In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
# A Series is similar to a list, or column in a table.
# Be default each item will be zero-indexed, just like a list
series = pd.Series([12, 'Data Science', 3.78, -142, 'Happy Holidays!'])
series
```

Out[2]:
```
0                 12
1       Data Science
2               3.78
3               -142
4    Happy Holidays!
dtype: object
```

In [3]:
```
# Unlike lists, you can specify the index for a Series
series = pd.Series([12, 'Data Science', 3.78, -142, 'Happy Holidays!'],
                   index=['A', 'B', 'C', 'D', 'E'])
series
```

Out[3]:
```
A                 12
B       Data Science
C               3.78
D               -142
E    Happy Holidays!
dtype: object
```

In [4]:
```
# We can also construct a Series from a dictionary, since it's just a key-va
d = {'Chicago': 1000, 'New York': 1300, 'Portland': 900, 'San Francisco': 11
     'Austin': 450, 'Boston': None}
cities = pd.Series(d)
cities
```

Out[4]:
```
Austin            450
Boston            NaN
Chicago          1000
New York         1300
Portland          900
San Francisco    1100
dtype: float64
```

In [5]:
```
# Just like Python data structures, we can reference items by their index/ke
cities['New York']
```

Out[5]:
```
1300.0
```

In [6]:
```
# Even better, we can do boolean indexing
cities[cities < 1000]
```

Out[6]:
```
Austin      450
Portland    900
dtype: float64
```

In [7]:
```
# What's happening is that cities < 1000 returns a Series of boolean values,
cities < 1000
```

Out[7]:
```
Austin            True
Boston           False
Chicago          False
New York         False
Portland          True
San Francisco    False
dtype: bool
```

In [8]:
```
# Series are mutable, meaning we can change values on the fly
print 'Old value:', cities['New York']
cities['New York'] = 1400
print 'New value:', cities['New York']
```

```
Old value: 1300.0
New value: 1400.0
```

In [9]:
```
# To check membership, it's like normal Python
print 'Stockholm' in cities
print 'New York' in cities
```

```
False
True
```

In [10]:
```
# We can apply functions and scalar arithmetic on the entire Series
print np.square(cities / 2)
print '\n'
print cities.map(lambda x: x * 2)
print '\n'
print cities.index.map(len)
```

```
Austin            50625
Boston              NaN
Chicago          250000
New York         490000
Portland         202500
San Francisco    302500
dtype: float64


Austin            900
Boston            NaN
Chicago          2000
New York         2800
Portland         1800
San Francisco    2200
dtype: float64


[ 6  6  7  8  8 13]
```

In [11]:
```
# Adding Series
print cities[['Chicago', 'New York', 'Portland']]
print'\n'
print cities[['Austin', 'New York']]
print'\n'
print cities[['Chicago', 'New York', 'Portland']] + cities[['Austin', 'New Yo
```

```
Chicago     1000
New York    1400
Portland     900
dtype: float64


Austin       450
New York    1400
dtype: float64


Austin       NaN
Chicago      NaN
New York    2800
Portland     NaN
dtype: float64
```

In [12]:
```
# We have convenience methods for selecting null and non-null values
print cities.notnull()
print '\n'
print cities.isnull()
```

```
Austin            True
Boston           False
Chicago           True
New York          True
Portland          True
San Francisco     True
dtype: bool


Austin           False
Boston            True
Chicago          False
New York         False
Portland         False
San Francisco    False
dtype: bool
```

In [13]:
```
# Dataframes consist of several series. If dataframes are tables, then serie
# Dataframes are the most common data structure in Pandas
data = {'year': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],
        'team': ['Bears', 'Bears', 'Bears', 'Packers', 'Packers', 'Lions', '
        'wins': [11, 8, 10, 15, 11, 6, 10, 4],
        'losses': [5, 8, 6, 1, 5, 10, 6, 12]}
football = pd.DataFrame(data, columns=['year', 'team', 'wins', 'losses'])
football
```

Out[13]:

	year	team	wins	losses
0	2010	Bears	11	5
1	2011	Bears	8	8
2	2012	Bears	10	6
3	2011	Packers	15	1
4	2012	Packers	11	5
5	2010	Lions	6	10
6	2011	Lions	10	6
7	2012	Lions	4	12

In [14]:
```
football.T
```

Out[14]:

	0	1	2	3	4	5	6	7
year	2010	2011	2012	2011	2012	2010	2011	2012
team	Bears	Bears	Bears	Packers	Packers	Lions	Lions	Lions
wins	11	8	10	15	11	6	10	4
losses	5	8	6	1	5	10	6	12

In [ ]: