pandas基础教程五\_多重索引

一、什么是多重索引对象

分层/多级索引(MultiIndex)可以理解为堆叠的一种索引结构，官方文档提及到它为一些相当复杂的数据分析和操作打开了大门，尤其是在处理高维数据的时候显得十分便利。从本质上讲,有了多级索引就可以在Series和DataFrame等低数据结构中存储和处理任意维数的数据。

二、多重索引的创建

多重索引的创建方式有很多，这里我们来看几个常见的方式：

（1）pd.MultiIndex.from\_arrays

（2）pd.MultiIndex.from\_frame

（3）pd.MultiIndex.from\_tuples

（4）pd.MultiIndex.from\_product

我们来看一下代码是怎样写的：

**1. pd.MultiIndex.from\_arrays()**

import pandas as pd  
import numpy as np  
pd\_test1 = pd.DataFrame(  
 np.random.randint(0,150,size=(4,3)),  
 columns=[**'java'**,**'html5'**,**'python'**],  
 index=pd.MultiIndex.from\_arrays  
 (  
 [  
 [**'a'**,**'b'**,**'c'**,**'d'**],  
 [**'A'**,**'B'**,**'C'**,**'D'**]  
 ]  
 )  
 )  
print(pd\_test1)  
print(pd\_test1.index)

#输出

java html5 python

a A 66 132 28

b B 77 110 94

c C 49 96 85

d D 137 94 136

MultiIndex([('a', 'A'),

('b', 'B'),

('c', 'C'),

('d', 'D')],

)

上述方式简单的将数组转换为MultiIndex。

**2. pd.MultiIndex.from\_frame()**

pd\_test2 = pd.DataFrame(  
 np.random.randint(0,150,size=(4,3)),  
 columns=[**'java'**,**'html5'**,**'python'**],  
 )  
  
print(pd\_test2)  
print(pd.MultiIndex.from\_frame(pd\_test2))  
print(pd.MultiIndex.from\_frame(pd\_test2,names=[**'J'**,**'H'**,**'P'**]))

# 输出

java html5 python

0 131 28 18

1 77 74 79

2 92 31 128

3 24 136 26

MultiIndex([(131, 28, 18),

( 77, 74, 79),

( 92, 31, 128),

( 24, 136, 26)],

names=['java', 'html5', 'python'])

MultiIndex([(131, 28, 18),

( 77, 74, 79),

( 92, 31, 128),

( 24, 136, 26)],

names=['J', 'H', 'P'])

使用pd.MultiIndex.from\_frame()方法，即可从DataFrame制作MultiIndex。

**3. pd.MultiIndex.from\_tuples()**

从上面两个方式就可以看出，第三种方式使用的是元组，我们来看一下实现：

*#3.pd.MultiIndex.from\_tuples*arr\_test3 = [(1, **'red'**),(1, **'blue'**),  
 (2, **'red'**), (2, **'blue'**),  
 (3, **'red'**), (3, **'blue'**)]  
index = pd.MultiIndex.from\_tuples(arr\_test3, names=(**'number'**, **'color'**))  
series\_test3 = pd.Series(arr\_test3, index=index)  
pd\_test3 = pd.DataFrame(arr\_test3, index=index)  
  
print(pd\_test3)  
print(series\_test3)  
print(pd\_test3.index)

# 输出

0 1

number color

1 red 1 red

blue 1 blue

2 red 2 red

blue 2 blue

3 red 3 red

blue 3 blue

number color

1 red (1, red)

blue (1, blue)

2 red (2, red)

blue (2, blue)

3 red (3, red)

blue (3, blue)

dtype: object

MultiIndex([(1, 'red'),

(1, 'blue'),

(2, 'red'),

(2, 'blue'),

(3, 'red'),

(3, 'blue')],

names=['number', 'color'])

**4. pd.MultiIndex.from\_product()**

*# 4.pd.MultiIndex.from\_product()*number = [1,2,3]  
color = [**'green'**, **'purple'**,**'blue'**]  
  
index = pd.MultiIndex.from\_product([number,color], names=[**'number'**, **'color'**])  
pd\_test4 = pd.DataFrame(np.random.randn(9), index=index)  
  
print(pd\_test4)  
print(pd\_test4.index)

# 输出：

0

number color

1 green 0.727379

purple -0.723411

blue 0.784745

2 green 1.202497

purple 0.540666

blue -0.565629

3 green 0.417987

purple -0.658018

blue -1.625875

MultiIndex([(1, 'green'),

(1, 'purple'),

(1, 'blue'),

(2, 'green'),

(2, 'purple'),

(2, 'blue'),

(3, 'green'),

(3, 'purple'),

(3, 'blue')],

names=['number', 'color'])

注：当考虑迭代两个对象的元素对时，可以优先考虑使用pd.MultiIndex.from\_product()来建立多重索引。

二、多重索引值获取

在数据分析和数据挖掘的过程中，我们很容易遇到怎么获取多重索引的值的需求，接下来我们一起来看一下怎么获取多重索引值的获取：

我们先创建我们的测试数据：

import pandas as pd  
import numpy as np  
  
iterables = [[**'1'**, **'2'**, **'3'**], [**'bule'**, **'green'**]]  
arrays = pd.MultiIndex.from\_product(iterables, names=[**'number'**, **'color'**])  
use\_df = pd.DataFrame(  
 np.random.randn(6,6),  
 index = arrays,  
 columns=arrays  
)  
print(use\_df)

# 输出为

number 1 2 3

color bule green bule green bule green

number color

1 bule 1.318232 0.178030 0.774973 -0.567643 -0.982961 -0.585139

green 1.397289 -0.571675 -2.372952 0.043259 0.385781 -0.232037

2 bule -1.321314 1.523664 0.419699 0.855854 -0.861650 1.089109

green -1.922272 0.219814 -1.032383 0.915663 -1.719036 1.472052

3 bule 0.302781 -1.035551 0.375275 0.792451 -1.590227 -0.925826

green -0.650767 -0.308584 -1.177814 1.757994 -0.413690 0.229166

1. **获取行列索引（全部）**

print(use\_df.index)  
print(use\_df.index.values)  
print(use\_df.columns.values)

# 输出

MultiIndex([('1', 'bule'),

('1', 'green'),

('2', 'bule'),

('2', 'green'),

('3', 'bule'),

('3', 'green')],

names=['number', 'color'])

[('1', 'bule') ('1', 'green') ('2', 'bule') ('2', 'green') ('3', 'bule')('3', 'green')]

[('1', 'bule') ('1', 'green') ('2', 'bule') ('2', 'green') ('3', 'bule')('3', 'green')]

另外可以通过pandas.MultiIndex.get\_level\_values(level)方法获取特定级别标签值的向量，这个向量的长度将等于索引的长度。参数level 是级别在MultiIndex中的整数位置，或者是级别的名称：

print(use\_df.index.get\_level\_values(0))  
print(use\_df.index.get\_level\_values(**'number'**))  
print(use\_df.index.get\_level\_values(1))  
print(use\_df.index.get\_level\_values(**'color'**))

# 输出

Index(['1', '1', '2', '2', '3', '3'], dtype='object', name='number')

Index(['1', '1', '2', '2', '3', '3'], dtype='object', name='number')

Index(['bule', 'green', 'bule', 'green', 'bule', 'green'], dtype='object', name='color')

Index(['bule', 'green', 'bule', 'green', 'bule', 'green'], dtype='object', name='color')

**2.获取指定的索引**

Pandas提供了可以获取指定索引的方法：

*# 获取标签或标签元组的位置*print(use\_df.index.get\_loc(**'1'**))  
*# 获取标签序列的位置*print(use\_df.index.get\_locs(**'1'**))  
print(use\_df.index.get\_locs(**'2'**))

# 输出

slice(0, 2, None)

[0 1]

[2 3]

print(use\_df[**'1'**].columns.values)  
print(use\_df[**'2'**].columns.values)  
print(use\_df[**'1'**].index.values)  
print(use\_df[**'2'**].index.values)  
print(use\_df[[**'2'**,**'3'**]].index.values)

# 输出

['bule' 'green']

['bule' 'green']

[('1', 'bule') ('1', 'green') ('2', 'bule') ('2', 'green') ('3', 'bule')('3', 'green')]

[('1', 'bule') ('1', 'green') ('2', 'bule') ('2', 'green') ('3', 'bule')('3', 'green')]

[('1', 'bule') ('1', 'green') ('2', 'bule') ('2', 'green') ('3', 'bule')('3', 'green')]

这里补充一个方法：pandas.MultiIndex.to\_flat\_index()，这个方法将MultiIndex转换为包含级别值的元组索引:

index = pd.MultiIndex.from\_product(  
 [[**'1'**, **'bule'**], [**'3'**, **'green'**]],  
 names=[**'number'**, **'color'**])  
print(index)  
print(index.to\_flat\_index())

# 输出

MultiIndex([( '1', '3'),

( '1', 'green'),

('bule', '3'),

('bule', 'green')],

names=['number', 'color'])

Index([('1', '3'), ('1', 'green'), ('bule', '3'), ('bule', 'green')], dtype='object')

Pandas还提供了其他的方法，大家可以自己研究一下。

三、取数操作

示范的数据为第二节的use\_df。

*# 取某一个索引的数据*

print(use\_df[**'1'**])

print(use\_df.loc[:][**'1'**])

print(use\_df[**'2'**])

*# 取多个索引的数据*

print(use\_df[[**'1'**,**'2'**]])

*# 列优先取数*print(use\_df.loc[**'1'**])

大家好好体会print(use\_df[**'1'**])和print(use\_df.loc[**'1'**])的不同，这里给出输出：

color bule green

number color

1 bule -0.181962 -0.241372

green -0.630053 0.676404

2 bule 0.696988 0.121653

green -0.386419 0.337361

3 bule -2.508173 -0.165031

green -0.939017 0.579207

number 1 2 3

color bule green bule green bule green

color

bule -0.181962 -0.241372 -0.054131 -1.291276 -1.323816 1.498784

green -0.630053 0.676404 -0.951133 -0.090582 -1.132037 -1.222341

使用iloc也可以获取意想不到的结果：

print(use\_df.iloc[1])  
print(use\_df.iloc[1][**'1'**])  
print(use\_df.iloc[1][1])

# 输出

number color

1 bule 0.919970

green -1.442989

2 bule -0.629463

green 1.040237

3 bule 0.596067

green 0.549045

Name: (1, green), dtype: float64

color

bule 0.919970

green -1.442989

Name: (1, green), dtype: float64

-1.4429891311997716

大家好好体会一下这个取数方法的不同，后期在数据分析的时候我们可以进一步体会这种取数的效果。

四、总结

本次文章的主要内容是关于pandas的多重索引的，大家掌握上述一些知识点，在数据分析的时候就可以进行多维数据的处理了。当然了，学习的小伙伴也要根据自己的实际数据分析需求进行练习。多练习和多思考就能有更多的收获。下期内容是关于pandas的广播操作。