DataFrame Manipulation

Let's go through some common tasks for manipulating a DataFrame.

Removing a Row/Column

In many cases during the data cleaning process, there may be a need to drop unwanted rows or data variables (i.e., columns). We typically do this using the **drop** function. The **drop** function has a parameter **axis** whose default is 0. If **axis** is set to 1, it drops columns in a dataset, but if left at the default, rows are dropped from the dataset.

Note that when a column or row is dropped, a new **DataFrame** or **Series** is returned without altering the original data structure. However, when the attribute **inplace** is set to **True**, the original DataFrame or Series is modified. Let's see some examples.

```
# the data frame
my_DF = pd.DataFrame({'age': [15,17,21,29,25], \
            'state_of_origin':['Lagos', 'Cross River', 'Kano', 'Abia',
            'Benue']})
my_DF
'Output':
   age state_of_origin
0   15           Lagos
1   17     Cross River
2   21            Kano
3   29            Abia
4   25           Benue
# drop the 3rd and 4th column
my_DF.drop([2,4])
'Output':
   age state_of_origin
0   15           Lagos
1   17     Cross River
3   29            Abia
# drop the `age` column
my_DF.drop('age', axis=1)
```

```
'Output':
  state_of_origin
0           Lagos
1     Cross River
2            Kano
3            Abia
4           Benue
# original DataFrame is unchanged
my_DF
'Output':
   age state_of_origin
0   15           Lagos
1   17     Cross River
2   21            Kano
3   29            Abia
4   25           Benue
# drop using 'inplace' - to modify the original DataFrame
my_DF.drop('age', axis=1, inplace=True)
# original DataFrame altered
my_DF
'Output':
  state_of_origin
0           Lagos
1     Cross River
2            Kano
3            Abia
4           Benue
```

Let's see examples of removing a row given a condition.

```
my_DF = pd.DataFrame({'age': [15,17,21,29,25], \
            'state_of_origin':['Lagos', 'Cross River', 'Kano', 'Abia',
            'Benue']})
my_DF
```

```
'Output':
   age state_of_origin
0   15           Lagos
1   17     Cross River
2   21            Kano
3   29            Abia
4   25           Benue

# drop all rows less than 20
my_DF.drop(my_DF[my_DF['age'] < 20].index, inplace=True)
my_DF

'Output':
   age state_of_origin
2   21            Kano
3   29            Abia
4   25           Benue
```

Adding a Row/Column

We can add a new column to a Pandas DataFrame by using the **assign** method.

```
# show dataframe
my_DF = pd.DataFrame({'age': [15,17,21,29,25], \
            'state_of_origin':['Lagos', 'Cross River', 'Kano', 'Abia',
            'Benue']})
my_DF
'Output':
   age state_of_origin
0   15           Lagos
1   17     Cross River
2   21            Kano
3   29            Abia
4   25           Benue
# add column to data frame
```