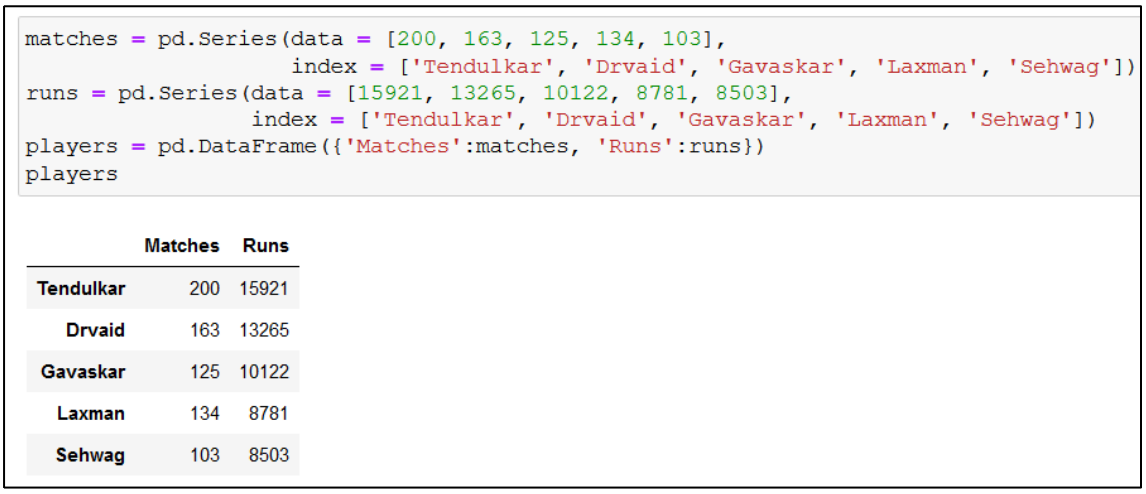
Objective: This worksheet demonstrates few additional ways to perform operations on Pandas Data Frames.

- To work on some data, let us first create a data frame from the following table:

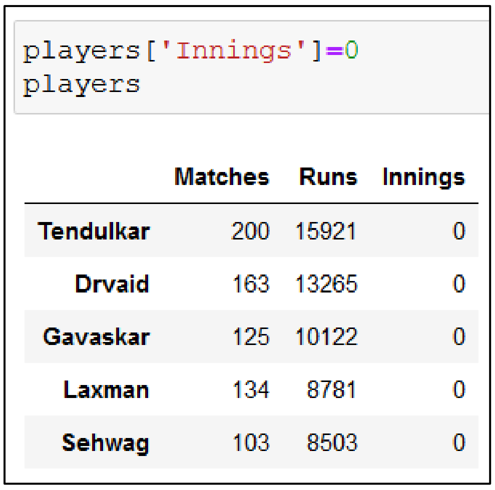
	Matches	Runs
Tendulkar	200	15921
Dravid	163	13265
Gavaskar	125	10122
Laxman	134	8781
Sehwag	103	8503



```
matches = pd.Series(data = [200, 163, 125, 134, 103],
                    index = ['Tendulkar', 'Drvaid', 'Gavaskar', 'Laxman', 'Sehwag'])
runs = pd.Series(data = [15921, 13265, 10122, 8781, 8503],
                 index = ['Tendulkar', 'Drvaid', 'Gavaskar', 'Laxman', 'Sehwag'])
players = pd.DataFrame({'Matches':matches, 'Runs':runs})
players
```

```
          Matches   Runs
Tendulkar     200  15921
   Drvaid     163  13265
 Gavaskar     125  10122
   Laxman     134   8781
   Sehwag     103   8503
```

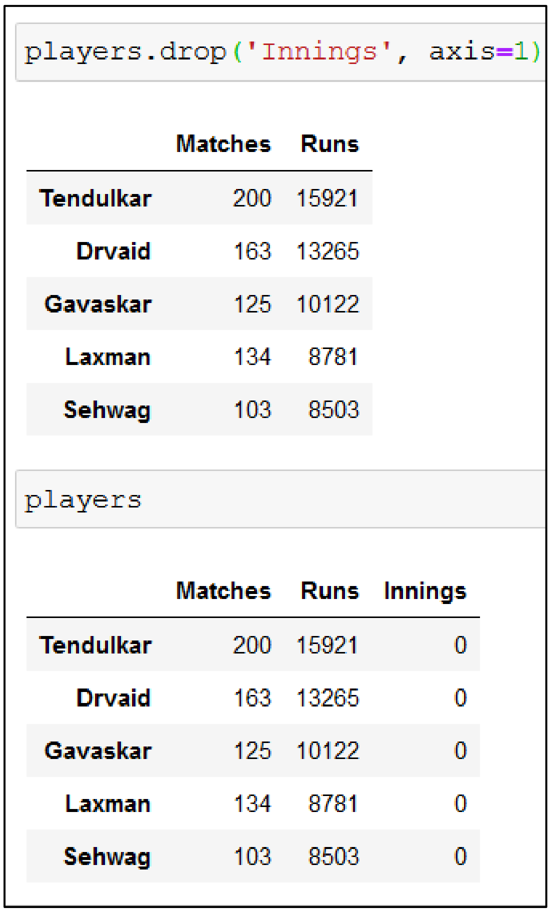
- Let us create a column for 'Innings' and assign all values to 0.



```
players['Innings']=0
players
```

```
          Matches   Runs  Innings
Tendulkar     200  15921        0
   Drvaid     163  13265        0
 Gavaskar     125  10122        0
   Laxman     134   8781        0
   Sehwag     103   8503        0
```

- Let us drop it the column for 'Innings' using the ***drop ()*** function and ***axis*** argument. Note that now when the value of players is printed, it still shows the 'Innings' column.



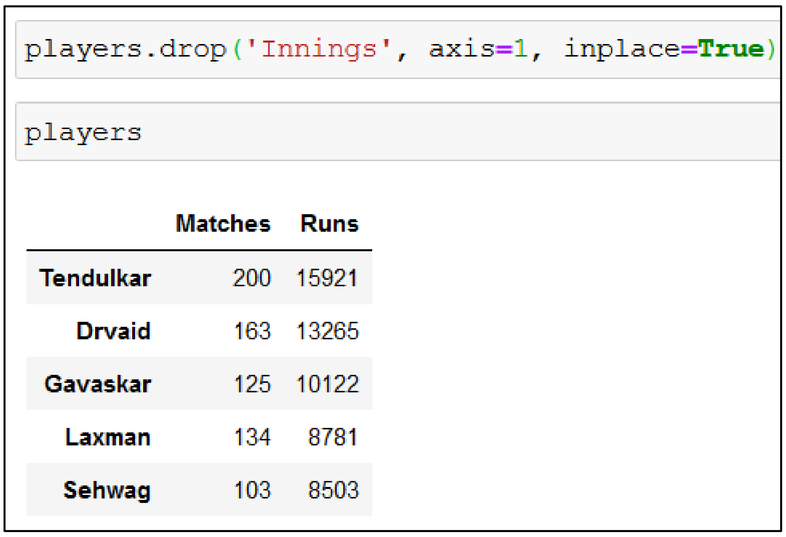
```python
players.drop('Innings', axis=1)
```

	Matches	Runs
Tendulkar	200	15921
Drvaid	163	13265
Gavaskar	125	10122
Laxman	134	8781
Sehwag	103	8503

```python
players
```

	Matches	Runs	Innings
Tendulkar	200	15921	0
Drvaid	163	13265	0
Gavaskar	125	10122	0
Laxman	134	8781	0
Sehwag	103	8503	0

- To actually remove the "Innings' column from the data frame, ***inplace*** argument with value ***True*** is used with the ***drop ()*** function.

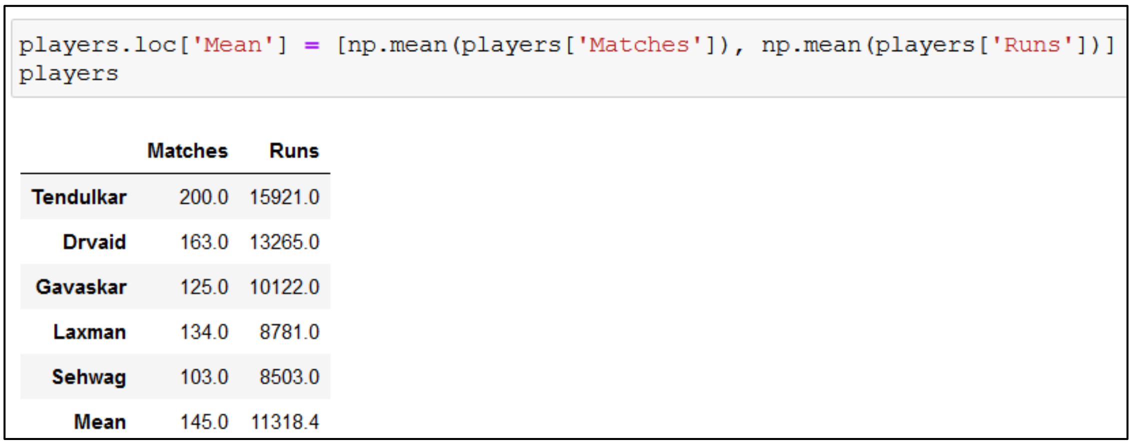


```python
players.drop('Innings', axis=1, inplace=True)
```

```python
players
```

	Matches	Runs
Tendulkar	200	15921
Drvaid	163	13265
Gavaskar	125	10122
Laxman	134	8781
Sehwag	103	8503

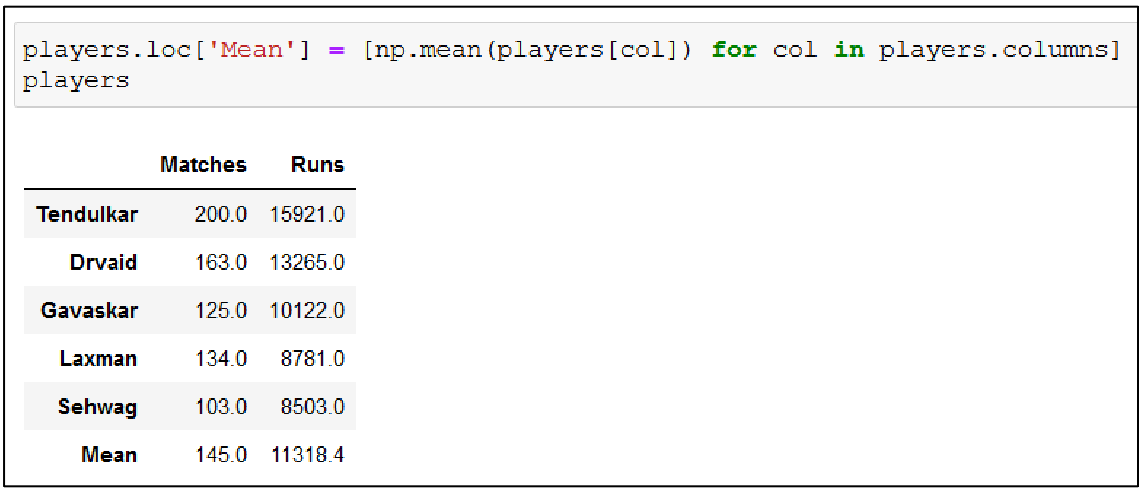
- Let us say we want to create a new row that contains the mean values of matches and runs for all players together (does not make any real sense though!). How this can be achieved?
- We can insert a new row for mean using ***append ()*** function as discussed in the previous worksheet. The point to be noted here is that each column ('Matches' and 'Runs') is a NumPy array and NumPy functions like mean can be applied on them individually. In this worksheet, a new row will be inserted using ***loc***. Note the usage of NumPy mean function.



```
players.loc['Mean'] = [np.mean(players['Matches']), np.mean(players['Runs'])]
players
```

	Matches	Runs
Tendulkar	200.0	15921.0
Drvaid	163.0	13265.0
Gavaskar	125.0	10122.0
Laxman	134.0	8781.0
Sehwag	103.0	8503.0
Mean	145.0	11318.4

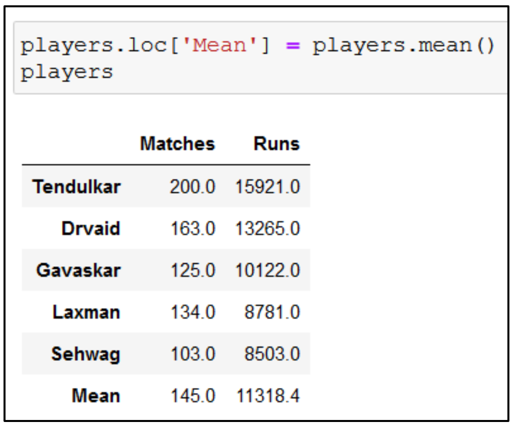
- The above can also be achieved using an ***for*** loop iteration as shown below:



```
players.loc['Mean'] = [np.mean(players[col]) for col in players.columns]
players
```

	Matches	Runs
Tendulkar	200.0	15921.0
Drvaid	163.0	13265.0
Gavaskar	125.0	10122.0
Laxman	134.0	8781.0
Sehwag	103.0	8503.0
Mean	145.0	11318.4

- You must be wondering if there is a better way supported by Pandas itself to calculate the mean on the columns. Since this is a very common operation across data tables. The answer is YES. We can directly use ***mean ()*** function of the DataFrame object (similarly, many more functions are available with data objects that can be explored using with dot (.) + TAB.

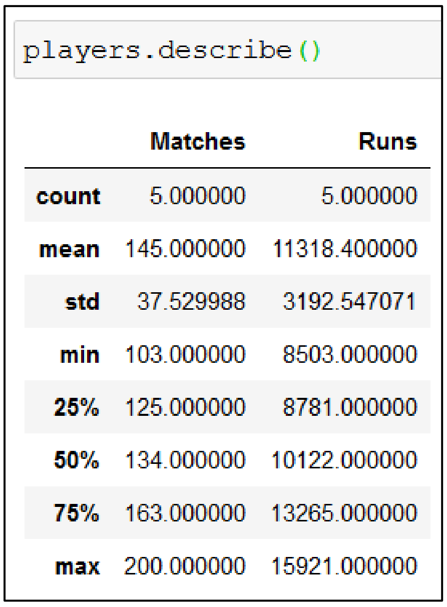


```
players.loc['Mean'] = players.mean()
players
```

	Matches	Runs
Tendulkar	200.0	15921.0
Drvaid	163.0	13265.0
Gavaskar	125.0	10122.0
Laxman	134.0	8781.0
Sehwag	103.0	8503.0
Mean	145.0	11318.4

Note: drop Mean row every time while trying out different ways as shown above to calculate the mean using a different method. Otherwise, it will be added in the mean calculations.

- Mean for the individual rows can also be calculated using axis=1 as an argument in the mean function.
- Pandas provides a very useful function ***describe ()*** for objects that provides mean, min, max, standard deviation and quartiles for the column data.



```
players.describe()
```

	Matches	Runs
count	5.000000	5.000000
mean	145.000000	11318.400000
std	37.529988	3192.547071
min	103.000000	8503.000000
25%	125.000000	8781.000000
50%	134.000000	10122.000000
75%	163.000000	13265.000000
max	200.000000	15921.000000