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

# PANDAS SERIES

In the course that I'm doing they're analysing *"the group of 7"*. Series is very similar to numpy arrays!

In [ ]:
```
#in millions
g7_pop=pd.Series([37.467, 63.951, 80.940, 60.665, 127.061, 64.511, 318.523])
g7_pop.name='G7 Population in millions' #naming the array
```

You can actually call/select objects like we can in a list, but the only difference is that we can change the indexing here! we could not have done that in lists. Basically we can name them.

In [ ]:
```
g7_pop
```

Out[ ]:
```
0     37.467
1     63.951
2     80.940
3     60.665
4    127.061
5     64.511
6    318.523
Name: G7 Population in millions, dtype: float64
```

In [ ]:
```
g7_pop.values #this is actually backed by numpy arrays
```

Out[ ]:
```
array([ 37.467,  63.951,  80.94 ,  60.665, 127.061,  64.511, 318.523])
```

In [ ]:
```
g7_pop.index #indexes to call (list like)
```

Out[ ]:
```
RangeIndex(start=0, stop=7, step=1)
```

In [ ]:
```
g7_pop.index=[
    'Canada',
    'France',
    'Germany',
    'Italy',
    'Japan',
    'United Kingdom',
    'United States'
]
g7_pop
```

Out[ ]:
```
Canada             37.467
France             63.951
Germany            80.940
Italy              60.665
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: G7 Population in millions, dtype: float64
```

We can also create the entire thing all at once by treating it as a "dictionary". Like:

In [ ]:
```
pd.Series({
```

```
    'Canada':37.467,
    'Frace': 63.951,
    'Germany': 80.940,
    'Italy':60.665,
    'Japan': 127.061,
    'United Kingdom':64.511,
    'United States':318.523
}, name='G7 Population in millions')
```

Out[ ]:
```
Canada             37.467
Frace              63.951
Germany            80.940
Italy              60.665
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: G7 Population in millions, dtype: float64
```

In [ ]:
```
g7_pop['Canada']+g7_pop['Japan']
```

Out[ ]:
```
164.52800000000002
```

In [ ]:
```

```

### iloc

attribute. Can be used in a similiar way to query for list positions. Gives us the *Value* of key:value pair. Like:

In [ ]:
```
g7_pop.iloc[-1]
```

Out[ ]:
```
318.523
```

## Slicing and selecting multiple elements is also doable:

using both names and iloc

In [ ]:
```
g7_pop[['Italy','Japan']]
```

Out[ ]:
```
Italy     60.665
Japan    127.061
Name: G7 Population in millions, dtype: float64
```

In [ ]:
```
g7_pop.iloc[[1,3]]
```

Out[ ]:
```
France    63.951
Italy     60.665
Name: G7 Population in millions, dtype: float64
```

Note that in python, the upper limit of a splice is not included, but in Pandas, it is included!

In [ ]:
```
g7_pop['Japan':'United States']  #Slicing
```

Out[ ]:
```
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: G7 Population in millions, dtype: float64
```

# Boolean series and Operations:

It is also possible to perform mathematical functions as well as boolean operations on Pandas series

In [ ]:
```
g7_pop
```

Out[ ]:
```
Canada             37.467
France             63.951
Germany            80.940
Italy              60.665
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: G7 Population in millions, dtype: float64
```

In [ ]:
```
g7_pop*1000000 #in numerical millions
```

Out[ ]:
```
Canada             37467000.0
France             63951000.0
Germany            80940000.0
Italy              60665000.0
Japan             127061000.0
United Kingdom     64511000.0
United States     318523000.0
Name: G7 Population in millions, dtype: float64
```

In [ ]:
```
#With some boolean stuff
g7_pop >100
```

Out[ ]:
```
Canada            False
France            False
Germany           False
Italy             False
Japan              True
United Kingdom    False
United States      True
Name: G7 Population in millions, dtype: bool
```

We can also ask Pandas to give us the data of the countries where population is more than 100 million, just by making the following query:

In [ ]:
```
g7_pop[g7_pop>100]
```

Out[ ]:
```
Japan            127.061
United States    318.523
Name: G7 Population in millions, dtype: float64
```

Some more random statistical tools we can use like: std, mean,etc.

In [ ]:
```
g7_pop.mean()
```

Out[ ]:
```
107.58828571428572
```

In [ ]:
```
g7_pop[g7_pop>g7_pop.std()]
```

Out[ ]:
```
Japan            127.061
United States    318.523
Name: G7 Population in millions, dtype: float64
```

In [ ]:
```
#we can also fo something like this:
g7_pop[(g7_pop>(g7_pop.mean() - g7_pop.std()/2)) | (g7_pop>(g7_pop.mean() + g7_pop
```

Out[ ]:
```
France              63.951
Germany             80.940
Italy               60.665
Japan              127.061
United Kingdom      64.511
United States      318.523
Name: G7 Population in millions, dtype: float64
```

We can also use traditional numpy functions here!

In [ ]:
```
np.log(g7_pop)
```

Out[ ]:
```
Canada             3.623461
France             4.158117
Germany            4.393708
Italy              4.105367
Japan              4.844667
United Kingdom     4.166836
United States      5.763695
Name: G7 Population in millions, dtype: float64
```

We can also modify the array

In [ ]:
```
g7_pop.iloc[-1]=500
g7_pop
```

Out[ ]:
```
Canada              37.467
France              63.951
Germany             80.940
Italy               60.665
Japan              127.061
United Kingdom      64.511
United States      500.000
Name: G7 Population in millions, dtype: float64
```