In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.DataFrame({'key':['A','B','C','A','B','C','A','B','C'],'data':[0,5,10,5,10,15,10,15,20]})
df
```

Out[2]:

	key	data
0	A	0
1	B	5
2	C	10
3	A	5
4	B	10
5	C	15
6	A	10
7	B	15
8	C	20

In [3]:
```
groupdf = df['data'].groupby(df['key'])
for name, group in groupdf:
    print(name)
    print(group)
```

```
A
0     0
3     5
6    10
Name: data, dtype: int64
B
1     5
4    10
7    15
Name: data, dtype: int64
C
2    10
5    15
8    20
Name: data, dtype: int64
```

In [5]:
```
df = pd.DataFrame({'key1' : ['a', 'b', 'b', 'a', 'a'],
 'key2' : ['one', 'two', 'one', 'two', 'one'],
 'data1' : np.random.randn(5),
'data2' : np.random.randn(5)})
```

In [ ]:
```
for (k1, k2), group in df.groupby(['key1', 'key2']):
    print((k1, k2))
    print(group)
```

In [ ]:
```
df['data2'].groupby(df['key1']).sum()
```

In [ ]:
```
means = df['data1'].groupby([df['key1'], df['key2']]).mean()
means
```

In [ ]:
```
means.unstack()
```

In [ ]:
```
states = np.array(['Ohio', 'California', 'California', 'Ohio', 'Ohio'])
```

In [ ]:
```
years = np.array([2005, 2006, 2005, 2006, 2005])
df['data1'].groupby([states, years]).mean()
```

In [ ]:
```
df['data1'].groupby([states, years]).mean()
```

In [ ]:
```
df['states']=states
df['years']=years
```

In [ ]:
```
df
```

In [ ]:
```
df['data1'].groupby([df['states'],df['years']]).mean()
```

In [ ]:
```
df['data1'].groupby([df['key1'], df['key2']]).mean()
```

In [ ]:
```
df.groupby(['key1', 'key2']).size()
```

In [ ]:
```
df
```

In [ ]:
```
df.groupby(['key1', 'key2']).size()
```

In [ ]:
```
pieces = dict(list(df.groupby('key1')))
```

In [ ]:
```
pieces
```

In [ ]:
```
df.groupby('key1')['data1'].sum()
```

Grouping with Dicts and Series

In [ ]:
```
 people = pd.DataFrame(np.random.randn(5, 5),
columns=['a', 'b', 'c', 'd', 'e'],
index=['Joe', 'Steve', 'Wes', 'Jim', 'Travis'])
```

In [ ]:
```
# Add a few NA values
people.iloc[2:3, [1, 2]] = np.nan
```

In [ ]:
```
people
```

Now, suppose I have a group correspondence for the columns and want to sum together the columns by group:

In [ ]:
```
mapping = {'a': 'red', 'b': 'red', 'c': 'blue',
'd': 'blue', 'e': 'red', 'f' : 'orange'}
```

In [ ]:
```
by_column = people.groupby(mapping,axis=1)
by_column.count()
```

In [ ]:
```
map_series = pd.Series(mapping)
```

In [ ]:
```
people.groupby(map_series, axis=1).count()
```

Grouping with Functions

In [ ]:
```
people.groupby(len).sum()
```

Grouping by Index Levels

In [ ]:
```
columns = pd.MultiIndex.from_arrays([['US', 'US', 'US', 'JP', 'JP'],[1, 3, 5, 1, 3]], names=['city', 'tenor'])
hier_df = pd.DataFrame(np.random.randn(4, 5), columns=columns)
```

In [ ]:
```
hier_df
```

In [ ]:
```
hier_df.groupby(level='city', axis=1).count()
```

Data Aggregation

In [6]:
```
df
```

Out[6]:

	key1	key2	data1	data2
0	a	one	-1.575168	-0.309040
1	b	two	0.202399	0.758345
2	b	one	-2.446542	1.165713
3	a	two	-0.541184	0.812300
4	a	one	-1.305165	-1.723706

In [55]:
```
grouped = df.groupby(['key1','key2'])
grouped['data1'].sum()
```

Out[55]:
```
key1  key2
a     one    -2.880333
      two    -0.541184
b     one    -2.446542
      two     0.202399
Name: data1, dtype: float64
```

In [56]:
```
grouped['data1'].quantile(0.9)
```

Out[56]:
```
key1  key2
a     one    -1.332165
      two    -0.541184
b     one    -2.446542
      two     0.202399
Name: data1, dtype: float64
```

In [32]:
```
df
```

Out[32]:

	key1	key2	data1	data2
0	a	one	-1.575168	-0.309040
1	b	two	0.202399	0.758345
2	b	one	-2.446542	1.165713
3	a	two	-0.541184	0.812300
4	a	one	-1.305165	-1.723706

In [59]:
```
def peak_to_peak(arr):
    return arr.max()-arr.min()
grouped['data1'].agg(peak_to_peak)
```

Out[59]:
```
key1  key2
a     one     0.270003
      two     0.000000
b     one     0.000000
      two     0.000000
Name: data1, dtype: float64
```

In [61]:
```
grouped.describe()
```

Out[61]:

		data1								data2							
		count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
key1	key2																
a	one	2.0	-1.440166	0.190921	-1.575168	-1.507667	-1.440166	-1.372666	-1.305165	2.0	-1.016373	1.00032	-1.723706	-1.370040	-1.016373	-0.662707	-0.309040
	two	1.0	-0.541184	NaN	-0.541184	-0.541184	-0.541184	-0.541184	-0.541184	1.0	0.812300	NaN	0.812300	0.812300	0.812300	0.812300	0.812300
b	one	1.0	-2.446542	NaN	-2.446542	-2.446542	-2.446542	-2.446542	-2.446542	1.0	1.165713	NaN	1.165713	1.165713	1.165713	1.165713	1.165713
	two	1.0	0.202399	NaN	0.202399	0.202399	0.202399	0.202399	0.202399	1.0	0.758345	NaN	0.758345	0.758345	0.758345	0.758345	0.758345

Column-Wise and Multiple Function Application

In [65]:
```
tips = pd.read_csv('tips.csv')
tips
```

Out[65]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [67]:
```
# Add tip percentage of total bill
tips['tip_pct'] = tips['tip'] / tips['total_bill']
```

In [68]:
```
tips[:6]
```

Out[68]:

	total_bill	tip	sex	smoker	day	time	size	tip_pct
0	16.99	1.01	Female	No	Sun	Dinner	2	0.059447
1	10.34	1.66	Male	No	Sun	Dinner	3	0.160542
2	21.01	3.50	Male	No	Sun	Dinner	3	0.166587
3	23.68	3.31	Male	No	Sun	Dinner	2	0.139780
4	24.59	3.61	Female	No	Sun	Dinner	4	0.146808
5	25.29	4.71	Male	No	Sun	Dinner	4	0.186240

In [75]:
```
grouped = tips.groupby(['day', 'smoker'])

grouped_pct = grouped['tip_pct']

grouped_pct.agg('mean')
```

Out[75]:
```
day   smoker
Fri   No        0.151650
      Yes       0.174783
Sat   No        0.158048
      Yes       0.147906
Sun   No        0.160113
      Yes       0.187250
Thur  No        0.160298
      Yes       0.163863
Name: tip_pct, dtype: float64
```

In [76]:
```
grouped_pct.agg(['mean', 'std', peak_to_peak])
```

Out[76]:

		mean	std	peak_to_peak
day	smoker			
Fri	No	0.151650	0.028123	0.067349
	Yes	0.174783	0.051293	0.159925
Sat	No	0.158048	0.039767	0.235193
	Yes	0.147906	0.061375	0.290095
Sun	No	0.160113	0.042347	0.193226
	Yes	0.187250	0.154134	0.644685
Thur	No	0.160298	0.038774	0.193350
	Yes	0.163863	0.039389	0.151240

In [77]:
```
grouped_pct.agg([('foo', 'mean'), ('bar', np.std)])
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_3828\345979284.py:1: FutureWarning: The provided callable <function std at 0x0000020625735120> is currently using SeriesGroupBy.std. In a future version of pandas, the provided callable will be used directly. To keep current behavior pass the string "std" instead.
  grouped_pct.agg([('foo', 'mean'), ('bar', np.std)])
```

Out[77]:

		foo	bar
day	**smoker**		
Fri	**No**	0.151650	0.028123
	Yes	0.174783	0.051293
Sat	**No**	0.158048	0.039767
	Yes	0.147906	0.061375
Sun	**No**	0.160113	0.042347
	Yes	0.187250	0.154134
Thur	**No**	0.160298	0.038774
	Yes	0.163863	0.039389

In [90]:
```
functions = ['count', 'mean', 'max']
result = grouped['tip_pct'].agg(functions)
result
```

Out[90]:

		count	mean	max
day	**smoker**			
Fri	**No**	4	0.151650	0.187735
	Yes	15	0.174783	0.263480
Sat	**No**	45	0.158048	0.291990
	Yes	42	0.147906	0.325733
Sun	**No**	57	0.160113	0.252672
	Yes	19	0.187250	0.710345
Thur	**No**	45	0.160298	0.266312
	Yes	17	0.163863	0.241255

In [ ]:
```

```