In [1]:
```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame

#Let's learn how to use dict or series with groupby
```

In [19]:
```
# Let's make a Dframe

animals = DataFrame(np.arange(16).reshape(4, 4),
                    columns=['W', 'X', 'Y', 'Z'],
                    index=['Dog', 'Cat', 'Bird', 'Mouse'])

#Now lets add some NAN values
animals.ix[1:2, ['W', 'Y']] = np.nan

#Show
animals
```

Out[19]:

	W	X	Y	Z
Dog	0	1	2	3
Cat	NaN	5	NaN	7
Bird	8	9	10	11
Mouse	12	13	14	15

In [20]:
```
# Now let's say I had a dictionary with ebhavior values in it
behavior_map = {'W': 'good', 'X': 'bad', 'Y': 'good','Z': 'bad'}
```

In [21]:
```
# Now we can groupby using that mapping
animal_col = animals.groupby(behavior_map, axis=1)

# Show the sum accroding to the groupby with the mapping
animal_col.sum()

# For example [dog][good] = [dog][Y]+[dog][W]
```

Out[21]:

	bad	good
Dog	4	2
Cat	12	NaN
Bird	20	18
Mouse	28	26

In [22]:
```
# Now let's try it with a Series
behav_series = Series(behavior_map)

#Show
behav_series
```

Out[22]:
```
W    good
X     bad
Y    good
Z     bad
dtype: object
```

In [23]:
```
# Now let's groupby the Series

animals.groupby(behav_series, axis=1).count()
```

Out[23]:

	bad	good
Dog	2	2
Cat	2	0
Bird	2	2
Mouse	2	2

In [26]:
```
# We can also groupby with functions!

#Show our dframe again
animals
```

Out[26]:

	W	X	Y	Z
Dog	0	1	2	3
Cat	NaN	5	NaN	7
Bird	8	9	10	11
Mouse	12	13	14	15

In [25]:
```
# Lets assume we wanted to group by the length of the animal names, we can pass t

# Show
animals.groupby(len).sum()

#Note the index is now number of letters in the animal name
```

Out[25]:

	W	X	Y	Z
3	0	6	2	10
4	8	9	10	11
5	12	13	14	15

In [29]:
```
# We can also mix functions with arrays,dicts, and Series for groupby methods

# Set a list for keys
keys = ['A', 'B', 'A', 'B']

# Now groupby length of name and the keys to show max values
animals.groupby([len, keys]).max()
```

Out[29]:

		W	X	Y	Z
3	**A**	0	1	2	3
	B	NaN	5	NaN	7
4	**A**	8	9	10	11
5	**B**	12	13	14	15

In [36]:
```
# We can also use groupby with hierarchaly index levels

#Create a hierarchal column index
hier_col = pd.MultiIndex.from_arrays([['NY','NY','NY','SF','SF'],[1,2,3,1,2]],nam

# Create a dframe with hierarchal index
dframe_hr = DataFrame(np.arange(25).reshape(5,5),columns=hier_col)

#Multiply values by 100 for clarity
dframe_hr = dframe_hr*100

#Show
dframe_hr
```

Out[36]:

City	NY			SF	
sub_value	**1**	**2**	**3**	**1**	**2**
0	0	100	200	300	400
1	500	600	700	800	900
2	1000	1100	1200	1300	1400
3	1500	1600	1700	1800	1900
4	2000	2100	2200	2300	2400

In [ ]:
```
#Up next: Data Aggregation!!
```