Introduction to Pandas

In this section of the course we will learn how to use pandas for data analysis. You can think of pandas as an extremely powerful version of Excel, with a lot more features. In this section of the course, you should go through the notebooks in this order:

- Introduction to Pandas
- Series
- DataFrames
- Missing Data
- GroupBy
- Merging,Joining,and Concatenating
- Operations
- Data Input and Output

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:

In [2]:
```
import numpy as np
import pandas as pd
```

Creating a Series

You can convert a list,numpy array, or dictionary to a Series:

In [3]:
```
labels = ['a','b','c']
my_list = [10,20,30]
arr = np.array([10,20,30])
d = {'a':10,'b':20,'c':30}
```

Using Lists

In [4]:
```
1 pd.Series(data=my_list)
```

Out[4]:
```
0    10
1    20
2    30
dtype: int64
```

In [5]:
```
1 pd.Series(data=my_list,index=labels)
```

Out[5]:
```
a    10
b    20
c    30
dtype: int64
```

In [6]:
```
1 pd.Series(my_list,labels)
```

Out[6]:
```
a    10
b    20
c    30
dtype: int64
```

NumPy Arrays

In [7]:
```
1 pd.Series(arr)
```

Out[7]:
```
0    10
1    20
2    30
dtype: int64
```

In [8]:
```
1 pd.Series(arr,labels)
```

Out[8]:
```
a    10
b    20
c    30
dtype: int64
```

Dictionary

In [9]:
```
1 pd.Series(d)
```

Out[9]:
```
a    10
b    20
c    30
dtype: int64
```

Data in a Series

A pandas Series can hold a variety of object types:

In [10]:
```
pd.Series(data=labels)
```

Out[10]:
```
0    a
1    b
2    c
dtype: object
```

In [11]:
```
# Even functions (although unlikely that you will use this)
pd.Series([sum,print,len])
```

Out[11]:
```
0      <built-in function sum>
1    <built-in function print>
2      <built-in function len>
dtype: object
```

Using an Index

The key to using a Series is understanding its index. Pandas makes use of these index names or numbers by allowing for fast look ups of information (works like a hash table or dictionary).

Let's see some examples of how to grab information from a Series. Let us create two sereis, ser1 and ser2:

In [12]:
```
ser1 = pd.Series([1,2,3,4],index = ['USA', 'Germany','USSR', 'Japan']
```

In [13]:
```
ser1
```

Out[13]:
```
USA        1
Germany    2
USSR       3
Japan      4
dtype: int64
```

In [14]:
```
ser2 = pd.Series([1,2,5,4],index = ['USA', 'Germany','Italy', 'Japan'
```

In [15]:
```
ser2
```

Out[15]:
```
USA        1
Germany    2
Italy      5
Japan      4
dtype: int64
```

In [16]:
```
1 ser1['USA']
```

Out[16]:
```
1
```

Operations are then also done based off of index:

In [17]:
```
1 ser1 + ser2
```

Out[17]:
```
Germany    4.0
Italy      NaN
Japan      8.0
USA        2.0
USSR       NaN
dtype: float64
```

Let's stop here for now and move on to DataFrames, which will expand on the concept of Series!

Great Job!