canada	1.564469e+12	1.596740e+12	1.612713e+12	1.565145e+12	
China	3.992331e+12	4.559041e+12	4.997775e+12	5.459247e+12	
France	2.607840e+12	2.669424e+12	2.674637e+12	2.595967e+12	
Germany	3.332891e+12	3.441561e+12	3.478809e+12	3.283340e+12	
India	1.265894e+12	1.374865e+12	1.428361e+12	1.549483e+12	
Iran	3.895523e+11	4.250646e+11	4.289909e+11	4.389208e+11	
Italy	2.202170e+12	2.234627e+12	2.211154e+12	2.089938e+12	
Japan	5.496542e+12	5.617036e+12	5.558527e+12	5.251308e+12	
Russian Federation	1.385793e+12	1.504071e+12	1.583004e+12	1.459199e+12	
South Korea	9.410199e+11	9.924316e+11	1.020510e+12	1.027730e+12	
Spain	1.414823e+12	1.468146e+12	1.484530e+12	1.431475e+12	
United Kingdom	2.419631e+12	2.482203e+12	2.470614e+12	2.367048e+12	
United States	1.479230e+13	1.505540e+13	1.501149e+13	1.459484e+13	

	2010	2011	2012	2013	\
Country					
Australia	1.142251e+12	1.169431e+12	1.211913e+12	1.241484e+12	
Brazil	2.208872e+12	2.295245e+12	2.339209e+12	2.409740e+12	
Canada	1.613406e+12	1.664087e+12	1.693133e+12	1.730688e+12	
China	6.039659e+12	6.612490e+12	7.124978e+12	7.672448e+12	
France	2.646995e+12	2.702032e+12	2.706968e+12	2.722567e+12	
Germany	3.417298e+12	3.542371e+12	3.556724e+12	3.567317e+12	
India	1.708459e+12	1.821872e+12	1.924235e+12	2.051982e+12	
Iran	4.677902e+11	4.853309e+11	4.532569e+11	4.445926e+11	
Italy	2.125185e+12	2.137439e+12	2.077184e+12	2.040871e+12	
Japan	5.498718e+12	5.473738e+12	5.569102e+12	5.644659e+12	
Russian Federation	1.524917e+12	1.589943e+12	1.645876e+12	1.666934e+12	
South Korea	1.094499e+12	1.134796e+12	1.160809e+12	1.194429e+12	
Spain	1.431673e+12	1.417355e+12	1.380216e+12	1.357139e+12	

```
United Kingdom      2.403504e+12  2.450911e+12  2.479809e+12  2.533370e+12
United States       1.496437e+13  1.520402e+13  1.554216e+13  1.577367e+13

                             2014          2015
Country
Australia           1.272520e+12  1.301251e+12
Brazil              2.412231e+12  2.319423e+12
Canada              1.773486e+12  1.792609e+12
China               8.230121e+12  8.797999e+12
France              2.729632e+12  2.761185e+12
Germany             3.624386e+12  3.685556e+12
India               2.200617e+12  2.367206e+12
Iran                4.639027e+11           NaN
Italy               2.033868e+12  2.049316e+12
Japan               5.642884e+12  5.669563e+12
Russian Federation  1.678709e+12  1.616149e+12
South Korea         1.234340e+12  1.266580e+12
Spain               1.375605e+12  1.419821e+12
United Kingdom      2.605643e+12  2.666333e+12
United States       1.615662e+13  1.654857e+13
```

1.0.2 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="2" fi
          <circle cx="200" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fi
          <circle cx="100" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fi
          <line x1="150" y1="125" x2="300" y2="150" stroke="black" stroke-width="2" fill="black"
          <text  x="300" y="165" font-family="Verdana" font-size="35">Everything but this!</text
        </svg>
```

```
<IPython.core.display.HTML object>
```

```
In [3]: def answer_two():
            import pandas as pd
            import numpy as np
            energy = pd.read_excel('Energy Indicators.xls', skip_footer=38, skiprows=17, parse_c
            col_names = ['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable']
            energy.columns = col_names
            energy.loc[energy['Energy Supply'] == '...'] = np.NaN
            energy[['Energy Supply', 'Energy Supply per Capita']] = energy[['Energy Supply', 'En
            energy['Energy Supply'] = energy['Energy Supply']*10**6
```

```
    energy['Country'] = energy['Country'].str.replace(r" \(.*\)","")
    energy['Country'] = energy['Country'].str.replace(r"([0-9]+)$","")
    replace_dict={"Republic of Korea": "South Korea",
                  "United States of America": "United States",
                  "United Kingdom of Great Britain and Northern Ireland": "United Kingdo
                  "China, Hong Kong Special Administrative Region": "Hong Kong"}
    energy['Country'].replace(to_replace=replace_dict, inplace=True)
    energy.reset_index()
    energy = energy.set_index('Country')
    en_shape = energy.shape

    GDP = pd.read_csv('world_bank.csv', skiprows=4)
    replace_dict = {"Korea, Rep.": "South Korea",
                    "Iran, Islamic Rep.": "Iran",
                    "Hong Kong SAR, China": "Hong Kong"
                   }
    GDP['Country Name'].replace(to_replace=replace_dict, inplace=True)
    years_to_keep = np.arange(2006, 2016).astype(str)
    GDP = GDP[np.append(['Country Name'],years_to_keep)]
    GDP.reset_index()
    GDP = GDP.rename(columns={'Country Name': 'Country'})
    GDP = GDP.set_index('Country')
    GDP_shape = GDP.shape

    ScimEn = pd.read_excel('scimagojr-3.xlsx', header=0)
    ScimEn.reset_index()
    ScimEn = ScimEn.set_index('Country')
    ScimEn_shape = ScimEn.shape

    first_merge = pd.merge(energy, GDP, how='outer', left_index=True, right_index=True)
    result = pd.merge(ScimEn, first_merge, how='outer', left_index=True, right_index=Tru
    #result = result.reset_index().dropna(thresh=result.shape[1]-10).set_index('Country'
    result = result.shape[0]-15

    return result
answer_two()
```

Out[3]: 307

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

1.1.1 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():
            import numpy as np
            Top15 = answer_one()
            years_to_keep = np.arange(2006, 2016).astype(str)
            Top15['avgGDP'] = Top15[years_to_keep].mean(axis=1)
            return Top15['avgGDP'].sort_values(ascending=False)

        answer_three()
```

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

1.1.2 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():
            import numpy as np
            Top15 = answer_one()
            years_to_keep = np.arange(2006, 2016).astype(str)
            Top15['avgGDP'] = Top15[years_to_keep].mean(axis=1)
            Top15 = Top15.sort_values(['avgGDP'], ascending=False)
            Top15['deltaGDP'] = Top15['2015'] - Top15['2006']
            Top15 = Top15.reset_index()
            return Top15.loc[5, 'deltaGDP']
        answer_four()
```

```
Out[5]: 246702696075.3999
```

1.1.3 Question 5 (6.6%)

What is the mean `Energy Supply per Capita`?

This function should return a single number.

```
In [6]: def answer_five():
            import numpy as np
            Top15 = answer_one()
            result = Top15['Energy Supply per Capita'].mean(axis=0)
            return result

        answer_five()
```

```
Out[6]: 157.59999999999999
```

1.1.4 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 [8]: def answer_six():
            import numpy as np
            Top15 = answer_one()
            result = Top15['% Renewable'].argmax()
            return (result, Top15.loc[result, '% Renewable'])

        answer_six()
```

```
Out[8]: ('Brazil', 69.648030000000006)
```

1.1.5 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 [9]: def answer_seven():
            import numpy as np
            Top15 = answer_one()
            Top15['CitationRatio'] = Top15['Self-citations']/Top15['Citations']
            result = Top15['CitationRatio'].argmax()
            return (result, Top15.loc[result, 'CitationRatio'])

        answer_seven()
```

```
Out[9]: ('China', 0.68931261793894216)
```

1.1.6 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 [10]: def answer_eight():
             import numpy as np
             Top15 = answer_one()
             Top15['PopEst'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
             Top15 = Top15.sort_values(['PopEst'], ascending=False)
             Top15 = Top15.reset_index()
             return Top15.loc[2, 'Country']

         answer_eight()

Out[10]: 'United States'
```

1.1.7 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 [11]: def answer_nine():
             import numpy as np
             Top15 = answer_one()
             Top15['PopEst'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
             Top15['Citable docs per Person'] = Top15['Citable documents']/Top15['PopEst']
             result = Top15.corr()
             return result.loc['Citable docs per Person', 'Energy Supply per Capita']

         answer_nine()

Out[11]: 0.79400104354429435
```

```
In [ ]: 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'
```

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

1.1.8 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 [12]: def answer_ten():
             import numpy as np
             Top15 = answer_one()
             Top15['median % Renewable'] = Top15['% Renewable'].median()
             Top15['HighRenew'] = Top15['% Renewable'] >= Top15['median % Renewable']
             return Top15['HighRenew'].sort_values(ascending=True)

         answer_ten()

Out[12]: Country
         Australia             False
         India                 False
         Iran                  False
         Japan                 False
         South Korea           False
         United Kingdom        False
         United States         False
         Brazil                 True
         Canada                 True
         China                  True
         France                 True
         Germany                True
         Italy                  True
         Russian Federation     True
         Spain                  True
         Name: HighRenew, dtype: bool
```

1.1.9 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 [13]:
```
def answer_eleven():
    import numpy as np
    import pandas as pd
    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 = Top15.reset_index()
    Top15['Continent'] = Top15['Country'].map(ContinentDict)
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    result = Top15.copy()
    result = result[['Continent', 'PopEst']]
    result = result.groupby('Continent')['PopEst'].agg({'size': np.size,'sum': np.sum,'
    #result = grouped.agg(['np.size', 'sum', 'mean', 'std'])
    idx = pd.IndexSlice
    #result = result.loc[:, idx['PopEst']]
    #result = result.reset_index()
    #result = result.set_index('Continent')
    return result

answer_eleven()
```

Out[13]:

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

1.1.10 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 [14]: def answer_twelve():
             import numpy as np
             import pandas as pd
             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 = Top15.reset_index()
             Top15['Continent'] = Top15['Country'].map(ContinentDict)
             Top15['% Renewable'] = pd.cut(Top15['% Renewable'], 5)
             result = Top15.groupby(['Continent', '% Renewable'])['Country'].count()
             result = result.reset_index()
             #result.drop('Country', axis=1, inplace=True)

             result = result.set_index(['Continent', '% Renewable'])
             return result['Country']

         answer_twelve()

Out[14]: Continent      % 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
         Name: Country, dtype: int64
```

1.1.11 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 [15]: def answer_thirteen():
             import numpy as np
             Top15 = answer_one()
             Top15['PopEst'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']
             Top15['PopEst'] = Top15['PopEst'].apply('{:,}'.format)
             return Top15['PopEst']

         answer_thirteen()
```

```
Out[15]: 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: PopEst, dtype: object
```

1.1.12 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','#377eb8','#4da
                               '#4daf4a','#e41a1c','#4daf4a','#4daf4a','#e41a1c','#dede00','#ff7
                            xticks=range(1,16), s=6*Top15['2014']/10**10, alpha=.75, figsize=[16

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

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