In [80]:
```
import numpy as np
import pandas as pd
```

In [81]:
```
s = [1,2,3,np.nan,5]
labels  = ['A','B','C','D','E']
pd.Series(data =s,index = labels)
```

Out[81]:
```
A    1.0
B    2.0
C    3.0
D    NaN
E    5.0
dtype: float64
```

In [82]:
```
dates = pd.date_range('20190602',periods = 7)
dates
```

Out[82]:
```
DatetimeIndex(['2019-06-02', '2019-06-03', '2019-06-04', '2019-06-05',
               '2019-06-06', '2019-06-07', '2019-06-08'],
              dtype='datetime64[ns]', freq='D')
```

In [83]:
```
df = pd.DataFrame(np.random.randn(7, 4), index=dates, columns=list('ABCD
'))
df
```

Out[83]:

	A	B	C	D
2019-06-02	0.374469	-0.250649	0.347215	-0.147779
2019-06-03	0.352550	0.391218	0.394606	-2.263850
2019-06-04	-0.100087	0.342009	-0.035109	-0.423535
2019-06-05	-0.604192	0.593251	-0.145405	0.786953
2019-06-06	0.109530	-0.085287	0.360249	1.063530
2019-06-07	-0.462695	0.622212	-1.562366	-1.022399
2019-06-08	-0.305086	0.723377	0.236061	-0.723989

In [84]:
```
df2 = pd.DataFrame({'A': 1.,
                    'B': pd.Timestamp('20130102'),
                    'C': pd.Series(1, index=list(range(4)), dtype='float
32'),
                    'D': np.array([3] * 4, dtype='int32'),
                    'E': pd.Categorical(["test", "train", "test", "trai
n"]),
                    'F': 'foo'})
```

In [85]:
```
df2
```

Out[85]:

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

In [86]:
```
#Types of the data type used in the DataFrame
df2.dtypes
```

Out[86]:
```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

In [87]:
```
#Viewing data for top use Head for bottom use Tail
df2.head(2)
```

Out[87]:

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo

In [88]:
```
df2.tail(2)
```

Out[88]:

	A	B	C	D	E	F
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

In [89]:
```
#To check quick summary
df.describe()
```

Out[89]:

	A	B	C	D
count	7.000000	7.000000	7.000000	7.000000
mean	-0.090787	0.333733	-0.057821	-0.390153
std	0.387524	0.370325	0.695430	1.124022
min	-0.604192	-0.250649	-1.562366	-2.263850
25%	-0.383891	0.128361	-0.090257	-0.873194
50%	-0.100087	0.391218	0.236061	-0.423535
75%	0.231040	0.607731	0.353732	0.319587
max	0.374469	0.723377	0.394606	1.063530

In [90]:
```
#Transpose of Data
df.T
```

Out[90]:

	2019-06-02 00:00:00	2019-06-03 00:00:00	2019-06-04 00:00:00	2019-06-05 00:00:00	2019-06-06 00:00:00	2019-06-07 00:00:00	2019-06-08 00:00:00
A	0.374469	0.352550	-0.100087	-0.604192	0.109530	-0.462695	-0.305086
B	-0.250649	0.391218	0.342009	0.593251	-0.085287	0.622212	0.723377
C	0.347215	0.394606	-0.035109	-0.145405	0.360249	-1.562366	0.236061
D	-0.147779	-2.263850	-0.423535	0.786953	1.063530	-1.022399	-0.723989

In [91]:
```
#sorting by axis
df.sort_index(axis=1,ascending=False)
```

Out[91]:

	D	C	B	A
2019-06-02	-0.147779	0.347215	-0.250649	0.374469
2019-06-03	-2.263850	0.394606	0.391218	0.352550
2019-06-04	-0.423535	-0.035109	0.342009	-0.100087
2019-06-05	0.786953	-0.145405	0.593251	-0.604192
2019-06-06	1.063530	0.360249	-0.085287	0.109530
2019-06-07	-1.022399	-1.562366	0.622212	-0.462695
2019-06-08	-0.723989	0.236061	0.723377	-0.305086

In [92]:
```
#sorting by axis
df.sort_index(axis=0,ascending=False)
```

Out[92]:

	A	B	C	D
2019-06-08	-0.305086	0.723377	0.236061	-0.723989
2019-06-07	-0.462695	0.622212	-1.562366	-1.022399
2019-06-06	0.109530	-0.085287	0.360249	1.063530
2019-06-05	-0.604192	0.593251	-0.145405	0.786953
2019-06-04	-0.100087	0.342009	-0.035109	-0.423535
2019-06-03	0.352550	0.391218	0.394606	-2.263850
2019-06-02	0.374469	-0.250649	0.347215	-0.147779

In [93]:
```
#sorting by values
df.sort_values(by="B",ascending=False)
```

Out[93]:

	A	B	C	D
2019-06-08	-0.305086	0.723377	0.236061	-0.723989
2019-06-07	-0.462695	0.622212	-1.562366	-1.022399
2019-06-05	-0.604192	0.593251	-0.145405	0.786953
2019-06-03	0.352550	0.391218	0.394606	-2.263850
2019-06-04	-0.100087	0.342009	-0.035109	-0.423535
2019-06-06	0.109530	-0.085287	0.360249	1.063530
2019-06-02	0.374469	-0.250649	0.347215	-0.147779

In [94]:
```
#Selection by [], which slice the row
df['A']
```

Out[94]:
```
2019-06-02    0.374469
2019-06-03    0.352550
2019-06-04   -0.100087
2019-06-05   -0.604192
2019-06-06    0.109530
2019-06-07   -0.462695
2019-06-08   -0.305086
Freq: D, Name: A, dtype: float64
```

In [95]:
```
df[0:3]
```

Out[95]:

	A	B	C	D
2019-06-02	0.374469	-0.250649	0.347215	-0.147779
2019-06-03	0.352550	0.391218	0.394606	-2.263850
2019-06-04	-0.100087	0.342009	-0.035109	-0.423535

In [96]:
```
#for getting cross section using the label
df.loc[dates[0]]
```

Out[96]:
```
A    0.374469
B   -0.250649
C    0.347215
D   -0.147779
Name: 2019-06-02 00:00:00, dtype: float64
```

In [97]:
```
#selecting multi-axis by label
df.loc[:,['A','B']]
```

Out[97]:

	A	B
2019-06-02	0.374469	-0.250649
2019-06-03	0.352550	0.391218
2019-06-04	-0.100087	0.342009
2019-06-05	-0.604192	0.593251
2019-06-06	0.109530	-0.085287
2019-06-07	-0.462695	0.622212
2019-06-08	-0.305086	0.723377

In [98]:
```
#for getting fast access to scaler
df.at[dates[0],'A']
```

Out[98]:
```
0.3744688036413159
```

In [99]:
```
#import weather dataset
data = pd.read_csv("weather_data.csv")
data.head()
```

Out[99]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain

In [100]:
```
data.sort_values(by = 'temperature', ascending= False)
```

Out[100]:

	day	temperature	windspeed	event
8	1/11/2017	40.0	12.0	Sunny
7	1/10/2017	34.0	8.0	Cloudy
0	1/1/2017	32.0	6.0	Rain
4	1/7/2017	32.0	NaN	Rain
2	1/5/2017	28.0	NaN	Snow
1	1/4/2017	NaN	9.0	Sunny
3	1/6/2017	NaN	7.0	NaN
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN

11th Jan was the hottest day of this data

In [101]:
```
# Fill the NaN data with the mean values
data.fillna(data.mean()['temperature':'windspeed'])
```

Out[101]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	33.2	9.0	Sunny
2	1/5/2017	28.0	8.4	Snow
3	1/6/2017	33.2	7.0	NaN
4	1/7/2017	32.0	8.4	Rain
5	1/8/2017	33.2	8.4	Sunny
6	1/9/2017	33.2	8.4	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [112]:
```
#Second Method
data1=data.where(pd.notna(data), data.mean(), axis='columns')
data1
```

Out[112]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	33.2	9.0	Sunny
2	1/5/2017	28.0	8.4	Snow
3	1/6/2017	33.2	7.0	NaN
4	1/7/2017	32.0	8.4	Rain
5	1/8/2017	33.2	8.4	Sunny
6	1/9/2017	33.2	8.4	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [110]:
```
#We can fill the missing value using the interpolation
data2=data.interpolate()
```

In [111]:
```
data2
```

Out[111]:

	day	temperature	windspeed	event
0	1/1/2017	32.000000	6.00	Rain
1	1/4/2017	30.000000	9.00	Sunny
2	1/5/2017	28.000000	8.00	Snow
3	1/6/2017	30.000000	7.00	NaN
4	1/7/2017	32.000000	7.25	Rain
5	1/8/2017	32.666667	7.50	Sunny
6	1/9/2017	33.333333	7.75	NaN
7	1/10/2017	34.000000	8.00	Cloudy
8	1/11/2017	40.000000	12.00	Sunny

In [128]:
```
data2.fillna(value='Cloudy')
```

Out[128]:

	day	temperature	windspeed	event
0	1/1/2017	32.000000	6.00	Rain
1	1/4/2017	30.000000	9.00	Sunny
2	1/5/2017	28.000000	8.00	Snow
3	1/6/2017	30.000000	7.00	Cloudy
4	1/7/2017	32.000000	7.25	Rain
5	1/8/2017	32.666667	7.50	Sunny
6	1/9/2017	33.333333	7.75	Cloudy
7	1/10/2017	34.000000	8.00	Cloudy
8	1/11/2017	40.000000	12.00	Sunny