Data Wrangling: Join, Combine, Reshaping

```
import numpy as np
import pandas as pd

data = pd.Series(np.random.uniform(size=9), index
=[["a","a","a","b","b","c","c","d","d"],[1,2,3,1,3,1,2,2,3]])
data

a  1    0.983110
   2    0.245524
   3    0.824464
b  1    0.891480
   3    0.560662
c  1    0.660139
   2    0.161066
d  2    0.440353
   3    0.972994
dtype: float64

data.index

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

data["b"]

1    0.891480
3    0.560662
dtype: float64

data["b":"c"]

b  1    0.891480
   3    0.560662
c  1    0.660139
   2    0.161066
dtype: float64

data.loc[["b","d"]]
```

```
b  1    0.891480
   3    0.560662
d  2    0.440353
   3    0.972994
dtype: float64
```

```
# inner level selection
data.loc[:,2]
```

```
a    0.245524
c    0.161066
d    0.440353
dtype: float64
```

```
#creating a data frame using unstack
data.unstack()
```

```
          1         2         3
a  0.983110  0.245524  0.824464
b  0.891480       NaN  0.560662
c  0.660139  0.161066       NaN
d       NaN  0.440353  0.972994
```

```
#data frame can have also an hierarchical index
frame = pd.DataFrame(np.arange(12).reshape((4,3)),
                     index=[["a","a","b","b"],[1,2,1,2]],
                     columns=[["Ohio","Ohio","Colorado"],
                              ["Green","Red","Greem"]])
frame
```

```
     Ohio     Colorado
    Green Red    Greem
a 1     0   1        2
  2     3   4        5
b 1     6   7        8
  2     9  10       11
```

```
#hierarichal levels have names
frame.index.names=["key1","key2"]
frame.columns.names = ["state","color"]
frame
```

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

```
frame.index.nlevels
```

```
2

#Reordering and Sorting Levels
frame.swaplevel("key1","key2")

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

frame.sort_index(level=1)

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

frame.swaplevel(0,1).sort_index(level=0)

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

#Indexing with a DataFrame's Columns
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
6  6  1  two  3

#using set_index to create a new DataFrame with one or more columns as
the index
frame2 = frame.set_index(["c","d"])
frame2
```

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

#remove columns
frame.set_index(["c","d"], drop=False)

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

#to move the index levels into columns
frame2.reset_index()

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

#Combine and Merging Datasets
#pandas.merge - connect rows in DF based on one or more keys
#pandas.concat - concatenate/stack objects along an axis
#combine_first - splice together overlapping data to fill in missing
values

df1 = pd.DataFrame({"key":["b","b","a","c","a","a","b"],
"data1":pd.Series(range(7), dtype="Int64")})
df2 = pd.DataFrame({"key":["a","b","d"], "data2":pd.Series(range(3),
dtype="Int64")})

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
```

```
#Many-to-one join. The df1 has multiple rows labeled a,b whereas df2
has only one row for each value in the key column
#Pandas.merge

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
```

```
pd.merge(df1,df2, on="key") #good practice to specify explicitly the
merging column
```

```
  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
```

```
df3 = pd.DataFrame({"lkey":["b","b","a","c","a","a","b"],
                    "data1":pd.Series(range(7), dtype="int64")})
df4 = pd.DataFrame({"rkey":["a","b","d"],
                    "data2":pd.Series(range(3), dtype="int64")})

df4
```

```
  rkey  data2
0    a      0
1    b      1
2    d      2
```

```
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
```

```
pd.merge(df1,df2, how="outer")
```

```
  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
6   c      3   <NA>
7   d   <NA>      2
```

```
pd.merge(df3,df4, left_on="lkey", right_on="rkey", how="outer")
```

```
  lkey  data1 rkey  data2
0    b    0.0    b    1.0
1    b    1.0    b    1.0
2    b    6.0    b    1.0
3    a    2.0    a    0.0
4    a    4.0    a    0.0
5    a    5.0    a    0.0
6    c    3.0  NaN    NaN
7  NaN    NaN    d    2.0
```

Summary

Option Behavior how="inner" use only the key combinations observed in both tables how="outer" use all key combinations observed in both tables together how="left" use all key combinations found in the left table how="right" use all key combinations found in the right table

```
#Merging on Index
left1 = pd.DataFrame({"key":["a","b","a","a","b","c"],
                     "value": pd.Series(range(6), dtype="int64")})

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

left1
```

```
  key  value
0   a      0
1   b      1
```

```
2   a      2
3   a      3
4   b      4
5   c      5

right1

   group_val
a        3.5
b        7.0

pd.merge(left1, right1, left_on="key", right_index=True)

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

#Concatenating Along an Axis
arr = np.arange(12).reshape((3,4))
arr

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

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]])

#Reshaping
#stack - rotates or pivots from the columns in the data to the rows
#unstack - pivots from rows into the columns
#melt - unpivots the data from long to wide format

data = pd.DataFrame(np.arange(6).reshape((2,3)),
                    index=pd.Index(["Cavite","Batangas"],
name="province"),
                    columns=pd.Index(["one","two","three"],
name="number"))
data

number    one  two  three
province
Cavite      0    1      2
Batangas    3    4      5
```

```
#using stack to pivot columns into rows
result = data.stack()

result

province  number
Cavite    one       0
          two       1
          three     2
Batangas  one       3
          two       4
          three     5
dtype: int32

result.unstack()

number    one  two  three
province
Cavite      0    1      2
Batangas    3    4      5

result.unstack(level=0)

province  Cavite  Batangas
number
one            0         3
two            1         4
three          2         5

result.unstack(level="province")

province  Cavite  Batangas
number
one            0         3
two            1         4
three          2         5

s1 = pd.Series([0,1,2,3], index=["a","b","c","d"], dtype="int64")
s2 = pd.Series([4,5,6], index = ["c","d","e"], dtype="int64")

data2 = pd.concat([s1,s2], keys=["one","two"])

data2

one  a    0
     b    1
     c    2
     d    3
two  c    4
     d    5
     e    6
dtype: int64
```

```
data2.unstack()
```

```
       a    b    c    d    e
one  0.0  1.0  2.0  3.0  NaN
two  NaN  NaN  4.0  5.0  6.0
```

```
data2.unstack().stack()
```

```
one  a    0.0
     b    1.0
     c    2.0
     d    3.0
two  c    4.0
     d    5.0
     e    6.0
dtype: float64
```

```
data2.unstack().stack(dropna=False)
```

```
one  a    0.0
     b    1.0
     c    2.0
     d    3.0
     e    NaN
two  a    NaN
     b    NaN
     c    4.0
     d    5.0
     e    6.0
dtype: float64
```