Hierarchical Indexing

```
import pandas as pd
import numpy as np
data=pd.Series(np.random.randn(9), index=[['a', 'a', 'a', 'b', 'b', 'c', 'c', 'd', 'd'], [1,
data
```

```
a  1   -1.096249
   2    0.124050
   3   -0.461791
b  1    1.157519
   3   -2.343330
c  1    0.451074
   2   -0.704817
d  2   -2.235014
   3   -0.249622
dtype: float64
```

```
In [11]: data.index
```

```
MultiIndex([('a', 1),
            ('a', 2),
            ('a', 3),
            ('b', 1),
            ('b', 3),
            ('c', 1),
            ('c', 2),
            ('d', 2),
            ('d', 3)],
           )
```

```
In [12]: data['b']
```

```
1    0.171279
3    0.314759
dtype: float64
```

```
In [13]: data['b':'c']
```

```
b  1    0.171279
   3    0.314759
c  1    0.249995
   2   -1.338926
dtype: float64
```

```
In [14]: data.loc[['b', 'd']]
```

```
b  1    0.171279
```

```
         3    0.314759
      d  2   -0.896418
         3   -0.091266
      dtype: float64
```

```
In [15]: data.loc[:, 2]
```

```
      a    1.307996
      c   -1.338926
      d   -0.896418
      dtype: float64
```

```
In [16]: data.unstack()
```

	1	2	3
a	1.853002	1.307996	1.168647
b	0.171279	NaN	0.314759
c	0.249995	-1.338926	NaN
d	NaN	-0.896418	-0.091266

```
In [17]: data.unstack().stack()
```

```
      a  1    1.853002
         2    1.307996
         3    1.168647
      b  1    0.171279
         3    0.314759
      c  1    0.249995
         2   -1.338926
      d  2   -0.896418
         3   -0.091266
      dtype: float64
```

```
In [18]: frame = pd.DataFrame(np.arange(12).reshape((4, 3)),
....: index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],
....: columns=[['Ohio', 'Ohio', 'Colorado'],
....: ['Green', 'Red', 'Green']])
In [19]: frame
```

	Ohio	Colorado

```
frame.index
```

```
MultiIndex([('a', 1),
            ('a', 2),
            ('b', 1),
            ('b', 2)],
           )
```

```
In [20]: frame.index.names = ['key1', 'key2']
In [21]: frame.columns.names = ['state', 'color']
In [22]: frame
```

state	Ohio		Colorado
color	Green	Red	Green
key1 key2			
a 1	0	1	2
2	3	4	5
b 1	6	7	8
2	9	10	11

```
frame.columns
```

```
MultiIndex([(    'Ohio', 'Green'),
            (    'Ohio',   'Red'),
            ('Colorado', 'Green')],
           names=['state', 'color'])
```

```
In [23]: frame['Ohio']
```

color	Green	Red
key1 key2		
a 1	0	1
2	3	4
b 1	6	7
2	9	10

Reordering and Sorting Levels

In [24]: frame.swaplevel('key1', 'key2')

	state	Ohio		Colorado
	color	Green	Red	Green
key2	key1			
1	a	0	1	2
2	a	3	4	5
1	b	6	7	8
2	b	9	10	11

frame

	state	Ohio		Colorado
	color	Green	Red	Green
key1	key2			
a	1	0	1	2
	2	3	4	5
b	1	6	7	8
	2	9	10	11

In [25]: frame.sort_index(level=1)

	state	Ohio		Colorado
	color	Green	Red	Green
key1	key2			
a	1	0	1	2
b	1	6	7	8
a	2	3	4	5
b	2	9	10	11

In [26]: frame.swaplevel(0, 1).sort_index(level=0)

	state	Ohio		Colorado
	color	Green	Red	Green
key2	key1			
1	a	0	1	2
	b	6	7	8
2	a	3	4	5

Summary Statistics by Level

```
In [27]: frame.sum(level='key2')
```

state	Ohio		Colorado
color	Green	Red	Green
key2			
1	6	8	10
2	12	14	16

```
In [28]: frame.sum(level='color', axis=1)
```

	color	Green	Red
key1	key2		
a	1	2	1
	2	8	4
b	1	14	7
	2	20	10

Indexing with a DataFrame's columns

```
In [29]: frame = pd.DataFrame({'a': range(7), 'b': range(7, 0, -1),
....: 'c': ['one', 'one', 'one', 'two', 'two',
....: 'two', 'two'],
....: 'd': [0, 1, 2, 0, 1, 2, 3]})
```

```
frame
```

	a	b	c	d
0	0	7	one	0
1	1	6	one	1
2	2	5	one	2
3	3	4	two	0
4	4	3	two	1
5	5	2	two	2

```
In [31]: frame2 = frame.set_index(['d'])
In [32]: frame2
```

	a	b	c
d			
0	0	7	one
1	1	6	one
2	2	5	one
0	3	4	two
1	4	3	two
2	5	2	two
3	6	1	two

```
In [34]: frame2.reset_index()
```

	d	a	b	c
0	0	0	7	one
1	1	1	6	one
2	2	2	5	one
3	0	3	4	two
4	1	4	3	two
5	2	5	2	two
6	3	6	1	two

Combining and Merging Datasets

```
In [35]: df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
```

```
....: 'data1': range(7)})
In [36]: df2 = pd.DataFrame({'key': ['a', 'b', 'd'],
....: 'data2': range(3)})
In [37]: df1
```

	key	data1
0	b	0
1	b	1
2	a	2
3	c	3
4	a	4
5	a	5
6	b	6

```
df2
```

	key	data2
0	a	0
1	b	1
2	d	2

```
In [39]: pd.merge(df1, df2)
```

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0
4	a	4	0
5	a	5	0

```
In [40]: pd.merge(df1, df2, on='key')
```

	key	data1	data2
0	b	0	1
1	b	1	1
2	b	6	1
3	a	2	0

```
In [41]: df3 = pd.DataFrame({'lkey': ['b', 'b', 'a', 'c', 'a', 'a', 'b'],
....: 'data1': range(7)})
```

```
In [42]: df4 = pd.DataFrame({'rkey': ['a', 'b', 'd'],
....: 'data2': range(3)})
```

```
In [43]: pd.merge(df3, df4, left_on='lkey', right_on='rkey')
```

	lkey	data1	rkey	data2
0	b	0	b	1
1	b	1	b	1
2	b	6	b	1
3	a	2	a	0
4	a	4	a	0
5	a	5	a	0

```
In [44]: pd.merge(df1, df2, how='right')
```

	key	data1	data2
0	a	2.0	0
1	a	4.0	0
2	a	5.0	0
3	b	0.0	1
4	b	1.0	1
5	b	6.0	1
6	d	NaN	2

```
In [51]: left = pd.DataFrame({'key1': ['foo', 'foo', 'bar'],
....: 'key2': ['one', 'two', 'one'],
....: 'lval': [1, 2, 3]})
In [52]: right = pd.DataFrame({'key1': ['foo', 'foo', 'bar', 'bar']
```

```
In [52]: right = pd.DataFrame({'key1': ['foo', 'foo', 'bar', 'bar'],
....: 'key2': ['one', 'one', 'one', 'two'],
....: 'rval': [4, 5, 6, 7]})
In [53]: pd.merge(left, right, on=['key1', 'key2'], how='outer')
```

	key1	key2	lval	rval
0	foo	one	1.0	4.0
1	foo	one	1.0	5.0
2	foo	two	2.0	NaN
3	bar	one	3.0	6.0
4	bar	two	NaN	7.0

```
In [54]: pd.merge(left, right, on='key1')
```

	key1	key2_x	lval	key2_y	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

```
In [55]: pd.merge(left, right, on='key1', suffixes=('_left', '_right'))
```

	key1	key2_left	lval	key2_right	rval
0	foo	one	1	one	4
1	foo	one	1	one	5
2	foo	two	2	one	4
3	foo	two	2	one	5
4	bar	one	3	one	6
5	bar	one	3	two	7

Merging on Index

```
In [56]: left1 = pd.DataFrame({'key': ['a', 'b', 'a', 'a', 'b', 'c'],
....: 'value': range(6)})
In [57]: right1 = pd.DataFrame({'group_val': [3.5, 7]}, index=['a', 'b'])
```

In [57]: right1 = pd.DataFrame({'group_val': [3.5, 7]}, index=['a', 'b'])

In [58]: left1

	key	value
0	a	0
1	b	1
2	a	2
3	a	3
4	b	4
5	c	5

In [59]: right1

	group_val
a	3.5
b	7.0

In [60]: pd.merge(left1, right1, right_on='key', left_index=True)

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-9-e5524697de09> in <module>()
----> 1 pd.merge(left1, right1, right_on='key', left_index=True)

                                3 frames
/usr/local/lib/python3.6/dist-packages/pandas/core/generic.py in
_get_label_or_level_values(self, key, axis)
   1561             values = self.axes[axis].get_level_values(key)._values
   1562         else:
-> 1563             raise KeyError(key)
   1564
   1565         # Check for duplicates

KeyError: 'key'
```

SEARCH STACK OVERFLOW

New Section

In [61]: pd.merge(left1, right1, left_on='key', right_index=True, how='outer')

In [62]: lefth = pd.DataFrame({'key1': ['Ohio', 'Ohio', 'Ohio',
    ...: 'Nevada', 'Nevada']

```
.....: 'Nevada', 'Nevada'],
....: 'key2': [2000, 2001, 2002, 2001, 2002],
....: 'data': np.arange(5.)})
In [63]: righth = pd.DataFrame(np.arange(12).reshape((6, 2)),
....: index=[['Nevada', 'Nevada', 'Ohio', 'Ohio',
....: 'Ohio', 'Ohio'],
....: [2001, 2000, 2000, 2000, 2001, 2002]],
....: columns=['event1', 'event2'])
In [64]: lefth
```

	key1	key2	data
0	Ohio	2000	0.0
1	Ohio	2001	1.0
2	Ohio	2002	2.0
3	Nevada	2001	3.0
4	Nevada	2002	4.0

```
In [65]: righth
```

		event1	event2
Nevada	**2001**	0	1
	2000	2	3
Ohio	**2000**	4	5
	2000	6	7
	2001	8	9
	2002	10	11

```
In [66]: pd.merge(lefth, righth, left_on=['key1', 'key2'], right_index=True)
```

	key1	key2	data	event1	event2
0	Ohio	2000	0.0	4	5
0	Ohio	2000	0.0	6	7
1	Ohio	2001	1.0	8	9
2	Ohio	2002	2.0	10	11
3	Nevada	2001	3.0	0	1

```
In [67]: pd.merge(lefth, righth, left_on=['key1', 'key2'],
....: right_index=True, how='outer')
```

	key1	key2	data	event1	event2
0	Ohio	2000	0.0	4.0	5.0
0	Ohio	2000	0.0	6.0	7.0
1	Ohio	2001	1.0	8.0	9.0
2	Ohio	2002	2.0	10.0	11.0
3	Nevada	2001	3.0	0.0	1.0
4	Nevada	2002	4.0	NaN	NaN
4	Nevada	2000	NaN	2.0	3.0

```
In [68]: left2 = pd.DataFrame([[1., 2.], [3., 4.], [5., 6.]],
....: index=['a', 'c', 'e'],
....: columns=['Ohio', 'Nevada'])
In [69]: right2 = pd.DataFrame([[7., 8.], [9., 10.], [11., 12.], [13, 14]],
....: index=['b', 'c', 'd', 'e'],
....: columns=['Missouri', 'Alabama'])
In [70]: left2
```

	Ohio	Nevada
a	1.0	2.0
c	3.0	4.0
e	5.0	6.0

```
In [71]: right2
```

	Missouri	Alabama
b	7.0	8.0
c	9.0	10.0
d	11.0	12.0
e	13.0	14.0

```
In [72]: pd.merge(left2, right2, how='outer', left_index=True, right_index=True)
```

	Ohio	Nevada	Missouri	Alabama
a	1.0	2.0	NaN	NaN
b	NaN	NaN	7.0	8.0

```
In [73]: right2.join(left2)
```

	Missouri	Alabama	Ohio	Nevada
b	7.0	8.0	NaN	NaN
c	9.0	10.0	3.0	4.0
d	11.0	12.0	NaN	NaN
e	13.0	14.0	5.0	6.0

```
In [74]: left1.join(right1, on='key')
```

	key	value	group_val
0	a	0	3.5
1	b	1	7.0
2	a	2	3.5
3	a	3	3.5
4	b	4	7.0
5	c	5	NaN

```
In [75]: another = pd.DataFrame([[7., 8.], [9., 10.], [11., 12.], [16., 17.]],
....: index=['a', 'c', 'e', 'f'],
....: columns=['New York', 'Oregon'])
In [76]: another
```

	New York	Oregon
a	7.0	8.0
c	9.0	10.0
e	11.0	12.0
f	16.0	17.0

```
In [77]: left2.join([right2, another])
```

	Ohio	Nevada	Missouri	Alabama	New York	Oregon
a	1.0	2.0	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0	9.0	10.0

```
In [78]: left2.join([right2, another], how='outer')
```

	Ohio	Nevada	Missouri	Alabama	New York	Oregon
a	1.0	2.0	NaN	NaN	7.0	8.0
c	3.0	4.0	9.0	10.0	9.0	10.0
e	5.0	6.0	13.0	14.0	11.0	12.0
b	NaN	NaN	7.0	8.0	NaN	NaN
d	NaN	NaN	11.0	12.0	NaN	NaN
f	NaN	NaN	NaN	NaN	16.0	17.0

Concatenating Along an Axis

```
In [79]: arr = np.arange(12).reshape((3, 4))
In [80]: arr

     array([[ 0,  1,  2,  3],
            [ 4,  5,  6,  7],
            [ 8,  9, 10, 11]])


In [81]: np.concatenate([arr, arr], axis=1)

     array([[ 0,  1,  2,  3,  0,  1,  2,  3],
            [ 4,  5,  6,  7,  4,  5,  6,  7],
            [ 8,  9, 10, 11,  8,  9, 10, 11]])


In [82]: s1 = pd.Series([0, 1], index=['a', 'b'])
In [83]: s2 = pd.Series([2, 3, 4], index=['c', 'd', 'e'])
In [84]: s3 = pd.Series([5, 6], index=['f', 'g'])



In [85]: pd.concat([s1, s2, s3])

     a    0
     b    1
     c    2
     d    3
     e    4
     f    5
```

```
g    6
dtype: int64
```

```
In [86]: pd.concat([s1, s2, s3], axis=1)
```

	0	1	2
a	0.0	NaN	NaN
b	1.0	NaN	NaN
c	NaN	2.0	NaN
d	NaN	3.0	NaN
e	NaN	4.0	NaN
f	NaN	NaN	5.0
g	NaN	NaN	6.0

```
In [87]: s4 = pd.concat([s1, s3])
In [88]: s4
```

```
a    0
b    1
f    5
g    6
dtype: int64
```

```
In [89]: pd.concat([s1, s4], axis=1)
```

	0	1
a	0.0	0
b	1.0	1
f	NaN	5
g	NaN	6

```
In [90]: pd.concat([s1, s4], axis=1, join='inner')
```

	0	1
a	0	0
b	1	1

```
In [92]: result = pd.concat([s1, s1, s3], keys=['one', 'two', 'three'])
In [93]: result
```

```
one    a    0
       b    1
two    a    0
       b    1
three  f    5
       g    6
dtype: int64
```

```
In [94]: result.unstack()
```

	a	b	f	g
one	0.0	1.0	NaN	NaN
two	0.0	1.0	NaN	NaN
three	NaN	NaN	5.0	6.0

```
In [95]: pd.concat([s1, s2, s3], axis=1, keys=['one', 'two', 'three'])
```

	one	two	three
a	0.0	NaN	NaN
b	1.0	NaN	NaN
c	NaN	2.0	NaN
d	NaN	3.0	NaN
e	NaN	4.0	NaN
f	NaN	NaN	5.0
g	NaN	NaN	6.0

Combining Data with Overlap

```
In [108]: a = pd.Series([np.nan, 2.5, np.nan, 3.5, 4.5, np.nan],
.....: index=['f', 'e', 'd', 'c', 'b', 'a'])
In [109]: b = pd.Series(np.arange(len(a), dtype=np.float64),
.....: index=['f', 'e', 'd', 'c', 'b', 'a'])
In [110]: b[-1] = np.nan
In [111]: a
```

```
f    NaN
e    2.5
d    NaN
c    3.5
b    4.5
```

```
a    NaN
dtype: float64
```

```
b
```

```
f    0.0
e    1.0
d    2.0
c    3.0
b    4.0
a    NaN
dtype: float64
```

```
In [113]: np.where(pd.isnull(a), b, a)
```

```
array([0. , 2.5, 2. , 3.5, 4.5, nan])
```

```
In [114]: a.combine_first(b)
```

```
f    0.0
e    2.5
d    2.0
c    3.5
b    4.5
a    NaN
dtype: float64
```

```
In [115]: df1 = pd.DataFrame({'a': [1., np.nan, 5., np.nan],
.....: 'b': [np.nan, 2., np.nan, 6.],
.....: 'c': range(2, 18, 4)})
In [116]: df2 = pd.DataFrame({'a': [5., 4., np.nan, 3., 7.],
.....: 'b': [np.nan, 3., 4., 6., 8.]})
In [117]: df1
```

	a	b	c
0	1.0	NaN	2
1	NaN	2.0	6
2	5.0	NaN	10
3	NaN	6.0	14

```
df2
```

	a	b
0	5.0	NaN
1	4.0	3.0

```
In [119]: df2.combine_first(df1)
```

	a	b	c
0	5.0	NaN	2.0
1	4.0	3.0	6.0
2	5.0	4.0	10.0
3	3.0	6.0	14.0
4	7.0	8.0	NaN

Reshaping and Pivoting

```
In [120]: data = pd.DataFrame(np.arange(6).reshape((2, 3)),
.....: index=pd.Index(['Ohio', 'Colorado'], name='state'),
.....: columns=pd.Index(['one', 'two', 'three'],
.....: name='number'))
In [121]: data
```

```
In [122]: result = data.stack()
result
```

```
result.unstack()
```

```
In [126]: result.unstack('state')
```

```
In [127]: s1 = pd.Series([0, 1, 2, 3], index=['a', 'b', 'c', 'd'])
In [128]: s2 = pd.Series([4, 5, 6], index=['c', 'd', 'e'])
In [129]: data2 = pd.concat([s1, s2], keys=['one', 'two'])
In [130]: data2
```

```
In [131]: data2.unstack()
```

```
In [131]: data2.unstack().stack()
```

```
In [134]: data2.unstack().stack(dropna=False)
```

```
In [135]: df = pd.DataFrame({'left': result, 'right': result + 5},
```

```
In [135]: df = pd.DataFrame({'left': result, 'right': result + 5},
.....: columns=pd.Index(['left', 'right'], name='side'))
In [136]: df


In [137]: df.unstack('state')


In [138]: df.unstack('state').stack('side')
```

	state	Colorado	Ohio
number	side		
one	left	3	0
	right	8	5
two	left	4	1
	right	9	6
three	left	5	2
	right	10	7