Pandas - Series

Hands on!

In [1]:
```
import pandas as pd
import numpy as np
```

Pandas Series

We'll start analyzing "The Group of Seven". Which is a political formed by Canada, France, Germany, Italy, Japan, the United Kingdom and the United States. We'll start by analyzing population, and for that, we'll use a `pandas.Series` object.

In [2]:
```
# In millions
g7_pop = pd.Series([35.467, 63.951, 80.940, 60.665, 127.061, 64.511, 318.523])
```

In [3]:
```
g7_pop
```

Out[3]:

	0
0	35.467
1	63.951
2	80.940
3	60.665
4	127.061
5	64.511
6	318.523

dtype: float64

Someone might not know we're representing population in millions of inhabitants. Series can have a `name`, to better document the purpose of the Series:

In [4]:
```
g7_pop.name = 'G7 Population in millions'
```

In [5]:
```
g7_pop
```

Out[5]:

	G7 Population in millions
0	35.467
1	63.951
2	80.940
3	60.665
4	127.061
5	64.511
6	318.523

dtype: float64

Series are pretty similar to numpy arrays:

In [6]:
```
g7_pop.dtype
```

Out[6]:
```
dtype('float64')
```

In [7]:
```
g7_pop.values
```

Out[7]:
```
array([ 35.467,  63.951,  80.94 ,  60.665, 127.061,  64.511, 318.523])
```

They're actually backed by numpy arrays:

In [8]:
```
type(g7_pop.values)
```

Out[8]:
```
numpy.ndarray
```

And they *look* like simple Python lists or Numpy Arrays. But they're actually more similar to Python `dict`s.

A Series has an `index`, that's similar to the automatic index assigned to Python's lists:

In [9]:
```
g7_pop
```

Out[9]:

	G7 Population in millions
0	35.467
1	63.951
2	80.940
3	60.665
4	127.061
5	64.511
6	318.523

dtype: float64

In [10]:
```
g7_pop[0]
```

Out[10]:
```
np.float64(35.467)
```

In [11]:
```
g7_pop[1]
```

Out[11]:
```
np.float64(63.951)
```

In [12]:
```
g7_pop.index
```

Out[12]:
```
RangeIndex(start=0, stop=7, step=1)
```

In [13]:
```
l = ['a', 'b', 'c']
```

But, in contrast to lists, we can explicitly define the index:

In [14]:
```
g7_pop.index = [
    'Canada',
    'France',
    'Germany',
    'Italy',
    'Japan',
    'United Kingdom',
    'United States',
]
```

In [15]:
```
g7_pop
```

Out[15]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

Compare it with the following table:

G7 Population	
(Expressed in millions)	
Canada	35.467
France	63.951
Germany	80.94
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

We can say that Series look like "ordered dictionaries". We can actually create Series out of dictionaries:

In [16]:

```
pd.Series({
    'Canada': 35.467,
    'France': 63.951,
    'Germany': 80.94,
    'Italy': 60.665,
    'Japan': 127.061,
    'United Kingdom': 64.511,
    'United States': 318.523
}, name='G7 Population in millions')
```

Out[16]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

In [17]:

```
pd.Series(
    [35.467, 63.951, 80.94, 60.665, 127.061, 64.511, 318.523],
    index=['Canada', 'France', 'Germany', 'Italy', 'Japan', 'United Kingdom',
       'United States'],
    name='G7 Population in millions')
```

Out[17]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

You can also create Series out of other series, specifying indexes:

In [18]:

```
pd.Series(g7_pop, index=['France', 'Germany', 'Italy', 'Spain'])
```

Out[18]:

	G7 Population in millions
France	63.951
Germany	80.940
Italy	60.665
Spain	NaN

dtype: float64

Indexing

Indexing works similarly to lists and dictionaries, you use the **index** of the element you're looking for:

In [19]:
```
g7_pop
```

Out[19]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

In [20]:
```
g7_pop['Canada']
```

Out[20]:
```
np.float64(35.467)
```

In [21]:
```
g7_pop['Japan']
```

Out[21]:
```
np.float64(127.061)
```

Numeric positions can also be used, with the `iloc` attribute:

In [22]:
```
g7_pop.iloc[0]
```

Out[22]: np.float64(35.467)

In [23]:
```
g7_pop.iloc[-1]
```

Out[23]: np.float64(318.523)

Selecting multiple elements at once:

In [24]:
```
g7_pop[['Italy', 'France']]
```

Out[24]:

	G7 Population in millions
Italy	60.665
France	63.951

dtype: float64

(The result is another Series)

In [25]:
```
g7_pop.iloc[[0, 1]]
```

Out[25]:

	G7 Population in millions
Canada	35.467
France	63.951

dtype: float64

Slicing also works, but **important**, in Pandas, the upper limit is also included:

In [26]:
```
g7_pop['Canada': 'Italy']
```

Out[26]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665

dtype: float64

Conditional selection (boolean arrays)

The same boolean array techniques we saw applied to numpy arrays can be used for Pandas `Series`:

In [27]:
```
g7_pop
```

Out[27]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

In [28]:
```
g7_pop > 70
```

Out[28]:

	G7 Population in millions
Canada	False
France	False
Germany	True
Italy	False
Japan	True
United Kingdom	False
United States	True

dtype: bool

In [29]:
```
g7_pop[g7_pop > 70]
```

Out[29]:

	G7 Population in millions
Germany	80.940
Japan	127.061
United States	318.523

dtype: float64

In [30]:
```
g7_pop.mean()
```

Out[30]:
```
np.float64(107.30257142857144)
```

In [31]:
```
g7_pop[g7_pop > g7_pop.mean()]
```

Out[31]:

	G7 Population in millions
Japan	127.061
United States	318.523

dtype: float64

In [32]:
```
g7_pop.std()
```

Out[32]:
```
97.24996987121581
```

In [33]:
```
# ~ not
# | or
# & and
```

In [34]:
```
g7_pop[(g7_pop < g7_pop.mean() - g7_pop.std()/2) | (g7_pop > g7_pop.mean() + g
```

Out[34]:

	G7 Population in millions
Canada	35.467
United States	318.523

dtype: float64

Operations and methods

Series also support vectorized operations and aggregation functions as Numpy:

In [35]:
```
g7_pop
```

Out[35]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

In [36]:
```
g7_pop * 1_000_000
```

Out[36]:

	G7 Population in millions
Canada	35467000.0
France	63951000.0
Germany	80940000.0
Italy	60665000.0
Japan	127061000.0
United Kingdom	64511000.0
United States	318523000.0

dtype: float64

In [37]:
```
g7_pop.mean()
```

Out[37]:
```
np.float64(107.30257142857144)
```

In [38]:
```
np.log(g7_pop)
```

Out[38]:

	G7 Population in millions
Canada	3.568603
France	4.158117
Germany	4.393708
Italy	4.105367
Japan	4.844667
United Kingdom	4.166836
United States	5.763695

dtype: float64

In [39]:

```
g7_pop['France': 'Italy'].mean()
```

Out[39]:

```
np.float64(68.51866666666666)
```

Boolean arrays

(Work in the same way as numpy)

In [40]:

```
g7_pop
```

Out[40]:

	G7 Population in millions
Canada	35.467
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

In [41]:

```
g7_pop > 80
```

Out[41]:

	G7 Population in millions
Canada	False
France	False
Germany	True
Italy	False
Japan	True
United Kingdom	False
United States	True

dtype: bool

In [42]:

```
g7_pop[g7_pop > 80]
```

Out[42]:

	G7 Population in millions
Germany	80.940
Japan	127.061
United States	318.523

dtype: float64

In [43]:

```
g7_pop[(g7_pop > 80) | (g7_pop < 40)]
```

Out[43]:

	G7 Population in millions
Canada	35.467
Germany	80.940
Japan	127.061
United States	318.523

dtype: float64

In [44]:

```
g7_pop[(g7_pop > 80) & (g7_pop < 200)]
```

Out[44]:

	G7 Population in millions
Germany	80.940
Japan	127.061

dtype: float64

Modifying series

In [45]:
```
g7_pop['Canada'] = 40.5
```

In [46]:
```
g7_pop
```

Out[46]:

	G7 Population in millions
Canada	40.500
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	318.523

dtype: float64

In [47]:
```
g7_pop.iloc[-1] = 500
```

In [48]:
```
g7_pop
```

Out[48]:

	G7 Population in millions
Canada	40.500
France	63.951
Germany	80.940
Italy	60.665
Japan	127.061
United Kingdom	64.511
United States	500.000

dtype: float64

In [49]:

```
g7_pop[g7_pop < 70]
```

Out[49]:

	G7 Population in millions
Canada	40.500
France	63.951
Italy	60.665
United Kingdom	64.511

dtype: float64

In [50]:

```
g7_pop[g7_pop < 70] = 99.99
```

In [51]:

```
g7_pop
```

Out[51]:

	G7 Population in millions
Canada	99.990
France	99.990
Germany	80.940
Italy	99.990
Japan	127.061
United Kingdom	99.990
United States	500.000

dtype: float64