Topics

- Introduction of Numpy
- Indexing & slicing
- Mathematical computation
- Array comparison
- array Manipulation
- transpose & swapcase
- insert and remove

Topics

- **What is Numpy?**
- **Why Numpy?**
- **What is Array?**
- **Dimensions in Arrays**
- **Initialization of an Array.**

What is Numerical Python?

- NumPy is the fundamental package for scientific computing with Python.
- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Why NumPy?

- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science.
- This is the main reason why Numpy is faster than lists. Also it is optimized to work with latest CPU architectures.

Arrays

- NumPy’s main object is the homogeneous multidimensional array.
- It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
- In NumPy dimensions are called *axes*. The number of axes is *rank*.
- NumPy’s array class is called **ndarray**. It is also known by the alias **array**.

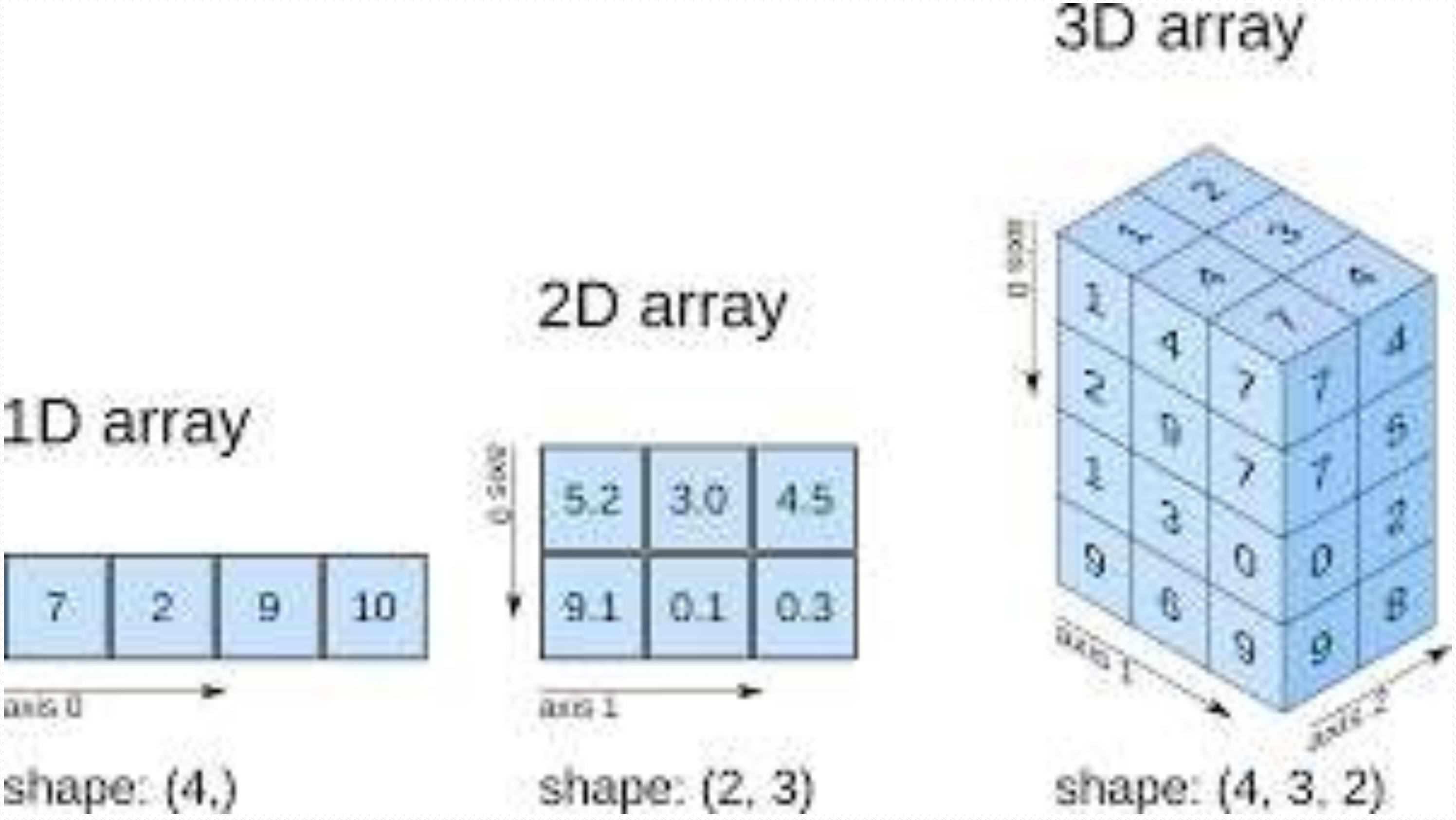
Dimensions in Arrays

- A dimension in arrays is one level of array depth

1-D Arrays

- An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.These are the most common and basic arrays.

-
  ```
  import numpy as np
  arr = np.array([1, 2, 3, 4, 5])
  print(arr)
  ```


1D array
7
2
9
10
axis 0
shape: (4,)
2D array
axis 0
5.2
3.0
4.5
9.1
0.1
0.3
axis 1
shape: (2, 3)
3D array
axis 0
axis 1
axis 2
shape: (4, 3, 2)

2-D Arrays

- An array that has 1-D arrays as its elements is called a 2-D array.These are often used to represent matrix or 2nd order tensors.
-
  ```
  import numpy as np
  arr = np.array([[1, 2, 3], [4, 5, 6]])
  print(arr)
  ```

3-D arrays

- An array that has 2-D arrays (matrices) as its elements is called 3-D array.These are often used to represent a 3rd order tensor.
-
  ```
  import numpy as np
  arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
  print(arr)
  ```

INITIALIZTION

np.array([1,2,3])	**1d array**
np.array([(1,2,3),(4,5,6)])	2d array
np.arange(start,stop,step)	range array
np.linspace(0,2,9)	Add evenly spaced values btw interval to array of length
np.zeros((1,2))	Create and array filled with zeros
np.ones((1,2))	Creates an array filled with ones
np.random.random((5,5))	Creates random array
np.empty((2,2))	Creates an empty array

ARRAY PROPERTIES

Syntax	Description
`array.shape`	Dimensions (Rows,Columns)
`len(array)`	Length of Array
`array.ndim`	Number of Array Dimensions
`array.size`	Number of Array Elements
`array.dtype`	Data Type
`type(array)`	Type of Array

COPYING/SORTING

`np.copy(array)`	**Creates copy of array**
`array.sort()`	Sorts an array
`array.sort(axis=0)`	Sorts axis of array

Operations

Operator	Description

`np.add(x,y)`
`x + y`
	Addition

`np.substract(x,y)`
`x − y`
	Subtraction

`np.divide(x,y)`
`x / y`
	Division

`np.multiply(x,y)`
`x @ y`
	Multiplication

`np.sqrt(x)`	Square Root
`np.sin(x)`	Element-wise sine
`np.cos(x)`	Element-wise cosine
`np.log(x)`	Element-wise natural log
`np.dot(x,y)`	Dot product
`np.roots([1,0,-4])`	Roots of a given polynomial coefficients

Data Analysis in Python using Pandas

Pandas

- Pyton Data analysis library
- Built on top of Numpy
- Abbreviation of **Pan**el **Da**ta **S**ystem
- Used in production in many companies

The Ideal tool for data Scientists

- Managing data
- Cleaning data
- Analyzing
- Modeling data
- Organizing the data in a form suitable for plotting or tabular display

DataFrame

- Python DataFrame is a data structure containing and ordered collecetions of columns.

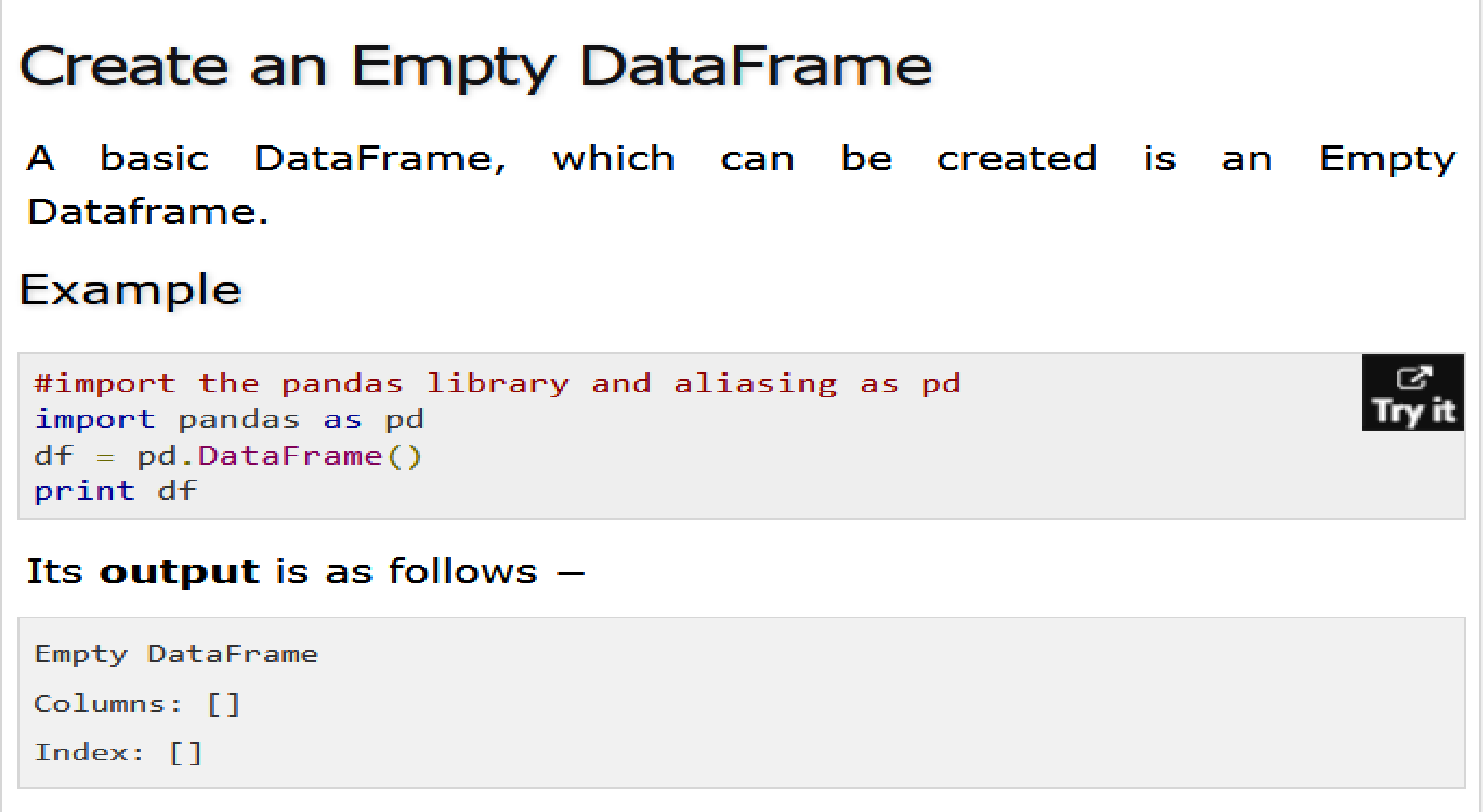
- Each column may hold numeric, string, boolean etc. Values

- DataFrame has both row and column index

Creating a DataFrame

○A pandas DataFrame can be created using various inputs like

- --Lists
- --Dict
- --Series
- --Numpy ndarrays
- --Another DataFrame



Create an Empty DataFrame

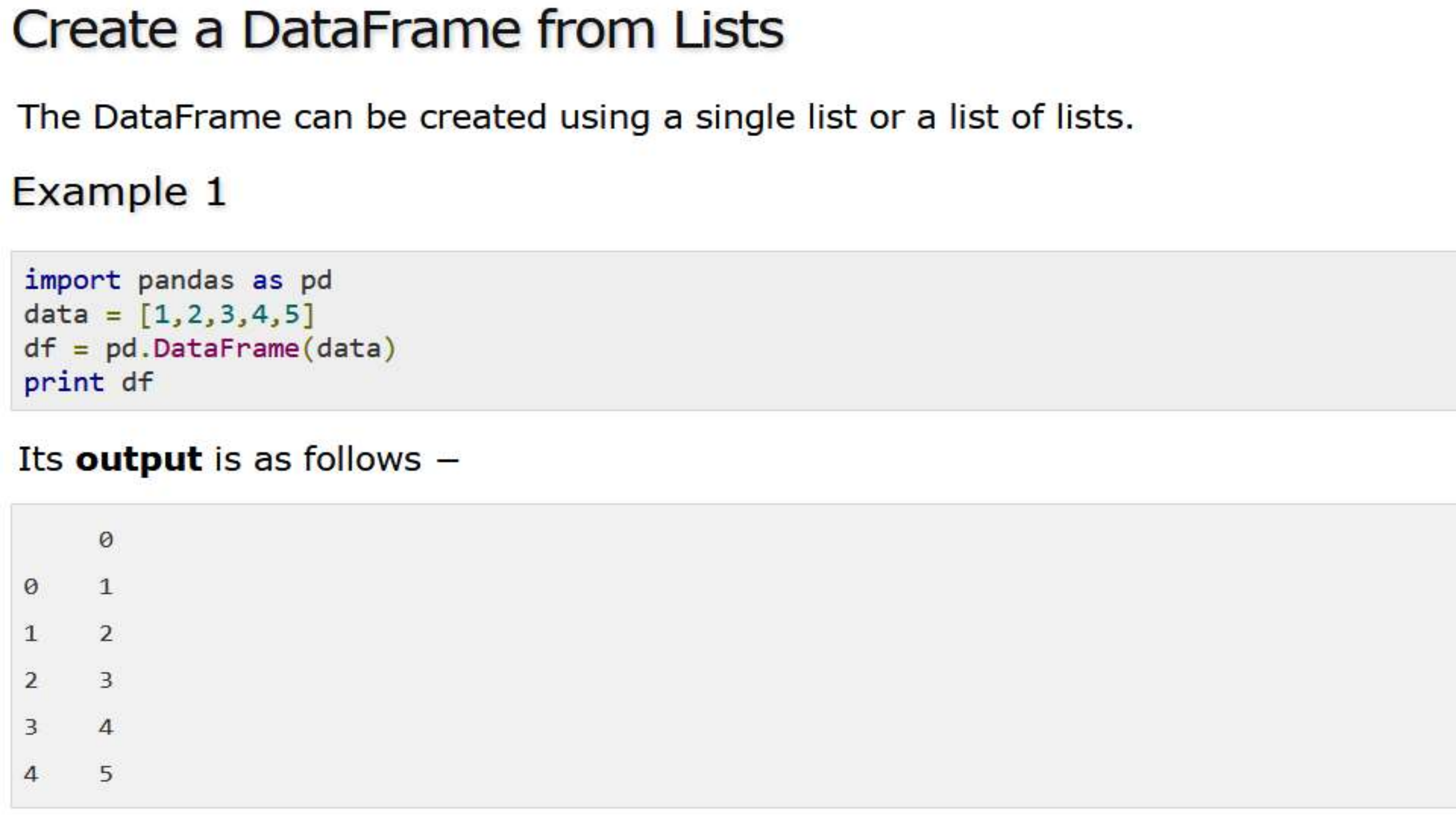
A basic DataFrame, which can be created is an Empty Dataframe.

Example

```
#import the pandas library and aliasing as pd
import pandas as pd
df = pd.DataFrame()
print df
```

Its **output** is as follows −

```
Empty DataFrame
Columns: []
Index: []
```



Create a DataFrame from Lists

The DataFrame can be created using a single list or a list of lists.

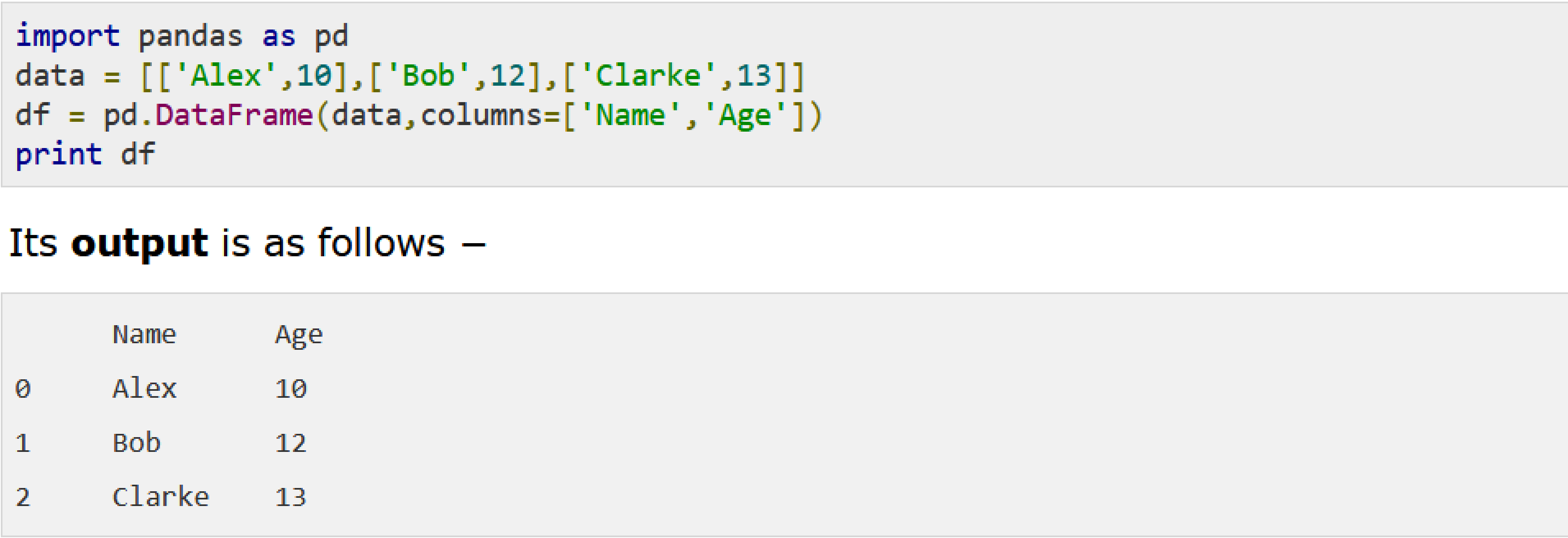
Example 1

```
import pandas as pd
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print df
```

Its **output** is as follows −

```
     0
0    1
1    2
2    3
3    4
4    5
```

Example 2

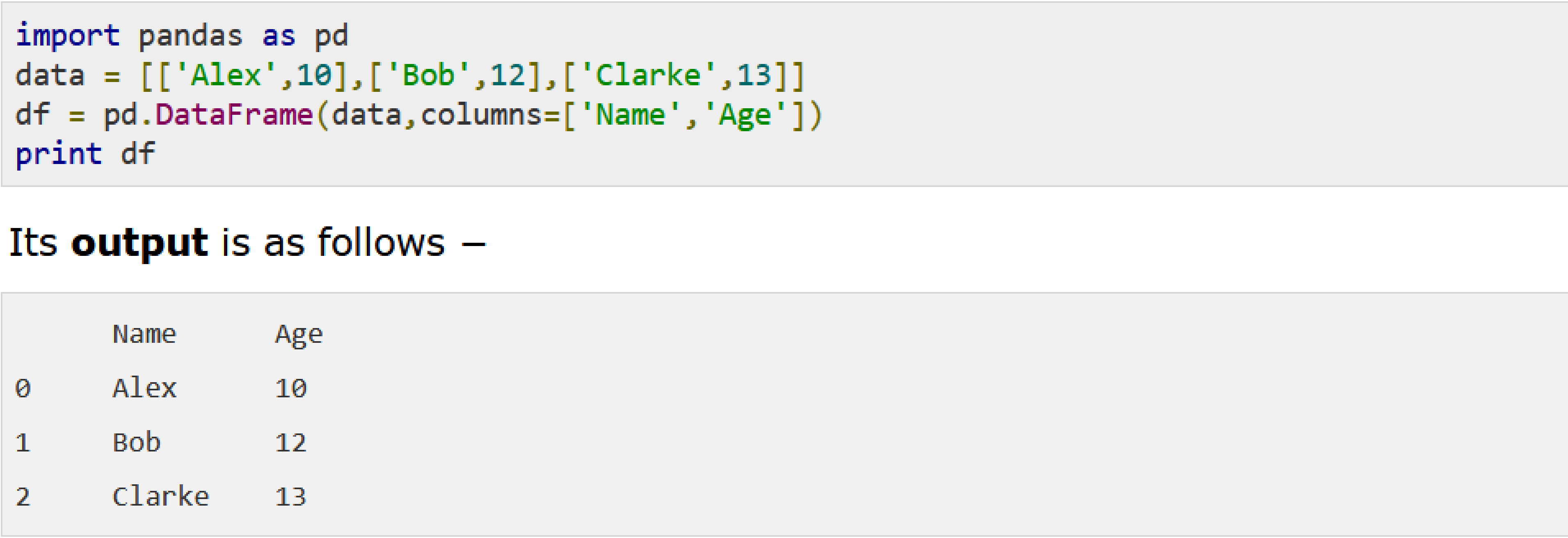


```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print df
```

Its **output** is as follows −

```
      Name      Age
0     Alex      10
1     Bob       12
2     Clarke    13
```

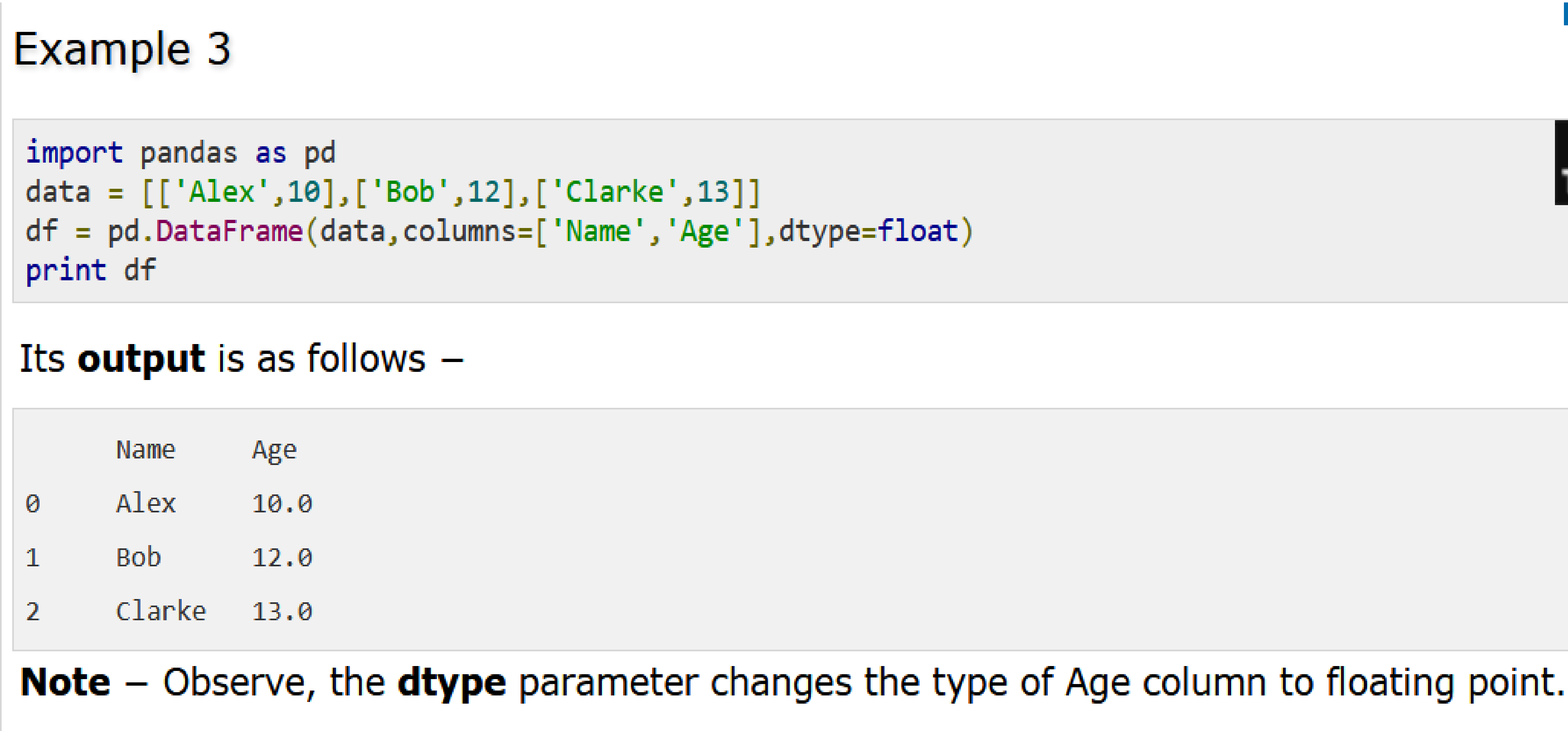
Example 2



```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print df
```

Its **output** is as follows −

```
      Name      Age
0     Alex      10
1     Bob       12
2     Clarke    13
```



Example 3

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print df
```

Its **output** is as follows −

```
      Name     Age
0     Alex     10.0
1     Bob      12.0
2     Clarke   13.0
```

Note − Observe, the **dtype** parameter changes the type of Age column to floating point.

Example 2

Let us now create an indexed DataFrame using arrays.

```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
print df
```

Its **output** is as follows −

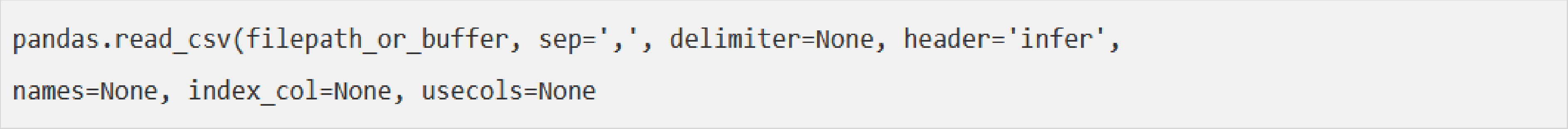
```
         Age    Name
rank1    28      Tom
rank2    34     Jack
rank3    29    Steve
rank4    42    Ricky
```

Note − Observe, the **index** parameter assigns an index to each row.

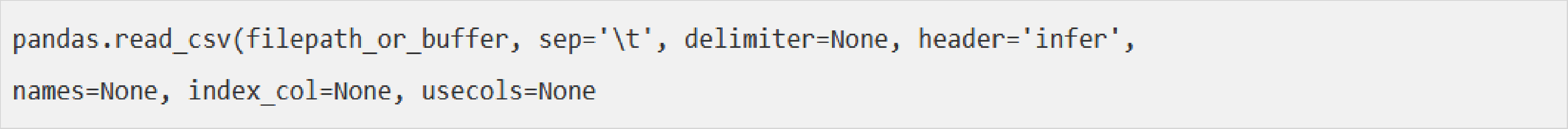
Python Pandas Input/Output TOOLS

○The **Pandas I/O API** is a set of top level reader functions accessed like **pd.read_csv()** that generally return a Pandas object.

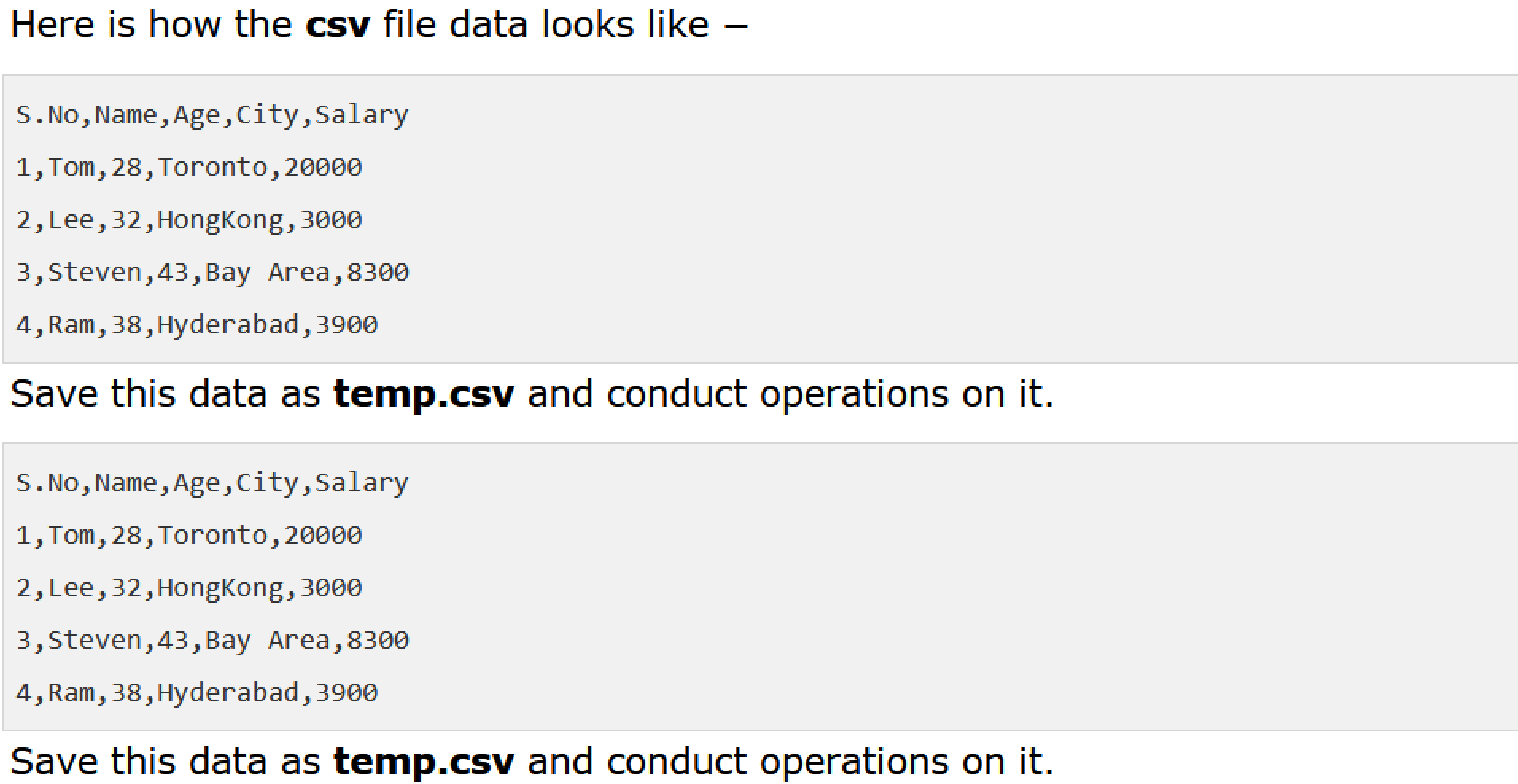
○The two functions for reading text files are **read_csv()** and **read_table()**. They both intelligently convert tabular data into a **DataFrame** object



```
pandas.read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer',

names=None, index_col=None, usecols=None
```



```
pandas.read_csv(filepath_or_buffer, sep='\t', delimiter=None, header='infer',

names=None, index_col=None, usecols=None
```



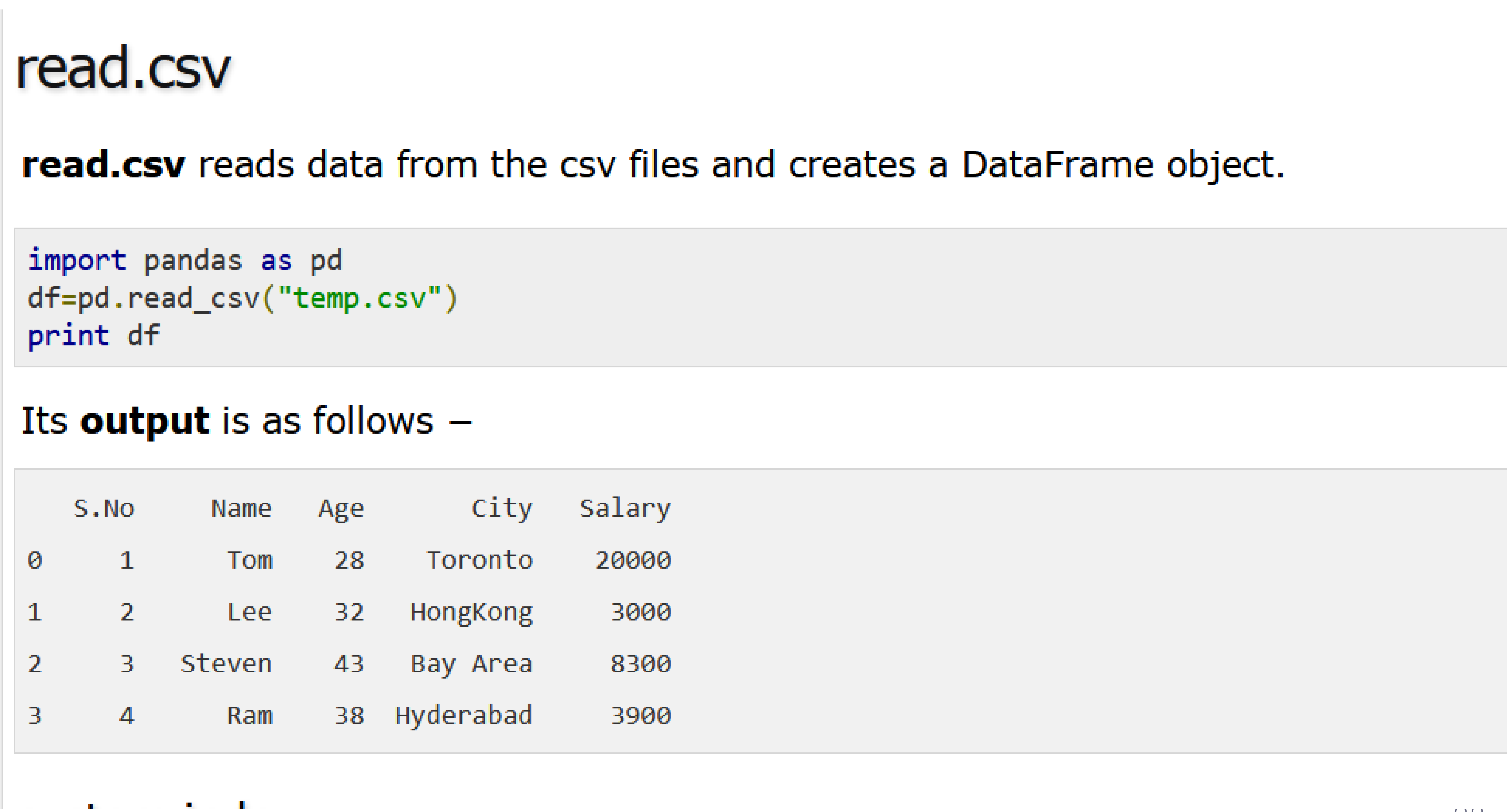
Here is how the **csv** file data looks like –

```
S.No,Name,Age,City,Salary
1,Tom,28,Toronto,20000
2,Lee,32,HongKong,3000
3,Steven,43,Bay Area,8300
4,Ram,38,Hyderabad,3900
```

Save this data as **temp.csv** and conduct operations on it.

```
S.No,Name,Age,City,Salary
1,Tom,28,Toronto,20000
2,Lee,32,HongKong,3000
3,Steven,43,Bay Area,8300
4,Ram,38,Hyderabad,3900
```

Save this data as **temp.csv** and conduct operations on it.



read.csv

read.csv reads data from the csv files and creates a DataFrame object.

```
import pandas as pd
df=pd.read_csv("temp.csv")
print df
```

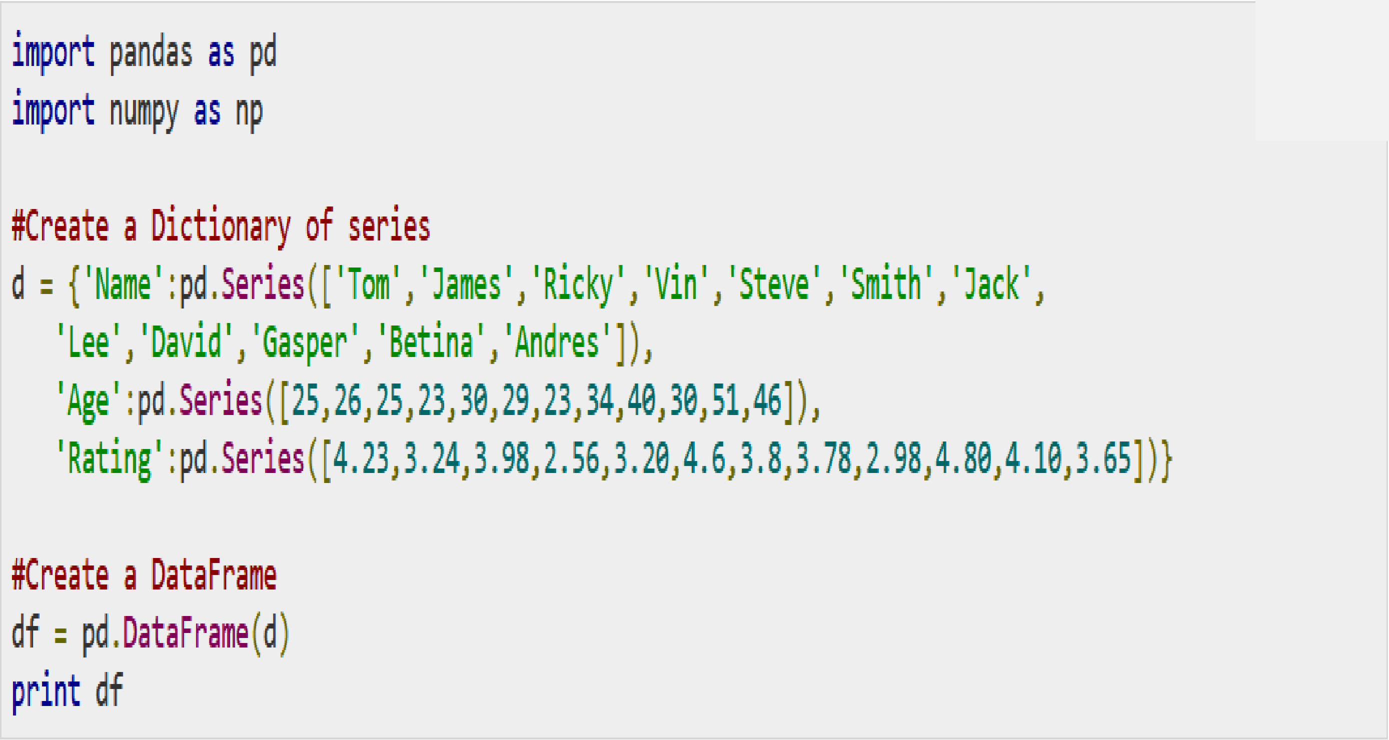
Its **output** is as follows −

```
   S.No     Name   Age       City   Salary
0     1      Tom    28    Toronto    20000
1     2      Lee    32   HongKong     3000
2     3   Steven    43   Bay Area     8300
3     4      Ram    38  Hyderabad     3900
```

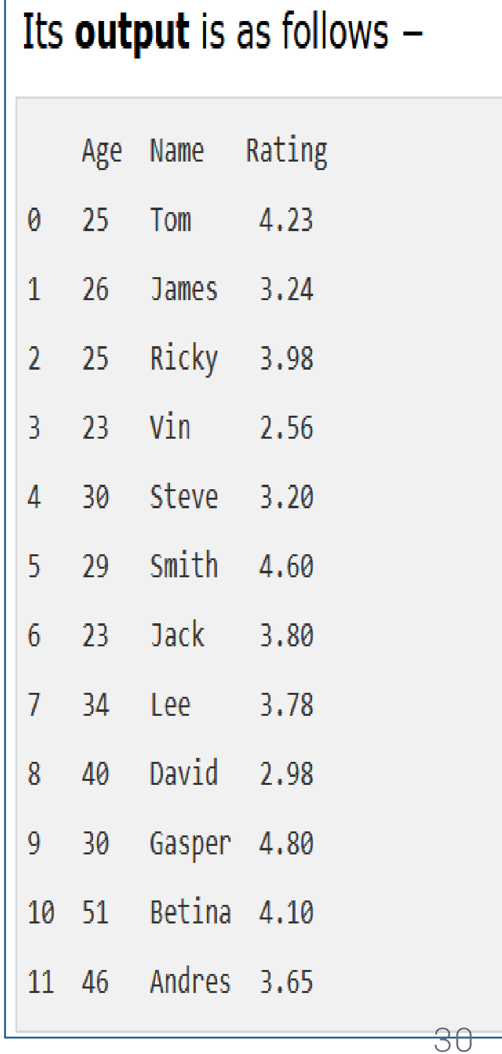
Python Pandas

Let us create a DataFrame and use this object throughout this chapter for all the operati

Example



```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df
```



Its **output** is as follows −

```
    Age  Name    Rating
0   25   Tom      4.23
1   26   James    3.24
2   25   Ricky    3.98
3   23   Vin      2.56
4   30   Steve    3.20
5   29   Smith    4.60
6   23   Jack     3.80
7   34   Lee      3.78
8   40   David    2.98
9   30   Gasper   4.80
10  51   Betina   4.10
11  46   Andres   3.65
```

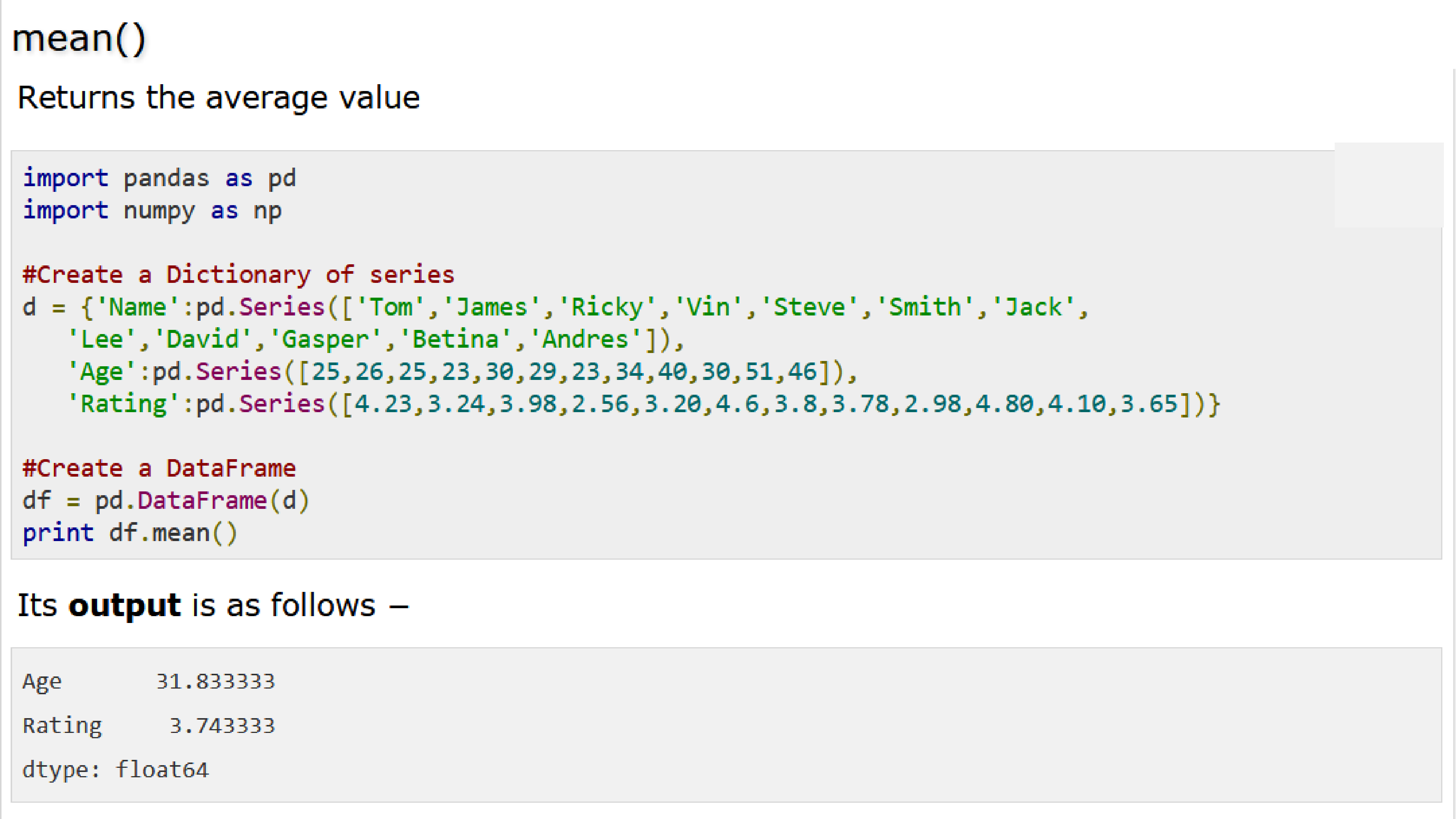
sum()

Returns the sum of the values for the requested axis. By default, axis is index (axis=0).

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.sum()
```

Its **output** is as follows −

```
Age                                                    382
Name     TomJamesRickyVinSteveSmithJackLeeDavidGasperBe...
Rating                                               44.92
dtype: object
```



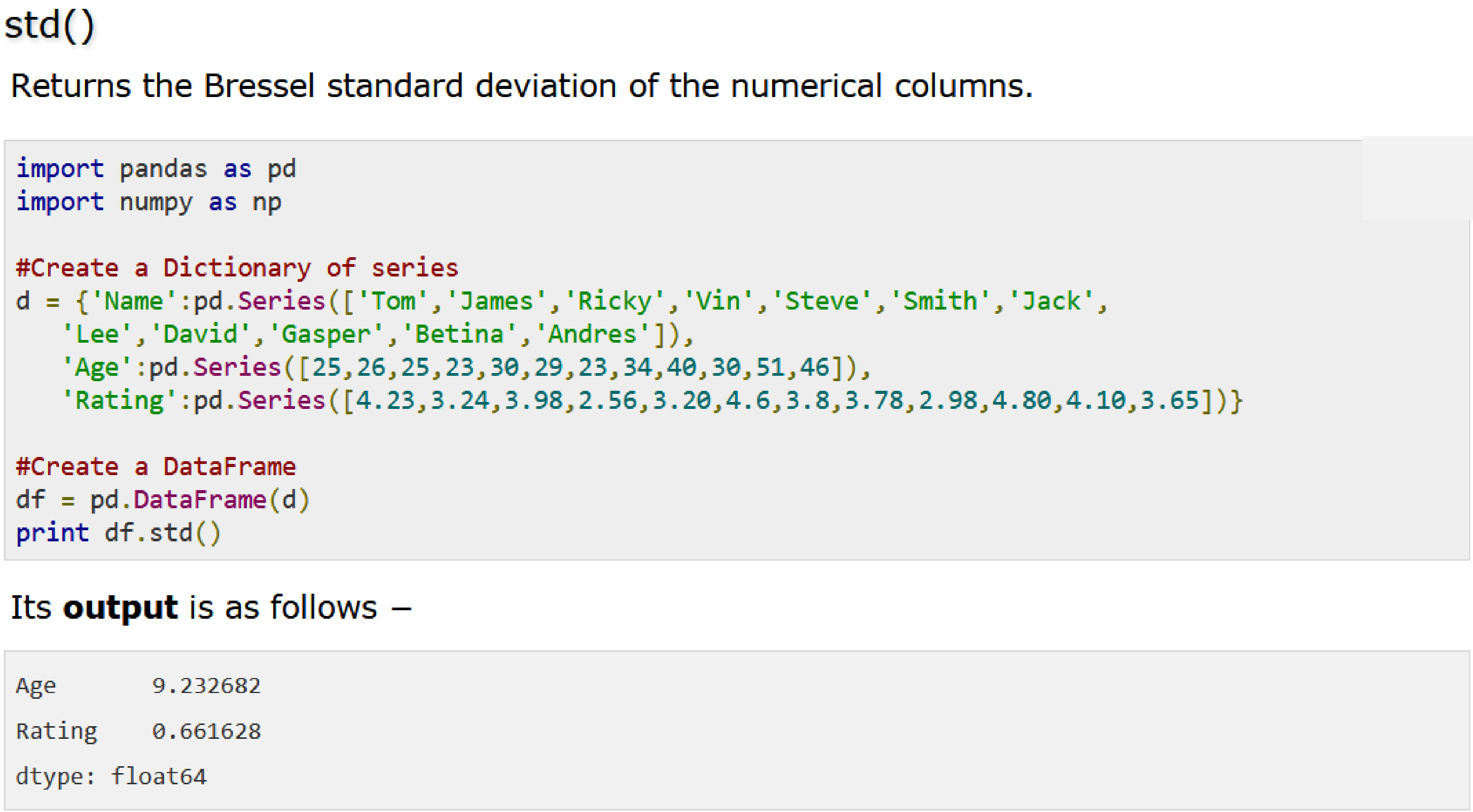
mean()

Returns the average value

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.mean()
```

Its **output** is as follows −

```
Age         31.833333

Rating       3.743333

dtype: float64
```



std()

Returns the Bressel standard deviation of the numerical columns.

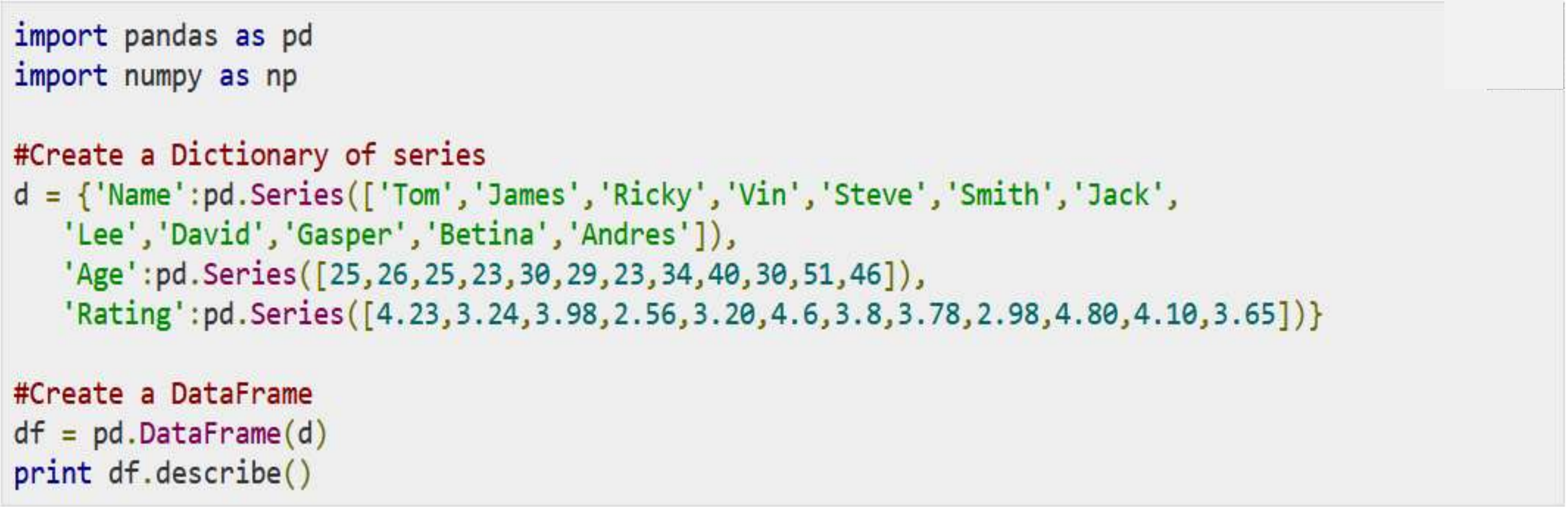
```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.std()
```

Its **output** is as follows −

```
Age         9.232682

Rating      0.661628

dtype: float64
```

Summarizing Data

The **describe()** function computes a summary of statistics pertaining to the DataFrame columns.



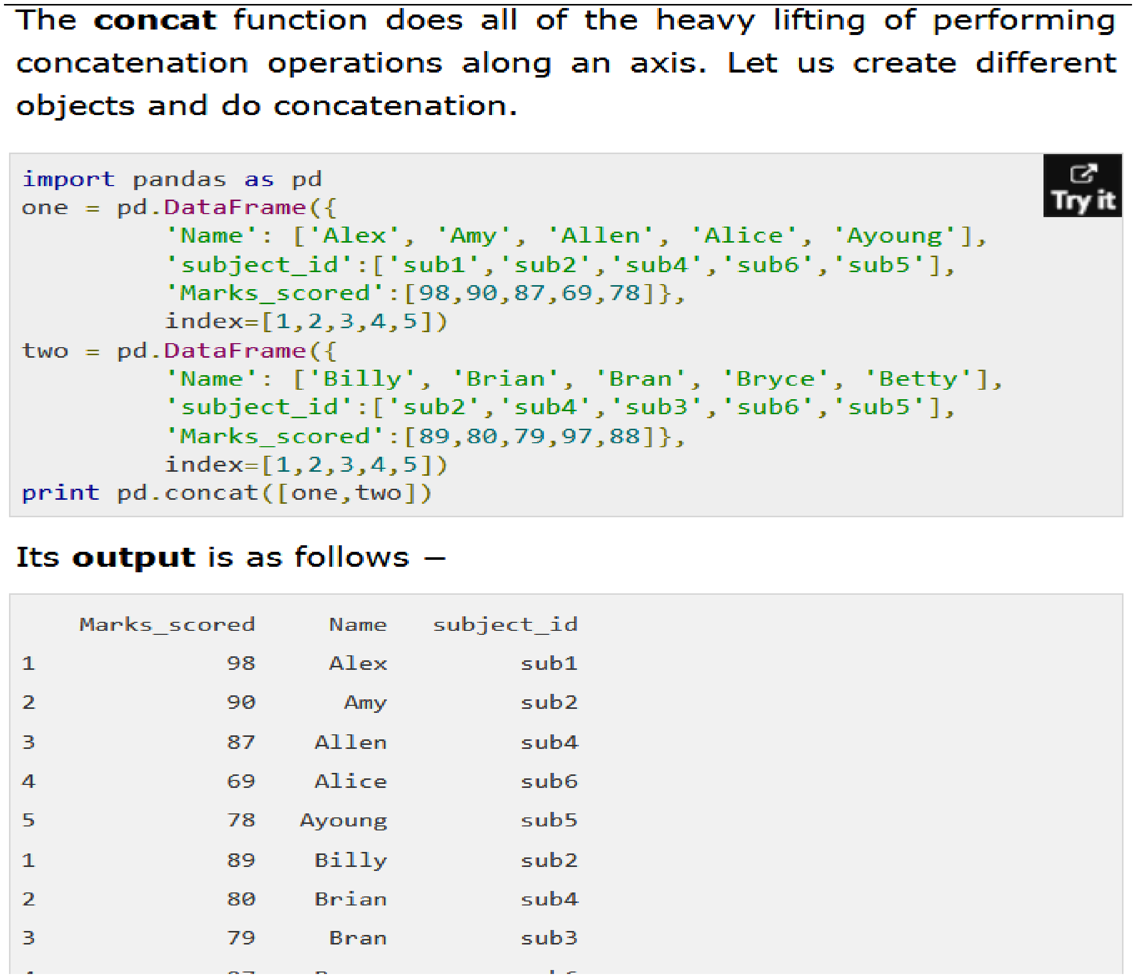
```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack',
   'Lee','David','Gasper','Betina','Andres']),
   'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
   'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print df.describe()
```

Its **output** is as follows −

```
               Age         Rating
count    12.000000      12.000000
mean     31.833333       3.743333
std       9.232682       0.661628
min      23.000000       2.560000
25%      25.000000       3.230000
50%      29.500000       3.790000
75%      35.500000       4.132500
max      51.000000       4.800000
```

This function gives the **mean, std** and **IQR** values. And, function excludes the character columns and given summary about numeric columns. **'include'** is the argument which is used to pass necessary information regarding what columns need to be considered for summarizing. Takes the list of values; by default, 'number'.

Python Pandas Concatenation



The **concat** function does all of the heavy lifting of performing concatenation operations along an axis. Let us create different objects and do concatenation.

```
import pandas as pd
one = pd.DataFrame({
         'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
         'subject_id':['sub1','sub2','sub4','sub6','sub5'],
         'Marks_scored':[98,90,87,69,78]},
         index=[1,2,3,4,5])
two = pd.DataFrame({
         'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
         'subject_id':['sub2','sub4','sub3','sub6','sub5'],
         'Marks_scored':[89,80,79,97,88]},
         index=[1,2,3,4,5])
print pd.concat([one,two])
```

Its **output** is as follows −

```
    Marks_scored     Name   subject_id
1             98     Alex         sub1
2             90      Amy         sub2
3             87    Allen         sub4
4             69    Alice         sub6
5             78   Ayoung         sub5
1             89    Billy         sub2
2             80    Brian         sub4
3             79     Bran         sub3
```