assignment3

December 10, 2020

1 Assignment 3

All questions are weighted the same in this assignment. 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. All questions are worth the same number of points except question 1 which is worth 17% of the assignment grade.

Note: Questions 2-13 rely on your question 1 answer.

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

# Filter all warnings. If you would like to see the warnings, please comment␣
 ↪the two lines below.
import warnings
warnings.filterwarnings('ignore')
```

1.0.1 Question 1

Load the energy data from the file `assets/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 (**Note: 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 parenthesis in their name. Be sure to remove these, e.g. `'Bolivia (Plurinational State of)'` should be `'Bolivia'`.

Next, load the GDP data from the file `assets/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 `assets/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, and the rows of the DataFrame should be sorted by "Rank".

[32]:
```
import pandas as pd
import numpy as np
import re

def answer_one():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                    #para sacar los '...',␣
↪recorro las columnas 1 y 2
```

```
      for i in range(len(energy)):                    #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
          if type(energy.iloc[i, j]) == str:
              energy.iloc[i, j] = np.nan

  energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

  for i in range(len(energy)):                        #para corregir los␣
↪nombres que están mal
      if 'Republic of Korea' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'South Korea'
      elif 'United States of America' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'United States'
      elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'United Kingdom'
      elif 'China, Hong Kong' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'Hong Kong'
      elif 'China2' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'China'
      elif 'Australia' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'Australia'
      elif 'Japan' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'Japan'
      elif 'France' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'France'
      elif 'Spain' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'Spain'
      elif 'Italy' in energy.iloc[i, 0]:
          energy.iloc[i, 0] = 'Italy'

  for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
      if '(' in energy.iloc[i, 0]:
          a = energy.iloc[i, 0]
          b = str(re.findall('(\s\(\D*\))', a))
          b = b[2:-2]
          energy.iloc[i, 0] = a.replace(b, '')

  Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

  #Energy.head()

  #primero debo leer el archivo y explorar las columas:

  #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
```

```
   #for i in range(df.shape[1]):
   #    print(i)
   #    print(df.iloc[4, i])

   columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

   for i in range(1960, 2016):                          #hago un for para␣
↪nombrar a las columnas de los respectivos años
       columns_names.append(i)

   df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                           #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df
```

```
    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    return (union_2)

    raise NotImplementedError()
```

[7]:
```
assert type(answer_one()) == pd.DataFrame, "Q1: You should return a DataFrame!"

assert answer_one().shape == (15,20), "Q1: Your DataFrame should have 20␣
↪columns and 15 entries!"
```

[4]:
```
# Cell for autograder.
```

1.0.2 Question 2

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.

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

[150]:
```
import pandas as pd
import numpy as np
import re
```

```
def answer_two():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan

    energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

    for i in range(len(energy)):                           #para corregir los␣
↪nombres que están mal
        if 'Republic of Korea' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'South Korea'
        elif 'United States of America' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United States'
        elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United Kingdom'
        elif 'China, Hong Kong' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Hong Kong'
        elif 'China2' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'China'
        elif 'Australia' in energy.iloc[i, 0]:
```

```
            energy.iloc[i, 0] = 'Australia'
        elif 'Japan' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Japan'
        elif 'France' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'France'
        elif 'Spain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

    GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
    GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
    GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
```

```
    GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                                2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

    GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
    GDP = GDP.drop('Indicator Name', axis=1)
    GDP = GDP.drop('Indicator Code', axis=1)

    #GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

    for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
        if 'Korea' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'South Korea'
        elif 'Iran' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Iran'
        elif 'Hong Kong' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Hong Kong'

    GDP = GDP.set_index('Country') #steo el nombre de pais como indice

    #GDP.head()

    df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='outer', left_index=True,␣
↪right_index=True)

    a = len(union_2) - len(ScimEn)

    return(a)

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_two()) == int, "Q2: You should return an int number!"
```

1.0.3 Question 3

What are the top 15 countries for average GDP over the last 10 years?

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

[41]:
```
import pandas as pd
import numpy as np
import re

def answer_three():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan

    energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

    for i in range(len(energy)):                           #para corregir los␣
↪nombres que están mal
        if 'Republic of Korea' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'South Korea'
        elif 'United States of America' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United States'
```

```
        elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United Kingdom'
        elif 'China, Hong Kong' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Hong Kong'
        elif 'China2' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'China'
        elif 'Australia' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Australia'
        elif 'Japan' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Japan'
        elif 'France' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'France'
        elif 'Spain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres
```

```
   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                    #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

   df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

   ScimEn = df.set_index('Country') #steo el nombre de pais como indice

   #ScimEn

   union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

   union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

   avgGDP = pd.Series(union_2.iloc[:15,10:20].mean(axis=1))
```

```
    avgGDP = avgGDP.sort_values(ascending=False)
    return(avgGDP)

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_three()) == pd.Series, "Q3: You should return a Series!"
```

1.0.4 Question 4

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.

[35]:
```
import pandas as pd
import numpy as np
import re

def answer_four():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan
```

```
   energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

   for i in range(len(energy)):                          #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'France'
       elif 'Spain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Spain'
       elif 'Italy' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Italy'

   for i in range(len(energy)):                          #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
       if '(' in energy.iloc[i, 0]:
           a = energy.iloc[i, 0]
           b = str(re.findall('(\s\(\D*\))', a))
           b = b[2:-2]
           energy.iloc[i, 0] = a.replace(b, '')

   Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

   #Energy.head()

   #primero debo leer el archivo y explorar las columas:

   #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
   #for i in range(df.shape[1]):
   #    print(i)
   #    print(df.iloc[4, i])
```

```
   columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

   for i in range(1960, 2016):                          #hago un for para␣
↪nombrar a las columnas de los respectivos años
       columns_names.append(i)

   df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

   df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

   ScimEn = df.set_index('Country') #steo el nombre de pais como indice
```

```
    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    año_2006 = union_2.iloc[3, 10]
    año_2015 = union_2.iloc[3, 19]
    return (año_2015 - año_2006)

    raise NotImplementedError()
```

[ ]:
```
# Cell for autograder.
```

1.0.5 Question 5

What is the mean energy supply per capita?
This function should return a single number.

[47]:
```
import pandas as pd
import numpy as np
import re

def answer_five():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad
```

```
   for j in range(1, 3):                                    #para sacar los '...',␣
↪recorro las columnas 1 y 2
       for i in range(len(energy)):                         #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
           if type(energy.iloc[i, j]) == str:
               energy.iloc[i, j] = np.nan

   energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

   for i in range(len(energy)):                             #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'France'
       elif 'Spain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Spain'
       elif 'Italy' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Italy'

   for i in range(len(energy)):                             #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
       if '(' in energy.iloc[i, 0]:
           a = energy.iloc[i, 0]
           b = str(re.findall('(\s\(\D*\))', a))
           b = b[2:-2]
           energy.iloc[i, 0] = a.replace(b, '')

   Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

   #Energy.head()
```

```
   #primero debo leer el archivo y explorar las columas:

   #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
   #for i in range(df.shape[1]):
   #    print(i)
   #    print(df.iloc[4, i])

   columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

   for i in range(1960, 2016):                          #hago un for para␣
↪nombrar a las columnas de los respectivos años
       columns_names.append(i)

   df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice
```

```
    #GDP.head()

    df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    a = union_2['Energy Supply per Capita'].mean(axis=0)
    return(a)

    raise NotImplementedError()
```

[ ]:
```
# Cell for autograder.
```

1.0.6 Question 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.

[141]:
```
import pandas as pd
import numpy as np
import re

def answer_six():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final
```

```
   energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

   energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

   for j in range(1, 3):                                #para sacar los '...',␣
↪recorro las columnas 1 y 2
       for i in range(len(energy)):                     #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
           if type(energy.iloc[i, j]) == str:
               energy.iloc[i, j] = np.nan

   energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

   for i in range(len(energy)):                         #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'France'
       elif 'Spain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Spain'
       elif 'Italy' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Italy'

   for i in range(len(energy)):                         #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
       if '(' in energy.iloc[i, 0]:
           a = energy.iloc[i, 0]
           b = str(re.findall('(\s\(\D*\))', a))
           b = b[2:-2]
```

```
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                          #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

    GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
    GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
    GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
    GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                                2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

    GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
    GDP = GDP.drop('Indicator Name', axis=1)
    GDP = GDP.drop('Indicator Code', axis=1)

    GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

    for i in range(len(GDP)):                            #para corregir los nombres␣
↪que están mal
        if 'Korea' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'South Korea'
        elif 'Iran' in GDP.iloc[i, 0]:
```

```
            GDP.iloc[i, 0] = 'Iran'
        elif 'Hong Kong' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Hong Kong'

    GDP = GDP.set_index('Country') #steo el nombre de pais como indice

    #GDP.head()

    df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    a = np.max(union_2['% Renewable'])
    b = union_2.sort_values('% Renewable', ascending=False).index[0]
    c = (b, a)
    return(c)

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_six()) == tuple, "Q6: You should return a tuple!"

assert type(answer_six()[0]) == str, "Q6: The first element in your result␣
↪should be the name of the country!"
```

1.0.7 Question 7

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.

[53]:
```
import pandas as pd
import numpy as np
import re

def answer_seven():
    # YOUR CODE HERE
```

```
   columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

   df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

   energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

   energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

   energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

   energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

   energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

   for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
       for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
           if type(energy.iloc[i, j]) == str:
               energy.iloc[i, j] = np.nan

   energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

   for i in range(len(energy)):                           #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
```

```
            energy.iloc[i, 0] = 'France'
        elif 'Spain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

    GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
    GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
    GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
    GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                                2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})
```

```
    GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
    GDP = GDP.drop('Indicator Name', axis=1)
    GDP = GDP.drop('Indicator Code', axis=1)

    GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

    for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
        if 'Korea' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'South Korea'
        elif 'Iran' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Iran'
        elif 'Hong Kong' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Hong Kong'

    GDP = GDP.set_index('Country') #steo el nombre de pais como indice

    #GDP.head()

    df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    union_2['Ratio of Citations'] = union_2['Self-citations'] /␣
↪union_2['Citations']
    a = np.max(union_2['Ratio of Citations'])
    b = union_2.sort_values('Ratio of Citations', ascending=False).index[0]
    c = (b, a)
    return(c)

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_seven()) == tuple, "Q7: You should return a tuple!"
```

```
assert type(answer_seven()[0]) == str, "Q7: The first element in your result␣
↪should be the name of the country!"
```

1.0.8 Question 8

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 the name of the country

[56]:
```
import pandas as pd
import numpy as np
import re

def answer_eight():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan

    energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

    for i in range(len(energy)):                           #para corregir los␣
↪nombres que están mal
```

```
        if 'Republic of Korea' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'South Korea'
        elif 'United States of America' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United States'
        elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United Kingdom'
        elif 'China, Hong Kong' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Hong Kong'
        elif 'China2' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'China'
        elif 'Australia' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Australia'
        elif 'Japan' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Japan'
        elif 'France' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'France'
        elif 'Spain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)
```

```
   df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

   df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

   ScimEn = df.set_index('Country') #steo el nombre de pais como indice

   #ScimEn

   union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)
```

```
    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    union_2['Population'] = union_2['Energy Supply'] / union_2['Energy Supply␣
↪per Capita']
    b = union_2.sort_values('Population', ascending=False).index[2]
    return(b)

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_eight()) == str, "Q8: You should return the name of the␣
↪country!"
```

1.0.9 Question 9

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)

[57]:
```
import pandas as pd
import numpy as np
import re

def answer_nine():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad
```

```
   for j in range(1, 3):                                    #para sacar los '...',␣
↪recorro las columnas 1 y 2
       for i in range(len(energy)):                         #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
           if type(energy.iloc[i, j]) == str:
               energy.iloc[i, j] = np.nan

   energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

   for i in range(len(energy)):                             #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'France'
       elif 'Spain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Spain'
       elif 'Italy' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Italy'

   for i in range(len(energy)):                             #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
       if '(' in energy.iloc[i, 0]:
           a = energy.iloc[i, 0]
           b = str(re.findall('(\s\(\D*\))', a))
           b = b[2:-2]
           energy.iloc[i, 0] = a.replace(b, '')

   Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

   #Energy.head()
```

```
   #primero debo leer el archivo y explorar las columas:

   #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
   #for i in range(df.shape[1]):
   #    print(i)
   #    print(df.iloc[4, i])

   columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

   for i in range(1960, 2016):                          #hago un for para␣
↪nombrar a las columnas de los respectivos años
       columns_names.append(i)

   df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice
```

```
    #GDP.head()

    df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    union_2['Population'] = union_2['Energy Supply'] / union_2['Energy Supply␣
↪per Capita']

    union_2['Energy Supply per Capita'] = union_2['Energy Supply per Capita'].
↪astype('float') #cambio el tipo de dato a float
    union_2['Citable documents per Capita'] = union_2['Citable documents'] /␣
↪union_2['Population']
    union_2['Citable documents per Capita'] = union_2['Citable documents per␣
↪Capita'].astype('float')#cambio el tipo de dato a float
    a = union_2['Citable documents per Capita'].corr(union_2['Energy Supply per␣
↪Capita'])
    return(a)

    raise NotImplementedError()
```

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

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

[ ]:
```
assert answer_nine() >= -1. and answer_nine() <= 1., "Q9: A valid correlation␣
↪should between -1 to 1!"
```

1.0.10 Question 10

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.*

[138]:
```
import pandas as pd
import numpy as np
import re

def answer_ten():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan

    energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

    for i in range(len(energy)):                           #para corregir los␣
↪nombres que están mal
        if 'Republic of Korea' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'South Korea'
        elif 'United States of America' in energy.iloc[i, 0]:
```

```
            energy.iloc[i, 0] = 'United States'
        elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United Kingdom'
        elif 'China, Hong Kong' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Hong Kong'
        elif 'China2' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'China'
        elif 'Australia' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Australia'
        elif 'Japan' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Japan'
        elif 'France' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'France'
        elif 'Spain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres
```

```
   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                        #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

   df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

   ScimEn = df.set_index('Country') #steo el nombre de pais como indice

   #ScimEn

   union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

   union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)
```

```
    union_2['Population'] = union_2['Energy Supply'] / union_2['Energy Supply␣
↪per Capita']

    union_2['HighRenew'] = 0
    for i in range(len(union_2)):
        if union_2['% Renewable'][i] >= union_2['% Renewable'].median():
            union_2['HighRenew'][i] = 1

    return(pd.Series(union_2['HighRenew']))

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_ten()) == pd.Series, "Q10: You should return a Series!"
```

1.0.11 Question 11

Use the following dictionary to group the Countries by Continent, then create a DataFrame 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']`

[129]:
```
import pandas as pd
import numpy as np
import re

def answer_eleven():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas
```

```
   df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

   energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

   energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

   energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

   energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

   energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

   for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
       for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
           if type(energy.iloc[i, j]) == str:
               energy.iloc[i, j] = np.nan

   energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

   for i in range(len(energy)):                           #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'France'
       elif 'Spain' in energy.iloc[i, 0]:
```

```
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

    GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
    GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
    GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
    GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                                2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

    GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
```

```
    GDP = GDP.drop('Indicator Name', axis=1)
    GDP = GDP.drop('Indicator Code', axis=1)

    GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

    for i in range(len(GDP)):                          #para corregir los nombres␣
↪que están mal
        if 'Korea' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'South Korea'
        elif 'Iran' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Iran'
        elif 'Hong Kong' in GDP.iloc[i, 0]:
            GDP.iloc[i, 0] = 'Hong Kong'

    GDP = GDP.set_index('Country') #steo el nombre de pais como indice

    #GDP.head()

    df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

    df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

    ScimEn = df.set_index('Country') #steo el nombre de pais como indice

    #ScimEn

    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    union_2['Population'] = union_2['Energy Supply'] / union_2['Energy Supply␣
↪per Capita']

    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 = pd.DataFrame([[key, ContinentDict[key]] for key in ContinentDict.
↪keys()], columns=['Country', 'Continent']) #creo df

   df = df.set_index('Country') #seteo index

   union_3 = pd.merge(union_2, df, how='outer', left_index=True,␣
↪right_index=True) #hago merge

   union_3 = union_3.reset_index() #reseteo indices

   union_3 = union_3.set_index(['Continent', 'Country']) #seteo indices
   #union_3 = union_3.set_index(['Continent']) #seteo indices

   union_3 = union_3.sort_index(level=0) #ordeno por el primer indice␣
↪(continente)

   a = union_3.groupby('Continent').agg({'Population': ('size', 'sum', 'std')})

   a['size'] = a['Population']['size']
   a['sum'] = a['Population']['sum']
   a['mean'] = a['Population']['sum'] / a['Population']['size']
   a['std'] = a['Population']['std']

   a = a.drop(['Population'], axis=1)
   #a = a.reset_index(level='Population')
   #a = a.drop(['Population'], axis=1)
   #a = a.set_index(['Continent'])
   return(a)

   raise NotImplementedError()
```

[ ]:
```
assert type(answer_eleven()) == pd.DataFrame, "Q11: You should return a␣
↪DataFrame!"

assert answer_eleven().shape[0] == 5, "Q11: Wrong row numbers!"

assert answer_eleven().shape[1] == 4, "Q11: Wrong column numbers!"
```

1.0.12 Question 12

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.

[76]:
```
import pandas as pd
import numpy as np
import re

def answer_twelve():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                   #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                        #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan

    energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo

    for i in range(len(energy)):                            #para corregir los␣
↪nombres que están mal
        if 'Republic of Korea' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'South Korea'
        elif 'United States of America' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United States'
        elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'United Kingdom'
```

```
        elif 'China, Hong Kong' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Hong Kong'
        elif 'China2' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'China'
        elif 'Australia' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Australia'
        elif 'Japan' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Japan'
        elif 'France' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'France'
        elif 'Spain' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Spain'
        elif 'Italy' in energy.iloc[i, 0]:
            energy.iloc[i, 0] = 'Italy'

    for i in range(len(energy)):                        #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
        if '(' in energy.iloc[i, 0]:
            a = energy.iloc[i, 0]
            b = str(re.findall('(\s\(\D*\))', a))
            b = b[2:-2]
            energy.iloc[i, 0] = a.replace(b, '')

    Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

    #Energy.head()

    #primero debo leer el archivo y explorar las columas:

    #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
    #for i in range(df.shape[1]):
    #    print(i)
    #    print(df.iloc[4, i])

    columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas

    for i in range(1960, 2016):                         #hago un for para␣
↪nombrar a las columnas de los respectivos años
        columns_names.append(i)

    df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

    GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
```

```
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                    #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

   df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

   ScimEn = df.set_index('Country') #steo el nombre de pais como indice

   #ScimEn

   union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

   union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

   union_2['Population'] = union_2['Energy Supply'] / union_2['Energy Supply␣
↪per Capita']
```

```
    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 = pd.DataFrame([[key, ContinentDict[key]] for key in ContinentDict.
↪keys()], columns=['Country', 'Continent']) #creo df

    df = df.set_index('Country') #seteo index

    union_3 = pd.merge(union_2, df, how='outer', left_index=True,␣
↪right_index=True) #hago merge

    #union_3['Bins'] = pd.cut(union_3['% Renewable'], 5)
    union_3['% Renewable'] = pd.cut(union_3['% Renewable'], 5)

    union_3 = union_3.reset_index() #reseteo indices

    #union_3['Bins2'] = union_3['Bins']
    union_3['Bins2'] = union_3['% Renewable']

    #union_3 = union_3.set_index(['Continent', 'Bins']) #seteo indices
    union_3 = union_3.set_index(['Continent', '% Renewable']) #seteo indices

    union_3 = union_3.sort_index()

    #a = union_3.groupby(['Continent', 'Bins']).agg({'Bins2': ('count')})
    a = union_3.groupby(['Continent', '% Renewable']).agg({'Bins2': ('count')})

    q12 = pd.Series(a.iloc[:15, 0])

    return(q12)

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_twelve()) == pd.Series, "Q12: You should return a Series!"

assert len(answer_twelve()) == 9, "Q12: Wrong result numbers!"
```

1.0.13 Question 13

Convert the Population Estimate series to a string with thousands separator (using commas). Use all significant digits (do not round the results).

e.g. 12345678.90 -> 12,345,678.90

This function should return a series `PopEst` *whose index is the country name and whose values are the population estimate string*

[71]:
```
import pandas as pd
import numpy as np
import re

def answer_thirteen():
    # YOUR CODE HERE

    columns_names = ['0', '1','Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] #nombres de columnas

    df = pd.read_excel('assets/Energy Indicators.xls', header=None ,␣
↪names=columns_names) #leo archivo y le pongo el nombre

    energy = df.drop(['0', '1'], axis=1) #elimino las dos primeras columnas

    energy = energy.drop(np.arange(18), axis=0) #elimino el encabezado

    energy = energy.drop(np.arange(245, 283, 1), axis=0) #elimino la parte del␣
↪final

    energy = energy.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos

    energy['Energy Supply'] = energy['Energy Supply'] * 1000000 #para cambiar␣
↪de unidad

    for j in range(1, 3):                                  #para sacar los '...',␣
↪recorro las columnas 1 y 2
        for i in range(len(energy)):                       #recorro todas las␣
↪filas, cuando hay un str lo cambio por np.nan
            if type(energy.iloc[i, j]) == str:
                energy.iloc[i, j] = np.nan

    energy = energy.drop(56, axis=0) #elimino la otra republic of korea que␣
↪esta jodiendo
```

```
   for i in range(len(energy)):                    #para corregir los␣
↪nombres que están mal
       if 'Republic of Korea' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'South Korea'
       elif 'United States of America' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United States'
       elif 'United Kingdom of Great Britain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'United Kingdom'
       elif 'China, Hong Kong' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Hong Kong'
       elif 'China2' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'China'
       elif 'Australia' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Australia'
       elif 'Japan' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Japan'
       elif 'France' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'France'
       elif 'Spain' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Spain'
       elif 'Italy' in energy.iloc[i, 0]:
           energy.iloc[i, 0] = 'Italy'

   for i in range(len(energy)):                    #para sacar todo lo que␣
↪esta dentro de los parentesis en algunos paises
       if '(' in energy.iloc[i, 0]:
           a = energy.iloc[i, 0]
           b = str(re.findall('(\s\(\D*\))', a))
           b = b[2:-2]
           energy.iloc[i, 0] = a.replace(b, '')

   Energy = energy.set_index('Country') #seteo el nombre de pais como indice  ␣
↪

   #Energy.head()

   #primero debo leer el archivo y explorar las columas:

   #df = pd.read_csv('assets/world_bank.csv', header=None) #leo df
   #for i in range(df.shape[1]):
   #    print(i)
   #    print(df.iloc[4, i])

   columns_names = ['Country', 'Country Code', 'Indicator Name', 'Indicator␣
↪Code'] #para los nombres de las columnas
```

```
   for i in range(1960, 2016):                          #hago un for para␣
↪nombrar a las columnas de los respectivos años
       columns_names.append(i)

   df = pd.read_csv('assets/world_bank.csv', header=None, names=columns_names)␣
↪#leo devuelta y le coloco bien los nombres

   GDP = df.drop(np.arange(5), axis=0) #elimino el encabezado
   GDP = GDP.reset_index().drop('index', axis=1) #reseteo los indices y␣
↪elimino los indices viejos
   GDP = GDP.drop(np.arange(1960, 2006), axis=1) #elimino columnas que no␣
↪sirven, datos de 1960 a 2005 inclusive
   GDP = GDP.rename(columns = {2006: '2006', 2007: '2007', 2008: '2008', 2009:␣
↪'2009', 2010: '2010',
                               2011: '2011', 2012: '2012', 2013: '2013', 2014:␣
↪'2014', 2015: '2015'})

   GDP = GDP.drop('Country Code', axis=1) #elimino otras columnas que no␣
↪sirven
   GDP = GDP.drop('Indicator Name', axis=1)
   GDP = GDP.drop('Indicator Code', axis=1)

   GDP = GDP.drop(191, axis=0) #elimino la otra korea que esta jodiendo

   for i in range(len(GDP)):                           #para corregir los nombres␣
↪que están mal
       if 'Korea' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'South Korea'
       elif 'Iran' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Iran'
       elif 'Hong Kong' in GDP.iloc[i, 0]:
           GDP.iloc[i, 0] = 'Hong Kong'

   GDP = GDP.set_index('Country') #steo el nombre de pais como indice

   #GDP.head()

   df = pd.read_excel('assets/scimagojr-3.xlsx') #leo df

   df = df.drop(np.arange(15, 191, 1), axis=0) #elimino las finlas que no me␣
↪interesan

   ScimEn = df.set_index('Country') #steo el nombre de pais como indice

   #ScimEn
```

```
    union_1 = pd.merge(ScimEn, Energy, how='left', left_index=True,␣
↪right_index=True)

    union_2 = pd.merge(union_1, GDP, how='left', left_index=True,␣
↪right_index=True)

    union_2['Population'] = union_2['Energy Supply'] / union_2['Energy Supply␣
↪per Capita']

    r = union_2['Population'].astype('str')

    union_2['PopEst'] = 0
    union_2['PopEst'] = union_2['PopEst'].astype('str')

    for i in range(len(union_2)):
        a = r[i]
        b = re.findall('(\w*)', a)
        d = b[0]
        if len(d) > 9:
            e = d[-12:-9] + ',' + d[-9:-6] + ',' + d[-6:-3] + ',' + d[-3:]
            union_2['PopEst'][i] = e + '.' + b[-2]
        else:
            e = d[-9:-6] + ',' + d[-6:-3] + ',' + d[-3:]
            union_2['PopEst'][i] = e + '.' + b[-2]

    return(pd.Series(union_2['PopEst']))

    raise NotImplementedError()
```

[ ]:
```
assert type(answer_thirteen()) == pd.Series, "Q13: You should return a Series!"

assert len(answer_thirteen()) == 15, "Q13: Wrong result numbers!"
```

1.0.14 Optional

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

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

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

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