Pandas for Machine Learning(Part 2)

AcadView

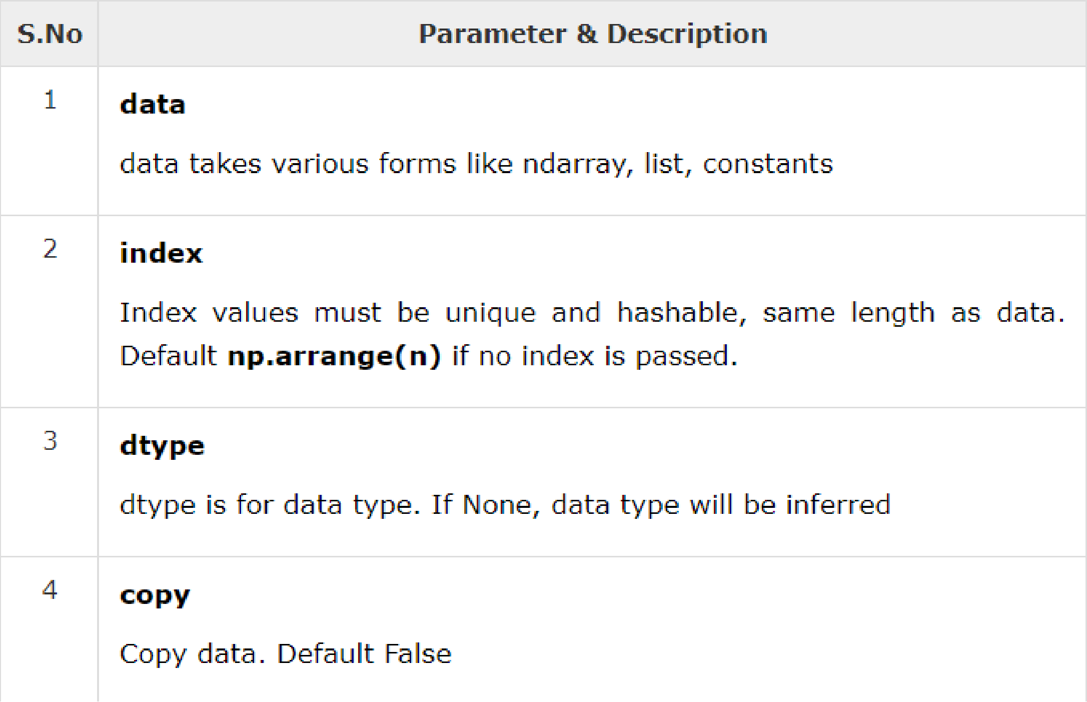
May 21, 2018

---

1 Series

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.
A pandas Series can be created using the following constructor :

$\mathbf{pandas}.\mathbf{Series}(\mathbf{data}, \mathbf{index}, \mathbf{dtype}, \mathbf{copy})$ The parameters of the constructor are as follows:

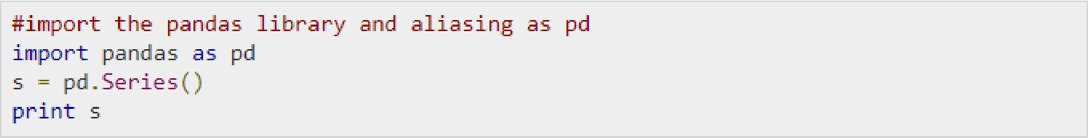


S.No	Parameter & Description
1	**data** data takes various forms like ndarray, list, constants
2	**index** Index values must be unique and hashable, same length as data. Default **np.arrange(n)** if no index is passed.
3	**dtype** dtype is for data type. If None, data type will be inferred
4	**copy** Copy data. Default False

1.1 Creating a Series

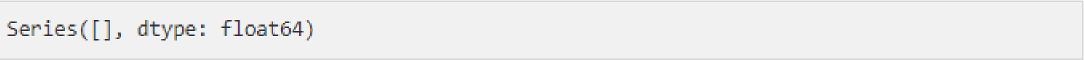
1.1.1 Creating Empty series

A basic series, which can be created is an Empty Series.
Example



```
#import the pandas library and aliasing as pd
import pandas as pd
s = pd.Series()
print s
```

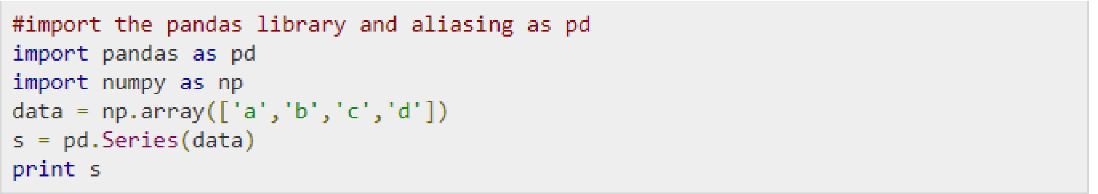
Its output is as follows:



```
Series([], dtype: float64)
```

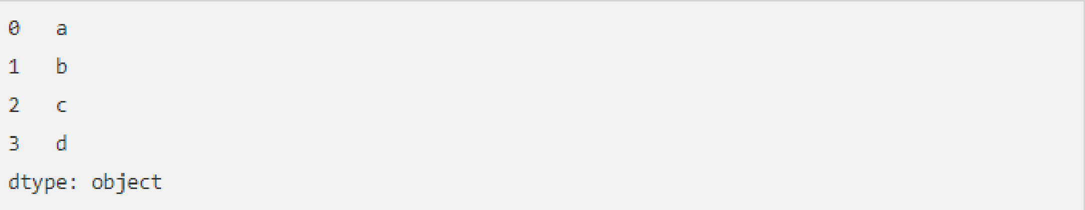
1.1.2 Create a Series from ndarray

If data is an ndarray, then index passed must be of the same length. If no index is passed, then by default index will be range(n) where n is array length, i.e., [0,1,2,3. range(len(array))-1].



```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print s
```

Its output is as follows:

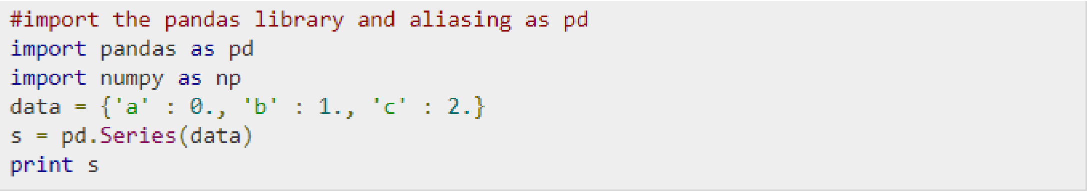


```
0   a
1   b
2   c
3   d
dtype: object
```

We did not pass any index, so by default, it assigned the indexes ranging from 0 to len(data)-1, i.e., 0 to 3.

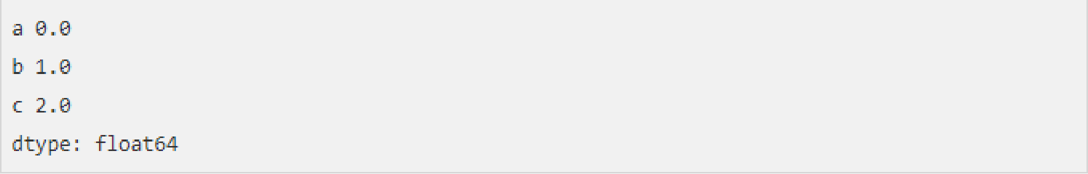
1.1.3 Create a Series from dict

A dict can be passed as input and if no index is specified, then the dictionary keys are taken in a sorted order to construct index. If index is passed, the values in data corresponding to the labels in the index will be pulled out.
Example1



```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data)
print s
```

Its output is as follows:

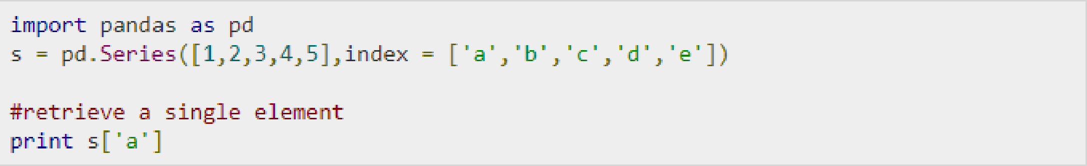


```
a 0.0
b 1.0
c 2.0
dtype: float64
```

1.2 Retrieve Data Using Label (Index)

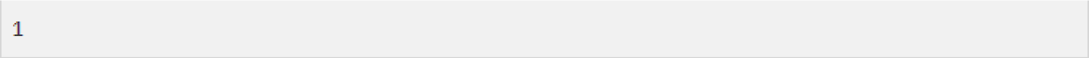
A Series is like a fixed-size dict in that you can get and set values by index label.

Example2



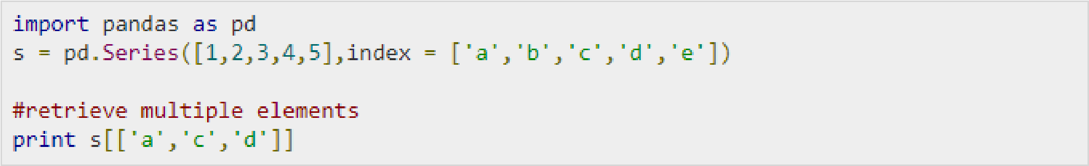
```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve a single element
print s['a']
```

Its output is as follows:



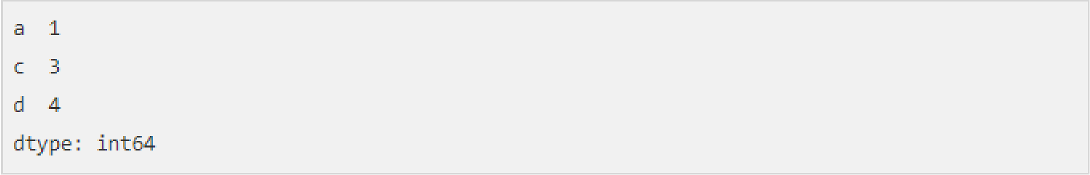
```
1
```

Example2



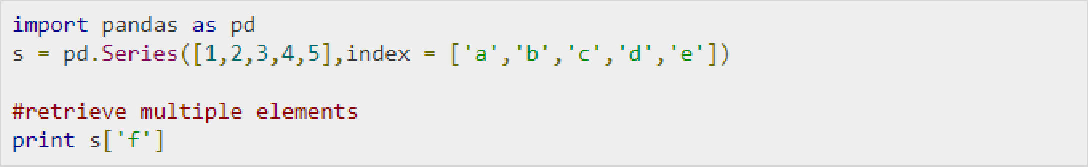
```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve multiple elements
print s[['a','c','d']]
```

Its output is as follows:



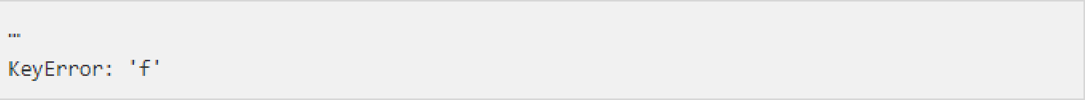
```
a  1
c  3
d  4
dtype: int64
```

Example 3



```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve multiple elements
print s['f']
```

Its output is as follows:



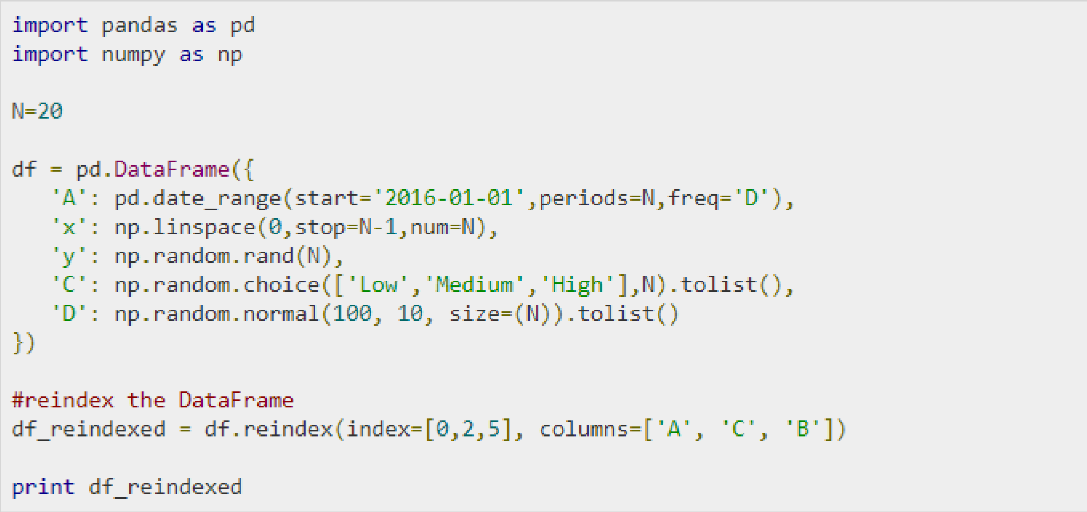
```
...
KeyError: 'f'
```

2 Reindexing

Reindexing changes the row labels and column labels of a DataFrame. To reindex means to conform the data to match a given set of labels along a particular axis.
Multiple operations can be accomplished through indexing like :

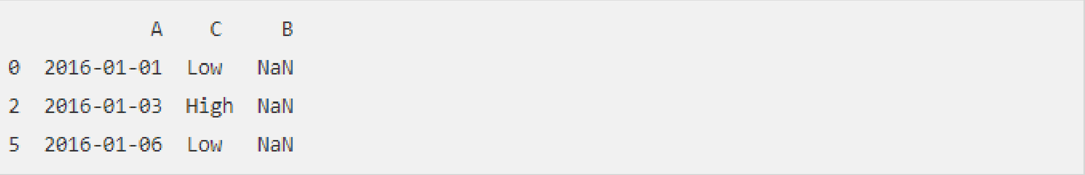
- Reorder the existing data to match a new set of labels.
- Insert missing value (NA) markers in label locations where no data for the label existed.

Example



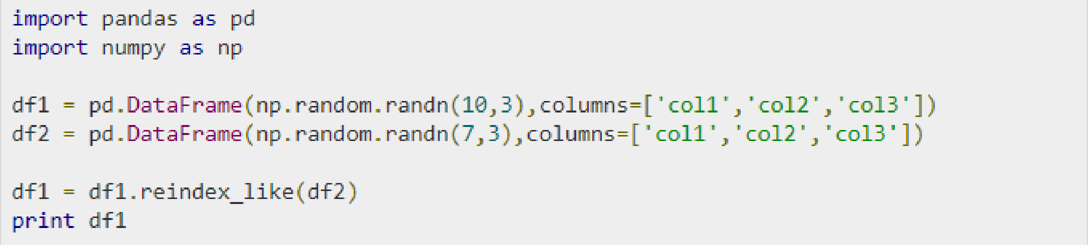
```
import pandas as pd
import numpy as np

N=20

df = pd.DataFrame({
   'A': pd.date_range(start='2016-01-01',periods=N,freq='D'),
   'x': np.linspace(0,stop=N-1,num=N),
   'y': np.random.rand(N),
   'C': np.random.choice(['Low','Medium','High'],N).tolist(),
   'D': np.random.normal(100, 10, size=(N)).tolist()
})

#reindex the DataFrame
df_reindexed = df.reindex(index=[0,2,5], columns=['A', 'C', 'B'])

print df_reindexed
```

Its output is as follows:



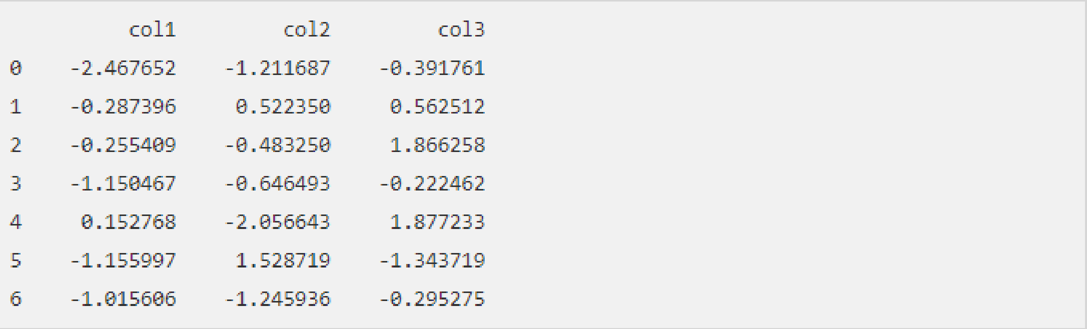
```
            A     C     B
0  2016-01-01  Low   NaN
2  2016-01-03  High  NaN
5  2016-01-06  Low   NaN
```

Reindex to Align with Other Objects You may wish to take an object and reindex its axes to be labeled the same as another object. Consider the following example to understand the same.
Example



```
import pandas as pd
import numpy as np

df1 = pd.DataFrame(np.random.randn(10,3),columns=['col1','col2','col3'])
df2 = pd.DataFrame(np.random.randn(7,3),columns=['col1','col2','col3'])

df1 = df1.reindex_like(df2)
print df1
```

Its output is as follows:



```
          col1        col2        col3
0    -2.467652   -1.211687   -0.391761
1    -0.287396    0.522350    0.562512
2    -0.255409   -0.483250    1.866258
3    -1.150467   -0.646493   -0.222462
4     0.152768   -2.056643    1.877233
5    -1.155997    1.528719   -1.343719
6    -1.015606   -1.245936   -0.295275
```

Note: Here, the df1 DataFrame is altered and reindexed like df2. The column names should be matched or else NAN will be added for the entire column label.

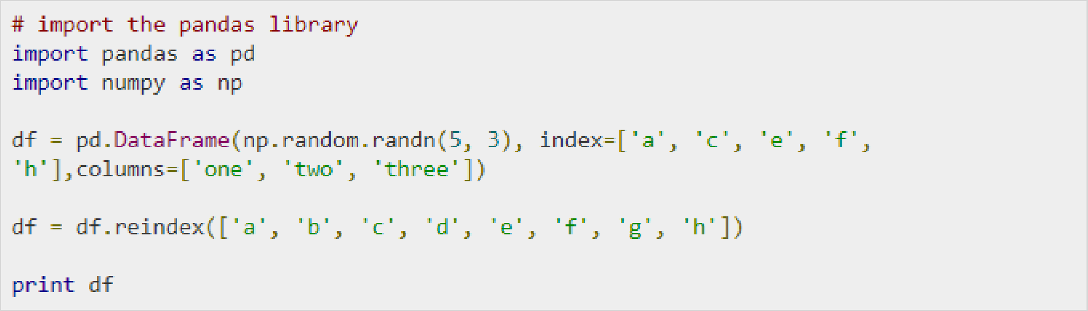
3 Missing Data

Missing data is always a problem in real life scenarios. Areas like machine learning and data mining face severe issues in the accuracy of their model predictions because of poor quality of data caused by missing values. In these areas, missing value treatment is a major point of focus to make their models more accurate and valid.

3.1 When and Why Is Data Missed?

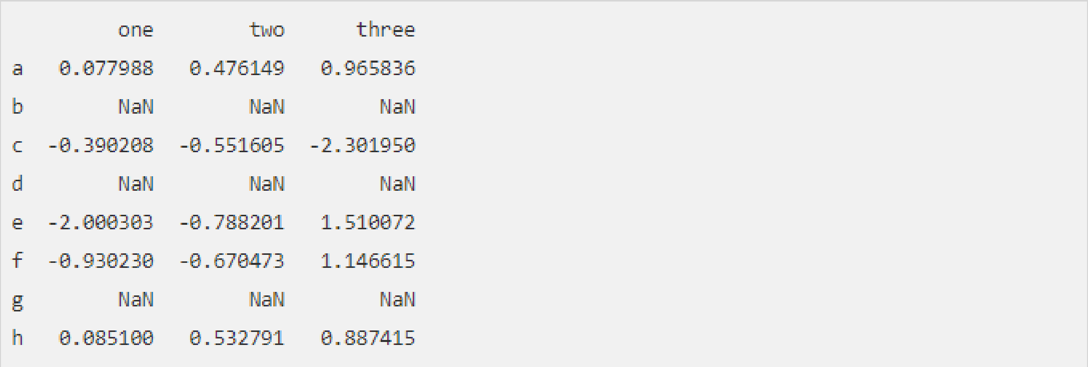
Let us consider an online survey for a product. Many a times, people do not share all the information related to them. Few people share their experience, but not how long they are using the product; few people share how long they are using the product, their experience but not their contact information. Thus, in some or the other way a part of data is always missing, and this is very common in real time.

Let us now see how we can handle missing values (say NA or NaN) using Pandas.
Example



```
# import the pandas library
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df
```

Its output is as follows:



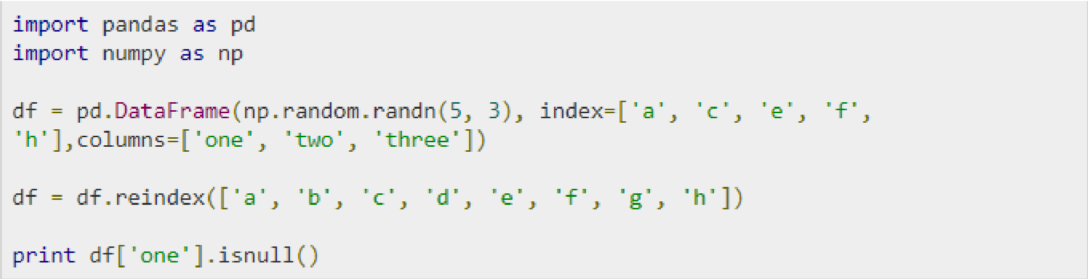
```
        one        two      three
a  0.077988   0.476149   0.965836
b       NaN        NaN        NaN
c -0.390208  -0.551605  -2.301950
d       NaN        NaN        NaN
e -2.000303  -0.788201   1.510072
f -0.930230  -0.670473   1.146615
g       NaN        NaN        NaN
h  0.085100   0.532791   0.887415
```

Using reindexing, we have created a DataFrame with missing values. In the output, NaN means Not a Number.

3.2 Check for Missing Values

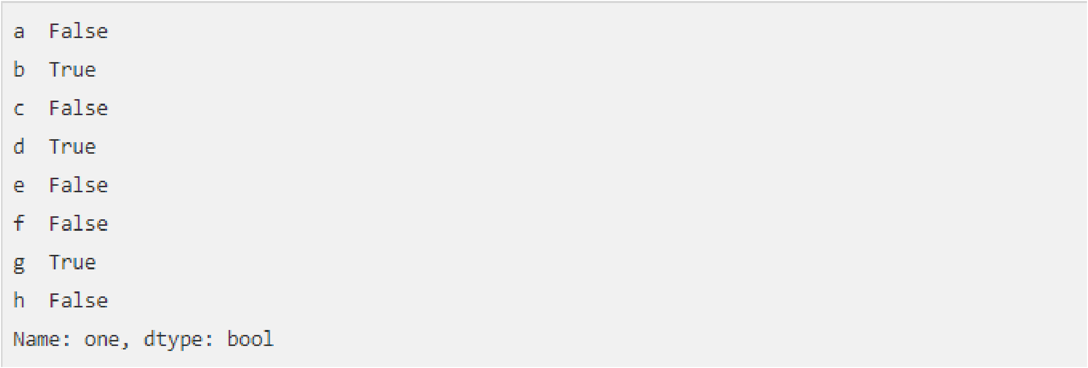
To make detecting missing values easier (and across different array dtypes), Pandas provides the isnull() and notnull() functions, which are also methods on Series and DataFrame objects.

Example 1



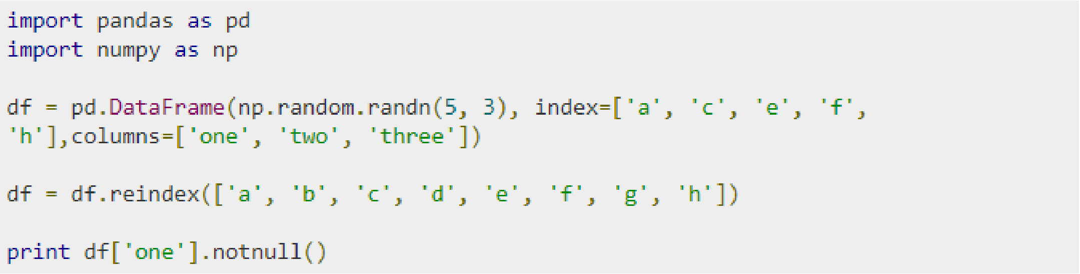
```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].isnull()
```

Its output is as follows:



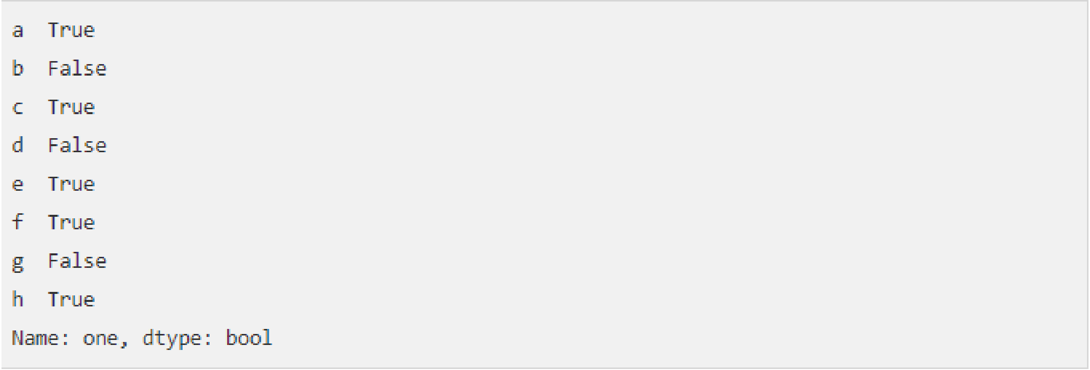
```
a  False
b  True
c  False
d  True
e  False
f  False
g  True
h  False
Name: one, dtype: bool
```

Example 2



```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].notnull()
```

Its output is as follows:

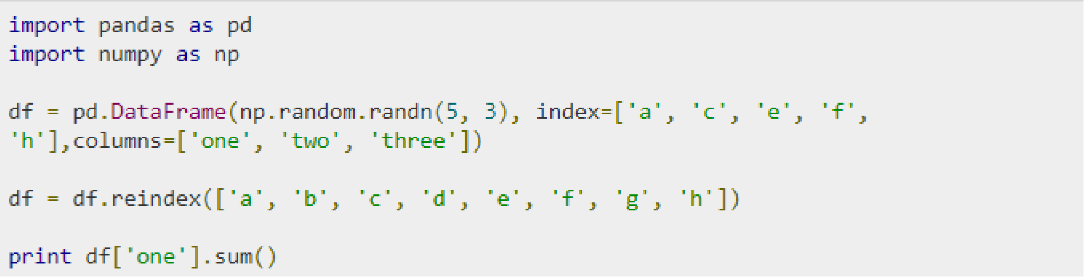


```
a  True
b  False
c  True
d  False
e  True
f  True
g  False
h  True
Name: one, dtype: bool
```

3.3 Calculations with Missing Data

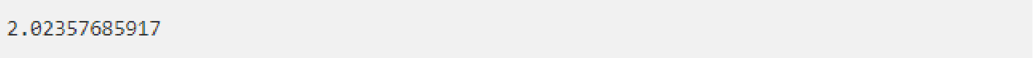
- When summing data, NA will be treated as Zero
- If the data are all NA, then the result will be NA

Example 1



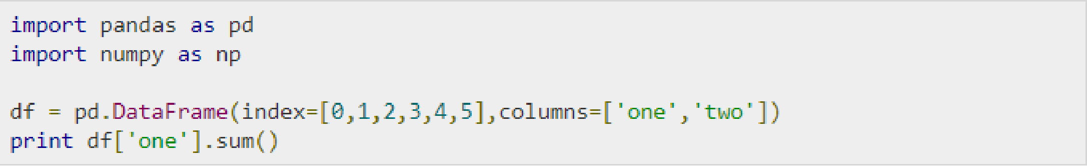
```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].sum()
```

Its output is as follows:



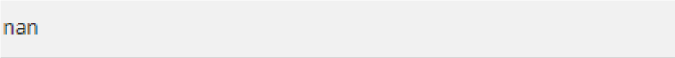
```
2.02357685917
```

Example 2



```
import pandas as pd
import numpy as np

df = pd.DataFrame(index=[0,1,2,3,4,5],columns=['one','two'])
print df['one'].sum()
```

Its output is as follows:



```
nan
```

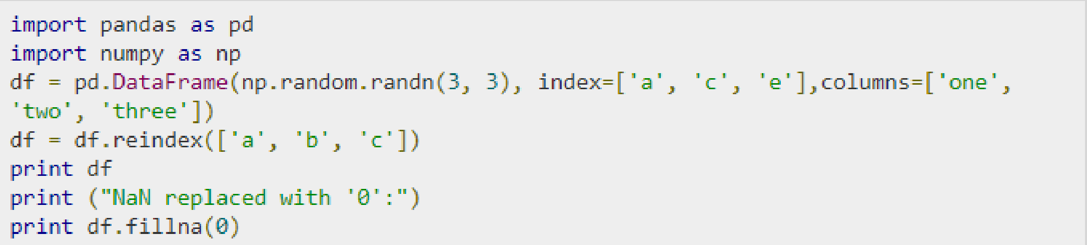
3.4 Cleaning / Filling Missing Data

Pandas provides various methods for cleaning the missing values. The fillna function can ”fill in” NA values with non-null data in a couple of ways, which we have illustrated in the following sections.

3.4.1 Replace NaN with a Scalar Value

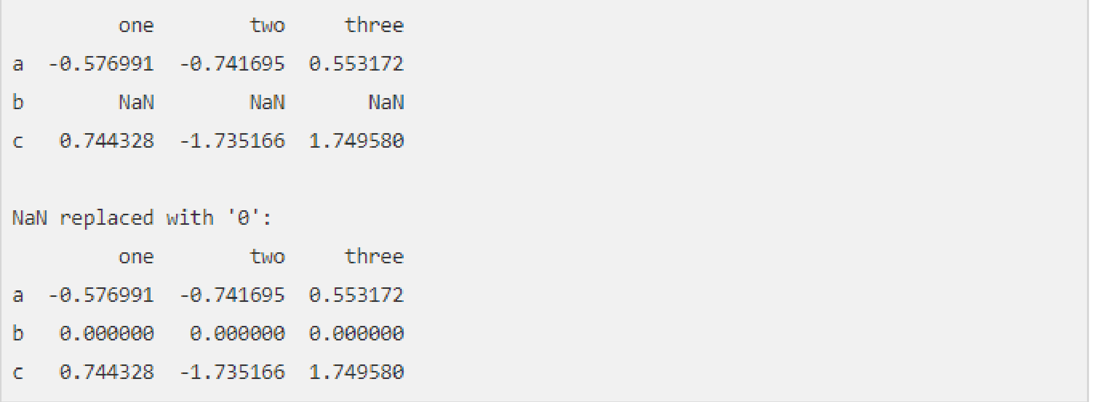
The following program shows how you can replace ”NaN” with ”0”.

Example



```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',
'two', 'three'])
df = df.reindex(['a', 'b', 'c'])
print df
print ("NaN replaced with '0':")
print df.fillna(0)
```

Its output is as follows:



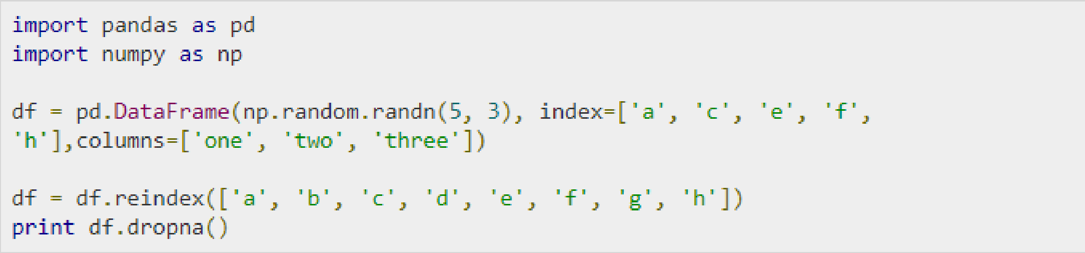
```
        one       two     three
a  -0.576991  -0.741695  0.553172
b        NaN        NaN       NaN
c   0.744328  -1.735166  1.749580

NaN replaced with '0':
        one       two     three
a  -0.576991  -0.741695  0.553172
b   0.000000   0.000000  0.000000
c   0.744328  -1.735166  1.749580
```

Here, we are filling with value zero; instead we can also fill with any other value.

3.5 Drop Missing Values

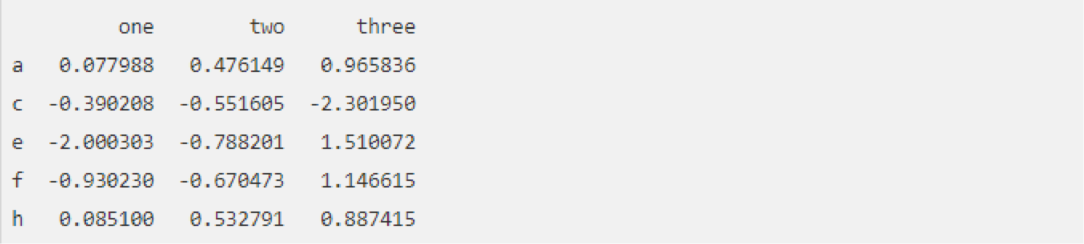
If you want to simply exclude the missing values, then use the dropna function along with the axis argument. By default, axis=0, i.e., along row, which means that if any value within a row is NA then the whole row is excluded.

Example 1



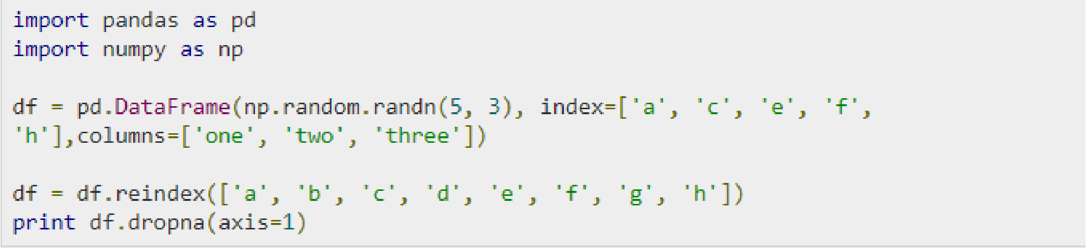
```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.dropna()
```

Its output is as follows:



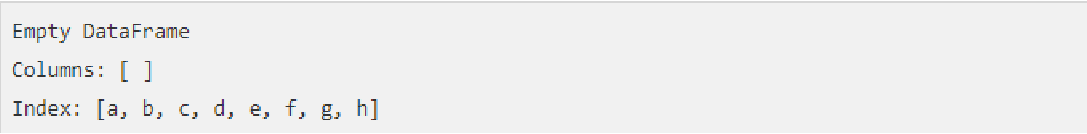
```
        one       two     three
a   0.077988   0.476149   0.965836
c  -0.390208  -0.551605  -2.301950
e  -2.000303  -0.788201   1.510072
f  -0.930230  -0.670473   1.146615
h   0.085100   0.532791   0.887415
```

Example 2



```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.dropna(axis=1)
```

Its output is as follows:



```
Empty DataFrame
Columns: [ ]
Index: [a, b, c, d, e, f, g, h]
```