pandas

> pandas可以出全新的数据格式（以一种列表的形式）

1. Series构建对象

```
s = pandas.Series([1, 2, 3, 'oj', 'oio', 8, 90.23, 12])
```

output

```
0        1
1        2
2        3
3       oj
4      oio
5        8
6    90.23
7       12
dtype: object
```

2. 创建日期

```
datas = pandas.date_range('20200226', periods=10)
print(datas)
```

output

```
DatetimeIndex(['2020-02-26', '2020-02-27', '2020-02-28', '2020-02-29',
               '2020-03-01', '2020-03-02', '2020-03-03', '2020-03-04',
               '2020-03-05', '2020-03-06'],
              dtype='datetime64[ns]', freq='D')
```

3.DataFrame构建对象

```
df = pandas.DataFrame(numpy.random.randn(4, 3), index=datas,
columns=list('127'))
print(df)
```

output

```
                   1         2         7
2020-02-26 -0.689954  1.068120  0.765515
2020-02-27 -0.591247  1.206699 -1.265164
2020-02-28 -1.500227 -0.564612 -2.410998
2020-02-29 -0.294010 -0.473683  0.638123
```

4.DataFrame构建对象(字典作为参数)

```
df2 = pandas.DataFrame(
    {
        'A': 1,
        'B': pandas.Timestamp('20200226'),
        'C': pandas.Series(1, index=list(range(4)), dtype=numpy.float32),
        'D': numpy.array([3] * 4, dtype=numpy.int),
        'E': pandas.Categorical(['test', 'train', 'test', 'train']),
        'F': 'foo',
    }
)
print(df2)
```

output

```
   A          B    C  D      E    F
0  1 2020-02-26  1.0  3   test  foo
1  1 2020-02-26  1.0  3  train  foo
2  1 2020-02-26  1.0  3   test  foo
3  1 2020-02-26  1.0  3  train  foo
```

5. 头部开始切片

```
print(df2.head(2))
```

output

```
   A          B    C  D      E    F
0  1 2020-02-26  1.0  3   test  foo
1  1 2020-02-26  1.0  3  train  foo
```

6.尾部开始切片

```
print(df2.tail(2))
```

output

```
   A          B    C  D      E    F
2  1 2020-02-26  1.0  3   test  foo
3  1 2020-02-26  1.0  3  train  foo
```

7. 获取索引

```
print(df2.index)
```

output

```
Int64Index([0, 1, 2, 3], dtype='int64')
```

8. 获取列名

```
print(df2.columns)
```

output

```
Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')
```

9. 转化为numpy

```
print(df2.to_numpy())
```

output

```
[[1 Timestamp('2020-02-26 00:00:00') 1.0 3 'test' 'foo']
 [1 Timestamp('2020-02-26 00:00:00') 1.0 3 'train' 'foo']
 [1 Timestamp('2020-02-26 00:00:00') 1.0 3 'test' 'foo']
 [1 Timestamp('2020-02-26 00:00:00') 1.0 3 'train' 'foo']]
```

10.矩阵转制

```
print(df2)
print(df2.T)
```

output

```
   A          B    C  D      E    F
0  1 2020-02-26  1.0  3   test  foo
1  1 2020-02-26  1.0  3  train  foo
2  1 2020-02-26  1.0  3   test  foo
3  1 2020-02-26  1.0  3  train  foo
                     0  ...                   3
A                    1  ...                   1
B  2020-02-26 00:00:00  ...  2020-02-26 00:00:00
C                    1  ...                   1
D                    3  ...                   3
E                 test  ...               train
F                  foo  ...                 foo

[6 rows x 4 columns]
```

11.输出数据统计摘要

```
print(df2.describe())
```

output

```
         A    C    D
count  4.0  4.0  4.0
mean   1.0  1.0  3.0
std    0.0  0.0  0.0
min    1.0  1.0  3.0
25%    1.0  1.0  3.0
50%    1.0  1.0  3.0
75%    1.0  1.0  3.0
max    1.0  1.0  3.0
```

12.按轴排序

> 列名

```
print(df)

print(df.sort_index(axis=1, ascending=False))
```

output

```
                   1         2         7
2020-02-26 -0.689954  1.068120  0.765515
2020-02-27 -0.591247  1.206699 -1.265164
2020-02-28 -1.500227 -0.564612 -2.410998
2020-02-29 -0.294010 -0.473683  0.638123
                   7         2         1
2020-02-26  0.765515  1.068120 -0.689954
2020-02-27 -1.265164  1.206699 -0.591247
2020-02-28 -2.410998 -0.564612 -1.500227
2020-02-29  0.638123 -0.473683 -0.294010
```

13. 按值排序

```
print(df)
print(df.sort_values(by='7'))
```

output

```
                   1         2         7
2020-02-26 -0.689954  1.068120  0.765515
2020-02-27 -0.591247  1.206699 -1.265164
2020-02-28 -1.500227 -0.564612 -2.410998
2020-02-29 -0.294010 -0.473683  0.638123
                   1         2         7
2020-02-28 -1.500227 -0.564612 -2.410998
2020-02-27 -0.591247  1.206699 -1.265164
2020-02-29 -0.294010 -0.473683  0.638123
2020-02-26 -0.689954  1.068120  0.765515
```

14. 数据获取

```
print(df['1'])
```

output

```
2020-02-26   -1.374345
2020-02-27   -0.106331
2020-02-28   -0.028206
2020-02-29   -0.878441
Freq: D, Name: 1, dtype: float64
```

15. 行切片

```
print(df)
# 0,1,2,3
# [1:3)
print(df[1:3])
```

output

```
                   1         2         7
2020-02-26 -0.101248 -1.656236 -0.155742
2020-02-27 -0.167526 -1.487569 -2.041528
2020-02-28  0.911294  1.415461 -0.826006
2020-02-29 -1.541064 -0.448371 -0.412446
                   1         2         7
2020-02-27 -0.167526 -1.487569 -2.041528
2020-02-28  0.911294  1.415461 -0.826006
```

16.行切片的另一种方式

```
print(df)
print(df['2020-02-27':'2020-02-28'])
```

output

```
1 2 7
2020-02-26 -0.101248 -1.656236 -0.155742
2020-02-27 -0.167526 -1.487569 -2.041528
2020-02-28 0.911294 1.415461 -0.826006
2020-02-29 -1.541064 -0.448371 -0.412446
 1 2 7
2020-02-27 -0.167526 -1.487569 -2.041528
2020-02-28 0.911294 1.415461 -0.826006
```

17. 用标签提取一行数据

```
print(df)
print(df.loc[datas[0]])
```

output

```
                   1         2         7
2020-02-26  0.595325 -1.225184 -0.929020
2020-02-27 -0.112272 -1.735965 -0.777457
2020-02-28 -1.088550  2.247177 -0.455900
2020-02-29 -1.068373 -0.363619  1.486763

1    0.595325
2   -1.225184
7   -0.929020
Name: 2020-02-26 00:00:00, dtype: float64
```

18.用标签提取一列数据

```
print(df)
print(df.loc[:, '1'])
```

output

```
                   1         2         7
2020-02-26  0.631671  0.701740 -1.351176
2020-02-27 -1.645152 -2.376652  1.636998
2020-02-28 -0.110709 -0.125051  0.288295
2020-02-29  0.094557 -1.281222 -1.313987
2020-02-26    0.631671
2020-02-27   -1.645152
2020-02-28   -0.110709
2020-02-29    0.094557
Freq: D, Name: 1, dtype: float64
```

19.用标签提取多列数据

```
print(df)
print(df.loc[:, ['1', '2']])
```

output

```
                   1         2         7
2020-02-26 -0.187202 -2.112106 -0.069009
2020-02-27  0.440817 -1.653357  0.271968
2020-02-28 -0.228939 -0.505399  0.784836
2020-02-29 -0.595871 -0.151807 -1.499617
                   1         2
2020-02-26 -0.187202 -2.112106
2020-02-27  0.440817 -1.653357
2020-02-28 -0.228939 -0.505399
2020-02-29 -0.595871 -0.151807
```

20. 通过标签切片

```
print(df)
print(df.loc['2020-02-27':'2020-02-28', ['1', '2']])
```

output

```
                   1         2         7
2020-02-26 -0.871265  0.284561  0.389396
2020-02-27 -0.410973 -1.210790  0.372238
2020-02-28 -0.400745  0.297670 -1.149716
2020-02-29  0.583281  0.110699 -0.629748
                   1        2
2020-02-27 -0.410973 -1.21079
2020-02-28 -0.400745  0.29767
```

21 返回对象将维

```
print(df)
print(df.loc['2020-02-27', ['1', '2']])
```

output

```
                   1         2         7
2020-02-26 -1.146999 -0.206506 -0.655373
2020-02-27 -0.787062  0.308879  0.024343
2020-02-28 -2.556434 -1.107197 -0.812660
2020-02-29 -0.058500  0.153493  0.632778

1   -0.787062
2    0.308879
Name: 2020-02-27 00:00:00, dtype: float64
```

22. 提取标量值I

> 行列坐标获取值（类似）

```
print(df)
print(df.loc['2020-02-27', '2'])
```

output

```
                   1         2         7
2020-02-26  2.175314  1.345178  1.091660
2020-02-27  0.742392 -0.297591 -0.733107
2020-02-28 -1.090019  0.323260 -0.803120
2020-02-29  0.503102 -0.290231  0.967283

-0.2975912923928904
```

23. 提取标量值II

```
print(df)print(df.at['2020-02-27', '2'])
```

output

```
                   1         2         7
2020-02-26  0.056902 -1.646469  3.068313
2020-02-27 -0.398578 -1.141463  0.617966
2020-02-28 -0.043816 -2.330776  0.136062
2020-02-29 -1.647084 -1.175585  0.114229
-1.1414631065251561
```

24.按位置选择数据

> 获取某一行数据

```
print(df)
print(df.iloc[2])
```

output

```
                   1         2         7
2020-02-26 -0.571376 -0.587768 -0.552440
2020-02-27  0.894636 -0.825456 -0.465294
2020-02-28  0.056505 -0.799793 -0.376725
2020-02-29 -0.982538 -0.575575 -0.034167
1    0.056505
2   -0.799793
7   -0.376725
Name: 2020-02-28 00:00:00, dtype: float64
```

25. 按位置切片

```
print(df)
print(df.iloc[[1, 2], [0, 2]])
```

output

```
                   1         2         7
2020-02-26 -0.851639  0.025686  0.875242
2020-02-27 -0.548420  0.187526 -0.539008
2020-02-28  0.180460  0.834249 -3.143706
2020-02-29  1.369735  0.843023  1.381880
                   1         7
2020-02-27 -0.54842 -0.539008
2020-02-28  0.18046 -3.143706
```

26. 获取某几行的切片操作

```
print(df)
print(df.iloc[1:3, :])
```

output

```
                   1         2         7
2020-02-26  0.605729  0.579234 -0.736682
2020-02-27 -0.946606 -2.729645 -0.396664
2020-02-28  1.793410 -1.866778  1.556704
2020-02-29 -0.886156  0.244965 -0.894693
                   1         2         7
2020-02-27 -0.946606 -2.729645 -0.396664
2020-02-28  1.793410 -1.866778  1.556704
```

27. 提取值

```
print(df)
print(df.iloc[1, 1])
```

output

```
                   1         2         7
2020-02-26 -0.373444 -0.496731  0.532798
2020-02-27  0.103117  0.905421  1.277110
2020-02-28 -1.533556 -1.272383  0.134391
2020-02-29 -0.164653 -0.767427  0.811411

0.9054212585451714
```

28. 布尔索引

```
print(df)
print(df[df['1'] > 0.0])
```

output

```
                   1         2         7
2020-02-26  0.505148 -0.305334 -0.055773
2020-02-27 -0.677972  2.079349 -0.283174
2020-02-28  0.107466  0.581837  0.813700
2020-02-29 -0.075352  0.669791  0.463767
                   1         2         7
2020-02-26  0.505148 -0.305334 -0.055773
2020-02-28  0.107466  0.581837  0.813700
```

29.获取所有达到某一个条件的值

```
print(df)
print(df[df > 0.0])
```

output

```
                   1         2         7
2020-02-26  0.382700 -0.990504  1.284774
2020-02-27  0.263575  0.954236 -0.379824
2020-02-28  1.115296 -0.336775  1.160863
2020-02-29 -1.156212 -2.096799 -0.023596
                   1         2         7
2020-02-26  0.382700       NaN  1.284774
2020-02-27  0.263575  0.954236       NaN
2020-02-28  1.115296       NaN  1.160863
2020-02-29       NaN       NaN       NaN
```

30.添加一列

```
print(df)
df['8'] = [1, 2, 3, 4]
print(df)
```

output

```
                   1         2         7
2020-02-26  1.466317 -1.232028  0.259794
2020-02-27  0.566937 -1.115348  1.663919
2020-02-28 -0.795268  2.231129 -1.311918
2020-02-29 -1.266320 -0.201466  0.796607

                   1         2         7  8
2020-02-26  1.466317 -1.232028  0.259794  1
2020-02-27  0.566937 -1.115348  1.663919  2
2020-02-28 -0.795268  2.231129 -1.311918  3
2020-02-29 -1.266320 -0.201466  0.796607  4
```

31.isin() 筛选

```
print(df)
print(df[df['8'].isin([2, 3])])
```

output

```
                   1         2         7  8
2020-02-26 -0.133331 -0.581567 -0.593335  1
2020-02-27  0.725303 -0.365224  1.607063  2
2020-02-28 -1.031799 -0.289436 -0.388104  3
2020-02-29 -0.110136  2.353057 -1.472018  4
                   1         2         7  8
2020-02-27  0.725303 -0.365224  1.607063  2
2020-02-28 -1.031799 -0.289436 -0.388104  3
```

32.按标签赋值

```
print(df)
# 按标签赋值
df.at['2020-02-27', '8'] = 22
print(df)
```

output

```
                   1         2         7  8
2020-02-26  0.141758  0.218805  0.427381  1
2020-02-27  2.321946 -0.514085 -1.253257  2
2020-02-28 -0.868238  0.437756 -0.114911  3
2020-02-29  1.469269 -0.362557  0.061980  4

                   1         2         7   8
2020-02-26  0.141758  0.218805  0.427381   1
2020-02-27  2.321946 -0.514085 -1.253257  22
2020-02-28 -0.868238  0.437756 -0.114911   3
2020-02-29  1.469269 -0.362557  0.061980   4
```

33. 按位置赋值

```
print(df)
df.iat[0, 3] = 11
print(df)
```

output

```
                   1         2         7   8
2020-02-26  0.141758  0.218805  0.427381   1
2020-02-27  2.321946 -0.514085 -1.253257  22
2020-02-28 -0.868238  0.437756 -0.114911   3
2020-02-29  1.469269 -0.362557  0.061980   4
                   1         2         7   8
2020-02-26  0.141758  0.218805  0.427381  11
2020-02-27  2.321946 -0.514085 -1.253257  22
2020-02-28 -0.868238  0.437756 -0.114911   3
2020-02-29  1.469269 -0.362557  0.061980   4
```

34. numpy 赋值

```
print(df)
# 按Numpy赋值
df.loc[:, '2'] = numpy.array([5] * len(df))
print(df)
```

output

```
                   1         2         7   8
2020-02-26  0.141758  0.218805  0.427381  11
2020-02-27  2.321946 -0.514085 -1.253257  22
2020-02-28 -0.868238  0.437756 -0.114911   3
2020-02-29  1.469269 -0.362557  0.061980   4
                   1  2         7   8
2020-02-26  0.141758  5  0.427381  11
2020-02-27  2.321946  5 -1.253257  22
2020-02-28 -0.868238  5 -0.114911   3
2020-02-29  1.469269  5  0.061980   4
```

35.where赋值

```
print(df)
# where赋值
df[df < 0] = df.abs()
print(df)
```

output

```
                   1  2         7   8
2020-02-26  0.141758  5  0.427381  11
2020-02-27  2.321946  5 -1.253257  22
2020-02-28 -0.868238  5 -0.114911   3
2020-02-29  1.469269  5  0.061980   4
                   1  2         7   8
2020-02-26  0.141758  5  0.427381  11
2020-02-27  2.321946  5  1.253257  22
2020-02-28  0.868238  5  0.114911   3
2020-02-29  1.469269  5  0.061980   4
```

36. 填充缺失值

```
print(df)
df = df.fillna(value=0.56)
print(df)
```

output

```
                   1  2   7   8
2020-02-26       NaN  5 NaN  11
2020-02-27       NaN  5 NaN  22
2020-02-28  1.333409  5 NaN   3
2020-02-29       NaN  5 NaN   4
                   1  2     7   8
2020-02-26  0.560000  5  0.56  11
2020-02-27  0.560000  5  0.56  22
2020-02-28  1.333409  5  0.56   3
2020-02-29  0.560000  5  0.56   4
```

37.统计 - I

```
print(df.mean())
```

output

```
                   1  2         7   8
2020-02-26  1.561291  5  1.326167  11
2020-02-27  1.525783  5  0.560000  22
2020-02-28  0.560000  5  0.560000   3
2020-02-29  0.560000  5  1.118203   4
1     1.051769
2     5.000000
7     0.891092
8    10.000000
```

38.统计 - II

```
print(df.mean(1))
```

output

```
                   1  2         7   8
2020-02-26  1.561291  5  1.326167  11
2020-02-27  1.525783  5  0.560000  22
2020-02-28  0.560000  5  0.560000   3
2020-02-29  0.560000  5  1.118203   4

2020-02-26    4.721865
2020-02-27    7.271446
2020-02-28    2.280000
2020-02-29    2.669551
Freq: D, dtype: float64
```

39.apply - I

> 类似lambda表达式的函数

```
print(df)
print(df.apply(numpy.cumsum))
```

output

```
                   1  2         7   8
2020-02-26  1.426124  5  1.137057  11
2020-02-27  0.560000  5  0.560000  22
2020-02-28  0.560000  5  2.375213   3
2020-02-29  1.281812  5  0.560000   4
                   1   2         7   8
2020-02-26  1.426124   5  1.137057  11
2020-02-27  1.986124  10  1.697057  33
2020-02-28  2.546124  15  4.072269  36
2020-02-29  3.827936  20  4.632269  40
```

40. apply-II

```
print(df)
print(df.apply(lambda x: x.max() - x.min()))
```

output

```
                   1  2         7   8
2020-02-26  1.426124  5  1.137057  11
2020-02-27  0.560000  5  0.560000  22
2020-02-28  0.560000  5  2.375213   3
2020-02-29  1.281812  5  0.560000   4

1     1.426124
2     5.000000
7     2.375213
8    22.000000
dtype: float64
```

41.str

```
ss = pandas.Series(['A', 'nihbi', 'ytgvhUGv', numpy.nan, 'ytuvyib', 12.90])
print(ss)
print(ss.str.lower())
```

output

```
0           A
1       nihbi
2    ytgvhUGv
3         NaN
4     ytuvyib
5        12.9
dtype: object
0           a
1       nihbi
2    ytgvhugv
3         NaN
4     ytuvyib
5         NaN
dtype: object
```

42. 按行分解

```
import pandas
import numpy

df = pandas.DataFrame(numpy.random.randn(10, 4))

print(df)

pieces = [df[:3], df[3:7], df[7:]]

for i in pieces:
    print(i)
```

output

```
          0         1         2         3
0 -2.019306  0.138412 -0.802316 -1.674366
1 -0.645525  1.435895 -2.875515 -0.578919
2  0.046742 -1.637722  0.037570  0.086485
3 -1.036658 -0.129502 -0.067573 -0.384879
4 -0.871106  0.807801  0.039728  1.803245
5  0.556266 -1.190668  0.123067 -0.428439
6  0.748610 -0.972963  0.631616  0.795010
7  1.745972 -0.179505  1.507371 -1.553459
8  0.070549 -0.172859  0.005718 -0.425660
9 -0.643688 -1.044556 -0.091081  0.866066

          0         1         2         3
0 -2.019306  0.138412 -0.802316 -1.674366
1 -0.645525  1.435895 -2.875515 -0.578919
2  0.046742 -1.637722  0.037570  0.086485

          0         1         2         3
3 -1.036658 -0.129502 -0.067573 -0.384879
4 -0.871106  0.807801  0.039728  1.803245
5  0.556266 -1.190668  0.123067 -0.428439
6  0.748610 -0.972963  0.631616  0.795010

          0         1         2         3
7  1.745972 -0.179505  1.507371 -1.553459
8  0.070549 -0.172859  0.005718 -0.425660
9 -0.643688 -1.044556 -0.091081  0.866066
```

43. 合并-I

```
df = pandas.concat(pieces)
print(df)
```

output

```
          0         1         2         3
0 -0.366790 -1.711347 -0.294656  0.426771
1 -0.460260  0.707317  0.824900 -0.207959
2  0.126244  1.749152 -1.211225  0.470419
3  0.815548  1.921412  0.946617 -0.045782
4 -0.518446  0.379627  0.118273  0.420653
5 -1.007861  0.840407  0.562251 -0.186715
6  1.433063  1.452031  0.619630 -0.338745
7 -1.644411 -0.737246 -0.863549  1.392585
8 -1.235217 -0.891394 -1.033121  0.045708
9 -0.621058  0.399198 -0.400415 -1.101105
```

44.连接（join)-I

```
right = pandas.DataFrame({
    'key': ['foo', 'foo'],
    'lval': [3, 4]
})

print(left)

print(right)

newDf = pandas.merge(left, right, on='key')
print(newDf)
```

output

```
   key  lval
0  foo     1
1  foo     2
   key  lval
0  foo     3
1  foo     4
   key  lval_x  lval_y
0  foo       1       3
1  foo       1       4
2  foo       2       3
3  foo       2       4
```

45.连接（join)-II

```
left = pandas.DataFrame({
    'key': ['foo', 'food'],
    'lval': [1, 2]
})

right = pandas.DataFrame({
    'key': ['foo', 'food'],
    'lval': [3, 4]
})

right1= pandas.DataFrame({
    'key': ['foo', 'food'],
    'lval': [5, 6]
})

print(left)

print(right)

newDf = pandas.merge(left, right, on='key')
print(newDf)

newDf = pandas.merge(newDf, right1, on='key')
print(newDf)
```

output

```
    key  lval
0   foo     1
1  food     2
    key  lval
0   foo     3
1  food     4
    key  lval_x  lval_y
0   foo       1       3
1  food       2       4
    key  lval_x  lval_y  lval
0   foo       1       3     5
1  food       2       4     6
```

46. Append

```
print(df)
s = df.iloc[3]
print(s)
df = df.append(s, ignore_index=True)
print(df)
```

output

```
          0         1         2         3
0 -0.274369  0.168474  0.805170  2.798873
1  0.016074 -0.673847  0.019831  0.747431
2 -0.378618  0.482296 -0.908284  0.018589
3  0.193489 -0.851349  0.265849 -0.377275
4 -0.057102  0.777787 -0.141706 -2.188727
5  0.083639  0.851072 -2.459198  1.276810
6  0.717780  0.359376  0.224964 -0.037942
7  2.044337 -0.098891  1.498120  1.330745
8  0.007174 -0.688353  0.404029 -1.161087
9  0.269644 -0.653162 -0.550746  0.220478
0    0.193489
1   -0.851349
2    0.265849
3   -0.377275
Name: 3, dtype: float64
           0         1         2         3
0  -0.274369  0.168474  0.805170  2.798873
1   0.016074 -0.673847  0.019831  0.747431
2  -0.378618  0.482296 -0.908284  0.018589
3   0.193489 -0.851349  0.265849 -0.377275
4  -0.057102  0.777787 -0.141706 -2.188727
5   0.083639  0.851072 -2.459198  1.276810
6   0.717780  0.359376  0.224964 -0.037942
7   2.044337 -0.098891  1.498120  1.330745
8   0.007174 -0.688353  0.404029 -1.161087
9   0.269644 -0.653162 -0.550746  0.220478
10  0.193489 -0.851349  0.265849 -0.377275
```

47. 分组

```
df = pandas.DataFrame({'A': ['foo', 'bar', 'foo', 'bar',
                             'foo', 'bar', 'foo', 'foo'],
                       'B': ['one', 'one', 'two', 'three',
                             'two', 'two', 'one', 'three'],
                        'C': numpy.random.randn(8),
                        'D': numpy.random.randn(8)})

print(df)


print(df.groupby('A').sum())

print(df.groupby(['A', 'B']).sum())
```

output

```
     A      B         C         D
0  foo    one  1.187615  1.115430
1  bar    one  2.792654  0.697883
2  foo    two -1.781985 -1.077277
3  bar  three -0.384596  0.945363
4  foo    two -1.758705  0.621603
5  bar    two -0.933051 -0.851648
6  foo    one  0.512358  0.710599
7  foo  three -0.234723 -1.471294
            C         D
A
bar  1.475007  0.791598
foo -2.075440 -0.100939
                  C         D
A   B
bar one    2.792654  0.697883
    three -0.384596  0.945363
    two   -0.933051 -0.851648
foo one    1.699973  1.826029
    three -0.234723 -1.471294
    two   -3.540690 -0.455674
```

48.把 DataFrame 列压缩至一层(stack())

```
print(df)
stacked = df.stack()
print(stacked)
```

output

```
            C         D
A
bar -0.853358 -2.056484
foo  1.481160 -2.995508

A
bar  C   -0.853358
     D   -2.056484
foo  C    1.481160
     D   -2.995508
dtype: float64
```

49.stack()逆操作

```
stacked = df.groupby('A').sum().stack()
print(stacked)
print(stacked.unstack())
```

output

```
A
bar  C   -2.180919
     D   -2.629716
foo  C    1.495516
     D   -2.061165
dtype: float64
            C         D
A
bar -2.180919 -2.629716
foo  1.495516 -2.061165
```

50.生成数据透视表

```
df = pandas.DataFrame({'A': ['one', 'one', 'two', 'three'] * 3,
 'B': ['A', 'B', 'C'] * 4,
 'C': ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] * 2,
 'D': numpy.random.randn(12),
 'E': numpy.random.randn(12)})

print(df)

newDf = pandas.pivot_table(df, values='D', index=['A', 'B'], columns=['C'])

print(newDf)
```

output

```
        A  B    C         D         E
0     one  A  foo  0.629920  1.443815
1     one  B  foo -0.195666 -0.369452
2     two  C  foo  0.619123  0.351196
3   three  A  bar  1.237938 -1.441836
4     one  B  bar  0.913854  0.223142
5     one  C  bar -0.750721 -0.174814
6     two  A  foo  1.265027 -0.675359
7   three  B  foo  0.192189  0.193948
8     one  C  foo -0.255528 -2.209672
9     one  A  bar -0.684274  1.130262
10    two  B  bar -1.842591 -0.648299
11  three  C  bar  0.446397  1.154587
C             bar       foo
A     B
one   A -0.684274  0.629920
      B  0.913854 -0.195666
      C -0.750721 -0.255528
three A  1.237938       NaN
      B       NaN  0.192189
      C  0.446397       NaN
two   A       NaN  1.265027
      B -1.842591       NaN
      C       NaN  0.619123
```

51. 时间序列

```
rng = pandas.date_range('2/27/2020 00:00', periods=5, freq='D')

print(rng)

ts = pandas.Series(numpy.random.randn(len(rng)), rng)

print(ts)
```

output

```
DatetimeIndex(['2020-02-27', '2020-02-28', '2020-02-29', '2020-03-01',
               '2020-03-02'],
              dtype='datetime64[ns]', freq='D')
2020-02-27    0.937204
2020-02-28    0.976350
2020-02-29   -0.511082
2020-03-01   -0.896374
2020-03-02    0.369202
Freq: D, dtype: float64
```

52. 时间表示形式修改

```
print(ts)
ts_utc = ts.tz_localize('UTC')
print(ts_utc)
```

output

```
2020-02-27   -0.716086
2020-02-28   -1.162248
2020-02-29    0.741673
2020-03-01   -1.203902
2020-03-02    0.981236
Freq: D, dtype: float64
2020-02-27 00:00:00+00:00   -0.716086
2020-02-28 00:00:00+00:00   -1.162248
2020-02-29 00:00:00+00:00    0.741673
2020-03-01 00:00:00+00:00   -1.203902
2020-03-02 00:00:00+00:00    0.981236
Freq: D, dtype: float64
```

53. 其他时区

```
ts_utc = ts_utc.tz_convert('US/Eastern')
```

output

```
2020-02-26 19:00:00-05:00   -0.716086
2020-02-27 19:00:00-05:00   -1.162248
2020-02-28 19:00:00-05:00    0.741673
2020-02-29 19:00:00-05:00   -1.203902
2020-03-01 19:00:00-05:00    0.981236
Freq: D, dtype: float64
```

54. 调整时间

```
prng = pandas.period_range('1999Q1', '2000Q4', freq='Q-NOV')
print(prng)
ts = pandas.Series(numpy.random.randn(len(prng)), prng)
print(ts)
# 季度频率转换为下一季度月末上午 9 点
ts.index = (prng.asfreq('M', 'e') + 1).asfreq('H', 's') + 9
print(ts)
```

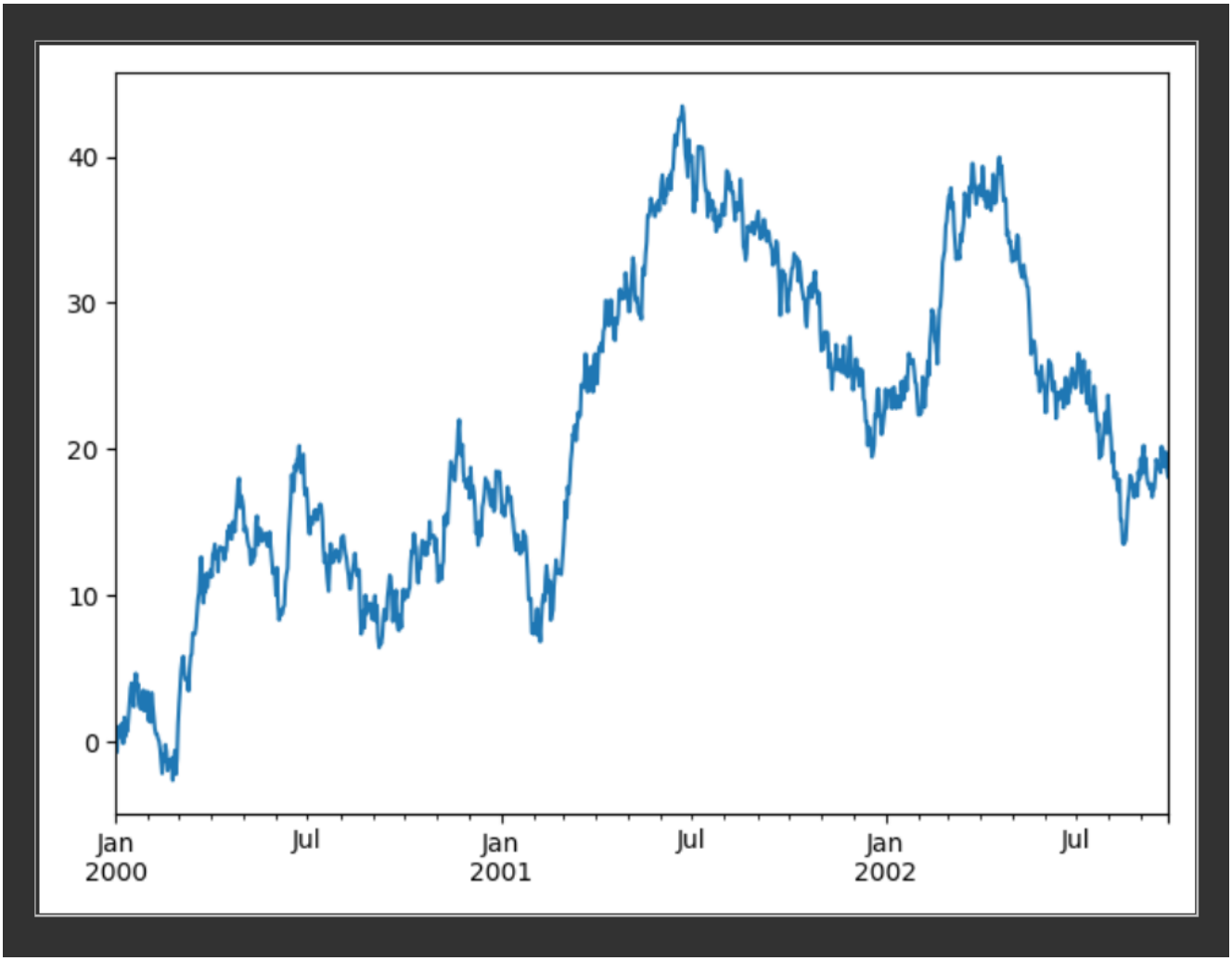
output

```
PeriodIndex(['1999Q1', '1999Q2', '1999Q3', '1999Q4', '2000Q1',
'2000Q2','2000Q3', '2000Q4'],
            dtype='period[Q-NOV]', freq='Q-NOV')
1999Q1    1.047781
1999Q2   -0.255710
1999Q3    1.251463
1999Q4   -1.535985
2000Q1    0.995607
2000Q2    1.249300
2000Q3   -1.742474
2000Q4    0.693920
Freq: Q-NOV, dtype: float64
1999-03-01 09:00    1.047781
1999-06-01 09:00   -0.255710
1999-09-01 09:00    1.251463
1999-12-01 09:00   -1.535985
2000-03-01 09:00    0.995607
2000-06-01 09:00    1.249300
2000-09-01 09:00   -1.742474
2000-12-01 09:00    0.693920
Freq: H, dtype: float64
```

55. pandas可视化

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

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000',
periods=1000))

ts = ts.cumsum()

ts.plot()

plt.show()
```

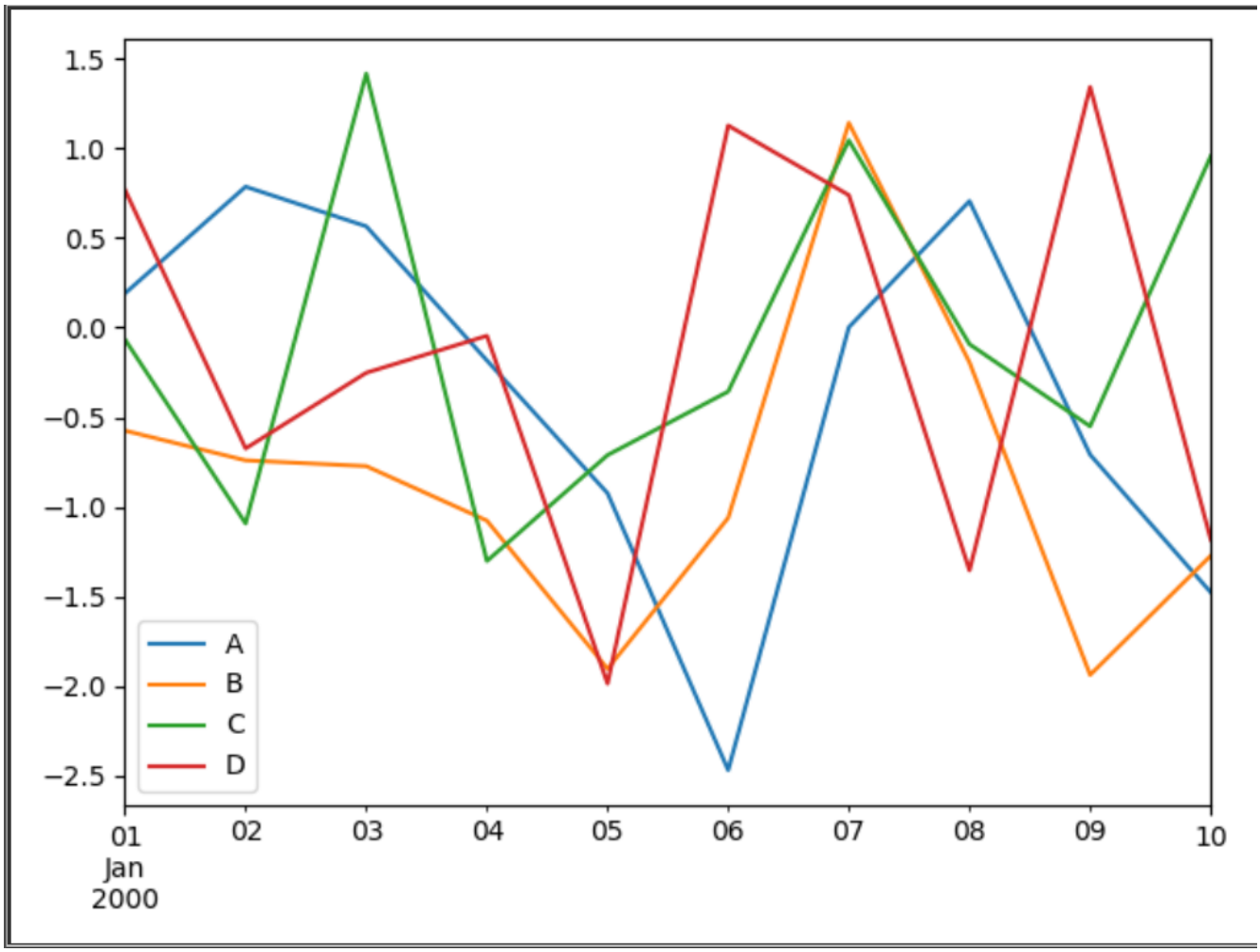
output



51. 带标签数据

```
df = pd.DataFrame(np.random.randn(10, 4), index=ts.index, columns=['A', 'B',
'C', 'D'])

plt.figure()

df.plot()

plt.legend(loc='best')

plt.show()
```

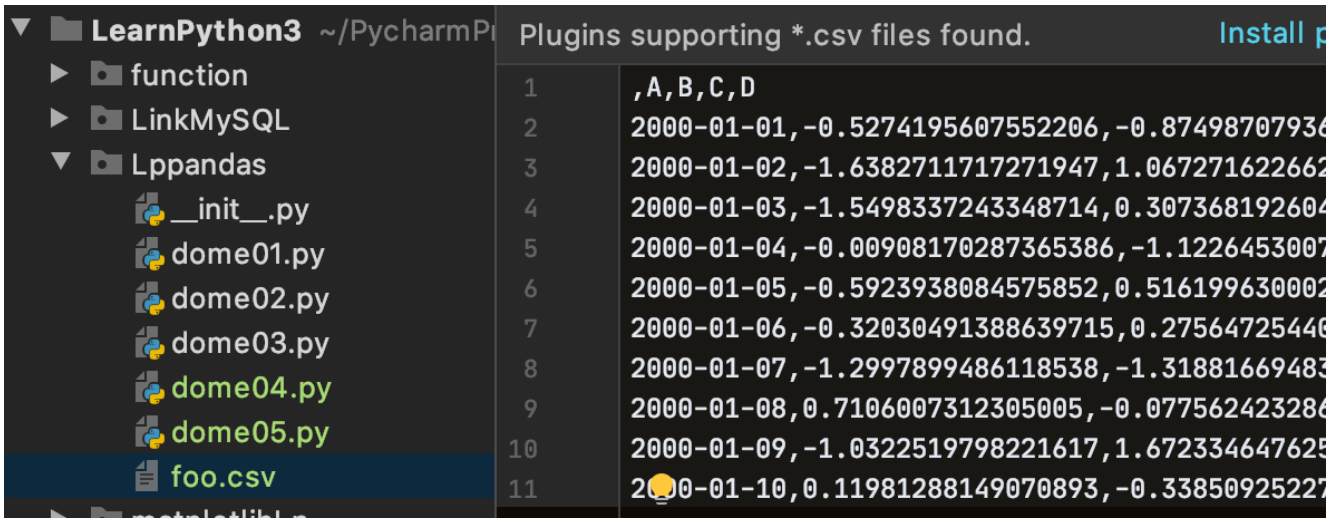
output



52. 数据写入文件-CSV

```
import numpy as np

ts = pd.Series(np.random.randn(10), index=pd.date_range('1/1/2000',
periods=10))

df = pd.DataFrame(np.random.randn(10, 4), index=ts.index, columns=['A', 'B',
'C', 'D'])

df.to_csv('foo.csv')
```

output



53.读取数据

```
print(pd.read_csv('foo.csv'))
```

output

```
   Unnamed: 0         A         B         C         D
0  2000-01-01 -0.527420 -0.874987 -0.143515 -0.577591
1  2000-01-02 -1.638271  1.067272 -0.187489  1.551672
2  2000-01-03 -1.549834  0.307368  0.054059 -1.341359
3  2000-01-04 -0.009082 -1.122645  2.011859  1.516419
4  2000-01-05 -0.592394  0.516200 -0.316947 -1.117220
5  2000-01-06 -0.320305  0.275647 -1.769232  0.303406
6  2000-01-07 -1.299790 -1.318817  0.310690 -0.114663
7  2000-01-08  0.710601 -0.077562  1.160157  2.411994
8  2000-01-09 -1.032252  1.672335  1.413137  0.983095
9  2000-01-10  0.119813 -0.338509  0.400005  0.795375
```

54. 数据写入文件-HDF5

```
ts = pd.Series(np.random.randn(10), index=pd.date_range('1/1/2000',
periods=10))

df = pd.DataFrame(np.random.randn(10, 4), index=ts.index, columns=['A', 'B',
'C', 'D'])

df.to_hdf('foo.h5', 'df')
```

output

55.读取文件-HDF5

```
print(pd.read_hdf('foo.h5', 'df'))
```

output

```
                   A         B         C         D
2000-01-01 -2.242973 -1.239371  1.634653 -1.093958
2000-01-02 -1.083398  1.799172 -0.480797 -0.591346
2000-01-03 -1.210755  0.185416 -0.885352  0.488429
2000-01-04 -0.707914 -0.179782  1.659592 -1.282103
2000-01-05 -0.706149 -1.432435 -0.952431 -0.929880
2000-01-06  0.885902  0.534051 -0.069661 -0.451097
2000-01-07  1.175877  2.032731 -1.135495 -0.563661
2000-01-08  1.691685  0.356640  0.826108 -1.025160
2000-01-09 -0.075771  0.690412  0.013092  1.148779
2000-01-10  0.228023  0.779857 -0.754926  2.329994
```

56.数据写入文件-Excel

```
df.to_excel('foo.xlsx', sheet_name='Sheet1')
```

output

	A	B	C	D	E
1		A	B	C	D
2	2000-01-01 00:00:00	-0.16476	0.602627	-0.85891	-0.32674
3	2000-01-02 00:00:00	-0.83432	-0.12451	-0.30357	-0.01648
4	2000-01-03 00:00:00	0.047414	0.130621	1.217756	0.023756
5	2000-01-04 00:00:00	-1.56003	-0.19992	1.615332	-1.4772
6	2000-01-05 00:00:00	-0.60308	-0.95402	0.789401	-0.74502
7	2000-01-06 00:00:00	0.255504	0.80949	0.907079	2.102616
8	2000-01-07 00:00:00	-0.49852	1.020483	0.444827	-0.29688
9	2000-01-08 00:00:00	0.069859	-0.6471	0.303071	-0.06296
10	2000-01-09 00:00:00	1.891164	-0.74363	0.119502	0.527606
11	2000-01-10 00:00:00	0.915416	0.973202	1.078322	-0.91406
12					

57. 读取Excel数据

```
print(pd.read_excel('foo.xlsx', 'Sheet1', index_col=None, na_values=['NA']))
```

output

```
  Unnamed: 0          A         B         C         D
0 2000-01-01 -0.545935 -0.099937  0.025618  1.328725
1 2000-01-02 -0.150113 -0.555841 -0.456048  0.830194
2 2000-01-03 -1.916913 -0.313361 -0.861359  0.619659
3 2000-01-04  0.957889 -1.191625  0.272261 -1.480473
4 2000-01-05  0.655982  2.234689  1.239589 -0.796676
5 2000-01-06 -0.225543  1.335624 -1.568939  1.431208
6 2000-01-07  0.457402  0.949207  0.926534  0.241123
7 2000-01-08  1.498727 -0.605570  0.984661  0.695882
8 2000-01-09  1.612117  0.030647  1.331574  0.192364
9 2000-01-10  1.709248  0.814548  0.027659 -0.629572
```