

# Python Pandas - Descriptive Statistics

A large number of methods collectively compute descriptive statistics and other related operations on DataFrame. Most of these are aggregations like **sum()**, **mean()**, but some of them, like **sumsum()**, produce an object of the same size. Generally speaking, these methods take an **axis** argument, just like *ndarray*.{*sum*, *std*, ...}, but the axis can be specified by name or integer

- **DataFrame** – “index” (axis=0, default), “columns” (axis=1)

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

## Example

[Live Demo](#)

```
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()
```

[Live Demo](#)

Its **output** is as follows –

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

Each individual column is added individually (Strings are appended).

## axis=1

This syntax will give the output as shown below.

```
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(1)
```

[Live Demo](#)

Its **output** is as follows –

```
0    29.23
1    29.24
2    28.98
```

```
3    25.56
4    33.20
5    33.60
6    26.80
7    37.78
8    42.98
9    34.80
10   55.10
11   49.65
dtype: float64
```

## 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()
```

[Live Demo](#)

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()
```

[Live Demo](#)

Its **output** is as follows –

```
Age      9.232682
Rating   0.661628
dtype: float64
```

## Functions & Description

Let us now understand the functions under Descriptive Statistics in Python Pandas. The following table list down the important functions –

Sr.No.	Function	Description
1	count()	Number of non-null observations
2	sum()	Sum of values
3	mean()	Mean of Values
4	median()	Median of Values
5	mode()	Mode of values
6	std()	Standard Deviation of the Values
7	min()	Minimum Value
8	max()	Maximum Value
9	abs()	Absolute Value
10	prod()	Product of Values
11	cumsum()	Cumulative Sum
12	cumprod()	Cumulative Product

**Note** – Since DataFrame is a Heterogeneous data structure. Generic operations don't work with all functions.

- Functions like **sum()**, **cumsum()** work with both numeric and character (or) string data elements without any error. Though in practice, character aggregations are never used generally, these functions do not throw any exception.
- Functions like **abs()**, **cumprod()** throw exception when the DataFrame contains character or string data because such operations cannot be performed.

## Summarizing Data

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

[Live Demo](#)

```
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'.

- **object** – Summarizes String columns
- **number** – Summarizes Numeric columns
- **all** – Summarizes all columns together (Should not pass it as a list value)

Now, use the following statement in the program and check the output –

[Live Demo](#)

```
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(include=['object'])
```

Its **output** is as follows –

	Name
count	12
unique	12
top	Ricky
freq	1

Now, use the following statement and check the output –

[Live Demo](#)



```
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(include='all')
```

Its **output** is as follows –

	Age	Name	Rating
count	12.000000	12	12.000000
unique	NaN	12	NaN
top	NaN	Ricky	NaN
freq	NaN	1	NaN
mean	31.833333	NaN	3.743333
std	9.232682	NaN	0.661628
min	23.000000	NaN	2.560000
25%	25.000000	NaN	3.230000
50%	29.500000	NaN	3.790000
75%	35.500000	NaN	4.132500
max	51.000000	NaN	4.800000