



MANIPULATING DATAFRAMES WITH PANDAS

Index objects and labeled data



pandas Data Structures

- Key building blocks
 - Indexes: Sequence of labels
 - Series: 1D array with Index
 - DataFrames: 2D array with Series as columns
- Indexes
 - Immutable (Like dictionary keys)
 - Homogenous in data type (Like NumPy arrays)



Creating a Series

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

```
In [2]: prices = [10.70, 10.86, 10.74, 10.71, 10.79]
```

```
In [3]: shares = pd.Series(prices)
```

```
In [4]: print(shares)
```

```
0    10.70
```

```
1    10.86
```

```
2    10.74
```

```
3    10.71
```

```
4    10.79
```

```
dtype: float64
```



Creating an index

```
In [5]: days = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri']
```

```
In [6]: shares = pd.Series(prices, index=days)
```

```
In [7]: print(shares)
```

```
Mon    10.70  
Tue    10.86  
Wed    10.74  
Thur    10.71  
Fri    10.79  
dtype: float64
```



Examining an index

```
In [8]: print(shares.index)
Index(['Mon', 'Tue', 'Wed', 'Thur', 'Fri'], dtype='object')
```

```
In [9]: print(shares.index[2])
Wed
```

```
In [10]: print(shares.index[:2])
Index(['Mon', 'Tue'], dtype='object')
```

```
In [11]: print(shares.index[-2:])
Index(['Thur', 'Fri'], dtype='object')
```

```
In [12]: print(shares.index.name)
None
```



Modifying index name

```
In [13]: shares.index.name = 'weekday'
```

```
In [14]: print(shares)
```

```
weekday
```

```
Monday      10.70
```

```
Tuesday     10.86
```

```
Wednesday   10.74
```

```
Thursday    10.71
```

```
Friday       10.79
```

```
dtype: float64
```



Modifying index entries

```
In [15]: shares.index[2] = 'Wednesday'
```

```
TypeError: Index does not support mutable operations
```

```
In [16]: shares.index[:4] = ['Monday', 'Tuesday',  
    ....:                    'Wednesday', 'Thursday']
```

```
TypeError: Index does not support mutable operations
```



Modifying all index entries

```
In [17]: shares.index = ['Monday', 'Tuesday', 'Wednesday',  
.....:                  'Thursday', 'Friday']
```

```
In [18]: print(shares)
```

Monday	10.70
Tuesday	10.86
Wednesday	10.74
Thursday	10.71
Friday	10.79

dtype: float64



Unemployment data

```
In [19]: unemployment = pd.read_csv('Unemployment.csv')
```

```
In [20]: unemployment.head()
```

```
Out[20]:
```

	Zip	unemployment	participants
0	1001	0.06	13801
1	1002	0.09	24551
2	1003	0.17	11477
3	1005	0.10	4086
4	1007	0.05	11362



Unemployment data

```
In [21]: unemployment.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33120 entries, 0 to 33119
Data columns (total 3 columns):
Zip                33120 non-null int64
unemployment       32556 non-null float64
participants       33120 non-null int64
dtypes: float64(1), int64(2)
memory usage: 776.3 KB
```



Assigning the index

```
In [22]: unemployment.index = unemployment['Zip']
```

```
In [23]: unemployment.head()
```

```
Out[23]:
```

	Zip	unemployment	participants
Zip			
1001	1001	0.06	13801
1002	1002	0.09	24551
1003	1003	0.17	11477
1005	1005	0.10	4086
1007	1007	0.05	11362



Removing extra column

```
In [24]: unemployment.head(3)
```

```
Out[24]:
```

	Zip	unemployment	participants
Zip			
1001	1001	0.06	13801
1002	1002	0.09	24551
1003	1003	0.17	11477

```
In [25]: del unemployment['Zip']
```

```
In [26]: unemployment.head(3)
```

```
Out[26]:
```

	unemployment	participants
Zip		
1001	0.06	13801
1002	0.09	24551
1003	0.17	11477



Examining index & columns

```
In [27]: print(unemployment.index)
Int64Index([1001, 1002, 1003, 1005, 1007, 1008, 1009, 1010, 1011, 1012,
          ...
          966, 968, 969, 971, 976, 979, 982, 983, 985, 987],
          dtype='int64', name='Zip', length=33120)
```

```
In [28]: print(unemployment.index.name)
Zip
```

```
In [29]: print(type(unemployment.index))
<class 'pandas.indexes.numeric.Int64Index'>
```

```
In [30]: print(unemployment.columns)
Index(['unemployment', 'participants'], dtype='object')
```



read_csv() with index_col

```
In [31]: unemployment = pd.read_csv('Unemployment.csv',  
    ....:                             index_col='Zip')
```

```
In [32]: unemployment.head()
```

```
Out[32]:
```

	unemployment	participants
Zip		
1001	0.06	13801
1002	0.09	24551
1003	0.17	11477
1005	0.10	4086
1007	0.05	11362



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Hierarchical Indexing



Stock data

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

```
In [2]: stocks = pd.read_csv('datasets/stocks.csv')
```

```
In [3]: print(stocks)
```

	Date	Close	Volume	Symbol
0	2016-10-03	31.50	14070500	CSCO
1	2016-10-03	112.52	21701800	AAPL
2	2016-10-03	57.42	19189500	MSFT
3	2016-10-04	113.00	29736800	AAPL
4	2016-10-04	57.24	20085900	MSFT
5	2016-10-04	31.35	18460400	CSCO
6	2016-10-05	57.64	16726400	MSFT
7	2016-10-05	31.59	11808600	CSCO
8	2016-10-05	113.05	21453100	AAPL

Repeated
values

Repeated
values



Setting index

```
In [4]: stocks = stocks.set_index(['Symbol', 'Date'])
```

```
In [5]: print(stocks)
```

		Close	Volume
Symbol	Date		
CSCO	2016-10-03	31.50	14070500
AAPL	2016-10-03	112.52	21701800
MSFT	2016-10-03	57.42	19189500
AAPL	2016-10-04	113.00	29736800
MSFT	2016-10-04	57.24	20085900
CSCO	2016-10-04	31.35	18460400
MSFT	2016-10-05	57.64	16726400
CSCO	2016-10-05	31.59	11808600
AAPL	2016-10-05	113.05	21453100



MultiIndex on DataFrame

```
In [6]: print(stocks.index)
MultiIndex(levels=[[ 'AAPL', 'CSCO', 'MSFT'], ['2016-10-03',
'2016-10-04', '2016-10-05']], labels=[[1, 0, 2, 0, 2, 1, 2, 1, 0],
[0, 0, 0, 1, 1, 1, 2, 2, 2]], names=['Symbol', 'Date'])
```

```
In [7]: print(stocks.index.name)
None
```

```
In [8]: print(stocks.index.names)
['Symbol', 'Date']
```



Sorting index

```
In [9]: stocks = stocks.sort_index()
```

```
In [10]: print(stocks)
```

		Close	Volume
AAPL	2016-10-03	112.52	21701800
	2016-10-04	113.00	29736800
	2016-10-05	113.05	21453100
CSCO	2016-10-03	31.50	14070500
	2016-10-04	31.35	18460400
	2016-10-05	31.59	11808600
MSFT	2016-10-03	57.42	19189500
	2016-10-04	57.24	20085900
	2016-10-05	57.64	16726400



Indexing (individual row)

```
In [11]: stocks.loc[('CSCO', '2016-10-04')]
```

```
Out[11]:
```

```
Close          31.35
```

```
Volume    18460400.00
```

```
Name: (CSCO, 2016-10-04), dtype: float64
```

```
In [12]: stocks.loc[('CSCO', '2016-10-04'), 'Volume']
```

```
Out[12]: 18460400.0
```



Slicing (outermost index)

```
In [13]: stocks.loc['AAPL']
```

```
Out[13]:
```

	Close	Volume
Date		
2016-10-03	112.52	21701800
2016-10-04	113.00	29736800
2016-10-05	113.05	21453100



Slicing (outermost index)

```
In [14]: stocks.loc['CSCO':'MSFT']
```

```
Out[14]:
```

		Close	Volume
CSCO	2016-10-03	31.50	14070500
	2016-10-04	31.35	18460400
	2016-10-05	31.59	11808600
MSFT	2016-10-03	57.42	19189500
	2016-10-04	57.24	20085900
	2016-10-05	57.64	16726400



Fancy indexing (outermost index)

```
In [15]: stocks.loc(['AAPL', 'MSFT'], '2016-10-05'), :]
```

```
Out[15]:
```

		Close	Volume
Symbol	Date		
AAPL	2016-10-05	113.05	21453100
MSFT	2016-10-05	57.64	16726400

```
In [16]: stocks.loc(['AAPL', 'MSFT'], '2016-10-05'), 'Close']
```

```
Out[16]:
```

Symbol	Date	
AAPL	2016-10-05	113.05
MSFT	2016-10-05	57.64

Name: Close, dtype: float64



Fancy indexing (innermost index)

```
In [17]: stocks.loc[('CSCO', ['2016-10-05', '2016-10-03']), :]  
Out[17]:
```

		Close	Volume
Symbol	Date		
CSCO	2016-10-03	31.50	14070500
	2016-10-05	31.59	11808600



Slicing (both indexes)

```
In [18]: stocks.loc[(slice(None), slice('2016-10-03', '2016-10-04')),:]  
Out[18]:
```

		Close	Volume
AAPL	2016-10-03	112.52	21701800
	2016-10-04	113.00	29736800
CSCO	2016-10-03	31.50	14070500
	2016-10-04	31.35	18460400
MSFT	2016-10-03	57.42	19189500
	2016-10-04	57.24	20085900



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!