Elements Of Data Processing - Week 2

Preliminaries

iPython

iPython is an interactive computing and development environment. Let's start with some basic examples in iPython. Launch iPython on the command line with the `ipython` command:

```
$ ipython
```

Basics

Python objects are formatted to be more readable in iPython:

In [1]:

```
import numpy as np

samples = {i:np.random.randn() for i in range(10)}
```

In [2]:

```
samples
```

Out[2]:

```
{0: -0.056261627008907 71,
 1: -1.8427630631717025,
 2: 1.007379938079604,
 3: -0.3746848703825882,
 4: -0.4496293900160743,
 5: 0.4534167381448738,
 6: 1.0886202998644463,
 7: 0.04953117232735405,
 8: -1.2363679034317279,
 9: -1.9865005261719912}
```

Object Introspection

You can find general information about an object, e.g., a variable, a function or an instance method using a question mark (?)

In [3]:

```
student_names = []
```

In [4]:

```
student_names?
```

In [5]:

```
def add_integers (x,y):
    '''
    (int,int) -> int
    -----------
    Adds two integers as input
    Returns the sum of input arguments
    '''
    return x+y
```

In [6]:

```
add_integers?
```

In [7]:

```
add_integers??
```

Magic Commands

Magic commands are designed to facilitate common tasks.

In [8]:

```
# run a .py file
%run hello.py
```

```
Hello World!
```

In [9]:

```
# check the execution time of a statement
import numpy as np
a = np.random.randn(10,10)
%timeit np.dot(a,a)
```

```
The slowest run took 101.70 times longer than the fastest. This co
uld mean that an intermediate result is being cached.
1000000 loops, best of 3: 1 µs per loop
```

Jupyter Notebook

The advantage of the notebooks (vs terminal) is that you can interactively read the questions, write the answers and see the result. The notebook application runs as server process on the command line. Start running the notebook in the command line:

$ jupyter notebook

Pandas

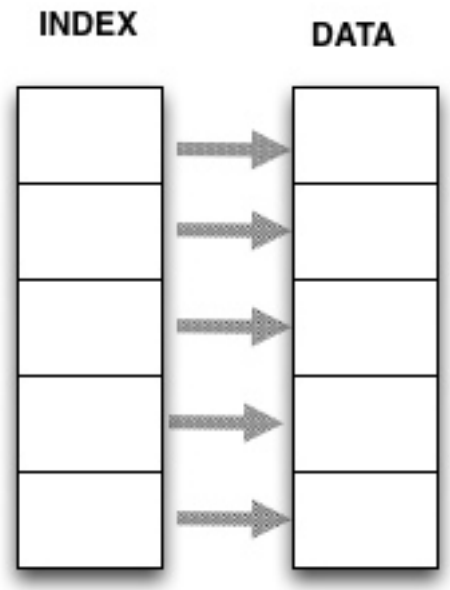
A library that contains high-level data structures and manipulation tools for faster analysis.

In [10]:

```
import pandas as pd
```

Series

One-dimensional array-like object containing the array of data and an associated array of data labels called index.



The basic method to create a Series:

- s = Series(data, index=index)

Here, data can be different things, including:

- a list
- an array
- a dictionary

Example 1 : Create a Basic Series Object

In [11]:

```
# series constructor with data as a list of integers

l = [4,3,-5,9,1,7]
s = pd.Series(l)
```

In [12]:

```
# the default indexing starts from zero
s.index
```

Out[12]:

```
Int64Index([0, 1, 2, 3, 4, 5], dtype='int64')
```

In [13]:

```
# retrieve the values of the series
s.values
```

Out[13]:

```
array([ 4,  3, -5,  9,  1,  7])
```

In [14]:

```
# create your own index using lists
newIndex = ['a','b','c','d','e','f']
s.index  = newIndex
```

In [15]:

```
# verify the index
s
```

Out[15]:

```
a    4
b    3
c   -5
d    9
e    1
f    7
dtype: int64
```

In [16]:

```
# Creating a series from a python dict

Aus_Emission = {'1990':15.45288167, '2000':17.20060983, '2007':17.86526004,
                '2008':18.16087566,'2009':18.20018196,'2010':16.92095367,
                '2011':16.86260095, '2012':16.51938578, '2013':16.34730205}

co2_Emission = pd.Series(Aus_Emission)
```

In [17]:

```
# retrieve the values of the series
co2_Emission.values
```

Out[17]:

```
array([ 15.45288167,  17.20060983,  17.86526004,  18.16087566,
        18.20018196,  16.92095367,  16.86260095,  16.51938578,  16
.34730205])
```

In [18]:

```
# verify the series object
co2_Emission
```

Out[18]:

```
1990    15.452882
2000    17.200610
2007    17.865260
2008    18.160876
2009    18.200182
2010    16.920954
2011    16.862601
2012    16.519386
2013    16.347302
dtype: float64
```

Slicing

You can **select** sections of list-like types (arrays, tuples, NumPy arrays) by using various slice notations:

In [19]:

```
# slicing the series using a boolean array operation
co2_Emission[co2_Emission>16.0]
```

Out[19]:

```
2000    17.200610
2007    17.865260
2008    18.160876
2009    18.200182
2010    16.920954
2011    16.862601
2012    16.519386
2013    16.347302
dtype: float64
```

In [20]:

```
# slicing the series using a time period
co2_Emission[:'2000']
```

Out[20]:

```
1990    15.452882
2000    17.200610
dtype: float64
```

In [21]:

```
# double the values of the series object
doubled = co2_Emission*2
doubled
```

Out[21]:

```
1990    30.905763
2000    34.401220
2007    35.730520
2008    36.321751
2009    36.400364
2010    33.841907
2011    33.725202
2012    33.038772
2013    32.694604
dtype: float64
```

In [22]:

```
# finding the average value of the series
co2_Emission.mean()
```

Out[22]:

```
17.05889462333333
```

In [23]:

```
# defining the column name
co2_Emission.name = 'CO2 Emission'
```

In [24]:

```
# defining the name of the index
co2_Emission.index.name = 'Year'
```

In [25]:

```
# verify the series object
co2_Emission
```

Out[25]:

```
Year
1990    15.452882
2000    17.200610
2007    17.865260
2008    18.160876
2009    18.200182
2010    16.920954
2011    16.862601
2012    16.519386
2013    16.347302
Name: CO2 Emission, dtype: float64
```

Exercise 1

Pandas Series objects have both *ndarray-like* and *dict-like properties*. Given the co2_Emission series object do the following:

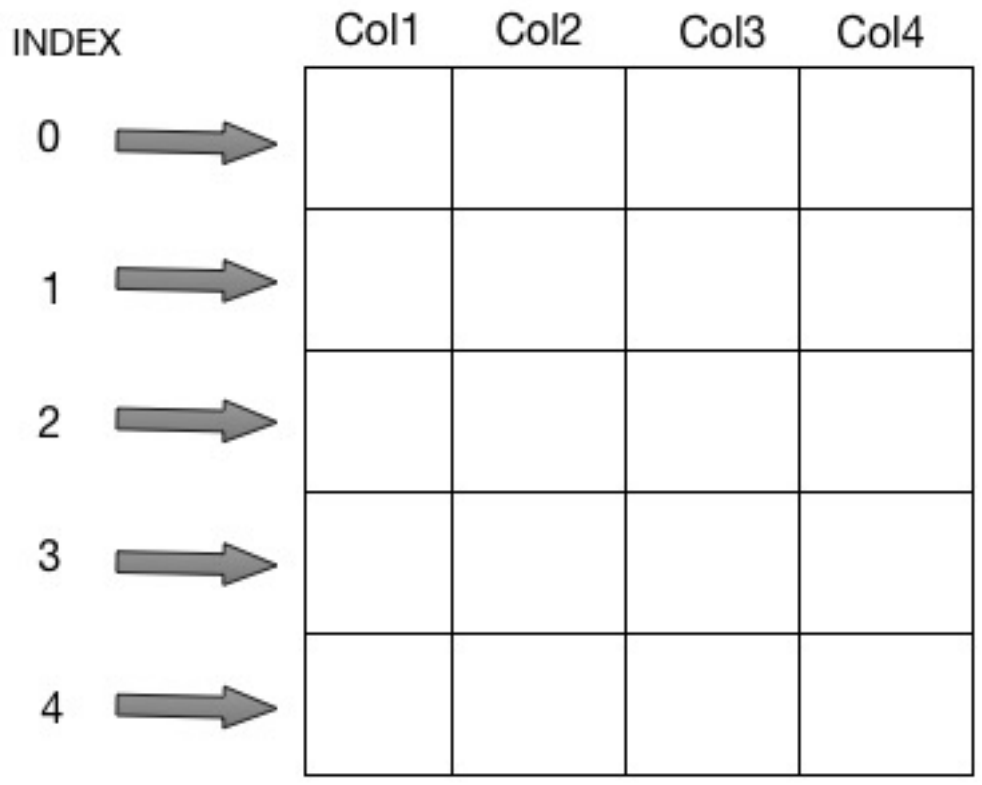
- Similar to the average of the series object, retrieve the maximum, median and cumulative sum of CO2 emission between 1960 to 2013 (max(), median() and cumsum() methods).
- Retrieve the CO2 emissions in Australia between 2000 to 2010.
- Given the population of Australia in 2013 is 23117353, retrieve the CO2 emission per capita for that year.

DataFrames

Represents a tabular data structure

Can be thought of as a dictionary of Series objects

Has both row and column indices



In [26]:

```
# create a new series of the population
Aus_Population = {'1990':17065100, '2000':19153000, '2007':20827600,
                  '2008':21249200,'2009':21691700,'2010':22031750,
                  '2011':22340024, '2012':22728254, '2013':23117353}
population = pd.Series(Aus_Population)
```

In [27]:

```
# verify the values in the series
population
```

Out[27]:

```
1990    17065100
2000    19153000
2007    20827600
2008    21249200
2009    21691700
2010    22031750
2011    22340024
2012    22728254
2013    23117353
dtype: int64
```

In [28]:

```
# create a DataFrame object from the series objects
australia = pd.DataFrame({'co2_emission':co2_Emission, 'Population':population
})
australia
```

Out[28]:

	Population	co2_emission
1990	17065100	15.452882
2000	19153000	17.200610
2007	20827600	17.865260
2008	21249200	18.160876
2009	21691700	18.200182
2010	22031750	16.920954
2011	22340024	16.862601
2012	22728254	16.519386
2013	23117353	16.347302

In [29]:

```
# create a DataFrame from a csv file
countries = pd.read_csv('countries.csv',encoding = 'ISO-8859-1')
```

In [30]:

```
# check the top 10 countries in the DataFrame
countries.head(10) # the default value is set to 5
```

Out[30]:

	Country	Region	IncomeGroup
0	Afghanistan	South Asia	Low income
1	Albania	Europe & Central Asia	Upper middle income
2	Algeria	Middle East & North Africa	Upper middle income
3	American Samoa	East Asia & Pacific	Upper middle income
4	Andorra	Europe & Central Asia	High income
5	Angola	Sub-Saharan Africa	Upper middle income
6	Antigua and Barbuda	Latin America & Caribbean	High income
7	Argentina	Latin America & Caribbean	Upper middle income
8	Armenia	Europe & Central Asia	Lower middle income
9	Aruba	Latin America & Caribbean	High income

In [31]:

```
# count the number of countries in each region
countries.Region.value_counts()
```

Out[31]:

```
Europe & Central Asia         58
Sub-Saharan Africa            48
Latin America & Caribbean     42
East Asia & Pacific           37
Middle East & North Africa    21
South Asia                     8
North America                  3
Name: Region, dtype: int64
```

In [32]:

```
# set the name of countries as the index
countries.set_index('Country')
```

Out[32]:

	Region	IncomeGroup
Country		
Afghanistan	South Asia	Low income
Albania	Europe & Central Asia	Upper middle income
Algeria	Middle East & North Africa	Upper middle income

American Samoa	East Asia & Pacific	Upper middle income
Andorra	Europe & Central Asia	High income
Angola	Sub-Saharan Africa	Upper middle income
Antigua and Barbuda	Latin America & Caribbean	High income
Argentina	Latin America & Caribbean	Upper middle income
Armenia	Europe & Central Asia	Lower middle income
Aruba	Latin America & Caribbean	High income
Australia	East Asia & Pacific	High income
Austria	Europe & Central Asia	High income
Azerbaijan	Europe & Central Asia	Upper middle income
Bahamas, The	Latin America & Caribbean	High income
Bahrain	Middle East & North Africa	High income
Bangladesh	South Asia	Lower middle income
Barbados	Latin America & Caribbean	High income
Belarus	Europe & Central Asia	Upper middle income
Belgium	Europe & Central Asia	High income
Belize	Latin America & Caribbean	Upper middle income
Benin	Sub-Saharan Africa	Low income
Bermuda	North America	High income
Bhutan	South Asia	Lower middle income
Bolivia	Latin America & Caribbean	Lower middle income
Bosnia and Herzegovina	Europe & Central Asia	Upper middle income
Botswana	Sub-Saharan Africa	Upper middle income
Brazil	Latin America & Caribbean	Upper middle income
British Virgin Islands	Latin America & Caribbean	High income
Brunei Darussalam	East Asia & Pacific	High income
Bulgaria	Europe & Central Asia	Upper middle income
...	...	...
Sweden	Europe & Central Asia	High income
Switzerland	Europe & Central Asia	High income
Syrian Arab Republic	Middle East & North Africa	Lower middle income
Tajikistan	Europe & Central Asia	Lower middle income
Tanzania	Sub-Saharan Africa	Low income
Thailand	East Asia & Pacific	Upper middle income

Timor-Leste	East Asia & Pacific	Lower middle income
Togo	Sub-Saharan Africa	Low income
Tonga	East Asia & Pacific	Lower middle income
Trinidad and Tobago	Latin America & Caribbean	High income
Tunisia	Middle East & North Africa	Lower middle income
Turkey	Europe & Central Asia	Upper middle income
Turkmenistan	Europe & Central Asia	Upper middle income
Turks and Caicos Islands	Latin America & Caribbean	High income
Tuvalu	East Asia & Pacific	Upper middle income
Uganda	Sub-Saharan Africa	Low income
Ukraine	Europe & Central Asia	Lower middle income
United Arab Emirates	Middle East & North Africa	High income
United Kingdom	Europe & Central Asia	High income
United States	North America	High income
Uruguay	Latin America & Caribbean	High income
Uzbekistan	Europe & Central Asia	Lower middle income
Vanuatu	East Asia & Pacific	Lower middle income
Venezuela, RB	Latin America & Caribbean	Upper middle income
Vietnam	East Asia & Pacific	Lower middle income
Virgin Islands (U.S.)	Latin America & Caribbean	High income
West Bank and Gaza	Middle East & North Africa	Lower middle income
Yemen, Rep.	Middle East & North Africa	Lower middle income
Zambia	Sub-Saharan Africa	Lower middle income
Zimbabwe	Sub-Saharan Africa	Low income

217 rows × 2 columns

In [33]:

```
# create a new DataFrame for the CO2 emission from a csv file
emission = pd.read_csv('emission.csv',encoding = 'ISO-8859-1')
emission.head()
```

Out[33]:

	Country	1990	2000	2007	2008	2009	2010	2011
0	Afghanistan	0.216661	0.039272	0.087858	0.158962	0.249074	0.302936	0.42526
1	Albania	1.615624	0.978175	1.322335	1.484311	1.495600	1.578574	1.80397
2	Algeria	3.007911	2.819778	3.195865	3.168524	3.430129	3.307164	3.30055
3	American Samoa	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Andorra	NaN	8.018181	6.350868	6.296125	6.049173	6.124770	5.96868

In [34]:

```
# Create a subset of emission dataset for Year 2010
yr2010 = emission['2010']
names  = emission['Country']
yr2010.index = names
type(yr2010)
```

Out[34]:

```
pandas.core.series.Series
```

In [35]:

```
# Sort column values using sort_values
yr2010.sort_values()
```

Out[35]:

```
Country
Burundi                        0.022480
Congo, Dem. Rep.               0.030197
Chad                           0.043463
Rwanda                         0.057354
Central African Republic       0.059398
Mali                           0.063586
Somalia                        0.063912
Niger                          0.072026
Ethiopia                       0.075215
Malawi                         0.080690
Madagascar                     0.092895
Eritrea                        0.109471
Mozambique                     0.112476
Uganda                         0.118253
Burkina Faso                   0.125501
Sierra Leone                   0.125706
Guinea-Bissau                  0.145855
```

```
Tanzania                        0.155682

Comoros                         0.183692
Nepal                           0.188153
Zambia                          0.194713
Liberia                         0.201046
Haiti                           0.212694
Timor-Leste                     0.220073
Guinea                          0.236422
Myanmar                         0.241924
Lao PDR                         0.261822
Gambia, The                     0.279411
Kenya                           0.301883
Afghanistan                     0.302936
                                   ...
Canada                         14.485639
Kazakhstan                     15.110081
Gibraltar                      15.153879
Australia                      16.920954
Oman                           17.112061
United States                  17.484803
Saudi Arabia                   18.531173
United Arab Emirates           19.306504
Brunei Darussalam              20.856947
Luxembourg                     21.635136
Bahrain                        23.101200
Aruba                          24.182702
Kuwait                         29.294309
Trinidad and Tobago            36.073741
Qatar                          41.131162
American Samoa                       NaN
Channel Islands                      NaN
Curacao                              NaN
Guam                                 NaN
Isle of Man                          NaN
Kosovo                               NaN
Monaco                               NaN
Northern Mariana Islands             NaN
Puerto Rico                          NaN
San Marino                           NaN
Sint Maarten (Dutch part)            NaN
South Sudan                          NaN
St. Martin (French part)             NaN
Tuvalu                               NaN
Virgin Islands (U.S.)                NaN
Name: 2010, dtype: float64
```

In [36]:

```
#Sort column values to find the top countries
yr2010.sort_values(ascending = False)
```

Out[36]:

```
Country
Qatar                          41.131162
Trinidad and Tobago            36.073741
Kuwait                         29.294309
```

Aruba	24.182702
Bahrain	23.101200
Luxembourg	21.635136
Brunei Darussalam	20.856947
United Arab Emirates	19.306504
Saudi Arabia	18.531173
United States	17.484803
Oman	17.112061
Australia	16.920954
Gibraltar	15.153879
Kazakhstan	15.110081
Canada	14.485639
New Caledonia	14.169288
Estonia	13.599689
Faroe Islands	12.986678
Norway	12.294955
Russian Federation	11.725820
Greenland	11.663773
Finland	11.543147
Korea, Rep.	11.469587
Turkmenistan	11.362475
Netherlands	10.932543
Czech Republic	10.652579
Palau	10.569272
Cayman Islands	10.173449
Belgium	9.889429
Libya	9.791235
	...
Burkina Faso	0.125501
Uganda	0.118253
Mozambique	0.112476
Eritrea	0.109471
Madagascar	0.092895
Malawi	0.080690
Ethiopia	0.075215
Niger	0.072026
Somalia	0.063912
Mali	0.063586
Central African Republic	0.059398
Rwanda	0.057354
Chad	0.043463
Congo, Dem. Rep.	0.030197
Burundi	0.022480
American Samoa	NaN
Channel Islands	NaN
Curacao	NaN
Guam	NaN
Isle of Man	NaN
Kosovo	NaN
Monaco	NaN
Northern Mariana Islands	NaN
Puerto Rico	NaN
San Marino	NaN
Sint Maarten (Dutch part)	NaN
South Sudan	NaN
St. Martin (French part)	NaN
Tuvalu	NaN

```
Virgin Islands (U.S.)                NaN
Name: 2010, dtype: float64
```

Exercise 2

- Retrieve the mean, median of CO2 emission generated in 2012 by all countries.
- Retrieve the top 5 countries with the most CO2 emission in 2012. How about the 5 countries with the least emission? (remember that sort_values has an **ascending** parameter that is set to True by default).
- Retrieve the sum of CO2 emission for all years and find the 2 years with the maximum CO2 emission.

More Sort Operations

In [37]:

```
# Sort column values of a DataFrame
sorted2012 = emission.sort_values( by = '2012',ascending = False )
sorted2012
```

Out[37]:

	Country	1990	2000	2007	2008	2009	2010
158	Qatar	24.712027	58.522169	53.672746	46.684906	43.504331	41.131
50	Curacao	NaN	NaN	NaN	NaN	NaN	NaN
196	Trinidad and Tobago	13.879875	18.844281	36.816763	35.455298	33.952984	36.073
105	Kuwait	23.466084	27.759023	29.636849	30.581264	30.300789	29.294
28	Brunei Darussalam	24.105188	14.255144	22.474463	23.950011	20.311171	20.856
14	Bahrain	25.058198	27.956644	26.001291	26.538479	22.815070	23.101
171	Sint Maarten (Dutch part)	NaN	NaN	NaN	NaN	NaN	NaN
115	Luxembourg	26.197659	18.885512	22.957283	22.385472	20.877125	21.635
204	United Arab Emirates	28.711160	36.904101	22.566623	22.804551	21.797583	19.306
165	Saudi Arabia	13.320831	13.880494	14.866777	16.085642	17.102371	18.531
147	Oman	6.283132	9.777453	16.846640	16.101568	15.963375	17.112
10	Australia	15.452882	17.200610	17.865260	18.160876	18.200182	16.920
206	United States	19.323368	20.207615	19.237460	18.489234	17.192367	17.484
74	Gibraltar	5.329743	12.200541	13.919080	14.137127	15.195251	15.153

99	Kazakhstan	NaN	7.934854	14.359712	11.879265	10.309399	15.110
140	New Caledonia	9.269475	10.473212	12.372273	11.849562	12.138518	14.169
35	Canada	15.659070	17.370452	16.446455	16.215164	14.837315	14.485
62	Estonia	NaN	10.636252	13.954884	13.081834	10.914320	13.599
160	Russian Federation	NaN	10.627121	11.672457	12.014507	11.023856	11.725
9	Aruba	27.850035	25.547679	25.613715	24.750133	24.876706	24.182
199	Turkmenistan	NaN	8.339388	11.535622	11.556826	10.110661	11.362
64	Faroe Islands	12.826216	14.828590	14.185395	12.970121	11.842902	12.986
103	Korea, Rep.	5.760374	9.520932	10.199582	10.369833	10.346506	11.469
131	Mongolia	4.573372	3.130988	4.653094	4.578057	4.907974	9.0909
149	Palau	15.553582	5.928706	10.389651	10.333663	10.274233	10.569
170	Singapore	15.405065	12.166624	4.342606	7.466762	7.422120	8.6562
139	Netherlands	10.594475	10.383090	10.468447	10.541492	10.234971	10.932
76	Greenland	9.958939	9.461121	11.282079	11.718151	10.221739	11.663
146	Norway	7.429148	8.834016	9.574850	11.679583	11.461829	12.294
97	Japan	8.873293	9.622352	9.782964	9.449534	8.619442	9.1500
...	...	...	...	...	...	...	...
169	Sierra Leone	0.124061	0.104753	0.119034	0.120200	0.115584	0.1257
30	Burkina Faso	0.066173	0.089717	0.115429	0.130385	0.127425	0.1255
81	Guinea-Bissau	0.163177	0.111505	0.151257	0.145619	0.146971	0.1458
134	Mozambique	0.076510	0.073884	0.101189	0.098553	0.106996	0.1124
177	South Sudan	NaN	NaN	NaN	NaN	NaN	NaN
118	Madagascar	0.082260	0.119013	0.093705	0.094772	0.086416	0.0928
202	Uganda	0.043664	0.060197	0.096346	0.102865	0.105434	0.1182
143	Niger	0.092696	0.062072	0.049726	0.053722	0.061771	0.0720
63	Ethiopia	0.062799	0.053424	0.074345	0.079184	0.077723	0.0752
119	Malawi	0.065085	0.079937	0.069546	0.078854	0.070120	0.0806
161	Rwanda	0.074757	0.065826	0.058789	0.055661	0.057431	0.0573
175	Somalia	0.002900	0.070009	0.068327	0.065851	0.063881	0.0639
37	Central African	0.067403	0.071843	0.060213	0.059112	0.058013	0.0593

	Republic						
122	Mali	0.049717	0.074356	0.073291	0.075282	0.054901	0.0635
38	Chad	0.024619	0.021097	0.042863	0.045756	0.042689	0.0434
44	Congo, Dem. Rep.	0.116420	0.016943	0.028927	0.029605	0.026937	0.0301
31	Burundi	0.052263	0.042809	0.021964	0.021615	0.020868	0.0224
3	American Samoa	NaN	NaN	NaN	NaN	NaN	NaN
39	Channel Islands	NaN	NaN	NaN	NaN	NaN	NaN
61	Eritrea	NaN	0.172191	0.131490	0.092069	0.111761	0.1094
78	Guam	NaN	NaN	NaN	NaN	NaN	NaN
93	Isle of Man	NaN	NaN	NaN	NaN	NaN	NaN
104	Kosovo	NaN	NaN	NaN	NaN	NaN	NaN
130	Monaco	NaN	NaN	NaN	NaN	NaN	NaN
145	Northern Mariana Islands	NaN	NaN	NaN	NaN	NaN	NaN
157	Puerto Rico	NaN	NaN	NaN	NaN	NaN	NaN
163	San Marino	NaN	NaN	NaN	NaN	NaN	NaN
182	St. Martin (French part)	NaN	NaN	NaN	NaN	NaN	NaN
201	Tuvalu	NaN	NaN	NaN	NaN	NaN	NaN
212	Virgin Islands (U.S.)	NaN	NaN	NaN	NaN	NaN	NaN

217 rows × 13 columns

In [38]:

```
# Sort column values using two columns
sorted2012 = emission.sort_values( by = ['2012','2013'],ascending = [False, True] )
sorted2012
```

Out[38]:

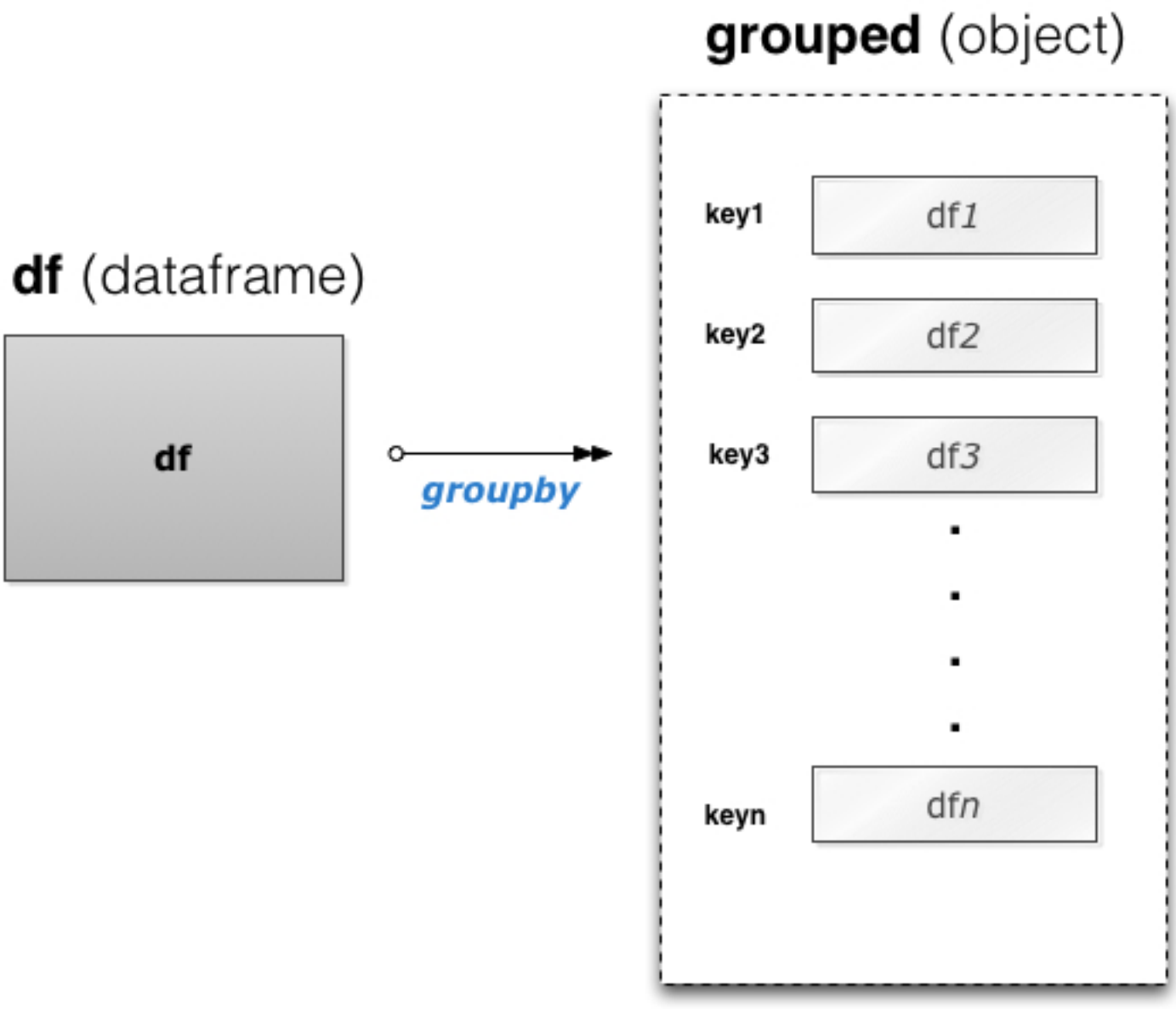
	Country	1990	2000	2007	2008	2009	2010
158	Qatar	24.712027	58.522169	53.672746	46.684906	43.504331	41.131
50	Curacao	NaN	NaN	NaN	NaN	NaN	NaN
196	Trinidad and	13.879875	18.844281	36.816763	35.455298	33.952984	36.073

	Tobago						
105	Kuwait	23.466084	27.759023	29.636849	30.581264	30.300789	29.294
28	Brunei Darussalam	24.105188	14.255144	22.474463	23.950011	20.311171	20.856
14	Bahrain	25.058198	27.956644	26.001291	26.538479	22.815070	23.101
171	Sint Maarten (Dutch part)	NaN	NaN	NaN	NaN	NaN	NaN
115	Luxembourg	26.197659	18.885512	22.957283	22.385472	20.877125	21.635
204	United Arab Emirates	28.711160	36.904101	22.566623	22.804551	21.797583	19.306
165	Saudi Arabia	13.320831	13.880494	14.866777	16.085642	17.102371	18.531
147	Oman	6.283132	9.777453	16.846640	16.101568	15.963375	17.112
10	Australia	15.452882	17.200610	17.865260	18.160876	18.200182	16.920
206	United States	19.323368	20.207615	19.237460	18.489234	17.192367	17.484
74	Gibraltar	5.329743	12.200541	13.919080	14.137127	15.195251	15.153
99	Kazakhstan	NaN	7.934854	14.359712	11.879265	10.309399	15.110
140	New Caledonia	9.269475	10.473212	12.372273	11.849562	12.138518	14.169
35	Canada	15.659070	17.370452	16.446455	16.215164	14.837315	14.485
62	Estonia	NaN	10.636252	13.954884	13.081834	10.914320	13.599
160	Russian Federation	NaN	10.627121	11.672457	12.014507	11.023856	11.725
9	Aruba	27.850035	25.547679	25.613715	24.750133	24.876706	24.182
199	Turkmenistan	NaN	8.339388	11.535622	11.556826	10.110661	11.362
64	Faroe Islands	12.826216	14.828590	14.185395	12.970121	11.842902	12.986
103	Korea, Rep.	5.760374	9.520932	10.199582	10.369833	10.346506	11.469
131	Mongolia	4.573372	3.130988	4.653094	4.578057	4.907974	9.0909
149	Palau	15.553582	5.928706	10.389651	10.333663	10.274233	10.569
170	Singapore	15.405065	12.166624	4.342606	7.466762	7.422120	8.6562
139	Netherlands	10.594475	10.383090	10.468447	10.541492	10.234971	10.932
76	Greenland	9.958939	9.461121	11.282079	11.718151	10.221739	11.663
146	Norway	7.429148	8.834016	9.574850	11.679583	11.461829	12.294
97	Japan	8.873293	9.622352	9.782964	9.449534	8.619442	9.1500
...	...	...	...	...	...	...	...

169	Sierra Leone	0.124061	0.104753	0.119034	0.120200	0.115584	0.1257
30	Burkina Faso	0.066173	0.089717	0.115429	0.130385	0.127425	0.1255
81	Guinea-Bissau	0.163177	0.111505	0.151257	0.145619	0.146971	0.1458
134	Mozambique	0.076510	0.073884	0.101189	0.098553	0.106996	0.1124
177	South Sudan	NaN	NaN	NaN	NaN	NaN	NaN
118	Madagascar	0.082260	0.119013	0.093705	0.094772	0.086416	0.0928
202	Uganda	0.043664	0.060197	0.096346	0.102865	0.105434	0.1182
143	Niger	0.092696	0.062072	0.049726	0.053722	0.061771	0.0720
63	Ethiopia	0.062799	0.053424	0.074345	0.079184	0.077723	0.0752
119	Malawi	0.065085	0.079937	0.069546	0.078854	0.070120	0.0806
161	Rwanda	0.074757	0.065826	0.058789	0.055661	0.057431	0.0573
175	Somalia	0.002900	0.070009	0.068327	0.065851	0.063881	0.0639
37	Central African Republic	0.067403	0.071843	0.060213	0.059112	0.058013	0.0593
122	Mali	0.049717	0.074356	0.073291	0.075282	0.054901	0.0635
38	Chad	0.024619	0.021097	0.042863	0.045756	0.042689	0.0434
44	Congo, Dem. Rep.	0.116420	0.016943	0.028927	0.029605	0.026937	0.0301
31	Burundi	0.052263	0.042809	0.021964	0.021615	0.020868	0.0224
3	American Samoa	NaN	NaN	NaN	NaN	NaN	NaN
39	Channel Islands	NaN	NaN	NaN	NaN	NaN	NaN
61	Eritrea	NaN	0.172191	0.131490	0.092069	0.111761	0.1094
78	Guam	NaN	NaN	NaN	NaN	NaN	NaN
93	Isle of Man	NaN	NaN	NaN	NaN	NaN	NaN
104	Kosovo	NaN	NaN	NaN	NaN	NaN	NaN
130	Monaco	NaN	NaN	NaN	NaN	NaN	NaN
145	Northern Mariana Islands	NaN	NaN	NaN	NaN	NaN	NaN
157	Puerto Rico	NaN	NaN	NaN	NaN	NaN	NaN
163	San Marino	NaN	NaN	NaN	NaN	NaN	NaN
	St. Martin						

182	(French part)	NaN	NaN	NaN	NaN	NaN	NaN
201	Tuvalu	NaN	NaN	NaN	NaN	NaN	NaN
212	Virgin Islands (U.S.)	NaN	NaN	NaN	NaN	NaN	NaN

Groupby



In [39]:

```
# Groupby using the Region column
grouped = countries.groupby('Region')
type(grouped)
```

Out[39]:

```
pandas.core.groupby.DataFrameGroupBy
```

In [40]:

```
# find the size of each group
for k,group in grouped:
        print (k)
        print (group.size)
```

```
East Asia & Pacific
111
Europe & Central Asia
174
Latin America & Caribbean
126
Middle East & North Africa
63
North America
9
South Asia
24
Sub-Saharan Africa
144
```

In [41]:

```
# Find the number of high income and low income countries by region
for k,group in grouped:
        print (k)
        print (group['IncomeGroup'].value_counts())
```

```
East Asia & Pacific
Lower middle income    15
High income            13
Upper middle income     8
Low income              1
Name: IncomeGroup, dtype: int64
Europe & Central Asia
High income            37
Upper middle income    14
Lower middle income     7
Name: IncomeGroup, dtype: int64
Latin America & Caribbean
Upper middle income    20
High income            16
Lower middle income     5
Low income              1
Name: IncomeGroup, dtype: int64
Middle East & North Africa
High income            8
Lower middle income    7
Upper middle income    6
Name: IncomeGroup, dtype: int64
North America
High income    3
Name: IncomeGroup, dtype: int64
South Asia
Lower middle income    5
Low income             2
Upper middle income    1
Name: IncomeGroup, dtype: int64
Sub-Saharan Africa
Low income             27
Lower middle income    13
Upper middle income     7
High income             1
Name: IncomeGroup, dtype: int64
```

Slicing using the .ix method

In [42]:

```
# Slicing using a range of rows and range of columns
emission.ix[2:5,2:6]
```

Out[42]:

	2000	2007	2008	2009
2	2.819778	3.195865	3.168524	3.430129
3	NaN	NaN	NaN	NaN
4	8.018181	6.350868	6.296125	6.049173
5	0.633625	1.311096	1.295687	1.354389

In [43]:

```
# Slicing using specific rows and specific columns
emission.ix[[3,5],['Country','1990']]
```

Out[43]:

	Country	1990
3	American Samoa	NaN
5	Angola	0.459698

In [44]:

```
# Specific rows and all columns

emission.ix[[3,5],:]
```

Out[44]:

	Country	1990	2000	2007	2008	2009	2010	2011
3	American Samoa	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	Angola	0.459698	0.633625	1.311096	1.295687	1.354389	1.369339	1.382752

In [45]:

```
# All rows and specific columns
emission.ix[:,['Country','1990']]
```

Out[45]:

	Country	1990
0	Afghanistan	0.216661
1	Albania	1.615624
2	Algeria	3.007911
3	American Samoa	NaN
4	Andorra	NaN
5	Angola	0.459698
6	Antigua and Barbuda	4.857267
7	Argentina	3.440711
8	Armenia	NaN
9	Aruba	27.850035
10	Australia	15.452882
11	Austria	7.513246

12	Azerbaijan	NaN
13	Bahamas, The	7.610436
14	Bahrain	25.058198
15	Bangladesh	0.146565
16	Barbados	4.126491
17	Belarus	NaN
18	Belgium	10.639672
19	Belize	1.661912
20	Benin	0.141510
21	Bermuda	10.013384
22	Bhutan	0.239671
23	Bolivia	0.797982
24	Bosnia and Herzegovina	NaN
25	Botswana	1.958655
26	Brazil	1.388940
27	British Virgin Islands	4.010329
28	Brunei Darussalam	24.105188
29	Bulgaria	8.690224
...	...	...
187	Sweden	6.069368
188	Switzerland	6.345085
189	Syrian Arab Republic	3.000384
190	Tajikistan	NaN
191	Tanzania	0.089017
192	Thailand	1.604832
193	Timor-Leste	NaN
194	Togo	0.204317
195	Tonga	0.809305
196	Trinidad and Tobago	13.879875
197	Tunisia	1.627000
198	Turkey	2.701355
199	Turkmenistan	NaN
200	Turks and Caicos Islands	NaN
201	Tuvalu	NaN

202	Uganda	0.043664
203	Ukraine	NaN
204	United Arab Emirates	28.711160
205	United Kingdom	9.710882
206	United States	19.323368
207	Uruguay	1.284045
208	Uzbekistan	NaN
209	Vanuatu	0.450144
210	Venezuela, RB	6.150573
211	Vietnam	0.324281
212	Virgin Islands (U.S.)	NaN
213	West Bank and Gaza	NaN
214	Yemen, Rep.	0.802312
215	Zambia	0.300362
216	Zimbabwe	1.478723

217 rows × 2 columns

Exercise 3

- Create a DataFrame object that has the name, region and IncomeGroup of the top 10 emitting countries in 2012.

If time permits, put all together:

Study the affect of population size on CO2 emission.

```
 - Create a new DataFrame object using  pd.read_csv('population.csv',in
dex_col = 'Country', encoding = 'ISO-8859-1').
 - Select ['Canada', 'United States', 'China', 'Australia'] and compare
their population growth. Use the following formula:
```

$$growth = \frac{1}{period} * \frac{(value_e - value_s) * 100}{value_e}$$

```
 - Compute the sum and mean of CO2 emission for the same countries.
 - Does an increase in population lead to an increase in the CO2 emissi
on for all of these countries?
 - Find the top 10 emitting (per capita) countries in each region for 2
010.
 - Is there any interesting trend in these countries with regard to the
ir IncomeGroup?
```

If you are bored:

- Create a Series object where the indices are the letters in the alphabet and the values of its **count** column is the number of countries that their name starts with that letter

In [ ]: