Data Analysis with Python

Intro to Pandas

Aim: learn why **Pandas** is the most important library for Data Processing in Python, and how its main data structure and Data Frame compares to other tools like spreadsheets or DFs used for Big Data

Pandas

- Will analyze “The Group of Seven”. This is a political formed by Canada, France, Germany, Italy, Spain, UK, US
- 1. Will analyze the population; will use pandas.Series object

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

import pandas as pd
import numpy as np

```
In [2]: # Analyzing the population.
        # In millions.

        g7_pop = pd.Series([35.467, 63.951, 80.940, 60.665, 127.061, 64.511, 318.523])
```

g7_pop = pd.Series([35.467, 63.951, 80.940, 60.665, 127.061, 64.511, 318.523])

```
In [3]: g7_pop

0     35.467
1     63.951
2     80.940
3     60.665
4    127.061
5     64.511
6    318.523
dtype: float64
```

g7_pop

```
In [ ]: # Others may not know that the population is represented in millions of inhabitants.
        # Will name the series to avoid confusion.

        g7_pop.name = 'G7 Populaion in millions'
```

g7_pop.name = ‘G7 Population in millions’

```
In [7]: g7_pop

0     35.467
1     63.951
2     80.940
3     60.665
4    127.061
5     64.511
6    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop

- Notice: Name: *67 Population in millions* is now present within the output
- Series are similar to Numpy arrays

```
In [8]: g7_pop.dtype

dtype('float64')
```

g7_pop.dtype

```
In [9]: g7_pop.values

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

g7_pop.values

- Series are actually backed by Numpy arrays

```
In [10]: type(g7_pop)

pandas.core.series.Series

In [11]: type(g7_pop.values)

numpy.ndarray
```

type(g7_pop)
type(g7_pop.values)

- They look like simple Python lists or Numpy arrays
- But they're more similar to Python dictionaries
- A Series has an index. This index is similar to the automatic index assigned to Python lists

```
In [13]: g7_pop[0]

35.467

In [14]: g7_pop[1]

63.951
```

g7_pop[0]
g7_pop[1]

```
In [15]: g7_pop.index

RangeIndex(start=0, stop=7, step=1)
```

g7_pop.index

- *start*: index you start with
- *stop*: index you stop at; excluding that index (i.e. should always be one less)
- *step*: 1 (difference between each index)

```
In [16]: l = ['a', 'b', 'c']
```

l = [‘a’, ‘b’, ‘c’]

- In contrast to lists, for Series, you can explicitly **define the index**

```
In [17]: g7_pop.index = [
             'Canada',
             'France',
             'Germany',
             'Italy',
             'Japan',
             'United Kingdom',
             'United States'
         ]
```

g7_pop.index = [
‘Canada’,
‘France’,
‘Germany’,
‘Italy’,
‘Japan’,
‘UK’,
‘US’
]

```
In [18]: g7_pop

Canada             35.467
France             63.951
Germany            80.940
Italy              60.665
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop

- Compare this 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

- Series look like “ordered dictionaries”
    - Can create Series out of dictionaries

```
In [19]: pd.Series({
             'Canada': 35.467,
             'France': 63.951,
             'Germany': 80.94,
             'Italy': 60.665,
             'Japan': 127.061,
             'UK': 64.511,
             'US': 318.523
         }, name = 'G7 Population in millions')

Canada      35.467
France      63.951
Germany     80.940
Italy       60.665
Japan      127.061
UK          64.511
US         318.523
Name: G7 Population in millions, dtype: float64
```

pd.Series({
‘Canada’: 35.467,
‘France’: 63.951,
‘Germany’: 80.94,
‘Italy’: 60.665,
‘Japan’: 127.061,
‘UK’: 64.511,
‘US’: 318.523
}, name = ‘G7 Population in millions’)

```
In [20]: pd.Series(
             [35.467, 63.951, 80.94, 60.665, 127.061, 64.511, 318.523],
             index=['Canada', 'France', 'Germany', 'Italy', 'Japan', 'UK', 'US'],
             name='G7 in millions')

Canada      35.467
France      63.951
Germany     80.940
Italy       60.665
Japan      127.061
UK          64.511
US         318.523
Name: G7 in millions, dtype: float64
```

pd.Series(
[35.467, 63.951, 80.94, 60.665, 127.061, 64.511, 318.523],

index=[‘Canada’, ‘France’, ‘Germany’, ‘Italy’, ‘Japan’, ‘UK’, ‘US’]
name=’G7 in millions’)

- This shows that you can create Series out of other series, specifying indexes

```
In [21]: pd.Series(g7_pop, index=['France', 'Germany', 'Italy', 'Spain'])

France     63.951
Germany    80.940
Italy      60.665
Spain         NaN
Name: G7 Populaion in millions, dtype: float64
```

pd.Series(g7_pop, index=[‘France’, ‘Germany’, ‘Italy’, ‘Spain’])

```
In [22]: # Indexing
         g7_pop

Canada             35.467
France             63.951
Germany            80.940
Italy              60.665
Japan             127.061
United Kingdom     64.511
United States     318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop

- Indexing works similarly to lists and dictionaries, you use the **index** of the element you are looking for

```
In [23]: g7_pop['Canada']

35.467

In [24]: g7_pop['Japan']

127.061
```

g7_pop[‘Canada’]
g7_pop[‘Japan’]

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

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

35.467

In [26]: g7_pop.iloc[-1]

318.523
```

g7_pop.iloc[0]
g7_pop.iloc[-1]

Can also select multiple elements at once:

```
In [27]: g7_pop[['Italy', 'France']]

Italy     60.665
France    63.951
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[[‘Italy’, ‘France’]]

Slicing also works:

```
In [28]: g7_pop['Canada': 'Italy']

Canada     35.467
France     63.951
Germany    80.940
Italy      60.665
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[‘Canada’: ‘Italy’]

- **Note:** in Pandas, the **upper limit** is also **INCLUDED**

Conditional Selection (boolean arrays)

Some boolean array techniques we saw applied to numpy arrays can be used for Pandas Series

```
In [30]: # Conditional Selection (boolean arrays)
         g7_pop

Canada            35.467
France            63.951
Germany           80.940
Italy             60.665
Japan            127.061
United Kingdom    64.511
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop

```
In [31]: g7_pop > 70

Canada           False
France           False
Germany           True
Italy            False
Japan             True
United Kingdom   False
United States     True
Name: G7 Populaion in millions, dtype: bool
```

g7_pop > 70

```
In [32]: g7_pop[g7_pop > 70]

Germany           80.940
Japan            127.061
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[g7 pop > 70]

```
In [33]: g7_pop.mean()

107.30257142857144
```

g7_pop.mean()

```
In [34]: g7_pop[g7_pop > g7_pop.mean()]

Japan            127.061
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[g7_pop > g7_pop.mean()]

```
In [36]: g7_pop.std()

97.24996987121581
```

g7_pop.std()

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

In [37]: g7_pop[(g7_pop > g7_pop.mean() - g7_pop.std() / 2) | (g7_pop > g7_pop.mean() + g7_pop.std() / 2)]

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

g7_pop[(g7_pop > g7_pop.mean() - g7_pop.std() /2) | (g7_pop > g7_pop.mean() + g7_pop.std()/2]

Operations and Methods

Series also support vectorized operations and aggregation functions as Numpy

```
In [39]: # Operations and Methods
         g7_pop

Canada           35.467
France           63.951
Germany          80.940
Italy            60.665
Japan           127.061
United Kingdom   64.511
United States   318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop

```
g7_pop * 1_000_000

Canada             35467000.0
France             63951000.0
Germany            80940000.0
Italy              60665000.0
Japan             127061000.0
United Kingdom     64511000.0
United States     318523000.0
Name: G7 Populaion in millions, dtype: float64
```

g7_pop * 1_000_000

```
In [41]: g7_pop.mean()

107.30257142857144
```

g7_pop.mean()

```
In [42]: np.log(g7_pop)

Canada           3.568603
France           4.158117
Germany          4.393708
Italy            4.105367
Japan            4.844667
United Kingdom   4.166836
United States    5.763695
Name: G7 Populaion in millions, dtype: float64
```

np.log(g7_pop)

```
In [43]: g7_pop['France': 'Italy'].mean()

68.51866666666666
```

g7_pop[‘France’: ‘Italy’].mean()

Boolean Arrays

Work the same way as Numpy

```
In [44]: # Boolean Arrays
         g7_pop

Canada            35.467
France            63.951
Germany           80.940
Italy             60.665
Japan            127.061
United Kingdom    64.511
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop

```
In [45]: g7_pop > 80

Canada            False
France            False
Germany            True
Italy             False
Japan              True
United Kingdom    False
United States      True
Name: G7 Populaion in millions, dtype: bool
```

g7_pop > 80

```
In [46]: g7_pop[g7_pop > 80]

Germany           80.940
Japan            127.061
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[g7_pop > 80]

```
In [47]: g7_pop[(g7_pop > 80) | (g7_pop < 40)]

Canada            35.467
Germany           80.940
Japan            127.061
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

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

```
In [48]: g7_pop[(g7_pop > 80) & (g7_pop < 200)]

Germany     80.940
Japan      127.061
Name: G7 Populaion in millions, dtype: float64
```

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

Modifying Series

```
In [49]: # Modufying Series

         g7_pop['Canada'] = 40.5

In [50]: g7_pop

Canada            40.500
France            63.951
Germany           80.940
Italy             60.665
Japan            127.061
United Kingdom    64.511
United States    318.523
Name: G7 Populaion in millions, dtype: float64
```

g7_pop['Canada'] = 40.5

- Notice: the Series has been modified, specifically the entry associated with 'Canada' as an index

```
In [51]: g7_pop.iloc[-1] = 500

In [52]: g7_pop

Canada            40.500
France            63.951
Germany           80.940
Italy             60.665
Japan            127.061
United Kingdom    64.511
United States    500.000
Name: G7 Populaion in millions, dtype: float64
```

g7_pop.iloc[-1] = 500

- Notice: modification in last index of the Series

```
In [53]: g7_pop[g7_pop < 70]

Canada           40.500
France           63.951
Italy            60.665
United Kingdom   64.511
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[g7_pop < 70]

```
In [54]: g7_pop[g7_pop < 70] = 99.99

In [55]: g7_pop

Canada           99.990
France           99.990
Germany          80.940
Italy            99.990
Japan           127.061
United Kingdom   99.990
United States   500.000
Name: G7 Populaion in millions, dtype: float64
```

g7_pop[g7_pop < 70] = 99.99

References

https://www.youtube.com/watch?v=r-uOLxNrNk8

https://github.com/ine-rmotr-curriculum/freecodecamp-intro-to-pandas/blob/master/1%20-%20Pandas%20-%20Series.ipynb

https://en.wikipedia.org/wiki/Group_of_Seven

https://docs.google.com/spreadsheets/d/1IlorV2-Oh9Da1JAZ7weVw86PQrQydSMp-ydVMH135il/edit#gid=0