





PANDAS FOUNDATIONS

Indexing time series



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 Coporation	Hardware	14



Parse dates

```
In [3]: sales.head()
```

```
Out[3]:
```

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

```
In [4]: 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

```
In [5]: sales.loc['2015-02-19 11:00:00', 'Company']  
Out[5]: 'Mediacore'
```



Selecting whole day

```
In [6]: sales.loc['2015-2-5']
```

```
Out[6]:
```

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

```
In [7]: sales.loc['2015-2']  
Out[7]:
```

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

```
In [8]: sales.loc['2015-2-16':'2015-2-20']
```

```
Out[8]:
```

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

```
In [9]: 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'])
```

```
In [10]: evening_2_11
```

```
Out[10]:
```

```
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

```
In [11]: sales.reindex(evening_2_11)
```

```
Out[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



```
In [12]: sales.reindex(evening_2_11, method='ffill')  
Out[12]:
```

		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

```
In [13]: sales.reindex(evening_2_11, method='bfill')  
Out[13]:
```

		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



PANDAS FOUNDATIONS

Let's practice!



PANDAS FOUNDATIONS

Resampling time series data



Sales data

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

```
In [2]: sales = pd.read_csv('sales-feb-2015.csv',  
    ....:                   parse_dates=True, index_col= 'Date')
```

```
In [3]: sales.head()
```

```
Out[3]:
```

		Company	Product	Units
Date				
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.
- Down-sampling
 - reduce datetime rows to slower frequency
- Up-sampling
 - increase datetime rows to faster frequency



Aggregating means

```
In [4]: daily_mean = sales.resample('D').mean()
```



```
In [5]: daily_mean
```

```
Out[5]:
```

	Units
Date	
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

```
In [6]: print(daily_mean.loc['2015-2-2'])  
Units      6.0  
Name: 2015-02-02 00:00:00, dtype: float64
```



```
In [7]: 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
```

```
In [8]: sales.loc['2015-2-2', 'Units'].mean()  
Out[8]: 6.0
```



Method chaining

```
In [9]: sales.resample('D').sum()
```

```
Out[9]:
```

	Units
Date	
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

```
In [10]: sales.resample('D').sum().max()  
Out[10]:  
Units      29.0  
dtype: float64
```



Resampling strings

```
In [11]: sales.resample('W').count()  
Out[11]:
```

	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

```
In [12]: sales.loc[:, 'Units'].resample('2W').sum()  
Out[12]:  
Date  
2015-02-08      82  
2015-02-22      79  
2015-03-08      14  
Freq: 2W-SUN, Name: Units, dtype: int64
```



Upsampling

```
In [13]: two_days = sales.loc['2015-2-4': '2015-2-5', 'Units']
```

```
In [13]: two_days
```

```
Out[13]:
```

```
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

```
In [14]: two_days.resample('4H').ffill()
```

```
Out[14]:
```

```
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
```



PANDAS FOUNDATIONS

Let's practice!



PANDAS FOUNDATIONS

Manipulating time series data



Sales data

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

```
In [2]: sales = pd.read_csv('sales-feb-2015.csv',  
....:                      parse_dates=['Date'])
```

```
In [3]: sales.head()
```

```
Out[3]:
```

	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

```
In [4]: sales['Company'].str.upper()
```

```
Out[4]:
```

```
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

```
In [5]: sales['Product'].str.contains('ware')
```

```
Out[5]:
```

```
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

```
In [6]: True + False  
Out[6]: 1
```

```
In [7]: True + True  
Out[7]: 2
```

```
In [8]: False + False  
Out[8]: 0
```



Boolean reduction

```
In [9]: sales['Product'].str.contains('ware').sum()  
Out[9]: 14
```





Datetime methods

```
In [9]: sales['Date'].dt.hour
```

```
Out[9]:
```

```
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

```
In [10]: central = sales['Date'].dt.tz_localize('US/Central')
```

```
In [11]: central
```

```
Out[11]:
```

```
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

```
In [12]: central.dt.tz_convert('US/Eastern')
```

```
Out[12]:
```

```
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

```
In [13]: sales['Date'].dt.tz_localize('US/Central').  
        ...: dt.tz_convert('US/Eastern')
```

```
Out[13]:
```

```
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

```
In [14]: population = pd.read_csv('world_population.csv',  
    ...: parse_dates=True, index_col= 'Date')
```

```
In [15]: population
```

```
Out[15]:
```

Date	Population
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

```
In [16]: population.resample('A').first()  
Out[16]:
```

	Population
Date	
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

```
In [17]: population.resample('A').first().interpolate('linear')  
Out[17]:
```

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



PANDAS FOUNDATIONS

Let's practice!



PANDAS FOUNDATIONS

Time series visualization



Topics

- Line types
- Plot types
- Subplots



S&P 500 Data

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

```
In [2]: import matplotlib.pyplot as plt
```

```
In [3]: sp500 = pd.read_csv('sp500.csv', parse_dates=True,  
....:                      index_col= 'Date')
```

```
In [4]: sp500.head()
```

```
Out[4]:
```

	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



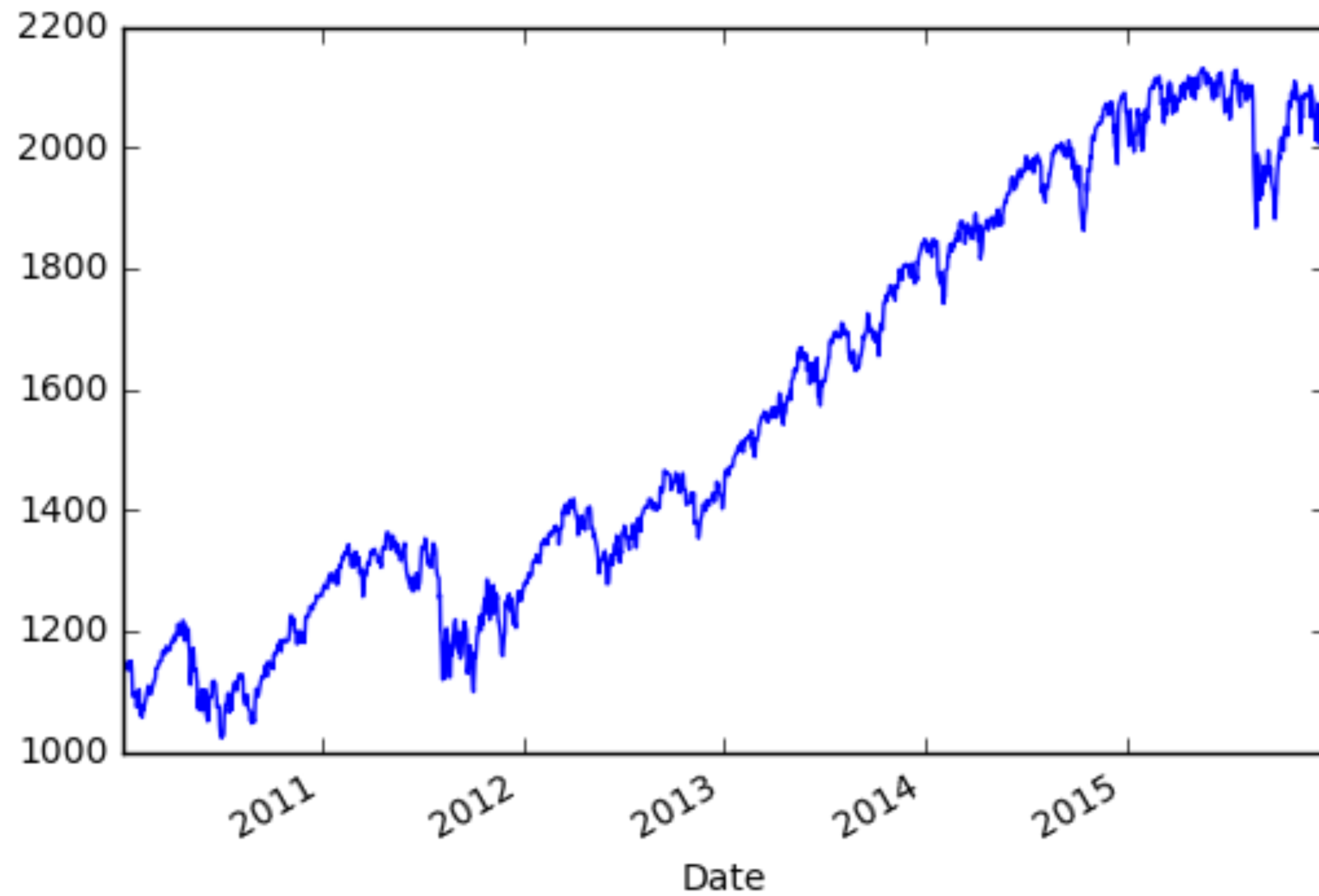
Pandas plot

```
In [5]: sp500['Close'].plot()
```

```
In [6]: plt.show()
```



Default plot





Labels and title

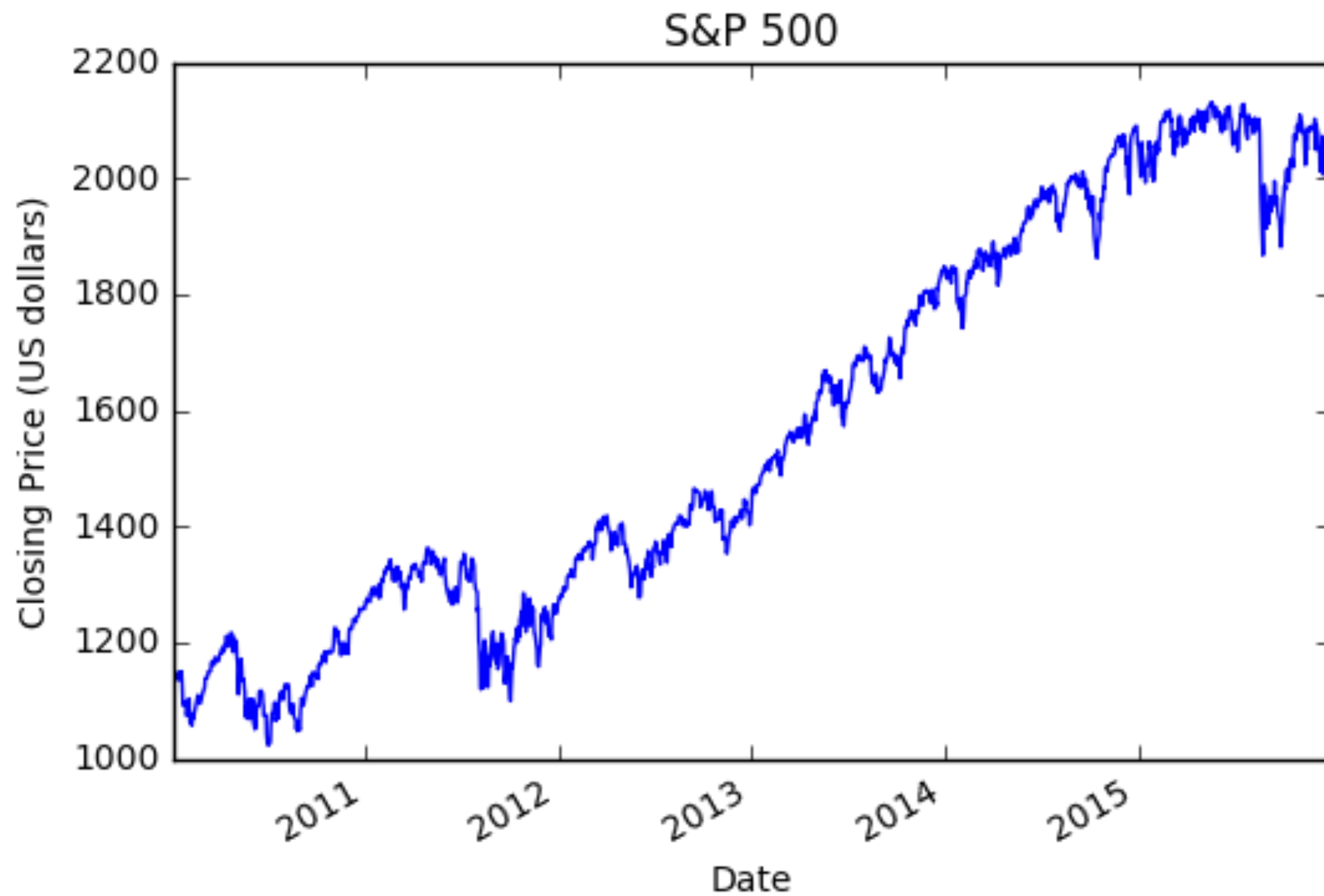
```
In [7]: sp500['Close'].plot(title='S&P 500')
```

```
In [8]: plt.ylabel('Closing Price (US Dollars)')
```

```
In [9]: plt.show()
```




Labels and title





One week

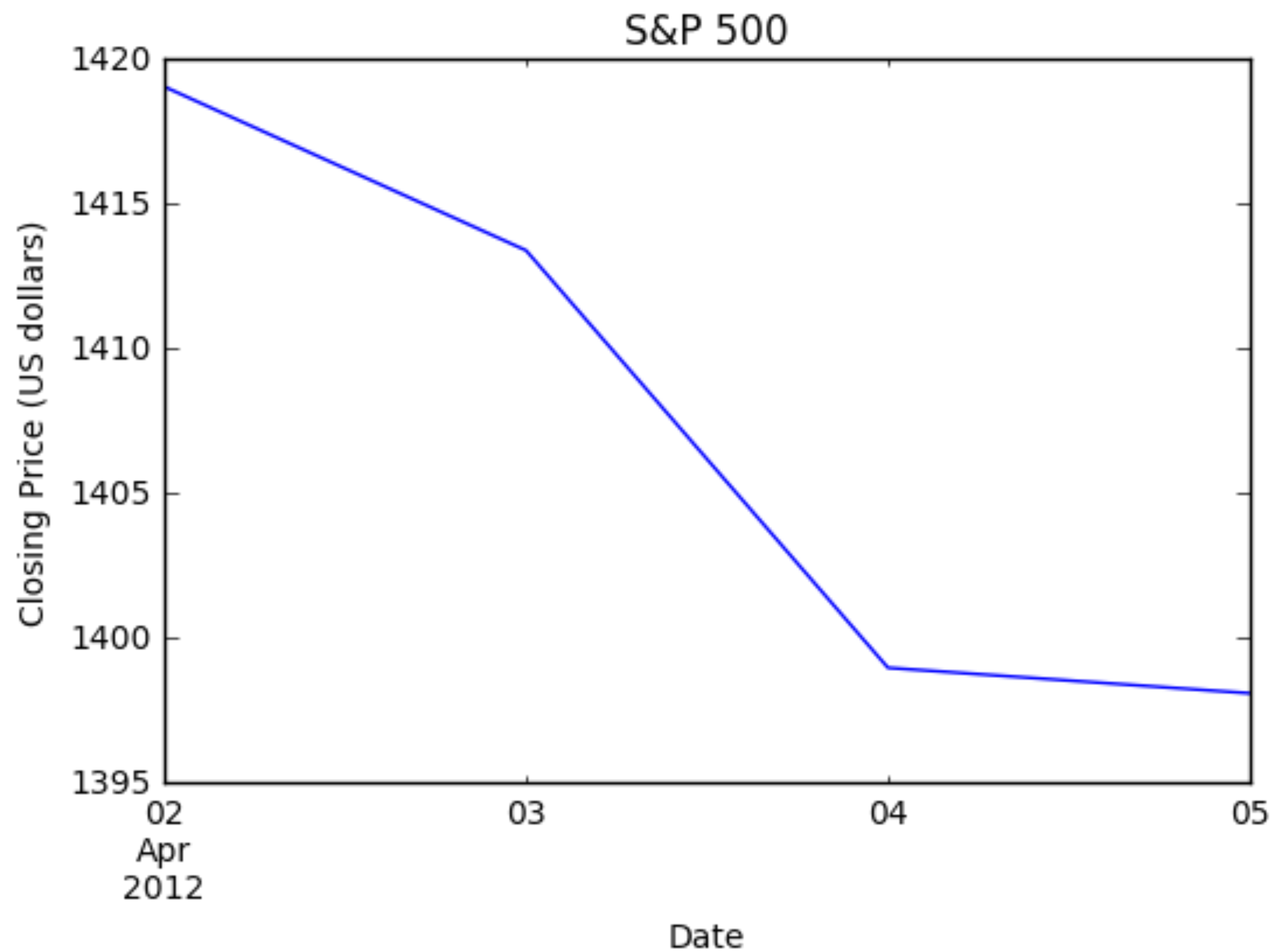
```
In [10]: sp500.loc['2012-4-1':'2012-4-7', 'Close'].plot(title='S&P  
...: 500')
```

```
In [11]: plt.ylabel('Closing Price (US Dollars)')
```

```
In [12]: plt.show()
```



One week





Plot styles

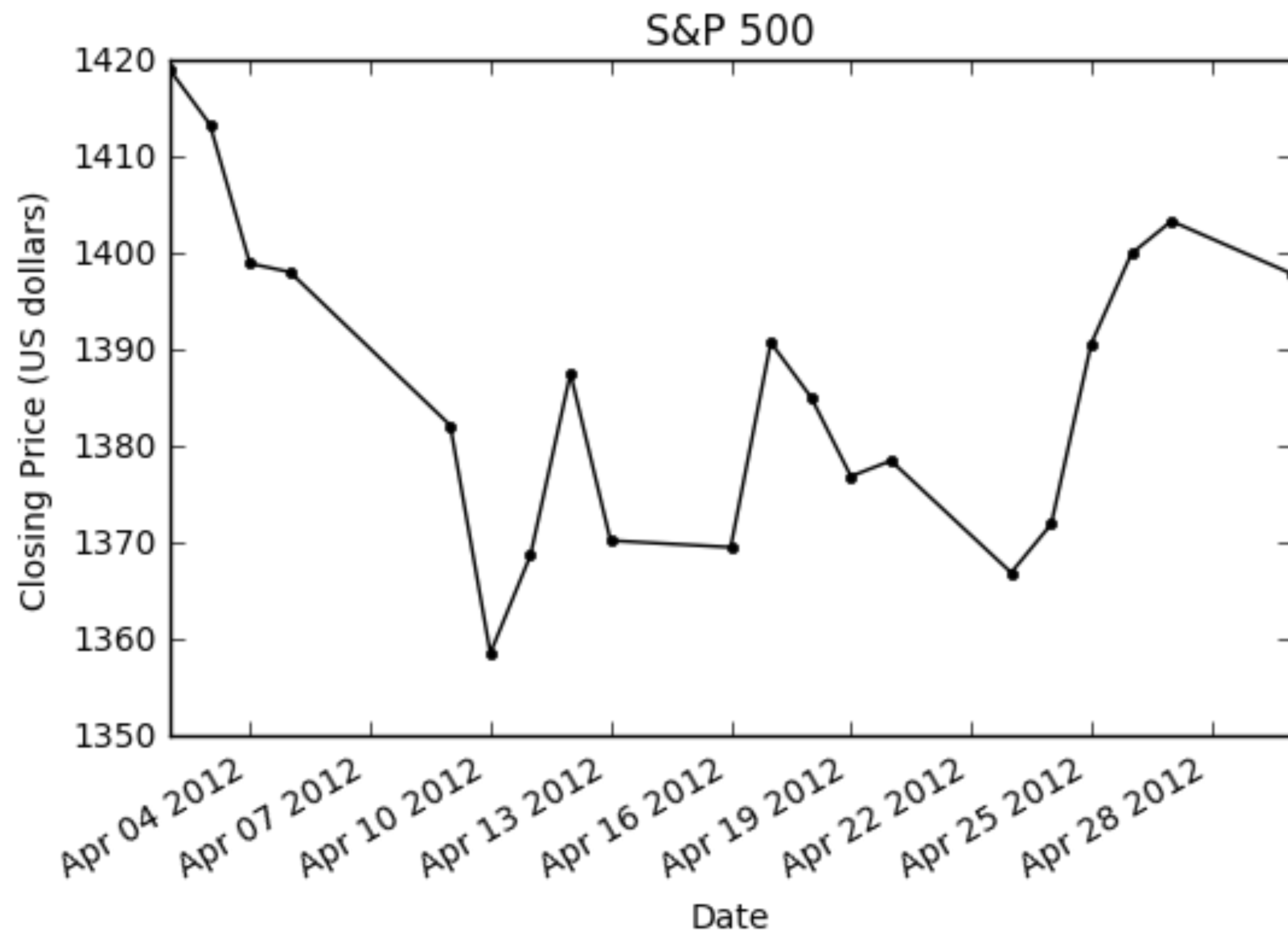
```
In [13]: sp500.loc['2012-4', 'Close'].plot(style='k.-',  
....:                                     title='S&P500')
```

```
In [14]: plt.ylabel('Closing Price (US Dollars)')
```

```
In [15]: plt.show()
```



One week





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

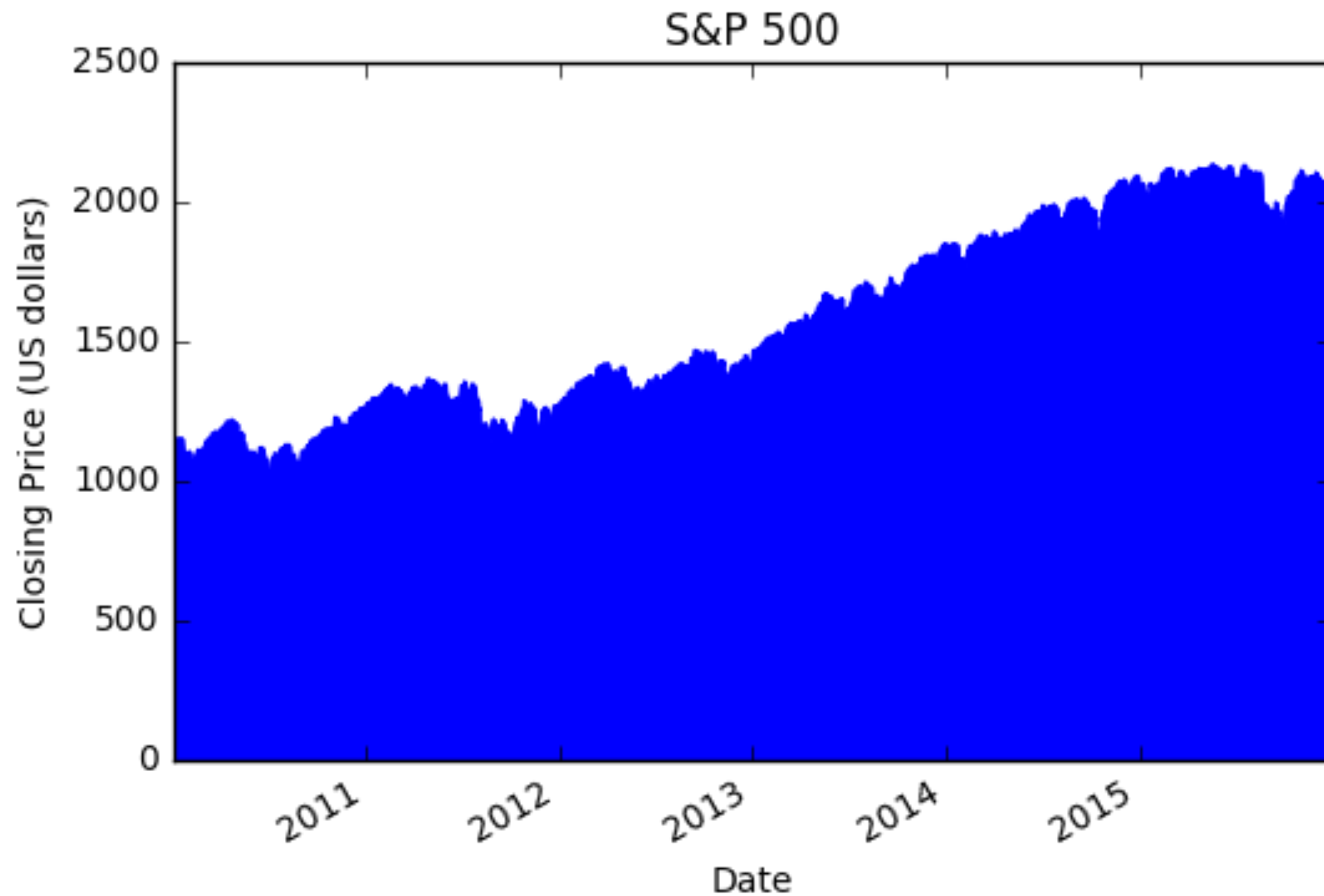
```
In [16]: sp500['Close'].plot(kind='area', title='S&P 500')
```

```
In [17]: plt.ylabel('Closing Price (US Dollars)')
```

```
In [18]: plt.show()
```




Area plot





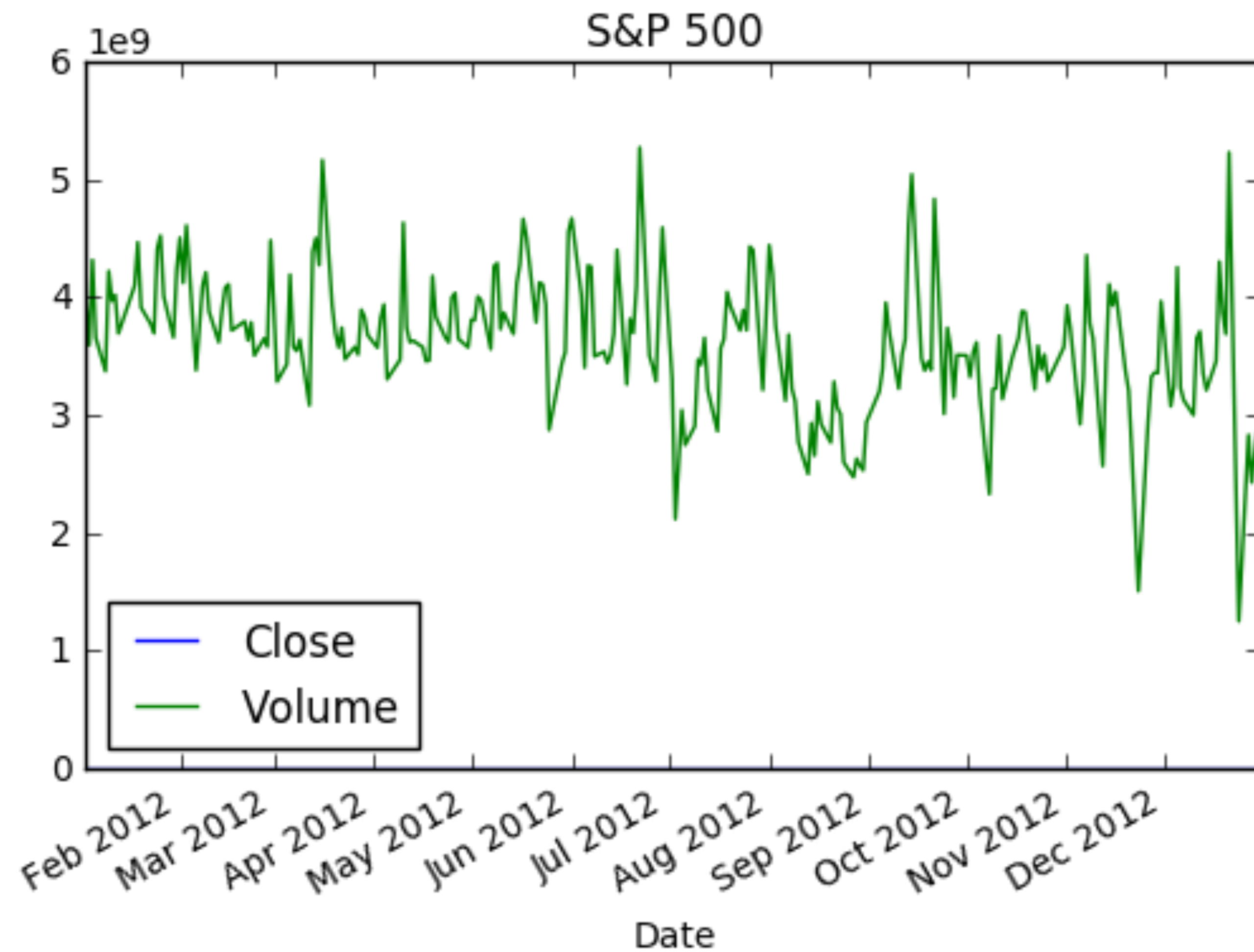
Multiple columns

```
In [19]: sp500.loc['2012', ['Close', 'Volume']].plot(title='S&P  
...: 500')
```

```
In [20]: plt.show()
```



Multiple columns



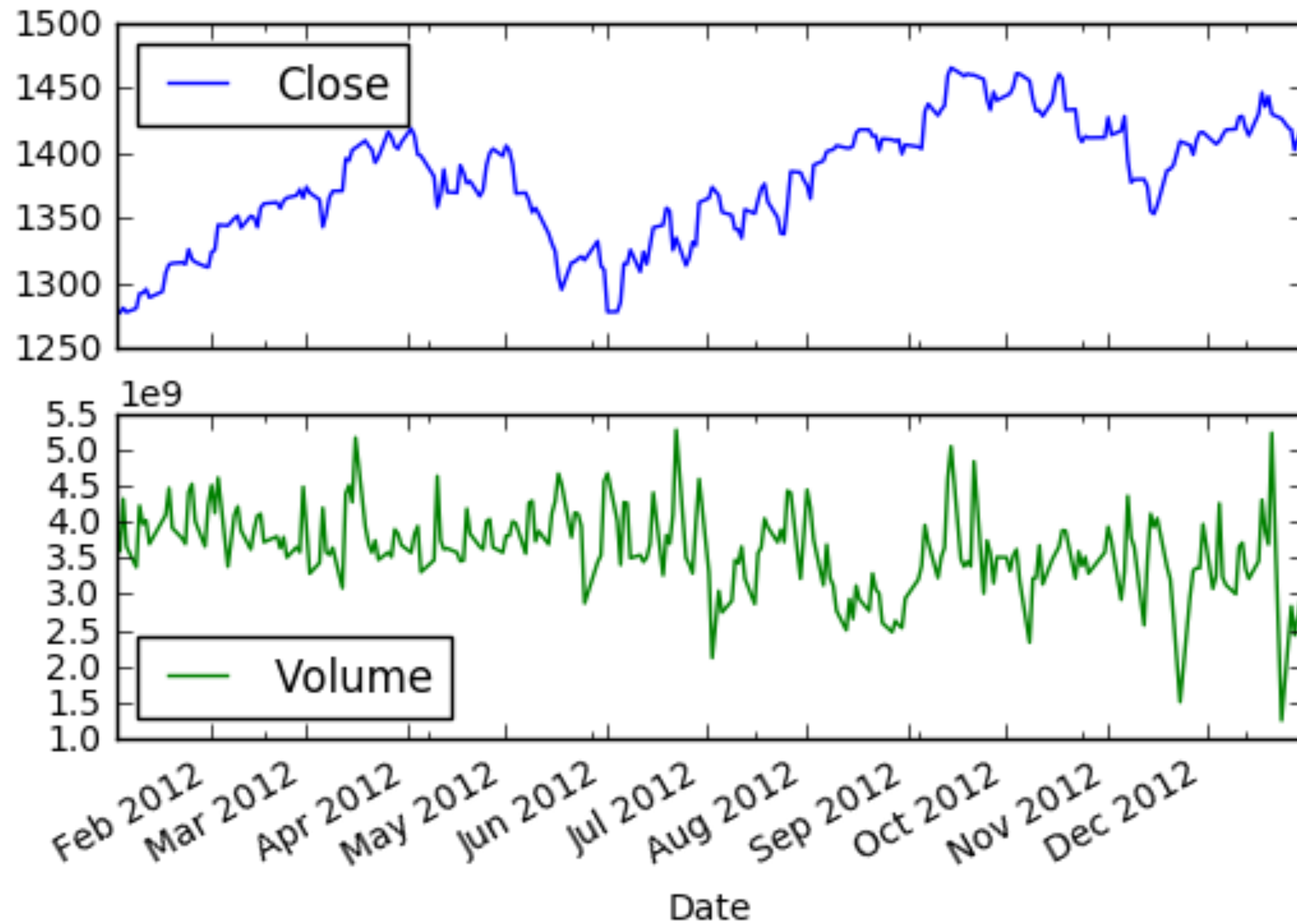


Subplots

```
In [21]: sp500.loc['2012', ['Close', 'Volume']].plot(subplots=True)  
  
In [22]: plt.show()
```



Subplots





PANDAS FOUNDATIONS

Let's practice!