# Series

Series is a one-dimensional array like object containing an array of data(any Numpy data type, and an associated array of data labels, called its index.

```
mjp= Series([5,4,3,2,1])# a simple series

print(mjp)       # A series is represented by index on the left and values on the right

print(mjp.values)# similar to dictionary. ".values" command returns values in a series

**********************************************************************************

print(mjp.index)  # returns the index values of the series

**********************************************************************************

jeeva = Series([5,4,3,2,1,-7,-29], index =['a','b','c','d','e','f','h'])   # The index is specified

print(jeeva)   # try jeeva.index and jeeva.values

print(jeeva['a'])  # selecting a particular value from a Series, by using index

**********************************************************************************

jeeva['d'] = 9   # change the value of a particular element in series

print(jeeva)

jeeva[['a','b','c']]   # select a group of values

**********************************************************************************

print(jeeva[jeeva>0])  # returns only the positive values

print(jeeva *2)  # multiplies 2 to each element of a series

**********************************************************************************

import numpy as np

np.mean(jeeva)  # you can apply numpy functions to a Series

**********************************************************************************

print('b' in jeeva)   # checks whether the index is present in Series or not

print('z' in jeeva)
```

**********

player_salary ={'Rooney': 50000, 'Messi': 75000, 'Ronaldo': 85000, 'Fabregas':40000, 'Van persie': 67000}

new_player = Series(player_salary) **# converting a dictionary to a series**

print(new_player) **# the series has keys of a dictionary**

**********

players =['Klose', 'Messi', 'Ronaldo', 'Van persie', 'Ballack']

player_1 =Series(player_salary, index= players)

print(player_1) **# I have changed the index of the Series. Since, no value was not found for Klose and Ballack, it appears as NAN**

**********

print(pd.isnull(player_1)) **#checks for Null values in player_1, pd denotes a pandas dataframe**

**********

print(pd.notnull(player_1)) **# Checks for null values that are not Null**

**********

player_1.name ='Bundesliga players' **# name for the Series**

player_1.index.name='Player names' **#name of the index**

print(player_1)

**********

player_1.index =['Neymar', 'Hulk', 'Pirlo', 'Buffon', 'Anderson'] **# is used to alter the index of Series**

print(player_1)

**********

# Data Frame

Data frame is a spread sheet like structure, containing ordered collection of columns. Each column can have different value type. Data frame has both row index and column index.

**********

```
states ={'State' :['Gujarat', 'Tamil Nadu', ' Andhra', 'Karnataka', 'Kerala'],
        'Population': [36, 44, 67,89,34],
        'Language' :['Gujarati', 'Tamil', 'Telugu', 'Kannada', 'Malayalam']}
india = DataFrame(states)   # creating a data frame
print(india)
**********
DataFrame(states, columns=['State', 'Language', 'Population'])  # change the sequence of column index
**********
new_farme = DataFrame(states, columns=['State', 'Language', 'Population', 'Per Capita Income'], index =['a','b','c','d','e'])
#if you pass a column that isnt in states, it will appear with Na values
**********
print(new_farme.columns)
print(new_farme['State'])   # retrieveing data like dictionary
**********
print(new_farme.Population)  # like Series
**********
print(new_farme.iloc[3])   # rows can be retrieved using .iloc function
# here I have retrieved 3rd row
**********
print(new_farme)
**********
new_farme['Per Capita Income'] = 99  # the empty per capita income column can be assigned a value
print(new_farme)
**********
new_farme['Per Capita Income'] = np.arange(5)   # assigning a value to the last column
```

```
print(new_farme)
```

**********

```
series = Series([44,33,22], index =['b','c','d'])
new_farme['Per Capita Income'] = series
```

**#when assigning list or arrays to a column, the values length should match the length of the DataFrame**

```
print(new_farme)
```
**# again the missing values are displayed as NAN**

**********

```
new_farme['Development'] = new_farme.State == 'Gujarat'
```
**# assigning a new column**

```
print(new_farme)
del new_farme['Development']
```
**# will delete the column 'Development'**

```
print(new_farme)
```

**********

```
new_data ={'Modi': {2010: 72, 2012: 78, 2014 : 98},'Rahul': {2010: 55, 2012: 34, 2014: 22}}
elections = DataFrame(new_data)
print(elections)
```
**# the outer dict keys are columns and inner dict keys are rows**

```
elections.T
```
**# transpose of a data frame**

**********

```
DataFrame(elections, index =[2012, 2014, 2016])
```
**# you can assign index for the data frame**

**********

```
ex= {'Gujarat':elections['Modi'][:-1], 'India': elections['Rahul'][:2]}
px =DataFrame(ex)
print(px)
```

**********

```
px.index.name = 'year'
px.columns.name = 'politicians'
```

```
print(px)
**********
print(px.values)
**********
jeeva = Series([5,4,3,2,1,-7,-29], index =['a','b','c','d','e','f','h'])
index = jeeva.index
print(index)
print(index[1:])  # returns all the index elements except a.
index[1] = 'f'   # you cannot modify an index element. It will generate an error. In other words, they are immutable
**********
print(px)
2013 in px.index  # checks if 2003 is an index in data frame px
**********
```

# Reindex

```
var = Series(['Python', 'Java', 'c', 'c++', 'Php'], index =[5,4,3,2,1])
print(var)
var1 = var.reindex([1,2,3,4,5])  # reindex creates a new object
print(var1)
**********
print(var.reindex([1,2,3,4,5,6,7]))  # introduces new indexes with values Nan
**********
print(var.reindex([1,2,3,4,5,6,7], fill_value =1))
**********
```

```
gh = Series(['Dhoni', 'Sachin', 'Kohli'], index =[0,2,4])

print(gh)

print(gh.reindex(range(6), method ='ffill')) #ffill is forward fill. It forward fills the values

**********

print(gh.reindex(range(6), method ='bfill'))# bfill, backward fills the values

**********

import numpy as np

fp = DataFrame(np.arange(9).reshape((3,3)),index =['a','b','c'], columns =['Gujarat','Tamil Nadu',
'Kerala'])

print(fp)

**********

print(states)

fp1 =fp.reindex(['a', 'b', 'c', 'd'], columns = states) # reindexing columns and indices

print(fp1)

**********
```

Other Reindexing arguments
**limit** When forward- or backfilling, maximum size gap to fill
**level** Match simple Index on level of MultiIndex, otherwise select subset of
**copy** Do not copy underlying data if new index is equivalent to old index. True by default (i.e. always copy data).

**********

# Dropping entries from an axis

```
er = Series(np.arange(5), index =['a','b','c','d','e'])

print(er)

er.drop(['a','b']) #drop method will return a new object  with values deleted from an axis

**********
```

```
states ={'State' :['Gujarat', 'Tamil Nadu', ' Andhra', 'Karnataka', 'Kerala'],
        'Population': [36, 44, 67,89,34],
        'Language' :['Gujarati', 'Tamil', 'Telugu', 'Kannada', 'Malayalam']}
india = DataFrame(states, columns =['State', 'Population', 'Language'])
print(india)
india.drop([0,1])# will drop index 0 and 1
**********
india.drop(['State', 'Population'], axis =1 )# the function dropped population and state columns. Apply the same concept with axis =0
**********
```

## #Selection, Indexing and Filtering

```
var = Series(['Python', 'Java', 'c', 'c++', 'Php'], index =[5,4,3,2,1])
print(var)
**********
print(var[5])
print(var[2:4])
**********
print(var[[3,2,1]])
**********
print(var[var == 'Php'])
**********
states ={'State' :['Gujarat', 'Tamil Nadu', ' Andhra', 'Karnataka', 'Kerala'],
        'Population': [36, 44, 67,89,34],
        'Language' :['Gujarati', 'Tamil', 'Telugu', 'Kannada', 'Malayalam']}
india = DataFrame(states, columns =['State', 'Population', 'Language'])
print(india)
```

```
**********************************************************************************

print(india[['Population', 'Language']])    # retrieve data from data frame

**********************************************************************************

print(india[india['Population'] > 50]) # returns data for population greater than 50

**********************************************************************************

print(india[:3]) # first three rows

**********************************************************************************

# for selecting specific rows and columns, you can use iloc function

import pandas as pd

states ={'State' :['Gujarat', 'Tamil Nadu', ' Andhra', 'Karnataka', 'Kerala'],

            'Population': [36, 44, 67,89,34],

            'Language' :['Gujarati', 'Tamil', 'Telugu', 'Kannada', 'Malayalam']}

india = DataFrame(states, columns =['State', 'Population', 'Language'], index =['a', 'b', 'c', 'd', 'e'])

print(india)

**********************************************************************************

print(india.loc[['a','b'], ['State','Language']]) # this is how you select subset of rows

**********************************************************************************
```