Chapter 2 - Data Preparation Basics

Segment 1 - Filtering and selecting data

```
import numpy as np
import pandas as pd

from pandas import Series, DataFrame
```

Selecting and retrieving data

You can write an index value in two forms.

- Label index or
- Integer index

```
series_obj = Series(np.arange(8), index=['row 1', 'row 2', 'row 3', 'row 4', 'row 5', 'row 6','row 7', 'row 8'])
series_obj
```

```
row 1    0
row 2    1
row 3    2
row 4    3
row 5    4
row 6    5
row 7    6
row 8    7
dtype: int64
```

```
series_obj['row 7']
```

```
6
```

```
series_obj[[0, 7]]
```

```
row 1    0
row 8    7
dtype: int32
```

```
np.random.seed(25)
DF_obj = DataFrame(np.random.rand(36).reshape((6,6)),
                   index=['row 1', 'row 2', 'row 3', 'row 4','row 5','row 6'],
                   columns=['column 1','column 2','column 3','column 4','column 5','column 6'])
DF_obj
```

	column 1	column 2	column 3	column 4	column 5	column 6
row 1	0.870124	0.582277	0.278839	0.185911	0.411100	0.117376
row 2	0.684969	0.437611	0.556229	0.367080	0.402366	0.113041
row 3	0.447031	0.585445	0.161985	0.520719	0.326051	0.699186
row 4	0.366395	0.836375	0.481343	0.516502	0.383048	0.997541
row 5	0.514244	0.559053	0.034450	0.719930	0.421004	0.436935
row 6	0.281701	0.900274	0.669612	0.456069	0.289804	0.525819

```
DF_obj.loc[['row 2', 'row 5'], ['column 5', 'column 2']]
```

	column 5	column 2
row 2	0.402366	0.437611
row 5	0.421004	0.559053

Data slicing

You can use slicing to select and return a slice of several values from a data set. Slicing uses index values so you can use the same square brackets when doing data slicing.

How slicing differs, however, is that with slicing you pass in two index values that are separated by a colon. The index value on the left side of the colon should be the first value you want to select. On the right side of the colon, you write the index value for the last value you want to retrieve. When you execute the code, the indexer then simply finds the first record and the last record and returns every record in between them.

```
series_obj['row 1':'row 5']
```

```
row 1    0
row 2    1
row 3    2
row 4    3
row 5    4
dtype: int64
```

Comparing with scalars

Now we're going to talk about comparison operators and scalar values. Just in case you don't know that a scalar value is, it's basically just a single numerical value. You can use comparison operators like greater than or less than to return true/false values for all records to indicate how each element compares to a scalar value.

```
DF_obj < .2
```

	column 1	column 2	column 3	column 4	column 5	column 6
row 1	False	False	False	True	False	True
row 2	False	False	False	False	False	True
row 3	False	False	True	False	False	False
row 4	False	False	False	False	False	False
row 5	False	False	True	False	False	False
row 6	False	False	False	False	False	False

Filtering with scalars

```
series_obj[series_obj > 6]
```

```
row 8    7
dtype: int32
```

Setting values with scalars

```
series_obj['row 1', 'row 5', 'row 8'] = 8
series_obj
```

```
row 1    8
row 2    1
row 3    2
row 4    3
row 5    8
row 6    5
row 7    6
row 8    8
dtype: int32
```

Filtering and selecting using Pandas is one of the most fundamental things you'll do in data analysis. Make sure you know how to use indexing to select and retrieve records.