assignment3

April 29, 2022

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.

[2]:
```
# 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 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 `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".

[3]:
```
import numpy as np
import re
import pandas as pd

def answer_one():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
 ↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                  #eliino rango␣
 ↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)               #eliino rango␣
 ↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
 ↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                           #relleno con␣
 ↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
 ↪#reemplazar con vacio '' donde cadena de numeros
    Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
 ↪#reemplazar con vacio '' donde cadena de numeros ()
    Energy['Country']=Energy['Country'].str.rstrip()           #elimino␣
 ↪espacios que sobran al fin de la cadena
    Energy['Country'].replace({'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'},␣
 ↪inplace=True) #reemplazo nombres
```

```
    Energy['Energy Supply']=Energy['Energy Supply']*1000000      #paso de peta a␣
↪giga joules
    Energy=Energy.set_index('Country')
    # print(Energy)
    # print(len(Energy)) #227

    GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
    GDP[0]=GDP[0].str.strip()                                    #remuevo␣
↪espacios de la columna 0
    axislist=[]
    for x in GDP.iloc[0]:                                        #creo lista con␣
↪los nombres de las cabeceras
        axislist.append(x)
    for k in range (4,len(axislist),1):
        axislist[k]=str(int(axislist[k]))
    GDP.drop([0],axis=0,inplace=True)                            #elimino la␣
↪fila 0
    GDP.set_axis(axislist, axis=1, inplace=True)                 #coloco lista␣
↪como nuevo axis
    GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
    GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn=ScimEn.set_index('Country')
    # print(ScimEn)
    # print(len(ScimEn)) #191

    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]
    lista=[]
    for x in df.columns:
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)
```

```
    answer_one=df                                #filtrada por los primeros 15␣
 ↪valores = (15 valores de rank)
    #print(type(answer_one))
    #print(answer_one.shape)
    #print(answer_one)
    return(answer_one)
```

[4]:
```
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!"
```

[5]:
```
# 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.

[6]:
```
%%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>
```

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

[10]:
```
import numpy as np
import re
import pandas as pd
def answer_two():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
 ↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                   #eliino rango␣
 ↪de filas 0 a 17 (header)
```

```
   Energy=Energy.drop(range(244,282,1), axis=0)                #eliino rango␣
↪de filas 244 a 282 (footer)
   Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
   listaener=[]
   for k in Energy.columns:
       str(k).strip()
       listaener.append(k)
   Energy.set_axis(listaener, axis=1, inplace=True)
   Energy[Energy == '...'] = np.nan                            #relleno con␣
↪NaN donde haya puntos (...)
   Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
   Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
   Energy['Country']=Energy['Country'].str.rstrip()            #elimino␣
↪espacios que sobran al fin de la cadena
   Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
   Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de peta a␣
↪giga joules
   Energy=Energy.set_index('Country')
   # print(Energy)
   # print(len(Energy)) #227

   GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
   GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
   axislist=[]
   for x in GDP.iloc[0]:                                       #creo lista con␣
↪los nombres de las cabeceras
       axislist.append(x)
   for k in range (4,len(axislist),1):
       axislist[k]=str(int(axislist[k]))
   GDP.drop([0],axis=0,inplace=True)                           #elimino la␣
↪fila 0
   GDP.set_axis(axislist, axis=1, inplace=True)                #coloco lista␣
↪como nuevo axis
   GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
   GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
```

```
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn['Country']=ScimEn['Country'].str.rstrip()
    ScimEn=ScimEn.set_index('Country')
    # print(ScimEn)
    # print(len(ScimEn)) #191


    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    outervalues=len(df)
    print('Outervalues length: ', outervalues)

    mergedf1=pd.merge(ScimEn, Energy, on="Country")  #uno GDP Y Energy␣
↪dataframes en un solo dataf: mergedf
    df1=pd.merge(mergedf1, GDP, on="Country")   #uno mergedf con Scim␣
↪dataframe= todos dataframe
    innervalues=len(df1)
    print('innervalues length: ',innervalues)


    answer_two=outervalues-innervalues
    print('lost Values: ', answer_two)

    return(answer_two)
```

[11]:
```
assert type(answer_two()) == int, "Q2: You should return an int number!"
```

```
Outervalues length:  318
innervalues length:  162
lost Values:  156
```

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.

[32]:
```
import numpy as np
import re
import pandas as pd
def answer_three():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                   #eliino rango␣
↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)                #eliino rango␣
↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                            #relleno con␣
↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
    Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
    Energy['Country']=Energy['Country'].str.rstrip()            #elimino␣
↪espacios que sobran al fin de la cadena
    Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
    Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de peta a␣
↪giga joules
    Energy=Energy.set_index('Country')
    # print(Energy)
    # print(len(Energy)) #227

    GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
    GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
    axislist=[]
    for x in GDP.iloc[0]:                                       #creo lista con␣
↪los nombres de las cabeceras
        axislist.append(x)
    for k in range (4,len(axislist),1):
```

```
        axislist[k]=str(int(axislist[k]))
    GDP.drop([0],axis=0,inplace=True)                                #elimino la␣
↪fila 0
    GDP.set_axis(axislist, axis=1, inplace=True)                     #coloco lista␣
↪como nuevo axis
    GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
    GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn=ScimEn.set_index('Country')
    # print(ScimEn)
    # print(len(ScimEn)) #191

    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                          #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas

    df['avgGDP']=df.iloc[:,10:].mean(axis=1)
    df=df['avgGDP'][0:15]
    df.sort_values(ascending=False, inplace=True)

    answer_three=df   #filtrada por los primeros 15 valores = (15 valores de␣
↪rank)

    print(answer_three)
    print(type(answer_three))

    return(answer_three)
```

[33]:
```
assert type(answer_three()) == pd.Series, "Q3: You should return a Series!"
```

```
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
<class 'pandas.core.series.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.

[5]:
```
import numpy as np
import re
import pandas as pd
def answer_four():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
 ↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                   #eliino rango␣
 ↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)                #eliino rango␣
 ↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
 ↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                            #relleno con␣
 ↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
 ↪#reemplazar con vacio '' donde cadena de numeros
```

```
   Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
   Energy['Country']=Energy['Country'].str.rstrip()            #elimino␣
↪espacios que sobran al fin de la cadena
   Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
   Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de peta a␣
↪giga joules
   Energy=Energy.set_index('Country')
   # print(Energy)
   # print(len(Energy)) #227

   GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
   GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
   axislist=[]
   for x in GDP.iloc[0]:                                       #creo lista con␣
↪los nombres de las cabeceras
       axislist.append(x)
   for k in range (4,len(axislist),1):
       axislist[k]=str(int(axislist[k]))
   GDP.drop([0],axis=0,inplace=True)                           #elimino la␣
↪fila 0
   GDP.set_axis(axislist, axis=1, inplace=True)                #coloco lista␣
↪como nuevo axis
   GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
   GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
   GDP=GDP.set_index('Country')
   # print(GDP)
   # print(len(GDP)) #264

   ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
   ScimEn=ScimEn.set_index('Country')
   # print(ScimEn)
   # print(len(ScimEn)) #191

   mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
   df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
```

```
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                          #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas
    df['avgGDP']=df.iloc[:,10:].mean(axis=1)
    #df.sort_values(ascending=False, inplace=True) descending average GDP␣
↪Countries
    print(df)
    # print(df.columns)
    answer_four=df.iloc[3,19] - df.iloc[3,10] #6th largest average GDP:
↪unitedkindome:2015 - 2006 =
    # print(answer_four)

    print(answer_four)
    print(type(answer_four))

    return(answer_four)
```

[6]:
```
# Cell for autograder.
```

1.0.5 Question 5

What is the mean energy supply per capita?

This function should return a single number.

[13]:
```
import numpy as np
import re
import pandas as pd
def answer_five():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                  #eliino rango␣
↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)               #eliino rango␣
↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
```

```
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                                #relleno con␣
↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
    Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
    Energy['Country']=Energy['Country'].str.rstrip()                #elimino␣
↪espacios que sobran al fin de la cadena
    Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
    Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de peta a␣
↪giga joules
    Energy=Energy.set_index('Country')
    # print(Energy)
    # print(len(Energy)) #227

    GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
    GDP[0]=GDP[0].str.strip()                                       #remuevo␣
↪espacios de la columna 0
    axislist=[]
    for x in GDP.iloc[0]:                                           #creo lista con␣
↪los nombres de las cabeceras
        axislist.append(x)
    for k in range (4,len(axislist),1):
        axislist[k]=str(int(axislist[k]))
    GDP.drop([0],axis=0,inplace=True)                               #elimino la␣
↪fila 0
    GDP.set_axis(axislist, axis=1, inplace=True)                    #coloco lista␣
↪como nuevo axis
    GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
    GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn=ScimEn.set_index('Country')
```

```
    # print(ScimEn)
    # print(len(ScimEn)) #191

    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                          #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas
    answer_five=df['Energy Supply per Capita'].mean() #157.6
    print(answer_five)
    print(type(answer_five))
    return(answer_five)
```

[14]:
```
# 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.

[5]:
```
import numpy as np
import re
import pandas as pd
def answer_six():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                  #eliino rango␣
↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)               #eliino rango␣
↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
```

```
   Energy.set_axis(listaener, axis=1, inplace=True)
   Energy[Energy == '...'] = np.nan                                  #relleno con␣
↪NaN donde haya puntos (...)
   Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
   Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
   Energy['Country']=Energy['Country'].str.rstrip()                  #elimino␣
↪espacios que sobran al fin de la cadena
   Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
   Energy['Energy Supply']=Energy['Energy Supply']*1000000           #paso de peta a␣
↪giga joules
   Energy=Energy.set_index('Country')
   # print(Energy)
   # print(len(Energy)) #227

   GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
   GDP[0]=GDP[0].str.strip()                                         #remuevo␣
↪espacios de la columna 0
   axislist=[]
   for x in GDP.iloc[0]:                                             #creo lista con␣
↪los nombres de las cabeceras
       axislist.append(x)
   for k in range (4,len(axislist),1):
       axislist[k]=str(int(axislist[k]))
   GDP.drop([0],axis=0,inplace=True)                                 #elimino la␣
↪fila 0
   GDP.set_axis(axislist, axis=1, inplace=True)                      #coloco lista␣
↪como nuevo axis
   GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
   GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
   GDP=GDP.set_index('Country')
   # print(GDP)
   # print(len(GDP)) #264

   ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
   ScimEn=ScimEn.set_index('Country')
   # print(ScimEn)
   # print(len(ScimEn)) #191
```

```
    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                          #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas
    index_max_valor=df.index[df['% Renewable']==df['% Renewable'].max()].
↪tolist() #entrega lista con index que cumpple la condicion
    index_max_valor=index_max_valor[0] #saco el valor de la lista
    valor=df.loc['Brazil']['% Renewable'] #encuentro el valor de Brazil # 69.
↪64803
    answer_six=(index_max_valor, valor)  #armo tupla
    #print(type(answer_six))

    return(answer_six)
```

[6]:
```
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!"
```

```
('Brazil', 69.64803)
('Brazil', 69.64803)
```

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.

[3]:
```
import numpy as np
import re
import pandas as pd
def answer_seven():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
```

```
   Energy=Energy.drop(range(0,17,1), axis=0)                    #eliino rango␣
↪de filas 0 a 17 (header)
   Energy=Energy.drop(range(244,282,1), axis=0)                 #eliino rango␣
↪de filas 244 a 282 (footer)
   Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
   listaener=[]
   for k in Energy.columns:
       str(k).strip()
       listaener.append(k)
   Energy.set_axis(listaener, axis=1, inplace=True)
   Energy[Energy == '...'] = np.nan                             #relleno con␣
↪NaN donde haya puntos (...)
   Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
   Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
   Energy['Country']=Energy['Country'].str.rstrip()             #elimino␣
↪espacios que sobran al fin de la cadena
   Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
   Energy['Energy Supply']=Energy['Energy Supply']*1000000      #paso de peta a␣
↪giga joules
   Energy=Energy.set_index('Country')
   # print(Energy)
   # print(len(Energy)) #227

   GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
   GDP[0]=GDP[0].str.strip()                                    #remuevo␣
↪espacios de la columna 0
   axislist=[]
   for x in GDP.iloc[0]:                                        #creo lista con␣
↪los nombres de las cabeceras
       axislist.append(x)
   for k in range (4,len(axislist),1):
       axislist[k]=str(int(axislist[k]))
   GDP.drop([0],axis=0,inplace=True)                            #elimino la␣
↪fila 0
   GDP.set_axis(axislist, axis=1, inplace=True)                 #coloco lista␣
↪como nuevo axis
   GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
```

```
    GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn=ScimEn.set_index('Country')
    # print(ScimEn)
    # print(len(ScimEn)) #191

    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                          #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas


    df['Ratio_self-citations']=df['Self-citations']/df['Citations']
    index_max_valor=df.
↪index[df['Ratio_self-citations']==df['Ratio_self-citations'].max()].tolist()␣
↪#entrega lista con index que cumpple la condicion
    index_max_valor=index_max_valor[0] #saco el valor de la lista

    valor=df.loc['China']['Ratio_self-citations'] #encuentro el valor de china
    answer_seven=(index_max_valor, valor)         #armo tupla

    print(answer_seven)
    return(answer_seven)
```

[4]:
```
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!"
```

```
('China', 0.6893126179389422)
('China', 0.6893126179389422)
```

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

[13]:
```
import numpy as np
import re
import pandas as pd
def answer_eight():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                   #eliino rango␣
↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)                #eliino rango␣
↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                            #relleno con␣
↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
    Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
    Energy['Country']=Energy['Country'].str.rstrip()            #elimino␣
↪espacios que sobran al fin de la cadena
    Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
    Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de peta a␣
↪giga joules
    Energy=Energy.set_index('Country')
    # print(Energy)
    # print(len(Energy)) #227

    GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
    GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
```

```
    axislist=[]
    for x in GDP.iloc[0]:                                   #creo lista con␣
↪los nombres de las cabeceras
        axislist.append(x)
    for k in range (4,len(axislist),1):
        axislist[k]=str(int(axislist[k]))
    GDP.drop([0],axis=0,inplace=True)                       #elimino la␣
↪fila 0
    GDP.set_axis(axislist, axis=1, inplace=True)            #coloco lista␣
↪como nuevo axis
    GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
    GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn=ScimEn.set_index('Country')
    # print(ScimEn)
    # print(len(ScimEn)) #191

    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                        #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas

    df['Population']=df['Energy Supply']/df['Energy Supply per Capita']
    df.sort_values(by=['Population'], inplace=True, ascending=False)

    index=df.index           #lista de index en orden descendente
    answer_eight=index[2]   #tercer index
    #print(answer_eight)
```

```
    return(answer_eight)
answer_eight()
```

[13]:
```
'United States'
```

[14]:
```
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)

[7]:
```
import numpy as np
import re
import pandas as pd
import scipy.stats as stats
def answer_nine():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                  #eliino rango␣
↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)               #eliino rango␣
↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                           #relleno con␣
↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
    Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
    Energy['Country']=Energy['Country'].str.rstrip()           #elimino␣
↪espacios que sobran al fin de la cadena
    Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
   Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de peta a␣
↪giga joules
   Energy=Energy.set_index('Country')
   # print(Energy)
   # print(len(Energy)) #227

   GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
   GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
   axislist=[]
   for x in GDP.iloc[0]:                                       #creo lista con␣
↪los nombres de las cabeceras
       axislist.append(x)
   for k in range (4,len(axislist),1):
       axislist[k]=str(int(axislist[k]))
   GDP.drop([0],axis=0,inplace=True)                           #elimino la␣
↪fila 0
   GDP.set_axis(axislist, axis=1, inplace=True)                #coloco lista␣
↪como nuevo axis
   GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
   GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
   GDP=GDP.set_index('Country')
   # print(GDP)
   # print(len(GDP)) #264

   ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
   ScimEn=ScimEn.set_index('Country')
   # print(ScimEn)
   # print(len(ScimEn)) #191

   mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
   df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
   df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
   df.sort_values(by=['Rank'], inplace=True)
   df=df.iloc[0:15]

   lista=[]
   for x in df.columns:                          #quita espacios de las columnas
```

```
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas

    df['Population']=df['Energy Supply']/df['Energy Supply per Capita']
    df['Citable docs per Capita'] = df['Citable documents'] / df['Population']
    corr, pval=stats.pearsonr(df['Citable docs per Capita'],df["Energy Supply␣
↪per Capita"])  #se devuelven dos variables coor, pval
    #print('Correlacion: ',corr)
    #print('pval:        ', pval)


    answer_nine=corr
    return(answer_nine)
answer_nine()
```

[7]:
```
0.7940010435442943
```

[8]:
```
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])
```

[3]:
```
assert answer_nine() >= -1. and answer_nine() <= 1., "Q9: A valid correlation␣
↪should between -1 to 1!"
```

```
Correlacion:  0.7940010435442943
pval:         0.0004083648953039715
Correlacion:  0.7940010435442943
pval:         0.0004083648953039715
```

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.*

[11]:
```
import numpy as np
import re
import pandas as pd
import scipy.stats as stats
```

```
def answer_ten():
    Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name = "Energy")
    Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
    Energy=Energy.drop(range(0,17,1), axis=0)                  #eliino rango␣
↪de filas 0 a 17 (header)
    Energy=Energy.drop(range(244,282,1), axis=0)               #eliino rango␣
↪de filas 244 a 282 (footer)
    Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per Capita',␣
↪'% Renewable'] # Renombro columnas
    listaener=[]
    for k in Energy.columns:
        str(k).strip()
        listaener.append(k)
    Energy.set_axis(listaener, axis=1, inplace=True)
    Energy[Energy == '...'] = np.nan                           #relleno con␣
↪NaN donde haya puntos (...)
    Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
    Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
    Energy['Country']=Energy['Country'].str.rstrip()           #elimino␣
↪espacios que sobran al fin de la cadena
    Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
    Energy['Energy Supply']=Energy['Energy Supply']*1000000    #paso de peta a␣
↪giga joules
    Energy=Energy.set_index('Country')
    # print(Energy)
    # print(len(Energy)) #227

    GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4) #elimino␣
↪header y las 4 priemras filas
    GDP[0]=GDP[0].str.strip()                                  #remuevo␣
↪espacios de la columna 0
    axislist=[]
    for x in GDP.iloc[0]:                                      #creo lista con␣
↪los nombres de las cabeceras
        axislist.append(x)
    for k in range (4,len(axislist),1):
        axislist[k]=str(int(axislist[k]))
    GDP.drop([0],axis=0,inplace=True)                          #elimino la␣
↪fila 0
```

```
    GDP.set_axis(axislist, axis=1, inplace=True)                #coloco lista␣
↪como nuevo axis
    GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio nombre␣
↪de cabecera
    GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic Rep.":
↪ "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
    GDP=GDP.set_index('Country')
    # print(GDP)
    # print(len(GDP)) #264

    ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
    ScimEn=ScimEn.set_index('Country')
    # print(ScimEn)
    # print(len(ScimEn)) #191

    mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
    df=pd.merge(mergedf, GDP, how='outer', left_index=True, right_index=True)  ␣
↪#uno mergedf con Scim dataframe= todos dataframes en una sola= df
    df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
    df.sort_values(by=['Rank'], inplace=True)
    df=df.iloc[0:15]

    lista=[]
    for x in df.columns:                          #quita espacios de las columnas
        str(x).strip()
        lista.append(x)
    df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos nombres␣
↪de columnas

    df['HighRenew']=np.where(df['% Renewable'] >= df['% Renewable'].median(), 1␣
↪, 0)

    answer_ten=df['HighRenew']
    print(type(answer_ten))

    return(answer_ten)
```

[12]:
```
assert type(answer_ten()) == pd.Series, "Q10: You should return a Series!"
```

```
<class 'pandas.core.series.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']

[9]:
```
import numpy as np
import re
import pandas as pd
def answer_eleven():
        Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name =␣
↪"Energy")
        Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino␣
↪columnas por su indice 0 y 1.
        Energy=Energy.drop(range(0,17,1), axis=0)                   #eliino␣
↪rango de filas 0 a 17 (header)
        Energy=Energy.drop(range(244,282,1), axis=0)                #eliino␣
↪rango de filas 244 a 282 (footer)
        Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per␣
↪Capita', '% Renewable'] # Renombro columnas
        listaener=[]
        for k in Energy.columns:
            str(k).strip()
            listaener.append(k)
        Energy.set_axis(listaener, axis=1, inplace=True)
        Energy[Energy == '...'] = np.nan                            #relleno␣
↪con NaN donde haya puntos (...)
        Energy['Country'].replace('[\d*]','', regex=True, inplace=True) ␣
↪#reemplazar con vacio '' donde cadena de numeros
        Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
        Energy['Country']=Energy['Country'].str.rstrip()            #elimino␣
↪espacios que sobran al fin de la cadena
```

```
        Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
        Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de␣
↪peta a giga joules
        Energy=Energy.set_index('Country')
        # print(Energy)
        # print(len(Energy)) #227

        GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4)␣
↪#elimino header y las 4 priemras filas
        GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
        axislist=[]
        for x in GDP.iloc[0]:                                       #creo lista␣
↪con los nombres de las cabeceras
            axislist.append(x)
        for k in range (4,len(axislist),1):
            axislist[k]=str(int(axislist[k]))
        GDP.drop([0],axis=0,inplace=True)                           #elimino la␣
↪fila 0
        GDP.set_axis(axislist, axis=1, inplace=True)                #coloco␣
↪lista como nuevo axis
        GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio␣
↪nombre de cabecera
        GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic␣
↪Rep.": "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
        GDP=GDP.set_index('Country')
        # print(GDP)
        # print(len(GDP)) #264

        ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
        ScimEn=ScimEn.set_index('Country')
        # print(ScimEn)
        # print(len(ScimEn)) #191

        mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
        df=pd.merge(mergedf, GDP, how='outer', left_index=True,␣
↪right_index=True)   #uno mergedf con Scim dataframe= todos dataframes en una␣
↪sola= df
        df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
```

```
        df.sort_values(by=['Rank'], inplace=True)
        df=df.iloc[0:15]

        lista=[]
        for x in df.columns:                          #quita espacios de las␣
↪columnas
            str(x).strip()
            lista.append(x)
        df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos␣
↪nombres de columnas

        df['Population']=df['Energy Supply']/df['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'}
        continents=pd.Series(ContinentDict)
        df['Continent']=continents
        list_continents=continents.unique()
        list_continents.sort()                          #lista continentes

        df.reset_index(inplace = True)
        df.set_index(['Continent','Country'], inplace=True)
        df.sort_values(by=['Continent'], inplace=True)

        size=[]
        sum=[]
        mean=[]
        std=[]
        for i in list_continents:
            size.append(len(df.loc[i]))                 #lista de size
            sum.append(df.loc[i]['Population'].sum())   #lista de sum ␣
↪population
            mean.append(df.loc[i]['Population'].mean()) #lista de mean␣
↪population
```

```
            std.append(df.loc[i]['Population'].std())   #lista de std
 ↪population

        col = {"size": size, "sum": sum, "mean": mean, "std":std}
        answer_eleven=pd.DataFrame(col, index=list_continents)


        return(answer_eleven)
```

[10]:
```
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.

[ ]:
```
import numpy as np
import re
import pandas as pd
def answer_twelve():
        Energy= pd.read_excel("assets/Energy Indicators.xls", sheet_name =
 ↪"Energy")
        Energy=Energy.drop(Energy.columns[[0, 1]], axis='columns')  #elimino
 ↪columnas por su indice 0 y 1.
        Energy=Energy.drop(range(0,17,1), axis=0)                   #eliino
 ↪rango de filas 0 a 17 (header)
        Energy=Energy.drop(range(244,282,1), axis=0)                #eliino
 ↪rango de filas 244 a 282 (footer)
        Energy.columns = ['Country', 'Energy Supply', 'Energy Supply per
 ↪Capita', '% Renewable'] # Renombro columnas
        listaener=[]
        for k in Energy.columns:
            str(k).strip()
            listaener.append(k)
        Energy.set_axis(listaener, axis=1, inplace=True)
        Energy[Energy == '...'] = np.nan                            #relleno
 ↪con NaN donde haya puntos (...)
        Energy['Country'].replace('[\d*]','', regex=True, inplace=True)
 ↪#reemplazar con vacio '' donde cadena de numeros
```

```
        Energy['Country'].replace('\(.*\)','', regex=True, inplace=True)␣
↪#reemplazar con vacio '' donde cadena de numeros ()
        Energy['Country']=Energy['Country'].str.rstrip()            #elimino␣
↪espacios que sobran al fin de la cadena
        Energy['Country'].replace({'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'},␣
↪inplace=True) #reemplazo nombres
        Energy['Energy Supply']=Energy['Energy Supply']*1000000     #paso de␣
↪peta a giga joules
        Energy=Energy.set_index('Country')
        # print(Energy)
        # print(len(Energy)) #227

        GDP= pd.read_csv('assets/world_bank.csv', header=None, skiprows=4)␣
↪#elimino header y las 4 priemras filas
        GDP[0]=GDP[0].str.strip()                                   #remuevo␣
↪espacios de la columna 0
        axislist=[]
        for x in GDP.iloc[0]:                                       #creo lista␣
↪con los nombres de las cabeceras
            axislist.append(x)
        for k in range (4,len(axislist),1):
            axislist[k]=str(int(axislist[k]))
        GDP.drop([0],axis=0,inplace=True)                           #elimino la␣
↪fila 0
        GDP.set_axis(axislist, axis=1, inplace=True)                #coloco␣
↪lista como nuevo axis
        GDP.rename(columns={'Country Name':'Country'}, inplace=True)#cambio␣
↪nombre de cabecera
        GDP['Country'].replace({"Korea, Rep." : "South Korea", "Iran, Islamic␣
↪Rep.": "Iran", "Hong Kong SAR, China": "Hong Kong"}, inplace=True)
        GDP=GDP.set_index('Country')
        # print(GDP)
        # print(len(GDP)) #264

        ScimEn= pd.read_excel("assets/scimagojr-3.xlsx", sheet_name = "Sheet1")
        ScimEn=ScimEn.set_index('Country')
        # print(ScimEn)
        # print(len(ScimEn)) #191

        mergedf=pd.merge(ScimEn, Energy, how='outer', left_index=True,␣
↪right_index=True)  #uno GDP Y Energy dataframes en un solo dataf: mergedf
```

```
        df=pd.merge(mergedf, GDP, how='outer', left_index=True,␣
↪right_index=True)   #uno mergedf con Scim dataframe= todos dataframes en una␣
↪sola= df
        df=df.drop(df.columns[10:59], axis='columns') #elimino columnas que no␣
↪necesito
        df.sort_values(by=['Rank'], inplace=True)
        df=df.iloc[0:15]

        lista=[]
        for x in df.columns:                          #quita espacios de las␣
↪columnas
            str(x).strip()
            lista.append(x)
        df.set_axis(lista, axis=1, inplace=True)  #reestablece los nuevos␣
↪nombres de columnas

        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'}
        continents=pd.Series(ContinentDict)
        df['Continent']=continents
        list_continents=continents.unique()
        list_continents.sort()                #lista continentes


        df.reset_index(inplace = True)
        df.set_index(['Continent','Country'], inplace=True)
        df.sort_values(by=['Continent'], inplace=True)

        print(df)
        return(answer_twelve)
answer_twelve()
```

[ ]:
```
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*

[ ]:
```
def answer_thirteen():
    # YOUR CODE HERE
    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.")
```