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 [1]:

```
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}
```

In [5]:

```
arr
```

Out[5]:

```
array([10, 20, 30])
```

In [7]:

```
pd.Series(arr)
```

Out[7]:

```
0    10
1    20
2    30
dtype: int64
```

In [8]:

```
pd.Series(data=my_list,index=labels)
```

Out[8]:

```
a    10
b    20
c    30
dtype: int64
```

** Using Lists**

In [4]:

```
pd.Series(data=my_list)
```

Out[4]:

```
0    10
1    20
2    30
dtype: int64
```

In [5]:

```
pd.Series(data=my_list,index=labels)
```

Out[5]:

```
a    10
b    20
c    30
dtype: int64
```

In [10]:

```
pd.Series(my_list,labels)
```

Out[10]:

```
a    10
b    20
c    30
dtype: int64
```

** NumPy Arrays **

In [9]:

```
pd.Series(arr)
```

Out[9]:

```
0    10
1    20
2    30
dtype: int64
```

In [ ]:

```
pd.Series(arr,labels)
```

In [11]:

```
pd.Series(arr,labels)
```

Out[11]:

```
a    10
b    20
c    30
dtype: int64
```

In [12]:

```
d = {'a':10,'b':20,'c':30}
```

In [13]:

```
pd.Series(d)
```

Out[13]:

```
a    10
b    20
c    30
dtype: int64
```

** Dictionary**

In [9]:

```
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 [14]:

```
pd.Series(data=labels)
```

Out[14]:

```
0    a
1    b
2    c
dtype: object
```

In [16]:

```
pd.Series(data=labels)
```

Out[16]:

```
0    a
1    b
2    c
dtype: object
```

In [17]:

```
pd.Series([sum,len,str,int])
```

Out[17]:

```
0    <built-in function sum>
1    <built-in function len>
2              <class 'str'>
3              <class 'int'>
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 [35]:

```
ser1 = pd.Series([1,2,3,4],index = ['USA', 'India','USSR', 'Japan'])
```

In [36]:

```
ser1
```

Out[36]:

```
USA      1
India    2
USSR     3
Japan    4
dtype: int64
```

In [37]:

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

In [38]:

```
ser2
```

Out[38]:

```
USA        1
Germany    2
Italy      5
Japan      4
dtype: int64
```

In [41]:

```
ser1['India']
```

Out[41]:

```
2
```

In [42]:

```
ser2['Italy']
```

Out[42]:

```
5
```

In [43]:

```
ser1 + ser2
```

Out[43]:

```
Germany    NaN
India      NaN
Italy      NaN
Japan      8.0
USA        2.0
USSR       NaN
dtype: float64
```

Operations are then also done based off of index:

In [44]:

```
ser1 + ser2
```

Out[44]:

```
Germany    NaN
India      NaN
Italy      NaN
Japan      8.0
USA        2.0
USSR       NaN
dtype: float64
```

In [51]:

```
ser1
```

Out[51]:

```
USA      1
India    2
USSR     3
Japan    4
dtype: int64
```

In [52]:

```
ser1.rename({'USSR':'Russia'})
```

Out[52]:

```
USA       1
India     2
Russia    3
Japan     4
dtype: int64
```