Python for Data Science

DataFrames - Part 3

PIERIAN DATA
www.pieriandata.com

Pandas - DataFrames Part 3

In [84]:
```
import numpy as np
import pandas as pd
```

In [85]:
```
# Index Levels
outside = ['G1','G1','G1','G2','G2','G2']
inside = [1,2,3,1,2,3]
hier_index = list(zip(outside,inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

In [ ]:

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

In [85]:
```
# Index Levels
outside = ['G1','G1','G1','G2','G2','G2']
inside = [1,2,3,1,2,3]
hier_index = list(zip(outside,inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

In [87]:
```
hier_index
```

Out[87]:
```
[1, 2, 3, 1, 2, 3]
```

In [88]:
```
list(zip(outside,inside))
```

Out[88]:
```
[('G1', 1), ('G1', 2), ('G1', 3), ('G2', 1), ('G2', 2), ('G2', 3)]
```

In [ ]:

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

```
import pandas as pd
```

In [85]:
```
# Index Levels
outside = ['G1','G1','G1','G2','G2','G2']
inside = [1,2,3,1,2,3]
hier_index = list(zip(outside,inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

In [89]:
```
hier_index
```

Out[89]:
```
MultiIndex(levels=[['G1', 'G2'], [1, 2, 3]],
           labels=[[0, 0, 0, 1, 1, 1], [0, 1, 2, 0, 1, 2]])
```

In [88]:

Out[88]:
```
[('G1', 1), ('G1', 2), ('G1', 3), ('G2', 1), ('G2', 2), ('G2', 3)]
```

In [ ]:

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

```
inside = [1,2,3,1,2,3]
hier_index = list(zip(outside,inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

In [90]:
```
df = pd.DataFrame(randn(6,2),hier_index,['A','B'])
```

In [91]:
```
df
```

Out[91]:

		A	B
G1	1	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

G1	**2**	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [92]:
```
df.loc['G1']
```

Out[92]:

	A	B
1	0.302665	1.693723
2	-1.706086	-1.159119
3	-0.134841	0.390528

In [ ]:

	1	0.302665	1.693723
G1	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [93]:
```
df.loc['G1'].loc[1]
```

Out[93]:
```
A    0.302665
B    1.693723
Name: 1, dtype: float64
```

In [ ]:

G1	**2**	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [96]:
```
df.index.names
```

Out[96]:
```
FrozenList([None, None])
```

In [ ]:

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

In [97]:
```
df.index.names = ['Groups','Num']
```

In [98]:
```
df
```

Out[98]:

		A	B
Groups	Num		
G1	1	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [ ]:

G1	**2**	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [99]:
```
df.loc['G2']
```

Out[99]:

	A	**B**
Num		
1	0.166905	0.184502
2	0.807706	0.072960
3	0.638787	0.329646

	1	0.302665	1.693723
G1	**2**	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [100]:
```
df.loc['G2'].loc[2]
```

Out[100]:
```
A    0.807706
B    0.072960
Name: 2, dtype: float64
```

In [ ]:

Groups	Num		
G1	1	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [101]:
```
df.loc['G2'].loc[2]['B']
```

Out[101]: 0.072959675317038689

In [ ]:

Groups	Num		
G1	1	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [101]:
```
df.xs
```

Out[101]:

```
Signature: df.xs(key, axis=0, level=None, copy=None, drop_level=True)
Docstring:
Returns a cross-section (row(s) or column(s)) from the
Series/DataFrame. Defaults to cross-section on the rows (axis=0).
```

In [ ]:

	3	0.638787	0.329646

In [102]:
```
df
```

Out[102]:

		A	B
Groups	**Num**		
G1	**1**	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [ ]:

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

	3	0.638787	0.329646

In [103]:
```
df.loc['G1']
```

Out[103]:

	A	B
Num		
1	0.302665	1.693723
2	-1.706086	-1.159119
3	-0.134841	0.390528

In [ ]:

G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [104]:
```
df.xs('G1')
```

Out[104]:

	A	B
Num		
1	0.302665	1.693723
2	-1.706086	-1.159119
3	-0.134841	0.390528

This is called “cross section”.

In [ ]:

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

In [98]:
```
df
```

Out[98]:

		A	B
Groups	**Num**		
G1	**1**	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [104]:
```
df.xs('G1')
```

Out[104]:

	A	B

Pandas

localhost:8888/notebooks/Pandas.ipynb

File Edit View Insert Cell Kernel Widgets Help | Python [conda env:py35]

In [98]:
```
df
```

Out[98]:

		A	B
Groups	**Num**		
G1	**1**	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [104]:
```
df.xs('G1')
```

Out[104]:

	A	B

	1	0.302665	1.693723
G1	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

In [105]:
```
df.xs(1,level='Num')
```

Out[105]:

	A	B
Groups		
G1	0.302665	1.693723
G2	0.166905	0.184502

In [ ]: