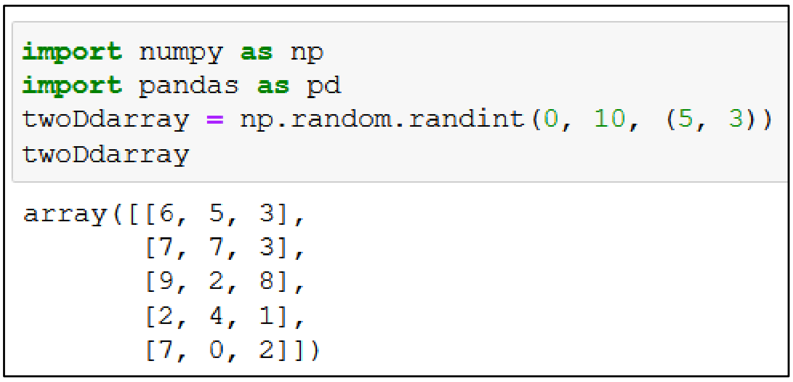
Objective: This worksheet introduces ***DataFrame*** object of Pandas. ***Series*** object reviewed in the previous worksheet is analogous to one-dimensional array and is helpful to create an object for time series type of data. ***DataFrame*** is analogous to two-dimensional array and it is helpful to store relational data.

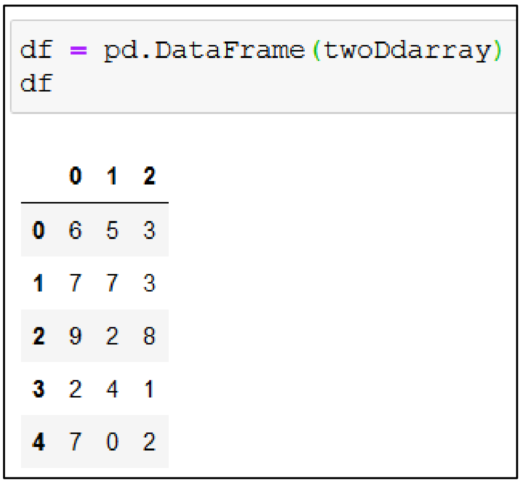
- To elaborate the concept, let us create a data frame using a two dimension NumPy array and ***DataFrame()*** function of Pandas package.
- First create a two dimensional NumPy array of size 5x3 of random numbers from 0 to 10.



```
import numpy as np
import pandas as pd
twoDdarray = np.random.randint(0, 10, (5, 3))
twoDdarray
```

```
array([[6, 5, 3],
       [7, 7, 3],
       [9, 2, 8],
       [2, 4, 1],
       [7, 0, 2]])
```

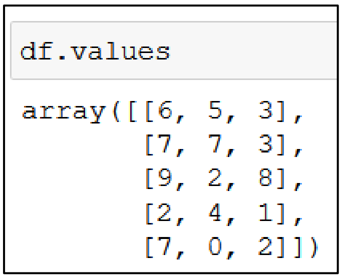
- Let us now create a Pandas data frame from this two dimensional array using ***DataFrame()*** function.



```
df = pd.DataFrame(twoDdarray)
df
```

	0	**1**	**2**
0	6	5	3
1	7	7	3
2	9	2	8
3	2	4	1
4	7	0	2

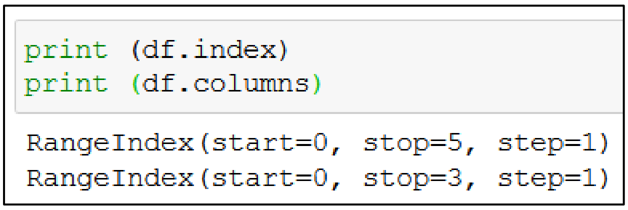
- Notice from the above output there is a name for each row (0 to 4) and a name (0 to 2) for each column.
- Data frames values can be accessed by the ***values*** property of the object. The returned values is a NumPy array.



```
df.values
```

```
array([[6, 5, 3],
       [7, 7, 3],
       [9, 2, 8],
       [2, 4, 1],
       [7, 0, 2]])
```

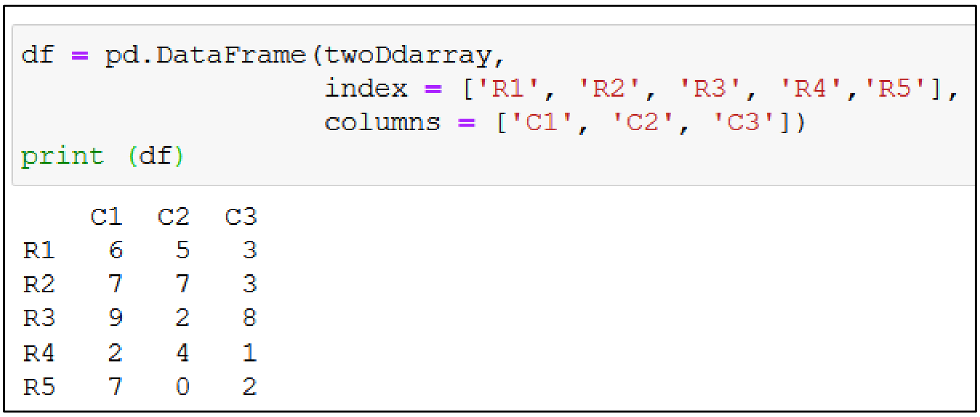
- In the data frames, row names are called index and column names are called columns that can also be accessed by ***index*** and ***columns*** properties respectively.



```
print (df.index)
print (df.columns)
```

```
RangeIndex(start=0, stop=5, step=1)
RangeIndex(start=0, stop=3, step=1)
```

- Notice from the above output that row is ranging from 0 to 5 with a step of 1, while columns are ranging from 0 to 3 with a step of 1. End-values 5 and 3 are exclusive.
- Programmer can also provide row and column names explicitly. In the example below, rows are named from R1 to R5 using and columns are named from C1 to C3.



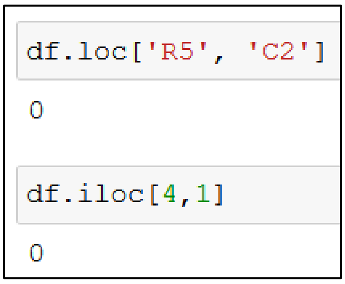
```
df = pd.DataFrame(twoDdarray,
                  index = ['R1', 'R2', 'R3', 'R4','R5'],
                  columns = ['C1', 'C2', 'C3'])
print (df)
```

```
    C1  C2  C3
R1   6   5   3
R2   7   7   3
R3   9   2   8
R4   2   4   1
R5   7   0   2
```

- If a data frame is already created (let us say in a variable x) with default index and column names, it can be changed later with explicit programmer provide names as shown below:

```
x.index = [‘Row-1’, ‘Row-2’, ‘Row-3’]
x.columns = [‘Col-1’, Col-2’, ‘Col-3’]
```

- Individual data element can be accessed using ***loc*** and ***iloc***. For example the data element (value 0) which is present in the 5th row and 2nd column can be accessed as follows:



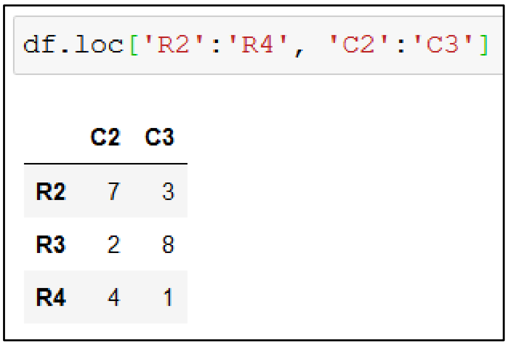
```
df.loc['R5', 'C2']
```

```
0
```

```
df.iloc[4,1]
```

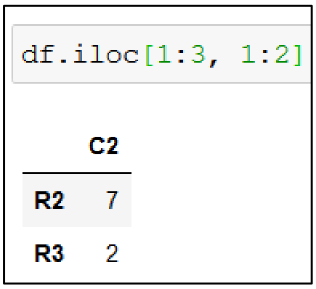
```
0
```

- Data elements can also be accessed using slicing. In case of ***loc*** the end value is inclusive but in case of ***iloc*** it is exclusive.



```
df.loc['R2':'R4', 'C2':'C3']
```

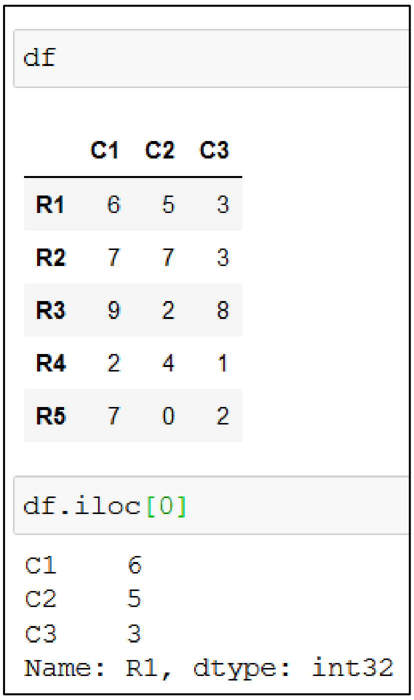
	C2	C3
R2	7	3
R3	2	8
R4	4	1



```
df.iloc[1:3, 1:2]
```

	C2
R2	7
R3	2

- Data frames can be considered and viewed as the collection of series. A data frame would have m series where m is the count of rows in the data frames.
- In the example below what is R1 alone? It is a pandas Series object where indices are C1, C2 and C3.



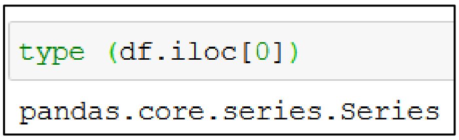
```
df
```

	C1	C2	C3
R1	6	5	3
R2	7	7	3
R3	9	2	8
R4	2	4	1
R5	7	0	2

```
df.iloc[0]
```

```
C1    6
C2    5
C3    3
Name: R1, dtype: int32
```

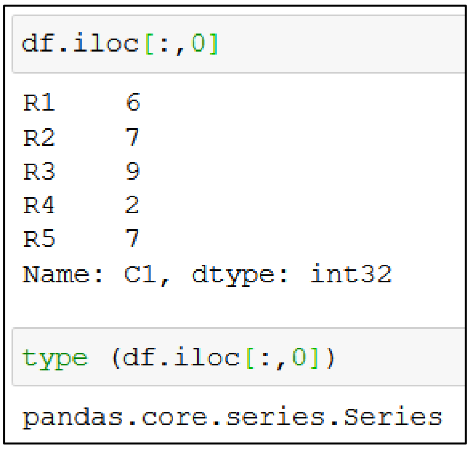
- Similarly, the elements of other rows (R2 to R5) are also Series objects where the element indices are same C1 to C3. This can also be verified checking the data type.



```
type (df.iloc[0])
```

```
pandas.core.series.Series
```

- With the same logic each a data frame would have n series where n is the count of columns in the data frames.
- In the example below what is C1 alone? It is pandas Series object where indices are R1 to R5. Notice the subtle difference while slicing the series from a column:



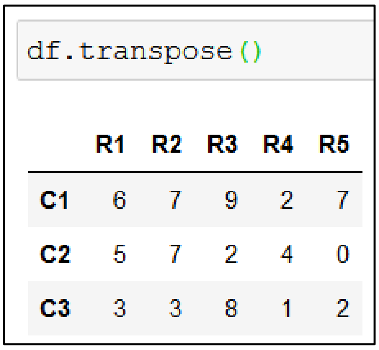
```
df.iloc[:,0]
```

```
R1    6
R2    7
R3    9
R4    2
R5    7
Name: C1, dtype: int32
```

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

```
pandas.core.series.Series
```

- A dataframe can be transposed also using the ***transpose()*** function.



```
df.transpose()
```

	R1	R2	R3	R4	R5
C1	6	7	9	2	7
C2	5	7	2	4	0
C3	3	3	8	1	2