DATAFRAMES with Pandas

In [ ]:

```
import pandas as pd
import numpy as np
```

We'll keep our analysis of G7 countries and looking now at DataFrames. As said, a DataFrame looks a lot like a table. Creating DataFrames manually can be tedious. 99% of the time you'll be pulling the data from a Database, a csv file or the web. But still, you can create a DataFrame by specifying the columns and values

In [ ]:

```
df = pd.DataFrame({
    'Population': [35.467, 63.951, 80.94 , 60.665, 127.061, 64.511, 318.523],
    'GDP': [
        1785387,
        2833687,
        3874437,
        2167744,
        4602367,
        2950039,
        17348075
    ],
    'Surface Area': [
        9984670,
        640679,
        357114,
        301336,
        377930,
        242495,
        9525067
    ],
    'HDI': [
        0.913,
        0.888,
        0.916,
        0.873,
        0.891,
        0.907,
        0.915
    ],
    'Continent': [
        'America',
        'Europe',
        'Europe',
        'Europe',
        'Asia',
        'Europe',
        'America'
    ]
}, columns=['Population', 'GDP', 'Surface Area', 'HDI', 'Continent'])
```

In [ ]:

```
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent
0	35.467	1785387	9984670	0.913	America
1	63.951	2833687	640679	0.888	Europe
2	80.940	3874437	357114	0.916	Europe
3	60.665	2167744	301336	0.873	Europe
4	127.061	4602367	377930	0.891	Asia
5	64.511	2950039	242495	0.907	Europe
6	318.523	17348075	9525067	0.915	America

A DataFrame column will be a pandas series. So we can think of a Dataframe as a combination of series

In [ ]:

```
df.index=[
    'Canada',
    'France',
    'Germany',
    'Italy',
    'Japan',
    'United Kingdom',
    'United States'
]
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent
Canada	35.467	1785387	9984670	0.913	America
France	63.951	2833687	640679	0.888	Europe
Germany	80.940	3874437	357114	0.916	Europe
Italy	60.665	2167744	301336	0.873	Europe
Japan	127.061	4602367	377930	0.891	Asia
United Kingdom	64.511	2950039	242495	0.907	Europe
United States	318.523	17348075	9525067	0.915	America

In [ ]:

```
df.info() #gives all information
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7 entries, Canada to United States
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Population    7 non-null      float64
 1   GDP           7 non-null      int64  
 2   Surface Area  7 non-null      int64  
 3   HDI           7 non-null      float64
 4   Continent     7 non-null      object 
dtypes: float64(2), int64(2), object(1)
memory usage: 336.0+ bytes
```

In [ ]:

```
print(df.size,'\n',df.shape)
```

```
35
 (7, 5)
```

In [ ]:

```
df.describe() #gives summary of statistics of numerical columns
```

Out[ ]:

	Population	GDP	Surface Area	HDI
count	7.000000	7.000000e+00	7.000000e+00	7.000000
mean	107.302571	5.080248e+06	3.061327e+06	0.900429
std	97.249970	5.494020e+06	4.576187e+06	0.016592
min	35.467000	1.785387e+06	2.424950e+05	0.873000
25%	62.308000	2.500716e+06	3.292250e+05	0.889500
50%	64.511000	2.950039e+06	3.779300e+05	0.907000
75%	104.000500	4.238402e+06	5.082873e+06	0.914000
max	318.523000	1.734808e+07	9.984670e+06	0.916000

In [ ]:

```
df.dtypes
```

Out[ ]:

```
Population      float64
GDP               int64
Surface Area      int64
HDI             float64
Continent        object
dtype: object
```

In [ ]:

```
df.dtypes.value_counts()
```

Out[ ]:

```
int64      2
float64    2
object     1
dtype: int64
```

INDEXING, SLICING AND SELECTION:

remember that each column is represented as a Series.

In [ ]:

```
df.loc['Canada'] #selection by index!
```

Out[ ]:

```
Population       35.467
GDP             1785387
Surface Area    9984670
HDI               0.913
Continent       America
Name: Canada, dtype: object
```

In [ ]:

```
df.iloc[-2] #United Kingdom # works with the numeric position
```

Out[ ]:

```
Population       64.511
GDP             2950039
Surface Area     242495
HDI               0.907
Continent        Europe
Name: United Kingdom, dtype: object
```

In [ ]:

```
df['Population'] #accessing a certain column
```

Out[ ]:

```
Canada             35.467
France             63.951
Germany            80.940
Italy              60.665
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: Population, dtype: float64
```

In [ ]:

```
df['GDP']
```

Out[ ]:

```
Canada             1785387
France             2833687
Germany            3874437
Italy              2167744
Japan              4602367
United Kingdom     2950039
United States     17348075
Name: GDP, dtype: int64
```

In [ ]:

```
df['Population'].to_frame() #converts series into dataframe
```

Out[ ]:

	Population
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

Slicing works at row level

In [ ]:

```
df[1:3] #row level selection-> prefer using loc and iloc
```

Out[ ]:

	Population	**GDP**	**Surface Area**	**HDI**	**Continent**
France	63.951	2833687	640679	0.888	Europe
Germany	80.940	3874437	357114	0.916	Europe

In [ ]:

```
df[['Population','Surface Area']]
```

Out[ ]:

	Population	Surface Area
Canada	35.467	9984670
France	63.951	640679
Germany	80.940	357114
Italy	60.665	301336
Japan	127.061	377930
United Kingdom	64.511	242495
United States	318.523	9525067

In [ ]:

```
df.loc['France':'Japan','Surface Area']
```

Out[ ]:

```
France      640679
Germany     357114
Italy       301336
Japan       377930
Name: Surface Area, dtype: int64
```

In [ ]:

```
df.iloc[[0,1,-2]]
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent
Canada	35.467	1785387	9984670	0.913	America
France	63.951	2833687	640679	0.888	Europe
United Kingdom	64.511	2950039	242495	0.907	Europe

In [ ]:

```
df.iloc[1:3,[1,2]] #row wise, dataframe ignores the upper limit
```

Out[ ]:

	GDP	Surface Area
France	2833687	640679
Germany	3874437	357114

CONDITIONAL SELECTION

In [ ]:

```
df['Population']>70
```

Out[ ]:

```
Canada            False
France            False
Germany            True
Italy             False
Japan              True
United Kingdom    False
United States      True
Name: Population, dtype: bool
```

In [ ]:

```
df.loc[df['Population']>70]
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent
Germany	80.940	3874437	357114	0.916	Europe
Japan	127.061	4602367	377930	0.891	Asia
United States	318.523	17348075	9525067	0.915	America

In [ ]:

```
df.loc[df['Population']>70,'Population'] # will give us the population, but now in
                                         # in data frame format.
```

Out[ ]:

```
Germany           80.940
Japan            127.061
United States    318.523
Name: Population, dtype: float64
```

DROPPING STUFF

In [ ]:

```
df.drop('Canada') #can drop multiple as well by df.drop(['Canada','Japan'])
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent
France	63.951	2833687	640679	0.888	Europe
Germany	80.940	3874437	357114	0.916	Europe
Italy	60.665	2167744	301336	0.873	Europe
Japan	127.061	4602367	377930	0.891	Asia
United Kingdom	64.511	2950039	242495	0.907	Europe
United States	318.523	17348075	9525067	0.915	America

In [ ]:

```
df.drop(columns=['Population','HDI']) #we can also use axis like we used in numpy,
```

Out[ ]:

	GDP	Surface Area	Continent
Canada	1785387	9984670	America
France	2833687	640679	Europe
Germany	3874437	357114	Europe
Italy	2167744	301336	Europe
Japan	4602367	377930	Asia
United Kingdom	2950039	242495	Europe
United States	17348075	9525067	America

Operations with Series

working at a column level

In [ ]:

```
crisis=pd.Series([-1000000, -0.3], index=['GDP','HDI'])
```

In [ ]:

```
df[['GDP','HDI']]+crisis #gets subtracted from all!
```

Out[ ]:

	GDP	HDI
Canada	785387.0	0.613
France	1833687.0	0.588
Germany	2874437.0	0.616
Italy	1167744.0	0.573
Japan	3602367.0	0.591
United Kingdom	1950039.0	0.607
United States	16348075.0	0.615

MODIYING DATAFRAMES:

In [ ]:

```
#ADDING A NEW COLUMN:
langs=pd.Series(['French','German','Italian'], index=['France','Germany','Italy'],
```

In [ ]:

```
df['Language']=langs
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
Canada	35.467	1785387	9984670	0.913	America	NaN
France	63.951	2833687	640679	0.888	Europe	French
Germany	80.940	3874437	357114	0.916	Europe	German
Italy	60.665	2167744	301336	0.873	Europe	Italian
Japan	127.061	4602367	377930	0.891	Asia	NaN
United Kingdom	64.511	2950039	242495	0.907	Europe	NaN
United States	318.523	17348075	9525067	0.915	America	NaN

REPLACING VALUES PER COLUMN:

In [ ]:

```
df['Language']='English' #all will get affected
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
Canada	35.467	1785387	9984670	0.913	America	English
France	63.951	2833687	640679	0.888	Europe	English
Germany	80.940	3874437	357114	0.916	Europe	English
Italy	60.665	2167744	301336	0.873	Europe	English
Japan	127.061	4602367	377930	0.891	Asia	English
United Kingdom	64.511	2950039	242495	0.907	Europe	English
United States	318.523	17348075	9525067	0.915	America	English

RENAMING COLUMNS:

In [ ]:

```
#Again remember that a new dataframe is created and the original one is never chang
df.rename(
    columns={
        'HDI' : 'Human Development Index',
        'Anual Popcorn Consumption': 'APC'
    },
    index={
        'United States': 'USA',
        'United Kingdom': 'UK',
        'Argentina': 'AR'
    }
)
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
Canada	35.467	1785387	9984670	0.913	America	English
France	63.951	2833687	640679	0.888	Europe	English
Germany	80.940	3874437	357114	0.916	Europe	English
Italy	60.665	2167744	301336	0.873	Europe	English
Japan	127.061	4602367	377930	0.891	Asia	English
United Kingdom	64.511	2950039	242495	0.907	Europe	English
United States	318.523	17348075	9525067	0.915	America	English

In [ ]:

```
df.rename(index=str.upper)  #making everything into capitals
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
CANADA	35.467	1785387	9984670	0.913	America	English
FRANCE	63.951	2833687	640679	0.888	Europe	English
GERMANY	80.940	3874437	357114	0.916	Europe	English
ITALY	60.665	2167744	301336	0.873	Europe	English
JAPAN	127.061	4602367	377930	0.891	Asia	English
UNITED KINGDOM	64.511	2950039	242495	0.907	Europe	English
UNITED STATES	318.523	17348075	9525067	0.915	America	English

In [ ]:

```
df.rename(index=lambda x: x.lower())  #using lambda function to make everything sho
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
canada	35.467	1785387	9984670	0.913	America	English
france	63.951	2833687	640679	0.888	Europe	English
germany	80.940	3874437	357114	0.916	Europe	English
italy	60.665	2167744	301336	0.873	Europe	English
japan	127.061	4602367	377930	0.891	Asia	English
united kingdom	64.511	2950039	242495	0.907	Europe	English
united states	318.523	17348075	9525067	0.915	America	English

Adding Values:

In [ ]:

```
df.append(pd.Series({
    'Population': 3,
    'GDP': 5
}, name='China'))
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
Canada	35.467	1785387.0	9984670.0	0.913	America	English
France	63.951	2833687.0	640679.0	0.888	Europe	English
Germany	80.940	3874437.0	357114.0	0.916	Europe	English
Italy	60.665	2167744.0	301336.0	0.873	Europe	English
Japan	127.061	4602367.0	377930.0	0.891	Asia	English
United Kingdom	64.511	2950039.0	242495.0	0.907	Europe	English
United States	318.523	17348075.0	9525067.0	0.915	America	English
China	3.000	5.0	NaN	NaN	NaN	NaN

In [ ]:

```
df.loc['China']=pd.Series({
    'Population': 1400.00,
    'Continent': 'Asia',
    'GDP': 17000000,
    'HDI': 0.889,
    'Language':'Chinese',
    'Surface Area': 100000000
})
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language
Canada	35.467	1785387	9984670.0	0.913	America	English
France	63.951	2833687	640679.0	0.888	Europe	English
Germany	80.940	3874437	357114.0	0.916	Europe	English
Italy	60.665	2167744	301336.0	0.873	Europe	English
Japan	127.061	4602367	377930.0	0.891	Asia	English
United Kingdom	64.511	2950039	242495.0	0.907	Europe	English
United States	318.523	17348075	9525067.0	0.915	America	English
China	1400.000	17000000	100000000.0	0.889	Asia	Chinese

In [ ]:

```
df.reset_index()
```

Out[ ]:

	index	Population	GDP	Surface Area	HDI	Continent	Language
0	Canada	35.467	1785387	9984670.0	0.913	America	English
1	France	63.951	2833687	640679.0	0.888	Europe	English
2	Germany	80.940	3874437	357114.0	0.916	Europe	English
3	Italy	60.665	2167744	301336.0	0.873	Europe	English
4	Japan	127.061	4602367	377930.0	0.891	Asia	English
5	United Kingdom	64.511	2950039	242495.0	0.907	Europe	English
6	United States	318.523	17348075	9525067.0	0.915	America	English
7	China	1400.000	17000000	100000000.0	0.889	Asia	Chinese

In [ ]:

```
df['GDP per capita']=df['GDP']/df['Population']
df
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language	GDP per capita
Canada	35.467	1785387	9984670.0	0.913	America	English	50339.385908
France	63.951	2833687	640679.0	0.888	Europe	English	44310.284437
Germany	80.940	3874437	357114.0	0.916	Europe	English	47868.013343
Italy	60.665	2167744	301336.0	0.873	Europe	English	35733.025633
Japan	127.061	4602367	377930.0	0.891	Asia	English	36221.712406
United Kingdom	64.511	2950039	242495.0	0.907	Europe	English	45729.239975
United States	318.523	17348075	9525067.0	0.915	America	English	54464.120330
China	1400.000	17000000	100000000.0	0.889	Asia	Chinese	12142.857143

STATISTICAL INFO

In [ ]:

```
df.head()
```

Out[ ]:

	Population	GDP	Surface Area	HDI	Continent	Language	GDP per capita
Canada	35.467	1785387	9984670.0	0.913	America	English	50339.385908
France	63.951	2833687	640679.0	0.888	Europe	English	44310.284437
Germany	80.940	3874437	357114.0	0.916	Europe	English	47868.013343
Italy	60.665	2167744	301336.0	0.873	Europe	English	35733.025633
Japan	127.061	4602367	377930.0	0.891	Asia	English	36221.712406

In [ ]:

```
population=df['Population']
```

In [ ]:

```
population  #extracted a series
```

Out[ ]:

```
Canada              35.467
France              63.951
Germany             80.940
Italy               60.665
Japan              127.061
United Kingdom      64.511
United States      318.523
China             1400.000
Name: Population, dtype: float64
```

In [ ]:

```
population.min(), population.max() #and others like std, mean,etc can also be perf
```

Out[ ]:

```
(35.467, 1400.0)
```

In [ ]: