**python之pandas的基本使用（1）**

2017年01月20日 16:57:50 [cxmscb](https://me.csdn.net/cxmscb) 阅读数：34103 标签： [pandas](http://so.csdn.net/so/search/s.do?q=pandas&t=blog)[python](http://so.csdn.net/so/search/s.do?q=python&t=blog)[数据分析](http://so.csdn.net/so/search/s.do?q=%E6%95%B0%E6%8D%AE%E5%88%86%E6%9E%90&t=blog) 更多

个人分类： [机器学习](https://blog.csdn.net/cxmscb/article/category/6147299)[pythonNote](https://blog.csdn.net/cxmscb/article/category/6094823)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/cxmscb/article/details/54632492

**一、pandas概述**

pandas ：pannel data analysis（面板数据分析）。pandas是基于numpy构建的，为时间序列分析提供了很好的支持。pandas中有两个主要的数据结构，一个是Series，另一个是DataFrame。

**二、数据结构 Series**

Series 类似于一维数组与字典(map)数据结构的结合。它由一组数据和一组与数据相对应的数据标签（索引index）组成。这组数据和索引标签的基础都是一个一维ndarray数组。可将index索引理解为行索引。 Series的表现形式为：索引在左，数据在右。

• 获取数据和索引：ser\_obj.index, ser\_obj.values

• 预览数据：ser\_obj.head(n), ser\_obj.tail(n)

Series的使用代码示例：

import pandas as pd

from pandas import Series,DataFrame

print '用一维数组生成Series'

x = Series([1,2,3,4])

print x

'''

0 1

1 2

2 3

3 4

'''

print x.values # [1 2 3 4]

# 默认标签为0到3的序号

print x.index # RangeIndex(start=0, stop=4, step=1)

print '指定Series的index' # 可将index理解为行索引

x = Series([1, 2, 3, 4], index = ['a', 'b', 'd', 'c'])

print x

'''

a 1

b 2

d 3

c 4

'''

print x.index # Index([u'a', u'b', u'd', u'c'], dtype='object')

print x['a'] # 通过行索引来取得元素值：1

x['d'] = 6 # 通过行索引来赋值

print x[['c', 'a', 'd']] # 类似于numpy的花式索引

'''

c 4

a 1

d 6

'''

print x[x > 2] # 类似于numpy的布尔索引

'''

d 6

c 4

'''

print 'b' in x # 类似于字典的使用：是否存在该索引：True

print 'e' in x # False

print '使用字典来生成Series'

data = {'a':1, 'b':2, 'd':3, 'c':4}

x = Series(data)

print x

'''

a 1

b 2

c 4

d 3

'''

print '使用字典生成Series,并指定额外的index，不匹配的索引部分数据为NaN。'

exindex = ['a', 'b', 'c', 'e']

y = Series(data, index = exindex) # 类似替换索引

print y

'''

a 1.0

b 2.0

c 4.0

e NaN

'''

print 'Series相加，相同行索引相加，不同行索引则数值为NaN'

print x+y

'''

a 2.0

b 4.0

c 8.0

d NaN

e NaN

'''

print '指定Series/索引的名字'

y.name = 'weight of letters'

y.index.name = 'letter'

print y

'''

letter

a 1.0

b 2.0

c 4.0

e NaN

Name: weight of letters, dtype: float64

'''

print '替换index'

y.index = ['a', 'b', 'c', 'f']

print y # 不匹配的索引部分数据为NaN

'''

a 1.0

b 2.0

c 4.0

f NaN

Name: weight of letters, dtype: float64

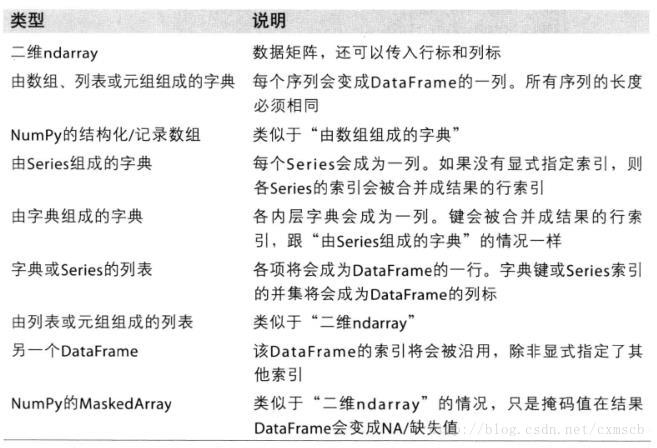
'''

**三、数据结构 DataFrame**

DataFrame是一个类似表格的数据结构，索引包括列索引和行索引，包含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔值等）。DataFrame的每一行和每一列都是一个Series，这个Series的name属性为当前的行索引名/列索引名。

* 通过列索引获取列数据（Series类型 ）：df\_obj[col\_idx] 或 df\_obj.col\_idx
* .ix，标签与位置混合索引

可输入给DataFrame构造器的数据：



DataFrame的使用代码示例：

print '使用字典生成DataFrame，key为列名字。'

data = {'state':['ok', 'ok', 'good', 'bad'],

'year':[2000, 2001, 2002, 2003],

'pop':[3.7, 3.6, 2.4, 0.9]}

print DataFrame(data) # 行索引index默认为0，1，2，3

'''

pop state year

0 3.7 ok 2000

1 3.6 ok 2001

2 2.4 good 2002

3 0.9 bad 2003

'''

# 指定列索引columns,不匹配的列为NaN

print DataFrame(data, columns = ['year', 'state', 'pop','debt'])

'''

year state pop

0 2000 ok 3.7

1 2001 ok 3.6

2 2002 good 2.4

3 2003 bad 0.9

'''

print '指定行索引index'

x = DataFrame(data,

columns = ['year', 'state', 'pop', 'debt'],

index = ['one', 'two', 'three', 'four'])

print x

'''

year state pop debt

one 2000 ok 3.7 NaN

two 2001 ok 3.6 NaN

three 2002 good 2.4 NaN

four 2003 bad 0.9 NaN

'''

import numpy

print 'DataFrame元素的索引与修改'

print x['state'] # 返回一个名为state的Series

'''

one ok

two ok

three good

four bad

Name: state, dtype: object

'''

print x.state # 可直接用.进行列索引

print x.ix['three'] # 用.ix[]来区分[]进行行索引

'''

year 2002

state good

pop 2.4

debt NaN

Name: three, dtype: object

'''

x['debt'] = 16.5 # 修改一整列数据

print x

'''

year state pop debt

one 2000 ok 3.7 16.5

two 2001 ok 3.6 16.5

three 2002 good 2.4 16.5

four 2003 bad 0.9 16.5

'''

x.debt = numpy.arange(4) # 用numpy数组修改元素

print x

'''

year state pop debt

one 2000 ok 3.7 0

two 2001 ok 3.6 1

three 2002 good 2.4 2

four 2003 bad 0.9 3

'''

print '用Series修改元素，没有指定的默认数据用NaN'

val = Series([-1.2, -1.5, -1.7,0], index = ['one', 'two', 'five','six'])

x.debt = val # DataFrame的行索引不变

print x

'''

year state pop debt

one 2000 ok 3.7 -1.2

two 2001 ok 3.6 -1.5

three 2002 good 2.4 NaN

four 2003 bad 0.9 NaN

'''

print '给DataFrame添加新列'

x['gain'] = (x.debt > 0) # 如果debt大于0为True

print x

'''

year state pop debt gain

one 2000 ok 3.7 -1.2 False

two 2001 ok 3.6 -1.5 False

three 2002 good 2.4 NaN False

four 2003 bad 0.9 NaN False

'''

print x.columns

# Index([u'year', u'state', u'pop', u'debt', u'gain'], dtype='object')

print 'DataFrame转置'

print x.T

'''

one two three four

year 2000 2001 2002 2003

state ok ok good bad

pop 3.7 3.6 2.4 0.9

debt -1.2 -1.5 NaN NaN

gain False False False False

'''

print '使用切片初始化数据,未被匹配的数据为NaN'

pdata = {'state':x['state'][0:3], 'pop':x['pop'][0:2]}

y = DataFrame(pdata)

print y

'''

pop state

one 3.7 ok

three NaN good

two 3.6 ok

'''

print '指定索引和列的名称'

# 与Series的index.name相区分

y.index.name = '序号'

y.columns.name = '信息'

print y

'''

信息 pop state

序号

one 3.7 ok

three NaN good

two 3.6 ok

'''

print y.values

'''

[[3.7 'ok']

[nan 'good']

[3.6 'ok']]

'''

**四、索引对象**

pandas的索引对象负责管理轴标签和轴名称等。构建Series或DataFrame时，所用到的任何数组或其他序列的标签都会被转换成一个Index对象。 Index对象是不可修改的，Series和DataFrame中的索引都是Index对象。

代码示例：

from pandas import Index

print '获取Index对象'

x = Series(range(3), index = ['a', 'b', 'c'])

index = x.index

print index

# Index([u'a', u'b', u'c'], dtype='object')

print index[0:2]

# Index([u'a', u'b'], dtype='object')

try:

index[0]='d'

except:

print "Index is immutable"

print '构造/使用Index对象'

index = Index(numpy.arange(3))

obj2 = Series([1.5, -2.5, 0], index = index)

print obj2

'''

0 1.5

1 -2.5

2 0.0

dtype: float64

'''

print obj2.index is index # True

print '判断列/行索引是否存在'

data = {'pop':{2.4, 2.9},

'year':{2001, 2002} }

x = DataFrame(data)

print x

'''

pop year

0 {2.4, 2.9} {2001, 2002}

1 {2.4, 2.9} {2001, 2002}

'''

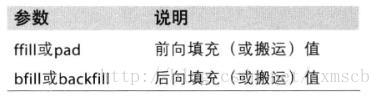
print 'pop' in x.columns # True

print 1 in x.index # True

**五、基本功能**

1. 对列/行索引重新指定索引（删除/增加：行/列）：reindex函数

reindex的method选项：



代码示例：

print '重新指定索引及NaN填充值'

x = Series([4, 7, 5], index = ['a', 'b', 'c'])

y = x.reindex(['a', 'b', 'c', 'd'])

print y

'''

a 4.0

b 7.0

c 5.0

d NaN

dtype: float64

'''

print x.reindex(['a', 'b', 'c', 'd'], fill\_value = 0)

# fill\_value 指定不存在元素NaN的默认值

'''

a 4

b 7

c 5

d 0

dtype: int64

'''

print '重新指定索引并指定填充NaN的方法'

x = Series(['blue', 'purple'], index = [0, 2])

print x.reindex(range(4), method = 'ffill')

'''

0 blue

1 blue

2 purple

3 purple

dtype: object

'''

print '对DataFrame重新指定行/列索引'

x = DataFrame(numpy.arange(9).reshape(3, 3),

index = ['a', 'c', 'd'],

columns = ['A', 'B', 'C'])

print x

'''

A B C

a 0 1 2

c 3 4 5

d 6 7 8

'''

x = x.reindex(['a', 'b', 'c', 'd'],method = 'bfill')

print x

'''

A B C

a 0 1 2

b 3 4 5

c 3 4 5

d 6 7 8

'''

print '重新指定column'

states = ['A', 'B', 'C','D']

x = x.reindex(columns = states,fill\_value = 0)

print x

'''

A B C D

a 0 1 2 0

b 3 4 5 0

d 6 7 8 0

c 3 4 5 0

'''

print x.ix[['a', 'b', 'd', 'c'], states]

'''

A B C D

a 0 1 2 0

b 3 4 5 0

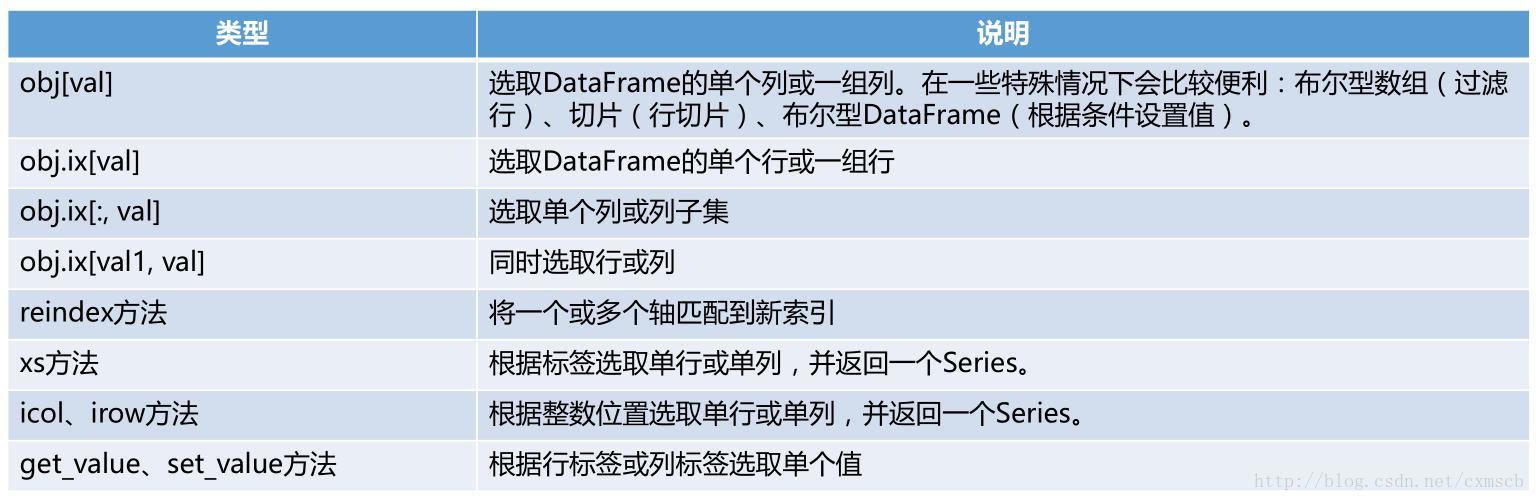
d 6 7 8 0

c 3 4 5 0

'''

1. 删除（丢弃）整一行/列的元素：drop函数
2. print 'Series根据行索引删除行'
3. x = Series(numpy.arange(4), index = ['a', 'b', 'c','d'])
4. print x.drop('c')
5. '''
6. a 0
7. b 1
8. d 3
9. dtype: int32
10. '''
11. print x.drop(['a', 'b']) # 花式删除
12. '''
13. c 2
14. d 3
15. dtype: int32
16. '''
17. print 'DataFrame根据索引行/列删除行/列'
18. x = DataFrame(numpy.arange(16).reshape((4, 4)),
19. index = ['a', 'b', 'c', 'd'],
20. columns = ['A', 'B', 'C', 'D'])
21. print x
22. '''
23. A B C D
24. a 0 1 2 3
25. b 4 5 6 7
26. c 8 9 10 11
27. d 12 13 14 15
28. '''
29. print x.drop(['A','B'],axis=1) # 在列的维度上删除AB两行
30. '''
31. C D
32. a 2 3
33. b 6 7
34. c 10 11
35. d 14 15
36. '''
37. print x.drop('a', axis = 0) # 在行的维度上删除行
38. '''
39. A B C D
40. b 4 5 6 7
41. c 8 9 10 11
42. d 12 13 14 15
43. '''
44. print x.drop(['a', 'b'], axis = 0)
45. '''
46. A B C D
47. c 8 9 10 11
48. d 12 13 14 15
49. '''
50. 索引、选取和过滤：

DataFrame的索引选项：



print 'Series的数组索引/字典索引'

x = Series(numpy.arange(4), index = ['a', 'b', 'c', 'd'])

print x['b'] # 1 像字典一样索引

print x[1] # 1 像数组一样索引

print x[[1, 3]] # 花式索引

'''

b 1

d 3

dtype: int32

'''

print x[x < 2] # 布尔索引

'''

a 0

b 1

dtype: int32

'''

print 'Series的数组切片'

print x['a':'c'] # 闭区间，索引顺序须为前后

'''

a 0

b 1

c 2

'''

x['a':'c'] = 5

print x

'''

a 5

b 5

c 5

d 3

'''

print 'DataFrame的索引'

data = DataFrame(numpy.arange(16).reshape((4, 4)),

index = ['a', 'b', 'c', 'd'],

columns = ['A', 'B', 'C', 'D'])

print data

'''

A B C D

a 0 1 2 3

b 4 5 6 7

c 8 9 10 11

d 12 13 14 15

'''

print data['A'] # 打印列

'''

a 0

b 4

c 8

d 12

Name: A, dtype: int32

'''

print data[['A', 'B']] # 花式索引

'''

A B

a 0 1

b 4 5

c 8 9

d 12 13

'''

print data[:2] # 切片索引,选择行

'''

A B C D

a 0 1 2 3

b 4 5 6 7

'''

print data.ix[:2, ['A', 'B']] # 指定行和列索引

'''

A B

a 0 1

b 4 5

'''

print data.ix[['a', 'b'], [3, 0, 1]] #行:字典索引，列:数组索引

'''

D A B

a 3 0 1

b 7 4 5

'''

print data.ix[2] # 打印第2行（从0开始）

'''

A 8

B 9

C 10

D 11

'''

print data.ix[:'b', 'A'] # 行从开始到b，第A列。

'''

a 0

b 4

Name: A, dtype: int32

'''

print '根据条件选择'

print data

'''

A B C D

a 0 1 2 3

b 4 5 6 7

c 8 9 10 11

d 12 13 14 15

'''

print data[data.A > 5] # 根据条件选择行

'''

A B C D

c 8 9 10 11

d 12 13 14 15

'''

print data < 5 # 打印True或者False

'''

A B C D

a True True True True

b True False False False

c False False False False

d False False False False

'''

data[data < 5] = 0 # 条件索引

print data

'''

A B C D

a 0 0 0 0

b 0 5 6 7

c 8 9 10 11

d 12 13 14 15

'''

1. 算术运算和数据对齐

代码示例：

print 'DataFrame算术:不重叠部分为NaN,重叠部分元素运算'

x = DataFrame(numpy.arange(9.).reshape((3, 3)),

columns = ['A','B','C'],

index = ['a', 'b', 'c'])

y = DataFrame(numpy.arange(12).reshape((4, 3)),

columns = ['A','B','C'],

index = ['a', 'b', 'c', 'd'])

print x

print y

print x + y

'''

A B C

a 0.0 2.0 4.0

b 6.0 8.0 10.0

c 12.0 14.0 16.0

d NaN NaN NaN

'''

print '对x/y的不重叠部分填充，不是对结果NaN填充'

print x.add(y, fill\_value = 0) # x不变化

'''

A B C

a 0.0 2.0 4.0

b 6.0 8.0 10.0

c 12.0 14.0 16.0

d 9.0 10.0 11.0

'''

print 'DataFrame与Series运算：每行/列进行运算'

frame = DataFrame(numpy.arange(9).reshape((3, 3)),

columns = ['A','B','C'],

index = ['a', 'b', 'c'])

series = frame.ix[0]

print frame

'''

A B C

a 0 1 2

b 3 4 5

c 6 7 8

'''

print series

'''

A 0

B 1

C 2

'''

print frame - series # 默认按行运算

'''

A B C

a 0 0 0

b 3 3 3

c 6 6 6

'''

series2 = Series(range(4), index = ['A','B','C','D'])

print frame + series2 # 按行运算：缺失列则为NaN

'''

A B C D

a 0 2 4 NaN

b 3 5 7 NaN

c 6 8 10 NaN

'''

series3 = frame.A

print series3

'''

a 0

b 3

c 6

'''

print frame.sub(series3, axis = 0) # 按列运算。

'''

A B C

a 0 1 2

b 0 1 2

c 0 1 2

'''

1. numpy函数应用与映射

代码示例：

print 'numpy函数在Series/DataFrame的应用'

frame = DataFrame(numpy.arange(9).reshape(3,3),

columns = ['A','B','C'],

index = ['a', 'b', 'c'])

print frame

'''

A B C

a 0 1 2

b 3 4 5

c 6 7 8

'''

print numpy.square(frame)

'''

A B C

a 0 1 4

b 9 16 25

c 36 49 64

'''

series = frame.A

print series

'''

a 0

b 3

c 6

'''

print numpy.square(series)

'''

a 0

b 9

c 36

'''

print 'lambda(匿名函数)以及应用'

print frame

'''

A B C

a 0 1 2

b 3 4 5

c 6 7 8

'''

print frame.max()

'''

A 6

B 7

C 8

'''

f = lambda x: x.max() - x.min()

print frame.apply(f) # 作用到每一列

'''

A 6

B 6

C 6

'''

print frame.apply(f, axis = 1) # 作用到每一行

'''

a 2

b 2

c 2

'''

def f(x): # Series的元素的类型为Series

return Series([x.min(), x.max()], index = ['min', 'max'])

print frame.apply(f)

'''

A B C

min 0 1 2

max 6 7 8

'''

print 'applymap和map：作用到每一个元素'

\_format = lambda x: '%.2f' % x

print frame.applymap(\_format) # 针对DataFrame

'''

A B C

a 0.00 1.00 2.00

b 3.00 4.00 5.00

c 6.00 7.00 8.00

'''

print frame['A'].map(\_format) # 针对Series

'''

a 0.00

b 3.00

c 6.00

Name: A, dtype: object

'''