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

In [8]:
```
#creating a dataframe using pandas DataFrame function

df= pd.DataFrame(np.arange(5,11).reshape(2,3), index= ['datainstance_row1', 'datainstanc
e_row2'], \
                  columns=['feature1', 'feature2', 'label'])
```

In [9]:
```
df
```

Out[9]:

	feature1	feature2	label
datainstance_row1	5	6	7
datainstance_row2	8	9	10

In [38]:
```
#Accesing the data- the most important part

df['feature1']
```

Out[38]:
```
datainstance_row1    5
datainstance_row2    8
Name: feature1, dtype: int32
```

In [42]:
```
df.label
```

Out[42]:
```
datainstance_row1     7
datainstance_row2    10
Name: label, dtype: int32
```

In [45]:
```
df[['feature1','label']]
```

Out[45]:

	feature1	label
datainstance_row1	5	7
datainstance_row2	8	10

In [43]:
```
df.feature1.datainstance_row1
```

Out[43]:
```
5
```

In [17]:
```
df.loc['datainstance_row1' ]
```

Out[17]:
```
feature1    5
feature2    6
label       7
Name: datainstance_row1, dtype: int32
```

In [15]:
```
df.iloc[:, :]
```

Out[15]:

	feature1	feature2	label
datainstance_row1	5	6	7
datainstance_row2	8	9	10

In [18]:
```
df.iloc[1:, 2:]
```

Out[18]:

	label
datainstance_row2	10

Understanding the difference between Series and dataframes:

A Pandas Series is one dimensioned whereas a DataFrame is two dimensioned. Therefore, a single column DataFrame can have a name for its single column but a Series cannot have a column name. In fact, each column of a DataFrame can be converted to a series.

In [19]:
```
type(df.iloc[1:, 2:])
```

Out[19]:
```
pandas.core.frame.DataFrame
```

In [20]:
```
type(df.loc['datainstance_row1' ])
```

Out[20]:
```
pandas.core.series.Series
```

In [21]:
```
type(df.iloc[:, 0])
```

Out[21]:
```
pandas.core.series.Series
```

In [22]:
```
df.iloc[:, 0]
```

Out[22]:
```
datainstance_row1    5
datainstance_row2    8
Name: feature1, dtype: int32
```

Thus, inshort a dataframe can be cut off as series if:

1) a single row is extracted from the df

2) a column is extracted without its column-name

DF to Array Conversion

In [25]:
```
array1=df.iloc[:,:]
array1
```

Out[25]:

	feature1	feature2	label
datainstance_row1	5	6	7
datainstance_row2	8	9	10

In [26]:
```
type(array1)
```

Out[26]:
```
pandas.core.frame.DataFrame
```

In [27]:
```
array1=df.iloc[:,:].values  #.values converts our df data into an array.
array1
```

Out[27]:
```
array([[ 5,  6,  7],
       [ 8,  9, 10]])
```

In [28]:
```
type(array1)
```

Out[28]:
```
numpy.ndarray
```

Pandas also provides us ways to identify and count null values in our dataset as shown below

In [32]:
```
df= pd.DataFrame(np.arange(5,11).reshape(2,3), index= ['datainstance_row1', 'datainstanc
e_row2'], \
                  columns=['feature1', 'feature2', 'label'])
df.isnull()
```

Out[32]:

	feature1	feature2	label
datainstance_row1	False	False	False
datainstance_row2	False	False	False

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
feature1    0
feature2    0
label       0
dtype: int64
```

In [34]:
```
# to identify the count of unique presence of the values present in a specific column
df['label'].value_counts()
```

Out[34]:
```
7     1
10    1
Name: label, dtype: int64
```

In [36]:
```
# to identify the values which are uniquely present i.e. without repitition in a specifi
c column

df.feature1.unique()
```

Out[36]:
```
array([5, 8], dtype=int64)
```

/