```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
column=['a','b','c','d','e']
index=['A','B','C','D','E']

# create a dataframe of random values of array
df1 = pd.DataFrame(np.random.rand(5,5),
            columns=column, index=index)
print(df1)
```

```
          a         b         c         d         e
A  0.630708  0.690999  0.961456  0.221983  0.347022
B  0.796255  0.644895  0.125046  0.950913  0.068460
C  0.937727  0.921140  0.824844  0.101292  0.421676
D  0.983416  0.913363  0.871646  0.874871  0.207836
E  0.418914  0.145288  0.083024  0.015487  0.606249
```

Reindexing

dataframe.reindex(keys, method, fill_value, limit)

key is mandatory: String or list containing row indexes or column labels

others are optional

```
print('\n\nDataframe after reindexing rows: \n',
df1.reindex(['B', 'D', 'A', 'C', 'E']))
```

```


Dataframe after reindexing rows: 
           a         b         c         d         e
B  0.796255  0.644895  0.125046  0.950913  0.068460
D  0.983416  0.913363  0.871646  0.874871  0.207836
A  0.630708  0.690999  0.961456  0.221983  0.347022
C  0.937727  0.921140  0.824844  0.101292  0.421676
E  0.418914  0.145288  0.083024  0.015487  0.606249
```

```
# create the new index for rows
new_index =['U', 'A', 'B', 'C', 'Z']
print(df1.reindex(new_index))
```

```
          a         b         c         d         e
U       NaN       NaN       NaN       NaN       NaN
A  0.630708  0.690999  0.961456  0.221983  0.347022
B  0.796255  0.644895  0.125046  0.950913  0.068460
C  0.937727  0.921140  0.824844  0.101292  0.421676
Z       NaN       NaN       NaN       NaN       NaN
```

```
print(df1.reindex(new_index,axis=0))
```

```
          a         b         c         d         e
U       NaN       NaN       NaN       NaN       NaN
A  0.630708  0.690999  0.961456  0.221983  0.347022
B  0.796255  0.644895  0.125046  0.950913  0.068460
C  0.937727  0.921140  0.824844  0.101292  0.421676
Z       NaN       NaN       NaN       NaN       NaN
```

```
print(df1.reindex(new_index,axis='rows'))
```

```
          a         b         c         d         e
U       NaN       NaN       NaN       NaN       NaN
A  0.630708  0.690999  0.961456  0.221983  0.347022
B  0.796255  0.644895  0.125046  0.950913  0.068460
C  0.937727  0.921140  0.824844  0.101292  0.421676
Z       NaN       NaN       NaN       NaN       NaN
```

```
column=['e','a','b','c','d']
# create the new index for columns
print(df1.reindex(column, axis='columns'))
```

```
          e         a         b         c         d
A  0.347022  0.630708  0.690999  0.961456  0.221983
B  0.068460  0.796255  0.644895  0.125046  0.950913
C  0.421676  0.937727  0.921140  0.824844  0.101292
D  0.207836  0.983416  0.913363  0.871646  0.874871
E  0.606249  0.418914  0.145288  0.083024  0.015487
```

```
column =['a', 'b', 'c', 'g', 'h']
# create the new index for columns
print(df1.reindex(column, axis ='columns'))
```

```
          a         b         c   g   h
A  0.630708  0.690999  0.961456 NaN NaN
B  0.796255  0.644895  0.125046 NaN NaN
C  0.937727  0.921140  0.824844 NaN NaN
D  0.983416  0.913363  0.871646 NaN NaN
E  0.418914  0.145288  0.083024 NaN NaN
```

```
column =['a', 'b', 'c', 'g', 'h']
# create the new index for columns
print(df1.reindex(column, axis ='columns', fill_value = 1.5))
```

```
          a         b         c    g    h
A  0.630708  0.690999  0.961456  1.5  1.5
B  0.796255  0.644895  0.125046  1.5  1.5
C  0.937727  0.921140  0.824844  1.5  1.5
D  0.983416  0.913363  0.871646  1.5  1.5
E  0.418914  0.145288  0.083024  1.5  1.5
```

```
column =['a', 'b', 'c', 'g', 'h']
# create the new index for columns
print(df1.reindex(column, axis ='columns', fill_value ='data missing'))
```

```
          a         b         c             g             h
A  0.630708  0.690999  0.961456  data missing  data missing
B  0.796255  0.644895  0.125046  data missing  data missing
C  0.937727  0.921140  0.824844  data missing  data missing
D  0.983416  0.913363  0.871646  data missing  data missing
E  0.418914  0.145288  0.083024  data missing  data missing
```

method: None, ‘backfill’/’bfill’, ‘pad’/’ffill’, ‘nearest’

Method to use for filling holes in reindexed DataFrame.

Only applicable to DataFrames/Series with a **monotonically increasing/decreasing index.**

- None (default): don’t fill gaps
- pad / ffill: Propagate last valid observation forward to next valid.
- backfill / bfill: Use next valid observation to fill gap.
- nearest: Use nearest valid observations to fill gap.

```
a={'name':['ramesh','suresh'],'age':[0,1]}
dfq=pd.DataFrame(a)
print(dfq)
```

```
     name  age
0  ramesh    0
1  suresh    1
```

```
dfq.reindex(['name','age','mobno'],axis=1)
```

	name	**age**	**mobno**
0	ramesh	0	NaN
1	suresh	1	NaN

```
ss=dfq.reindex([0,1,2],axis=0,method='nearest')
print(ss)
```

```
     name  age
0  ramesh    0
1  suresh    1
2  suresh    1
```

```
ss=dfq.reindex([0,1,2],axis=0,method='ffill')
print(ss)
```

```
     name  age
0  ramesh    0
1  suresh    1
2  suresh    1
```

```
ss=dfq.reindex([0,1,2],axis=0,method='bfill')
print(ss)
```

```
     name  age
0  ramesh  0.0
1  suresh  1.0
2     NaN  NaN
```

```
print(dfq)
```

```
     name  age
0  ramesh    0
1  suresh    1
```

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

print(df1)
```

```
       col1      col2      col3
0  0.263229 -0.053976  0.003595
1 -0.721784 -0.052665 -1.197194
2  0.715990  0.040279  0.132420
3  1.138192 -0.755675  0.466553
4 -1.537607  0.955755  0.937493
5 -1.075502 -0.120526 -0.046793
6 -1.100591  0.690980 -0.108555
7 -0.804782 -0.629512  0.433028
8 -0.678589 -1.699126 -0.029498
9 -0.707616  0.525920 -0.959048
```

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

```
       col1      col2      col3
0 -0.103445  1.013431  1.806733
1  0.328299  0.136150 -0.467214
2 -0.083452 -0.560955 -1.098263
3 -0.300195  1.009559  1.192445
4 -2.422225  0.796601  1.451036
5 -0.273899  0.436058 -1.647098
6  0.390150 -1.402130 -0.811021
```

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

```
       col1      col2      col3
0  0.263229 -0.053976  0.003595
1 -0.721784 -0.052665 -1.197194
2  0.715990  0.040279  0.132420
3  1.138192 -0.755675  0.466553
4 -1.537607  0.955755  0.937493
5 -1.075502 -0.120526 -0.046793
6 -1.100591  0.690980 -0.108555
```

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

# Padding NAN's
print (df2.reindex_like(df1))

# Now Fill the NAN's with preceding Values
print ("Data Frame with Forward Fill:")
ssd=df2.reindex_like(df1,method='ffill')
print(ssd)
```

```
       col1      col2      col3
0  0.432344  1.726540 -0.937331
1 -1.182160  0.020489  1.318318
2 -0.978342 -0.423633  0.263409
3 -0.012252  0.061163 -0.024201
4 -0.886422 -1.954940 -0.090159
5 -1.653464 -0.006140 -1.877122
6  1.301150  0.307100 -0.206475
7       NaN       NaN       NaN
8       NaN       NaN       NaN
9       NaN       NaN       NaN
Data Frame with Forward Fill:
       col1      col2      col3
0  0.432344  1.726540 -0.937331
1 -1.182160  0.020489  1.318318
2 -0.978342 -0.423633  0.263409
3 -0.012252  0.061163 -0.024201
4 -0.886422 -1.954940 -0.090159
5 -1.653464 -0.006140 -1.877122
6  1.301150  0.307100 -0.206475
7  1.301150  0.307100 -0.206475
8  1.301150  0.307100 -0.206475
9  1.301150  0.307100 -0.206475
```

```
date_index = pd.date_range('1/1/2010', periods=6, freq='D')
df2 = pd.DataFrame({"prices": [100, 101, np.nan, 100, 89, 88]},
                   index=date_index)
df2
```

	prices
2010-01-01	100.0
2010-01-02	101.0
2010-01-03	NaN
2010-01-04	100.0
2010-01-05	89.0
2010-01-06	88.0

```
date_index2 = pd.date_range('12/29/2009', periods=10, freq='D')
df2.reindex(date_index2)
```

	prices
2009-12-29	NaN
2009-12-30	NaN
2009-12-31	NaN
2010-01-01	100.0
2010-01-02	101.0
2010-01-03	NaN
2010-01-04	100.0
2010-01-05	89.0
2010-01-06	88.0
2010-01-07	NaN

```
df2.reindex(date_index2, method='bfill')
```

	prices
2009-12-29	100.0
2009-12-30	100.0
2009-12-31	100.0
2010-01-01	100.0
2010-01-02	101.0
2010-01-03	NaN
2010-01-04	100.0
2010-01-05	89.0
2010-01-06	88.0
2010-01-07	NaN

```
df2.reindex(date_index2, method='ffill')
```

	prices
2009-12-29	NaN
2009-12-30	NaN
2009-12-31	NaN
2010-01-01	100.0

Check for Missing Values

2010-01-03 NaN

```
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f','h'],columns
print(df)
print('\n')
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print(df)
```

```
        one       two     three
a  0.178669 -0.022223  1.134353
c -0.218136  1.119158 -0.542021
e  0.607355 -0.926195  1.212720
f  1.239645  0.557774 -0.325448
h -0.608266  0.124757  2.749352


        one       two     three
a  0.178669 -0.022223  1.134353
b       NaN       NaN       NaN
c -0.218136  1.119158 -0.542021
d       NaN       NaN       NaN
e  0.607355 -0.926195  1.212720
f  1.239645  0.557774 -0.325448
g       NaN       NaN       NaN
h -0.608266  0.124757  2.749352
```

```
df.isnull()
```

	one	two	three
a	False	False	False
b	True	True	True
c	False	False	False
d	True	True	True
e	False	False	False
f	False	False	False
g	True	True	True
h	False	False	False

```
df.isna()
```

	one	two	three
a	False	False	False
b	True	True	True
c	False	False	False
d	True	True	True
e	False	False	False
f	False	False	False
g	True	True	True
h	False	False	False

```
df.isna().any()#give result columnwise
```

```
one      True
two      True
three    True
dtype: bool
```

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

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

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

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

```
print (df.fillna(0))
```

```
        one       two     three
a  0.178669 -0.022223  1.134353
b  0.000000  0.000000  0.000000
c -0.218136  1.119158 -0.542021
```

```
d  0.000000  0.000000  0.000000
e  0.607355 -0.926195  1.212720
f  1.239645  0.557774 -0.325448
g  0.000000  0.000000  0.000000
h -0.608266  0.124757  2.749352
```

```
print (df.fillna(method='pad'))
```

```
        one       two     three
a  0.178669 -0.022223  1.134353
b  0.178669 -0.022223  1.134353
c -0.218136  1.119158 -0.542021
d -0.218136  1.119158 -0.542021
e  0.607355 -0.926195  1.212720
f  1.239645  0.557774 -0.325448
g  1.239645  0.557774 -0.325448
h -0.608266  0.124757  2.749352
```

```
a= df.fillna(method='backfill')
print(a)
```

```
        one       two     three
a  0.178669 -0.022223  1.134353
b -0.218136  1.119158 -0.542021
c -0.218136  1.119158 -0.542021
d  0.607355 -0.926195  1.212720
e  0.607355 -0.926195  1.212720
f  1.239645  0.557774 -0.325448
g -0.608266  0.124757  2.749352
h -0.608266  0.124757  2.749352
```

Drop Missing Values

By default, axis=0: row

```
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)
print('\n after droping missing value')
print (df.dropna())
```

```
        one       two     three
a  0.596687 -0.237725 -2.893086
b       NaN       NaN       NaN
c  1.036576  0.047866 -1.560327
d       NaN       NaN       NaN
e -0.183099 -0.879034  1.447953
f  0.585059  0.522077  1.169595
g       NaN       NaN       NaN
h  0.146995 -1.385961  0.452158

 after droping missing value
        one       two     three
```

```
a  0.596687 -0.237725 -2.893086
c  1.036576  0.047866 -1.560327
e -0.183099 -0.879034  1.447953
f  0.585059  0.522077  1.169595
h  0.146995 -1.385961  0.452158
```

```
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))
```

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

```
df = pd.DataFrame({'one':[10,20,30,40,50,2000], 'two':[1000,0,30,40,50,60]})
print(df)
print (df.replace({1000:10,2000:60}))
```

```
    one   two
0    10  1000
1    20     0
2    30    30
3    40    40
4    50    50
5  2000    60
   one  two
0   10   10
1   20    0
2   30   30
3   40   40
4   50   50
5   60   60
```

Group

```
ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
   'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
   'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
   'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
   'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
df = pd.DataFrame(ipl_data)

print (df)
df.plot()
```

```
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
<matplotlib.axes._subplots.AxesSubplot at 0x7f80bad2c090>
```

2000
1750
1500
1250
1000
Rank
Year

```
1 print (df.groupby('Team'))
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f80bab8b7d0>
```

```
1 print (df.groupby('Team').groups)
```

```
{'Devils': [2, 3], 'Kings': [4, 6, 7], 'Riders': [0, 1, 8, 11], 'Royals': [9,
```

Colab paid products - Cancel contracts here

✓ 0s completed at 8:26 AM