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

In [2]:
```
#Now we'll learn about binning
```

In [3]:
```
years = [1990,1991,1992,2008,2012,2015,1987,1969,2013,2008,1999]
```

In [4]:
```
# We can seperate these years by decade
decade_bins = [1960,1970,1980,1990,2000,2010,2020]
```

In [7]:
```
#Now we'll use cut to get somethign called a Category object
decade_cat = pd.cut(years,decade_bins)
```

In [8]:
```
#Show
decade_cat
```

Out[8]:
```
[(1980, 1990], (1990, 2000], (1990, 2000], (2000, 2010], (2010, 2020], ..., (19
80, 1990], (1960, 1970], (2010, 2020], (2000, 2010], (1990, 2000]]
Length: 11
Categories (6, object): [(1960, 1970] < (1970, 1980] < (1980, 1990] < (1990, 20
00] < (2000, 2010] < (2010, 2020]]
```

In [13]:
```
# We can check the categories using .categories
decade_cat.categories
```

Out[13]:
```
Index([u'(1960, 1970]', u'(1970, 1980]', u'(1980, 1990]', u'(1990, 2000]', u'(2
000, 2010]', u'(2010, 2020]'], dtype='object')
```

In [16]:
```
# Then we can check the value counts in each category
pd.value_counts(decade_cat)
```

Out[16]:
```
(2010, 2020]    3
(1990, 2000]    3
(2000, 2010]    2
(1980, 1990]    2
(1960, 1970]    1
(1970, 1980]    0
dtype: int64
```

In [30]:
```
# We can also pass data values to the cut.

#For instance, if we just wanted to make two bins, evenly spaced based on max and
pd.cut(years,2,precision=1)
```

Out[30]:
```
[(1969, 1992], (1969, 1992], (1969, 1992], (1992, 2015], (1992, 2015], ..., (19
69, 1992], (1969, 1992], (1992, 2015], (1992, 2015], (1992, 2015]]
Length: 11
Categories (2, object): [(1969, 1992] < (1992, 2015]]
```

In [1]:
```
# Thats about it for binning basics
# One last thing to note, jus tlike in standard math notation, when setting up bi
# () means open, while [] means closed/inclusive
```

In [ ]:
```
# Next up: Finding Outliers and Describing Data!
```