In [2]:
```
# Now we'll learn about concatenating along an axis
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
```

In [3]:
```
# First in just Numpy
```

In [4]:
```
# Create a matrix
arr1 = np.arange(9).reshape((3,3))
```

In [5]:
```
# Show
arr1
```

Out[5]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [6]:
```
# Concatenate along axis 1
np.concatenate([arr1,arr1],axis=1)
```

Out[6]:
```
array([[0, 1, 2, 0, 1, 2],
       [3, 4, 5, 3, 4, 5],
       [6, 7, 8, 6, 7, 8]])
```

In [7]:
```
# Let's see other axis options
np.concatenate([arr1,arr1],axis=0)
```

Out[7]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8],
       [0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [8]:
```
# Now let's see how this works in pandas
```

In [9]:
```
# Lets create two Series with no overlap
ser1 =  Series([0,1,2],index=['T','U','V'])

ser2 = Series([3,4],index=['X','Y'])

#Now let use concat (default is axis=0)
pd.concat([ser1,ser2])
```

Out[9]:
```
T    0
U    1
V    2
X    3
Y    4
dtype: int64
```

In [10]:
```
# Now passing along another axis will produce a DataFrame
pd.concat([ser1,ser2],axis=1)
```

Out[10]:

| | 0 | 1 |
|---|---|---|
| **T** | 0 | NaN |
| **U** | 1 | NaN |
| **V** | 2 | NaN |
| **X** | NaN | 3 |
| **Y** | NaN | 4 |

In [17]:
```
# We can specify which specific axes to be used
pd.concat([ser1,ser2],axis=1,join_axes=[['U','V','Y']])
```

Out[17]:

| | 0 | 1 |
|---|---|---|
| **U** | 1 | NaN |
| **V** | 2 | NaN |
| **Y** | NaN | 4 |

In [11]:
```
# Lets say we wanted to add markers.keys to the concatenation result

# WE can do this with a hierarchical index
pd.concat([ser1,ser2],keys=['cat1','cat2'])
```

Out[11]:
```
cat1  T    0
      U    1
      V    2
cat2  X    3
      Y    4
dtype: int64
```

In [12]:
```
# Along the axis=1 then these Keys become column headers
pd.concat([ser1,ser2],axis=1,keys=['cat1','cat2'])
```

Out[12]:

| | cat1 | cat2 |
|---|---|---|
| **T** | 0 | NaN |
| **U** | 1 | NaN |
| **V** | 2 | NaN |
| **X** | NaN | 3 |
| **Y** | NaN | 4 |

In [14]:
```
#Lastly, everything works similarly in DataFrames

dframe1 = DataFrame(np.random.randn(4,3), columns=['X', 'Y', 'Z'])
dframe2 = DataFrame(np.random.randn(3, 3), columns=['Y', 'Q', 'X'])
```

In [16]:
```
#Concat on DataFrame
pd.concat([dframe1,dframe2])
```

Out[16]:

| | Q | X | Y | Z |
|---|---|---|---|---|
| **0** | NaN | 1.096040 | -1.366980 | 0.546707 |
| **1** | NaN | -1.406425 | 0.484748 | -1.156143 |
| **2** | NaN | 1.155464 | 1.166407 | -0.245477 |
| **3** | NaN | -0.153330 | 2.185743 | 0.307704 |
| **0** | 0.789881 | 1.616933 | -0.961830 | NaN |
| **1** | 0.201265 | 0.293210 | -0.277847 | NaN |
| **2** | -0.121395 | 0.959849 | -1.360611 | NaN |

In [17]:
```
#If we dont care about the index info and just awnt to make a complete DataFrame,
pd.concat([dframe1,dframe2],ignore_index=True)
```

Out[17]:

| | Q | X | Y | Z |
|---|---|---|---|---|
| **0** | NaN | 1.096040 | -1.366980 | 0.546707 |
| **1** | NaN | -1.406425 | 0.484748 | -1.156143 |
| **2** | NaN | 1.155464 | 1.166407 | -0.245477 |
| **3** | NaN | -0.153330 | 2.185743 | 0.307704 |
| **4** | 0.789881 | 1.616933 | -0.961830 | NaN |
| **5** | 0.201265 | 0.293210 | -0.277847 | NaN |
| **6** | -0.121395 | 0.959849 | -1.360611 | NaN |

In [18]:
```
#For more info in documentation:
url='http://pandas.pydata.org/pandas-docs/stable/generated/pandas.concat.html'
```

In [ ]:
```
#Next up: More on Combining DataFrames with Overlapping Indexes!
```