

Indexing time series

PANDAS FOUNDATIONS



Dhavide Aruliah

Director of Training, Anaconda

Using pandas to read datetime objects

- `read_csv()` function
 - Can read strings into datetime objects
 - Need to specify `'parse_dates=True'`
- ISO 8601 format
 - `yyyy-mm-dd hh:mm:ss`

Product sales CSV

	Date	Company	Product	Units
0	2015-02-02 08:30:00	Hooli	Software	3
1	2015-02-02 21:00:00	Mediacore	Hardware	9
2	2015-02-03 14:00:00	Initech	Software	13
3	2015-02-04 15:30:00	Streeplex	Software	13
4	2015-02-04 22:00:00	Acme Corporation	Hardware	14

Parse dates

```
import pandas as pd
sales = pd.read_csv('sales-feb-2015.csv',
                    parse_dates=True, index_col= 'Date')
```

Parse dates

```
sales.head()
```

Date	Company	Product	Units
2015-02-02 08:30:00	Hooli	Software	3
2015-02-02 21:00:00	Mediacore	Hardware	9
2015-02-03 14:00:00	Initech	Software	13
2015-02-04 15:30:00	Streeplex	Software	13
2015-02-04 22:00:00	Acme Coporation	Hardware	14

Parse dates

```
sales.info()
```

```
DatetimeIndex: 19 entries, 2015-02-02 08:30:00 to 2015-02-26 09:00:00  
Data columns (total 3 columns):  
Company      19 non-null object  
Product      19 non-null object  
Units        19 non-null int64  
dtypes: int64(1), object(2)  
memory usage: 608.0+ bytes
```

Selecting single datetime

```
sales.loc['2015-02-19 11:00:00', 'Company']
```

```
'Mediacore'
```

Selecting whole day

```
sales.loc['2015-2-5']
```

Date		Company	Product	Units
2015-02-05 02:00:00	Acme Coporation	Software	19	
2015-02-05 22:00:00	Hooli	Service	10	

Partial datetime string selection

- Alternative formats:
 - `sales.loc['February 5, 2015']`
 - `sales.loc['2015-Feb-5']`
- Whole month: `sales.loc['2015-2']`
- Whole year: `sales.loc['2015']`

Selecting whole month

```
sales.loc['2015-2']
```

Date	Company	Product	Units
2015-02-02 08:30:00	Hooli	Software	3
2015-02-02 21:00:00	Mediacore	Hardware	9
2015-02-03 14:00:00	Initech	Software	13
2015-02-04 15:30:00	Streeplex	Software	13
2015-02-04 22:00:00	Acme Coporation	Hardware	14
2015-02-05 02:00:00	Acme Coporation	Software	19
2015-02-05 22:00:00	Hooli	Service	10
2015-02-07 23:00:00	Acme Coporation	Hardware	1
2015-02-09 09:00:00	Streeplex	Service	19
2015-02-09 13:00:00	Mediacore	Software	7
2015-02-11 20:00:00	Initech	Software	7
2015-02-11 23:00:00	Hooli	Software	4
2015-02-16 12:00:00	Hooli	Software	10
2015-02-19 11:00:00	Mediacore	Hardware	16
...			

Slicing using dates/times

```
sales.loc['2015-2-16':'2015-2-20']
```

Date		Company	Product	Units
2015-02-16 12:00:00		Hooli	Software	10
2015-02-19 11:00:00		Mediacore	Hardware	16
2015-02-19 16:00:00		Mediacore	Service	10

Convert strings to datetime

```
evening_2_11 = pd.to_datetime(['2015-2-11 20:00',  
                               '2015-2-11 21:00', '2015-2-11 22:00', '2015-2-11 23:00'])  
evening_2_11
```

```
DatetimeIndex(['2015-02-11 20:00:00', '2015-02-11 21:00:00',  
              '2015-02-11 22:00:00', '2015-02-11 23:00:00'],  
              dtype='datetime64[ns]', freq=None)
```

Reindexing DataFrame

```
sales.reindex(evening_2_11)
```

		Company	Product	Units
2015-02-11	20:00:00	Initech	Software	7.0
2015-02-11	21:00:00	NaN	NaN	NaN
2015-02-11	22:00:00	NaN	NaN	NaN
2015-02-11	23:00:00	Hooli	Software	4.0

Filling missing values

```
sales.reindex(evening_2_11, method='ffill')
```

		Company	Product	Units
2015-02-11	20:00:00	Initech	Software	7
2015-02-11	21:00:00	Initech	Software	7
2015-02-11	22:00:00	Initech	Software	7
2015-02-11	23:00:00	Hooli	Software	4

```
sales.reindex(evening_2_11, method='bfill')
```

		Company	Product	Units
2015-02-11	20:00:00	Initech	Software	7
2015-02-11	21:00:00	Hooli	Software	4
2015-02-11	22:00:00	Hooli	Software	4
2015-02-11	23:00:00	Hooli	Software	4

Let's practice!
PANDAS FOUNDATIONS

Resampling time series data

PANDAS FOUNDATIONS



Dhavide Aruliah

Director of Training, Anaconda

Sales data

```
import pandas as pd
sales = pd.read_csv('sales-feb-2015.csv',
                    parse_dates=True, index_col= 'Date')
sales.head()
```

Date	Company	Product	Units
2015-02-02 08:30:00	Hooli	Software	3
2015-02-02 21:00:00	Mediacore	Hardware	9
2015-02-03 14:00:00	Initech	Software	13
2015-02-04 15:30:00	Streeplex	Software	13
2015-02-04 22:00:00	Acme Coporation	Hardware	14

Resampling

- Statistical methods over different time intervals
 - `mean()`, `sum()`, `count()`, etc.
- Downsampling
 - reduce datetime rows to slower frequency
- Upsampling
 - increase datetime rows to faster frequency

Aggregating means

```
daily_mean = sales.resample('D').mean()  
daily_mean
```

Date	Units
2015-02-02	6.0
2015-02-03	13.0
2015-02-04	13.5
2015-02-05	14.5
2015-02-06	NaN
2015-02-07	1.0
2015-02-08	NaN
2015-02-09	13.0
2015-02-10	NaN
2015-02-11	5.5
2015-02-12	NaN
2015-02-13	NaN
2015-02-14	NaN

Verifying

```
print(daily_mean.loc['2015-2-2'])
```

```
Units      6.0  
Name: 2015-02-02 00:00:00, dtype: float64
```

```
print(sales.loc['2015-2-2', 'Units'])
```

```
Date  
2015-02-02 08:30:00      3  
2015-02-02 21:00:00      9  
Name: Units, dtype: int64
```

```
sales.loc['2015-2-2', 'Units'].mean()
```

```
6.0
```

Method chaining

```
sales.resample('D').sum()
```

Date	Units
2015-02-02	6.0
2015-02-03	13.0
2015-02-04	13.5
2015-02-05	14.5
2015-02-06	NaN
2015-02-07	1.0
2015-02-08	NaN
2015-02-09	13.0
2015-02-10	NaN
2015-02-11	5.5
2015-02-12	NaN
2015-02-13	NaN

Method chaining

```
sales.resample('D').sum().max()
```

```
Units      29.0  
dtype: float64
```

Resampling strings

```
sales.resample('W').count()
```

	Company	Product	Units
Date			
2015-02-08	8	8	8
2015-02-15	4	4	4
2015-02-22	5	5	5
2015-03-01	2	2	2

Resampling frequencies

Input	Description
'min', 'T'	minute
'H'	hour
'D'	day
'B'	business day
'W'	week
'M'	month
'Q'	quarter
'A'	year

Multiplying frequencies

```
sales.loc[:, 'Units'].resample('2W').sum()
```

```
Date
2015-02-08    82
2015-02-22    79
2015-03-08    14
Freq: 2W-SUN, Name: Units, dtype: int64
```

Upsampling

```
two_days = sales.loc['2015-2-4': '2015-2-5', 'Units']  
two_days
```

```
Date  
2015-02-04 15:30:00    13  
2015-02-04 22:00:00    14  
2015-02-05 02:00:00    19  
2015-02-05 22:00:00    10  
Name: Units, dtype: int64
```

Upsampling and filling

```
two_days.resample('4H').ffill()
```

```
Date
Date
2015-02-04 12:00:00      NaN
2015-02-04 16:00:00    13.0
2015-02-04 20:00:00    13.0
2015-02-05 00:00:00    14.0
2015-02-05 04:00:00    19.0
2015-02-05 08:00:00    19.0
2015-02-05 12:00:00    19.0
2015-02-05 16:00:00    19.0
2015-02-05 20:00:00    19.0
Freq: 4H, Name: Units, dtype: float64
```

Let's practice!
PANDAS FOUNDATIONS

Manipulating time series data

PANDAS FOUNDATIONS



Dhavide Aruliah

Director of Training, Anaconda

Sales data

```
import pandas as pd
sales = pd.read_csv('sales-feb-2015.csv',
                    parse_dates=['Date'])

sales.head()
```

	Date	Company	Product	Units
0	2015-02-02 08:30:00	Hooli	Software	3
1	2015-02-02 21:00:00	Mediacore	Hardware	9
2	2015-02-03 14:00:00	Initech	Software	13
3	2015-02-04 15:30:00	Streeplex	Software	13
4	2015-02-04 22:00:00	Acme Coporation	Hardware	14

String methods

```
sales['Company'].str.upper()
```

```
0          HOOLI
1        MEDIACORE
2         INITECH
3        STREEPLEX
4    ACME COPORATION
5    ACME COPORATION
6          HOOLI
7    ACME COPORATION
8        STREEPLEX
9        MEDIACORE
10         INITECH
11          HOOLI
12          HOOLI
13        MEDIACORE
14        MEDIACORE
15        MEDIACORE
...
```

Substring matching

```
sales['Product'].str.contains('ware')
```

```
0      True
1      True
2      True
3      True
4      True
5      True
6     False
7      True
8     False
9      True
10     True
11     True
12     True
13     True
14     False
...
```


Boolean arithmetic

True + False

1

True + True

2

False + False

0

Boolean reduction

```
sales['Product'].str.contains('ware').sum()
```

14

Datetime methods

```
sales['Date'].dt.hour
```

```
0      8
1     21
2     14
3     15
4     22
5      2
6     22
7     23
8      9
9     13
10    20
11    23
12    12
13    11
14    16
...
```

Set timezone

```
central = sales['Date'].dt.tz_localize('US/Central')
central
```

```
0    2015-02-02 08:30:00-06:00
1    2015-02-02 21:00:00-06:00
2    2015-02-03 14:00:00-06:00
3    2015-02-04 15:30:00-06:00
4    2015-02-04 22:00:00-06:00
5    2015-02-05 02:00:00-06:00
6    2015-02-05 22:00:00-06:00
7    2015-02-07 23:00:00-06:00
8    2015-02-09 09:00:00-06:00
9    2015-02-09 13:00:00-06:00
10   2015-02-11 20:00:00-06:00
11   2015-02-11 23:00:00-06:00
12   2015-02-16 12:00:00-06:00
...
Name: Date, dtype: datetime64[ns, US/Central]
```

Convert timezone

```
central.dt.tz_convert('US/Eastern')
```

```
0    2015-02-02 09:30:00-05:00
1    2015-02-02 22:00:00-05:00
2    2015-02-03 15:00:00-05:00
3    2015-02-04 16:30:00-05:00
4    2015-02-04 23:00:00-05:00
5    2015-02-05 03:00:00-05:00
6    2015-02-05 23:00:00-05:00
7    2015-02-08 00:00:00-05:00
8    2015-02-09 10:00:00-05:00
9    2015-02-09 14:00:00-05:00
10   2015-02-11 21:00:00-05:00
11   2015-02-12 00:00:00-05:00
12   2015-02-16 13:00:00-05:00
13   2015-02-19 12:00:00-05:00
14   2015-02-19 17:00:00-05:00
...
Name: Date, dtype: datetime64[ns, US/Eastern]
```

Method chaining

```
sales['Date'].dt.tz_localize('US/Central').  
    dt.tz_convert('US/Eastern')
```

```
0    2015-02-02 09:30:00-05:00  
1    2015-02-02 22:00:00-05:00  
2    2015-02-03 15:00:00-05:00  
3    2015-02-04 16:30:00-05:00  
4    2015-02-04 23:00:00-05:00  
5    2015-02-05 03:00:00-05:00  
6    2015-02-05 23:00:00-05:00  
7    2015-02-08 00:00:00-05:00  
8    2015-02-09 10:00:00-05:00  
9    2015-02-09 14:00:00-05:00  
10   2015-02-11 21:00:00-05:00  
11   2015-02-12 00:00:00-05:00  
12   2015-02-16 13:00:00-05:00  
13   2015-02-19 12:00:00-05:00  
14   2015-02-19 17:00:00-05:00  
...  
Name: Date, dtype: datetime64[ns, US/Eastern]
```

World Population

```
population = pd.read_csv('world_population.csv',  
                          parse_dates=True, index_col= 'Date')  
population
```

	Population
Date	
1960-12-31	2.087485e+10
1970-12-31	2.536513e+10
1980-12-31	3.057186e+10
1990-12-31	3.644928e+10
2000-12-31	4.228550e+10
2010-12-31	4.802217e+10

Upsample population

```
population.resample('A').first()
```

Date	Population
1960-12-31	2.087485e+10
1961-12-31	NaN
1962-12-31	NaN
1963-12-31	NaN
1964-12-31	NaN
1965-12-31	NaN
1966-12-31	NaN
1967-12-31	NaN
1968-12-31	NaN
1969-12-31	NaN
1970-12-31	2.536513e+10
1971-12-31	NaN
1972-12-31	NaN

Interpolate missing data

```
population.resample('A').first().interpolate('linear')
```

Date	Population
1960-12-31	2.087485e+10
1961-12-31	2.132388e+10
1962-12-31	2.177290e+10
1963-12-31	2.222193e+10
1964-12-31	2.267096e+10
1965-12-31	2.311999e+10
1966-12-31	2.356902e+10
1967-12-31	2.401805e+10
1968-12-31	2.446707e+10
1969-12-31	2.491610e+10
1970-12-31	2.536513e+10
1971-12-31	2.588580e+10
1972-12-31	2.640648e+10

Let's practice!
PANDAS FOUNDATIONS

Time series visualization

PANDAS FOUNDATIONS



Dhavide Aruliah

Director of Training, Anaconda

Topics

- Line types
- Plot types
- Subplots

S&P 500 Data

```
import pandas as pd
import matplotlib.pyplot as plt
sp500 = pd.read_csv('sp500.csv', parse_dates=True,
                    index_col= 'Date')

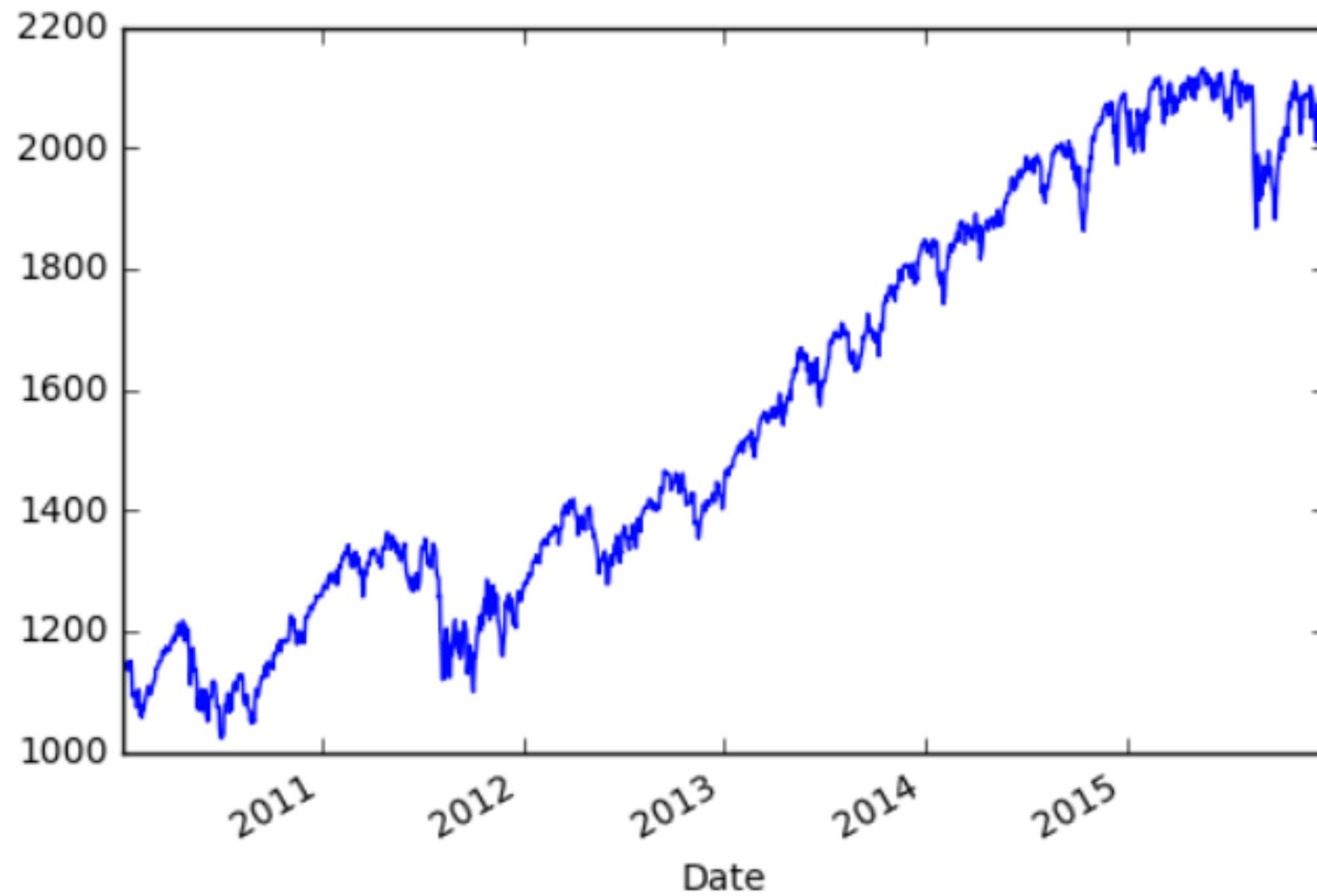
sp500.head()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2010-01-04	1116.560059	1133.869995	1116.560059	1132.989990	3991400000	1132.989990
2010-01-05	1132.660034	1136.630005	1129.660034	1136.520020	2491020000	1136.520020
2010-01-06	1135.709961	1139.189941	1133.949951	1137.140015	4972660000	1137.140015
2010-01-07	1136.270020	1142.459961	1131.319946	1141.689941	5270680000	1141.689941
2010-01-08	1140.520020	1145.390015	1136.219971	1144.979980	4389590000	1144.979980

Pandas plot

```
sp500['Close'].plot()  
plt.show()
```

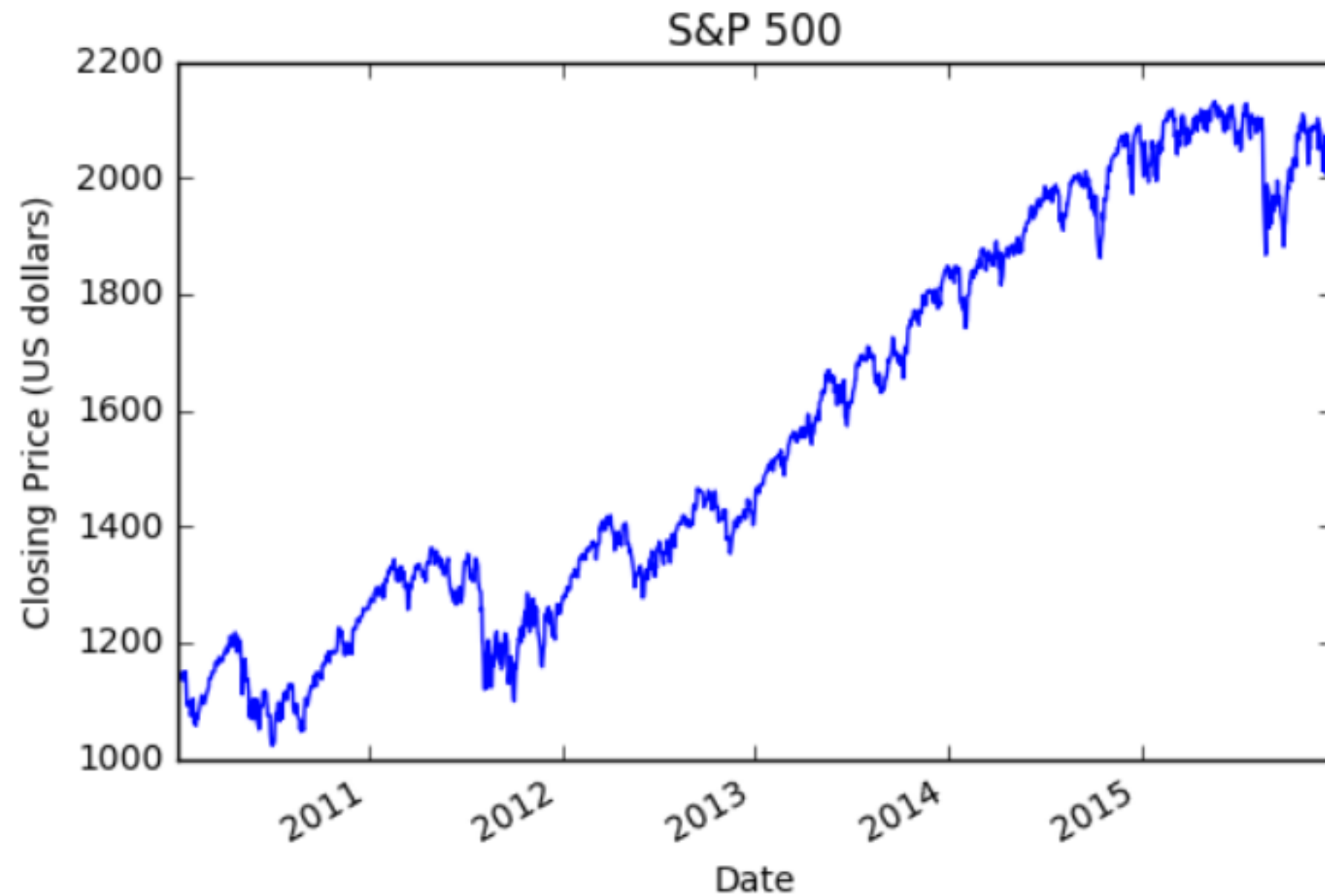
Default plot



Labels and title

```
sp500['Close'].plot(title='S&P 500')  
plt.ylabel('Closing Price (US Dollars)')  
plt.show()
```

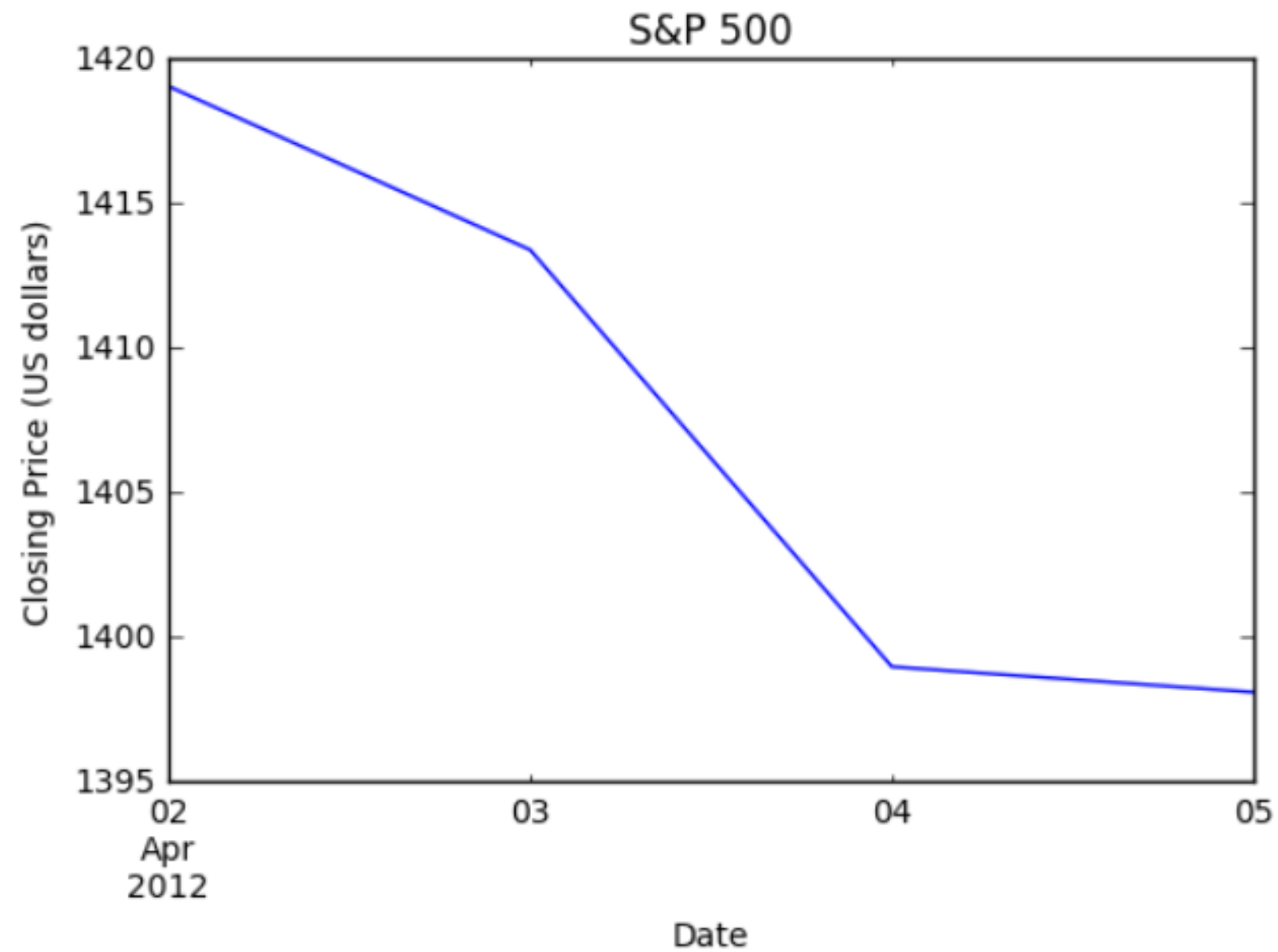

Labels and title



One week

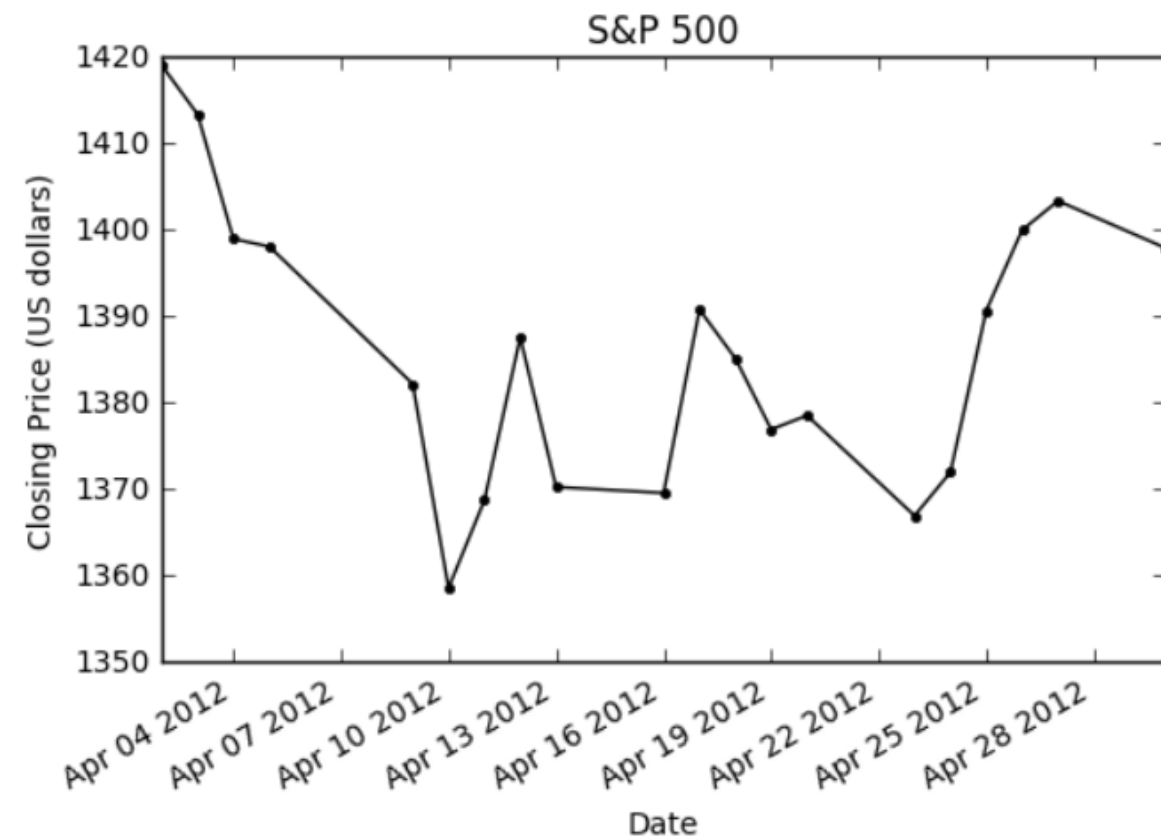
```
sp500.loc['2012-4-1':'2012-4-7', 'Close'].plot(title='S&P  
500')  
plt.ylabel('Closing Price (US Dollars)')  
plt.show()
```

One week



Plot styles

```
sp500.loc['2012-4', 'Close'].plot(style='k.-',  
                                  title='S&P500')  
  
plt.ylabel('Closing Price (US Dollars)')  
plt.show()
```



More plot styles

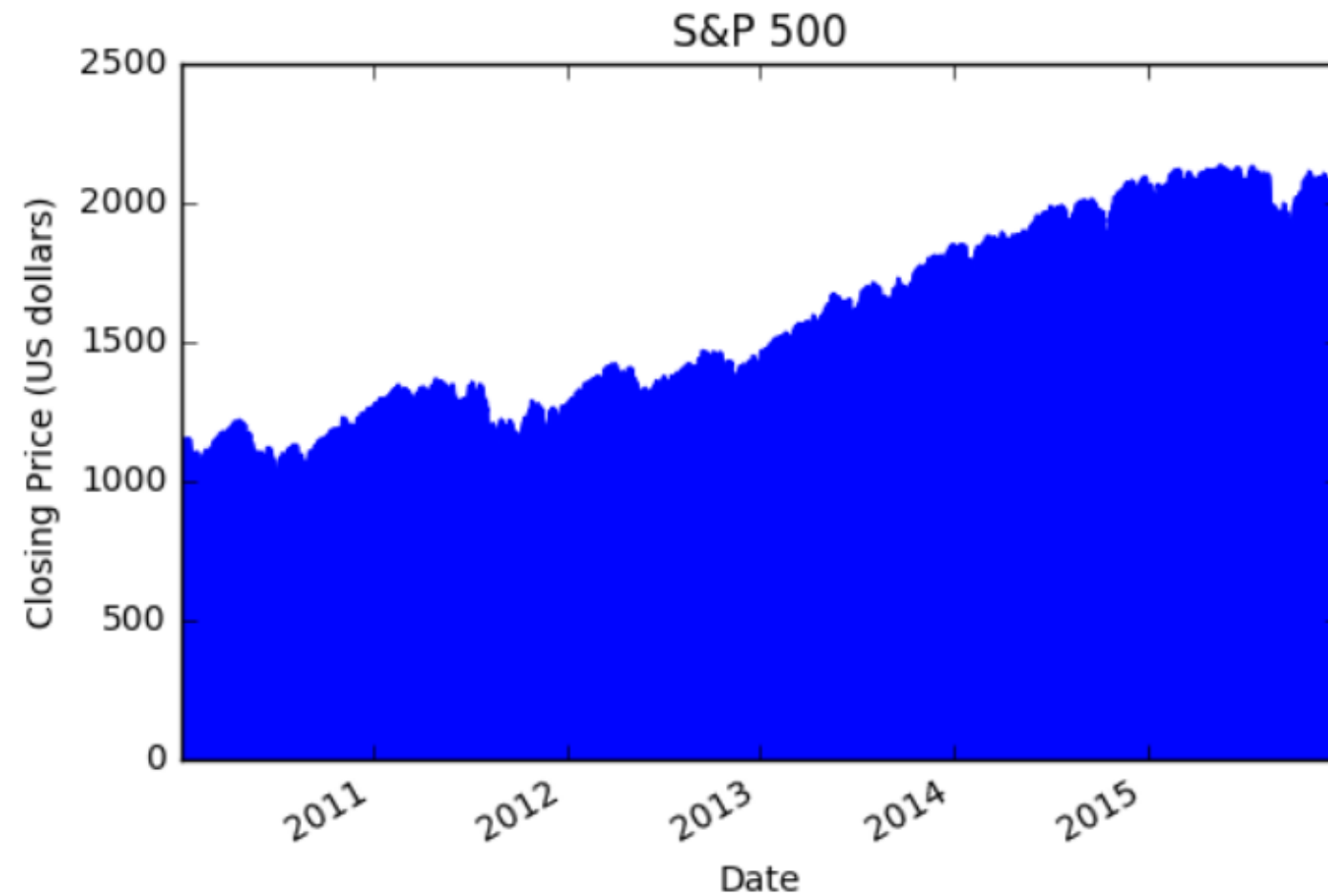
- Style format string
 - color (k: black)
 - marker (. : dot)
 - line type (-: solid)

More plot styles

Color	Marker	Line
b: blue	o: circle	: dotted
g: green	*: star	-: dashed
r: red	s: square	
c: cyan	+: plus	

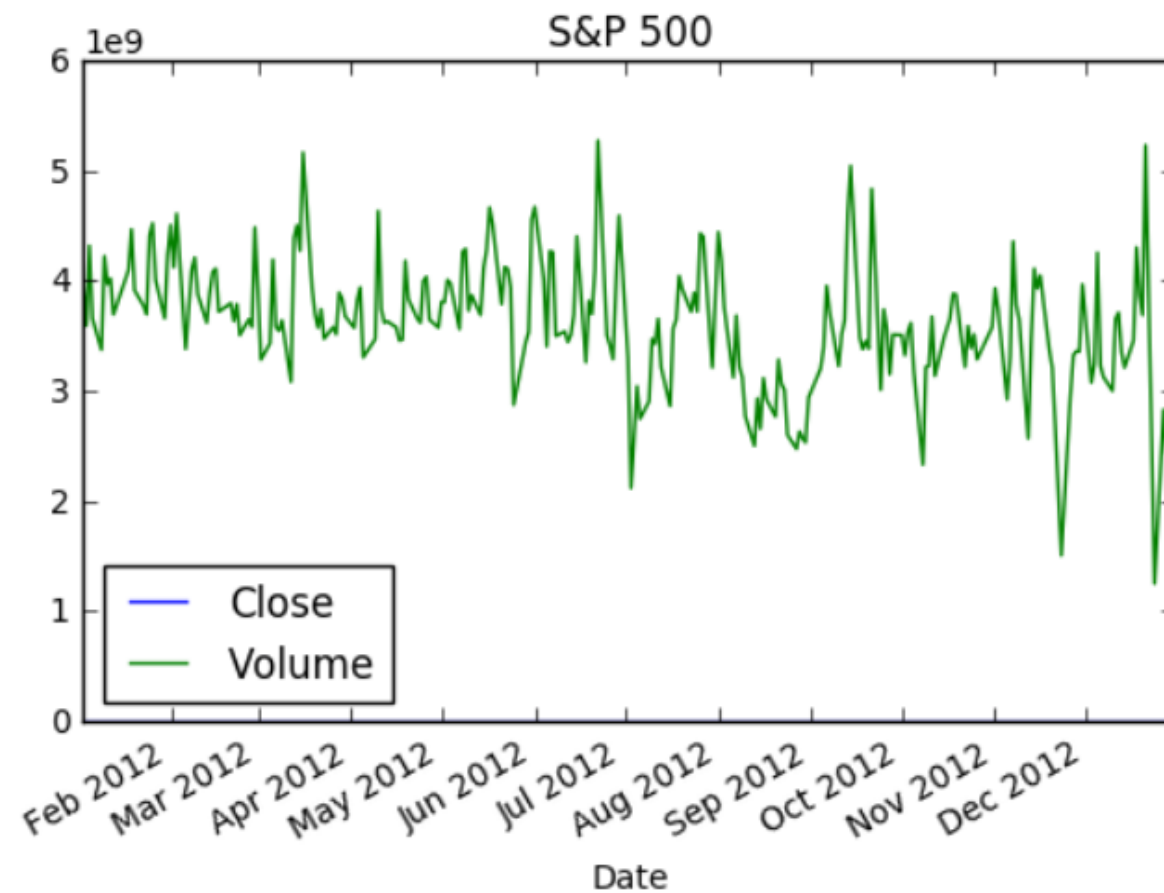
Area plot

```
sp500['Close'].plot(kind='area', title='S&P 500')  
plt.ylabel('Closing Price (US Dollars)')  
plt.show()
```



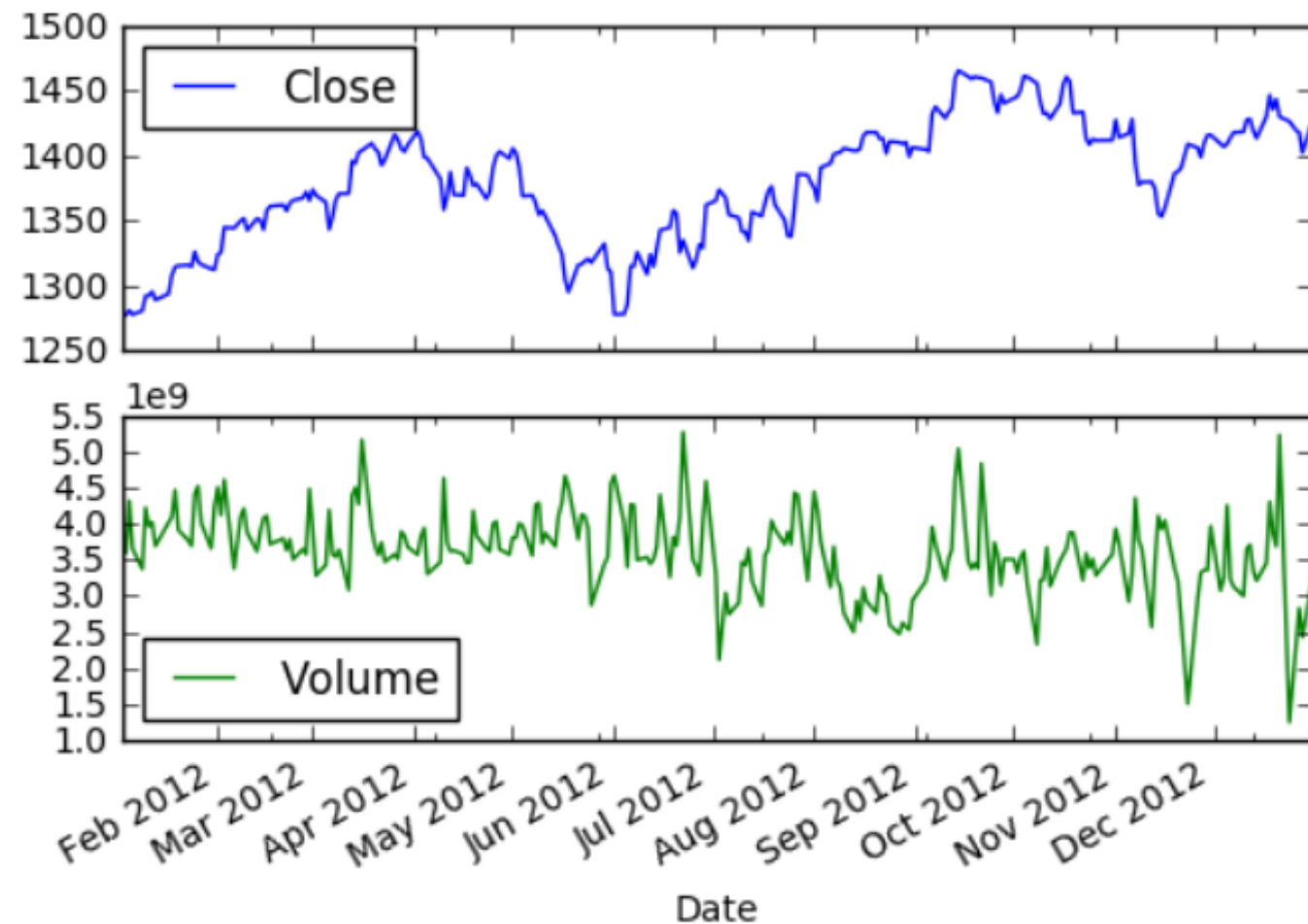
Multiple columns

```
sp500.loc['2012', ['Close', 'Volume']].plot(title='S&P  
500')  
plt.show()
```



Subplots

```
sp500.loc['2012', ['Close', 'Volume']].plot(subplots=True)  
plt.show()
```



Let's practice!
PANDAS FOUNDATIONS