In [1]:
```
# Now we'll learn how to merge data sets by linking rows by keys.

import numpy as np
import pandas as pd
from pandas import Series, DataFrame
```

In [6]:
```
# Let's make a dframe

dframe1 = DataFrame({'key':['X','Z','Y','Z','X','X'],'data_set_1': np.arange(6)})

#Show
dframe1
```

Out[6]:

	data_set_1	key
0	0	X
1	1	Z
2	2	Y
3	3	Z
4	4	X
5	5	X

In [13]:
```
#Now lets make another dframe

dframe2 = DataFrame({'key':['Q','Y','Z'],'data_set_2':[1,2,3]})

#Show
dframe2
```

Out[13]:

	data_set_2	key
0	1	Q
1	2	Y
2	3	Z

In [14]:
```
# Now we can use merge the dataframes, this is a "many-to-one" situation

# Merge will automatically choose overlapping columns to merge on
pd.merge(dframe1,dframe2)

#Note no overlapping 'X's
```

Out[14]:

	data_set_1	key	data_set_2
0	1	Z	3
1	3	Z	3
2	2	Y	2

In [16]:
```
# We could have also specified which column to merge on
pd.merge(dframe1,dframe2,on='key')
```

Out[16]:

	data_set_1	key	data_set_2
0	1	Z	3
1	3	Z	3
2	2	Y	2

In [17]:
```
# We can choose which DataFrame's keys to use, this will choose left (dframe1)
pd.merge(dframe1,dframe2,on='key',how='left')
```

Out[17]:

	data_set_1	key	data_set_2
0	0	X	NaN
1	1	Z	3
2	2	Y	2
3	3	Z	3
4	4	X	NaN
5	5	X	NaN

In [18]:
```
# Choosing the one on the right (dframe2)
pd.merge(dframe1,dframe2,on='key',how='right')
```

Out[18]:

	data_set_1	key	data_set_2
0	1	Z	3
1	3	Z	3
2	2	Y	2
3	NaN	Q	1

In [19]:
```
#Choosing the "outer" method selects the union of both keys
pd.merge(dframe1,dframe2,on='key',how='outer')
```

Out[19]:

	data_set_1	key	data_set_2
0	0	X	NaN
1	4	X	NaN
2	5	X	NaN
3	1	Z	3
4	3	Z	3
5	2	Y	2
6	NaN	Q	1

In [30]:
```
#Now we'll learn about a many to many merge

# Nnote that these DataFrames contain more than one instance of the key in BOTH d

dframe3 = DataFrame({'key': ['X', 'X', 'X', 'Y', 'Z', 'Z'],
                     'data_set_3': range(6)})
dframe4 = DataFrame({'key': ['Y', 'Y', 'X', 'X', 'Z'],
                     'data_set_4': range(5)})

#Show the merge
pd.merge(dframe3, dframe4)
```

Out[30]:

	data_set_3	key	data_set_4
0	0	X	2
1	0	X	3
2	1	X	2
3	1	X	3
4	2	X	2
5	2	X	3
6	3	Y	0
7	3	Y	1
8	4	Z	4
9	5	Z	4

So what happened? A many to many merge results in the product of the rows. Because there were 3 'X's in dframe3 and 2 'X's in dframe4 there ended up being a total of 6 'X' rows in the result (2*3=6)! Note how dframe3 repeats its 0,1,2 values for 'X' and dframe4 repeats its '2,3' pairs throughout the key set.

In [33]:
```
# We can also merge with multiple keys!

# Dframe on left
df_left = DataFrame({'key1': ['SF', 'SF', 'LA'],
                     'key2': ['one', 'two', 'one'],
                     'left_data': [10,20,30]})

#Dframe on right
df_right = DataFrame({'key1': ['SF', 'SF', 'LA', 'LA'],
                      'key2': ['one', 'one', 'one', 'two'],
                      'right_data': [40,50,60,70]})

#Merge
pd.merge(df_left, df_right, on=['key1', 'key2'], how='outer')
```

Out[33]:

	key1	key2	left_data	right_data
0	SF	one	10	40
1	SF	one	10	50
2	SF	two	20	NaN
3	LA	one	30	60
4	LA	two	NaN	70

In [32]:
```
# Now using the above you can check mulitple data sets for multiple key combos, f
# Answer =  60
```

In [35]:
```
#Note that the left and right DataFrames have overlapping key names (key1 and key
# pandas automatically adds suffixes to them

pd.merge(df_left,df_right,on='key1')
```

Out[35]:

	key1	key2_x	left_data	key2_y	right_data
0	SF	one	10	one	40
1	SF	one	10	one	50
2	SF	two	20	one	40
3	SF	two	20	one	50
4	LA	one	30	one	60
5	LA	one	30	two	70

In [36]:
```
# We can also specify what the suffix becomes
pd.merge(df_left,df_right, on='key1',suffixes=('_lefty','_righty'))
```

Out[36]:

	key1	key2_lefty	left_data	key2_righty	right_data
0	SF	one	10	one	40
1	SF	one	10	one	50
2	SF	two	20	one	40
3	SF	two	20	one	50
4	LA	one	30	one	60
5	LA	one	30	two	70

In [37]:
```
# For more info on merge parameters check out:
url = 'http://pandas.pydata.org/pandas-docs/dev/generated/pandas.DataFrame.merge.

# Next we'll learn how to merge on Index!
```

In [ ]:
```

```