(http://www.pieriandata.com)

Series

The first main data type we will learn about for pandas is the Series data type. Let's import Pandas and explore the Series object.

A Series is very similar to a NumPy array (in fact it is built on top of the NumPy array object). What differentiates the NumPy array from a Series, is that a Series can have axis labels, meaning it can be indexed by a label, instead of just a number location. It also doesn't need to hold numeric data, it can hold any arbitrary Python Object.

Let's explore this concept through some examples:

Imports

In [12]:
```
import numpy as np
import pandas as pd
```

Creating a Series from Python Objects

In [13]:
```
help(pd.Series)
```

```
Help on class Series in module pandas.core.series:

class Series(pandas.core.base.IndexOpsMixin, pandas.core.generic.NDFram
e)
 |  One-dimensional ndarray with axis labels (including time series).
 |
 |  Labels need not be unique but must be a hashable type. The object
 |  supports both integer- and label-based indexing and provides a host
of
 |  methods for performing operations involving the index. Statistical
 |  methods from ndarray have been overridden to automatically exclude
 |  missing data (currently represented as NaN).
 |
 |  Operations between Series (+, -, /, *, **) align values based on th
eir
 |  associated index values-- they need not be the same length. The res
ult
 |  index will be the sorted union of the two indexes.
 |
```

Index and Data Lists

We can create a Series from Python lists (also from NumPy arrays)

In [14]:
```
myindex = ['USA','Canada','Mexico']
```

In [15]:
```
mydata = [1776,1867,1821]
```

In [16]:
```
myser = pd.Series(data=mydata)
```

In [17]:
```
myser
```

Out[17]:
```
0    1776
1    1867
2    1821
dtype: int64
```

In [18]:
```
pd.Series(data=mydata,index=myindex)
```

Out[18]:
```
USA       1776
Canada    1867
Mexico    1821
dtype: int64
```

In [23]:
```
ran_data = np.random.randint(0,100,4)
```

In [24]:
```
ran_data
```

Out[24]:
```
array([39, 35, 37, 23])
```

In [26]:
```
names = ['Andrew','Bobo','Claire','David']
```

In [27]:
```
ages = pd.Series(ran_data,names)
```

In [28]:
```
ages
```

Out[28]:
```
Andrew    39
Bobo      35
Claire    37
David     23
dtype: int32
```

From a Dictionary

In [29]:
```
ages = {'Sammy':5,'Frank':10,'Spike':7}
```

In [30]:
```
ages
```

Out[30]:
```
{'Frank': 10, 'Sammy': 5, 'Spike': 7}
```

In [31]:
```
pd.Series(ages)
```

Out[31]:
```
Sammy     5
Frank    10
Spike     7
dtype: int64
```

Key Ideas of a Series

Named Index

In [32]:
```
# Imaginary Sales Data for 1st and 2nd Quarters for Global Company
q1 = {'Japan': 80, 'China': 450, 'India': 200, 'USA': 250}
q2 = {'Brazil': 100,'China': 500, 'India': 210,'USA': 260}
```

In [33]:
```
# Convert into Pandas Series
sales_Q1 = pd.Series(q1)
sales_Q2 = pd.Series(q2)
```

In [34]:
```
sales_Q1
```

Out[34]:
```
Japan     80
China    450
India    200
USA      250
dtype: int64
```

In [35]:
```
# Call values based on Named Index
sales_Q1['Japan']
```

Out[35]:
```
80
```

In [36]:
```
# Integer Based Location information also retained!
sales_Q1[0]
```

Out[36]:
```
80
```

Be careful with potential errors!

In [37]:
```
# Wrong Name
# sales_Q1['France']
```

In [38]:
```
# Accidental Extra Space
# sales_Q1['USA ']
```

In [39]:
```
# Capitalization Mistake
# sales_Q1['usa']
```

Operations

In [40]:
```
# Grab just the index keys
sales_Q1.keys()
```

Out[40]:
```
Index(['Japan', 'China', 'India', 'USA'], dtype='object')
```

In [41]:
```
# Can Perform Operations Broadcasted across entire Series
sales_Q1 * 2
```

Out[41]:
```
Japan    160
China    900
India    400
USA      500
dtype: int64
```

In [42]:
```
sales_Q2 / 100
```

Out[42]:
```
Brazil    1.0
China     5.0
India     2.1
USA       2.6
dtype: float64
```

Between Series

In [43]:
```
# Notice how Pandas informs you of mismatch with NaN
sales_Q1 + sales_Q2
```

Out[43]:
```
Brazil      NaN
China     950.0
India     410.0
Japan       NaN
USA       510.0
dtype: float64
```

In [44]:
```
# You can fill these with any value you want
sales_Q1.add(sales_Q2,fill_value=0)
```

Out[44]:
```
Brazil    100.0
China     950.0
India     410.0
Japan      80.0
USA       510.0
dtype: float64
```

That is all we need to know about Series, up next, DataFrames!