INTRO TO TIME SERIES ANALYSIS

Learning Objectives:

After this lesson, you will be able to:

- Understand what time series analysis is and its uses
- Use pandas to model and manipulate a time series data
- Explain the functionality afforded to the DateTime object

What is time series analysis?

How is it different from our prior analyses?

Time Series Analysis

- Up until this point, we have studied classification and regression where each observation existed simultaneously without respect to any notion of time or ordering.

Time Series Analysis

- Today, we look at incorporating time into our analysis. Our observations for a given variable will be ordered and tied to a given interval.

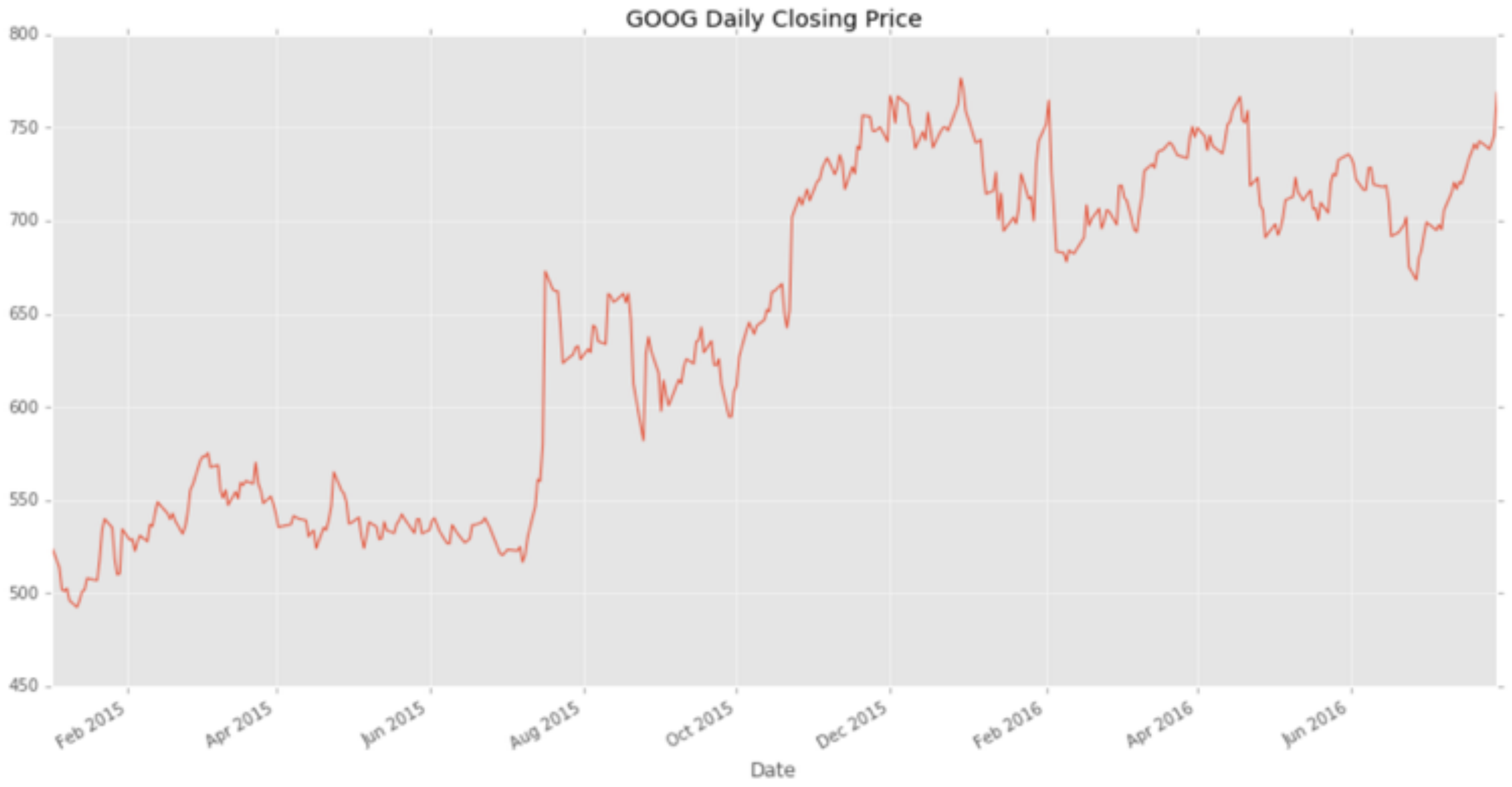
Time Series Analysis

- What are some use-cases for time-series analysis?

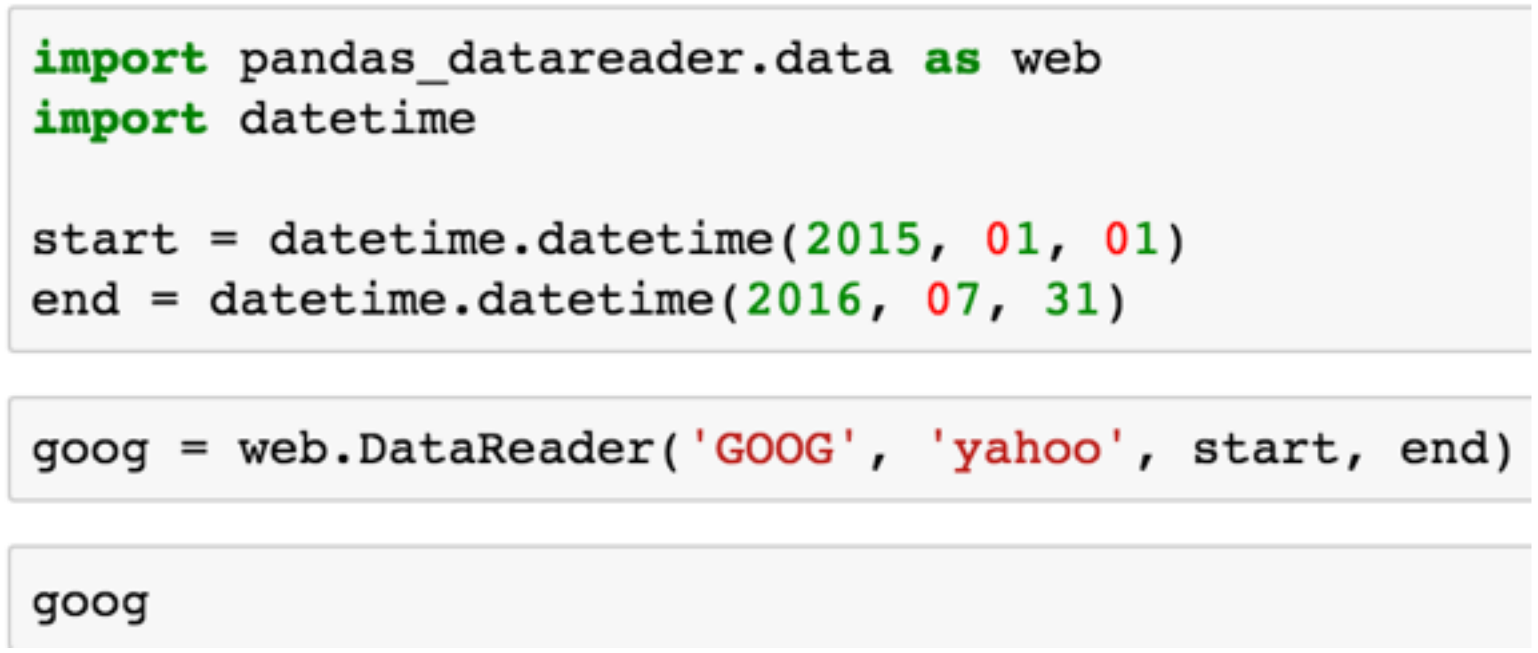
- What are some use-cases for time-series analysis?
  - Forecasting quarterly sales, profits, etc.
  - Weather forecasting
  - Epidemiological forecasting
  - Signal Processing

Time Series Analysis

- What a time series looks like:



‣ How we get that in pandas:



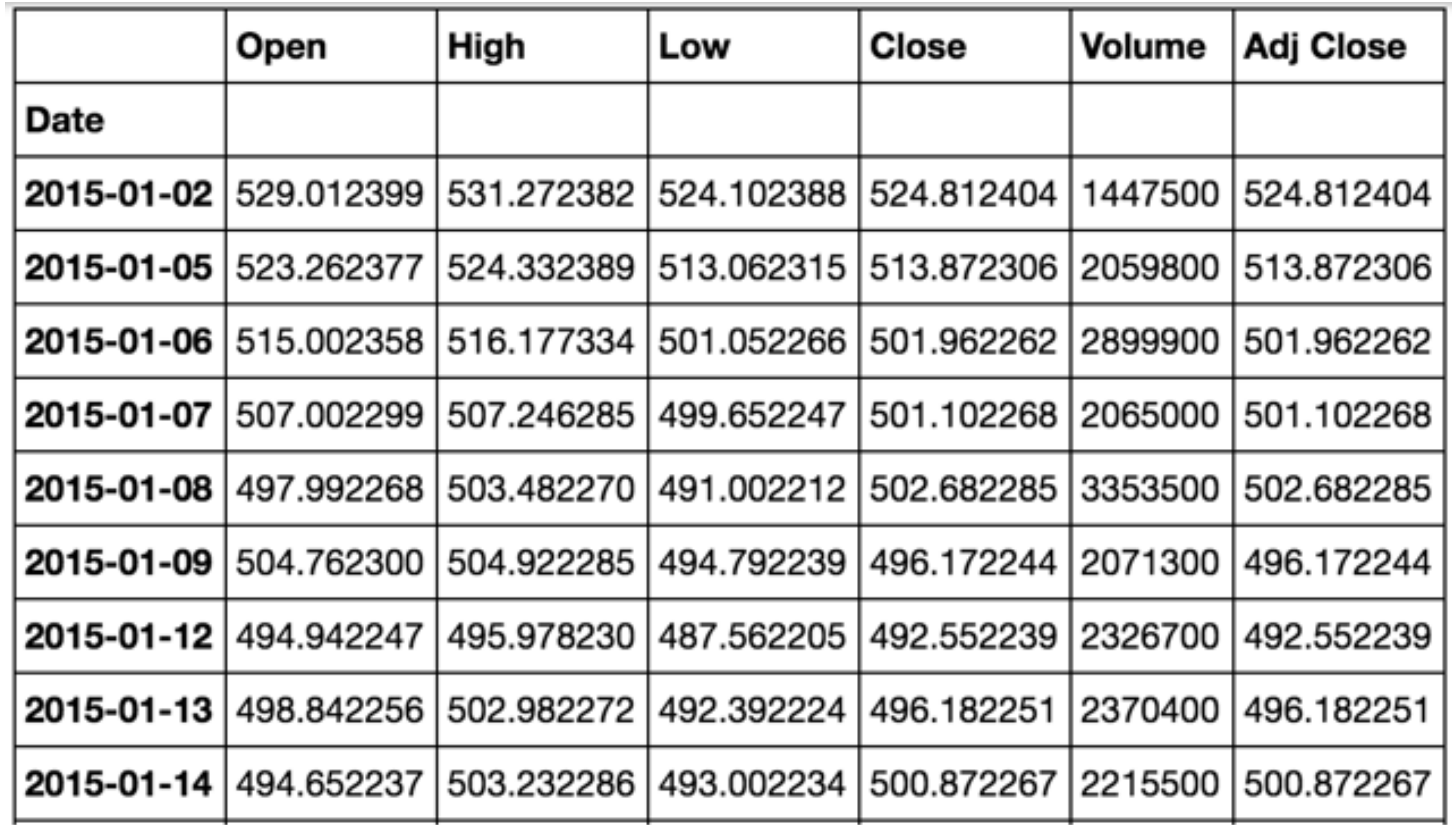
```
import pandas_datareader.data as web
import datetime

start = datetime.datetime(2015, 01, 01)
end = datetime.datetime(2016, 07, 31)
```

```
goog = web.DataReader('GOOG', 'yahoo', start, end)
```

```
goog
```

‣ That results in:

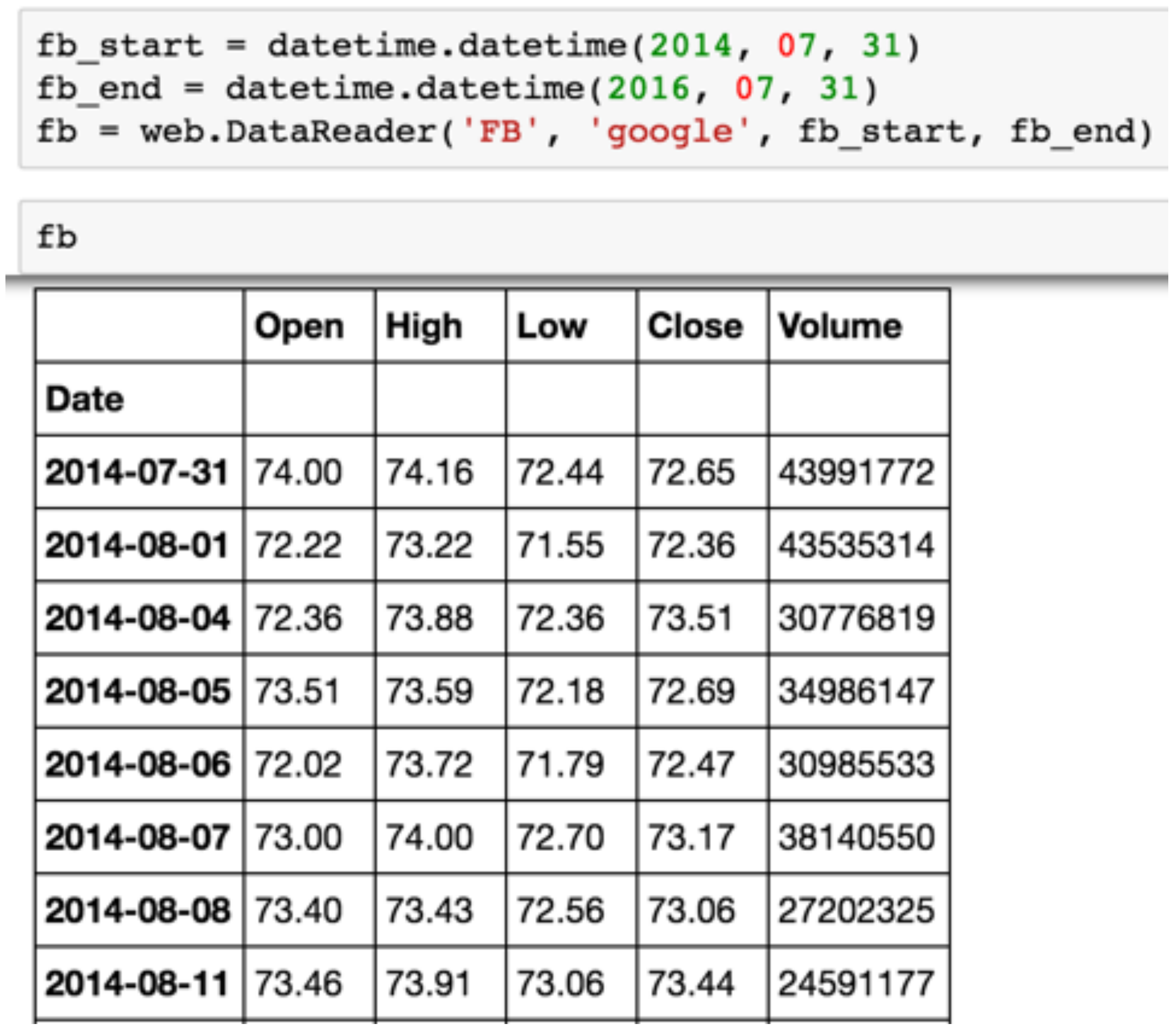


	Open	High	Low	Close	Volume	Adj Close
Date						
2015-01-02	529.012399	531.272382	524.102388	524.812404	1447500	524.812404
2015-01-05	523.262377	524.332389	513.062315	513.872306	2059800	513.872306
2015-01-06	515.002358	516.177334	501.052266	501.962262	2899900	501.962262
2015-01-07	507.002299	507.246285	499.652247	501.102268	2065000	501.102268
2015-01-08	497.992268	503.482270	491.002212	502.682285	3353500	502.682285
2015-01-09	504.762300	504.922285	494.792239	496.172244	2071300	496.172244
2015-01-12	494.942247	495.978230	487.562205	492.552239	2326700	492.552239
2015-01-13	498.842256	502.982272	492.392224	496.182251	2370400	496.182251
2015-01-14	494.652237	503.232286	493.002234	500.872267	2215500	500.872267

- Exercise:
  - Use the pandas_datareader to download the stock data for Facebook for the last 2 years.
  - Use Google as the source.
  - Note: You will need to make sure you are on pandas 0.18 (conda update pandas at the command line)

Time Series Analysis

Exercise:

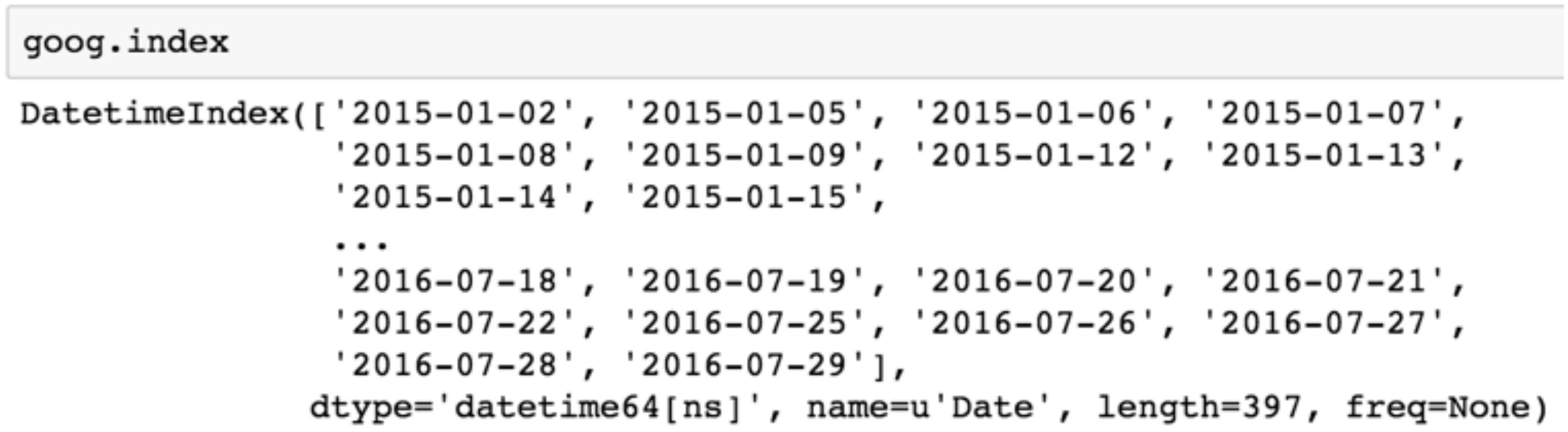


```
fb_start = datetime.datetime(2014, 07, 31)
fb_end = datetime.datetime(2016, 07, 31)
fb = web.DataReader('FB', 'google', fb_start, fb_end)
```

```
fb
```

	Open	High	Low	Close	Volume
Date					
2014-07-31	74.00	74.16	72.44	72.65	43991772
2014-08-01	72.22	73.22	71.55	72.36	43535314
2014-08-04	72.36	73.88	72.36	73.51	30776819
2014-08-05	73.51	73.59	72.18	72.69	34986147
2014-08-06	72.02	73.72	71.79	72.47	30985533
2014-08-07	73.00	74.00	72.70	73.17	38140550
2014-08-08	73.40	73.43	72.56	73.06	27202325
2014-08-11	73.46	73.91	73.06	73.44	24591177

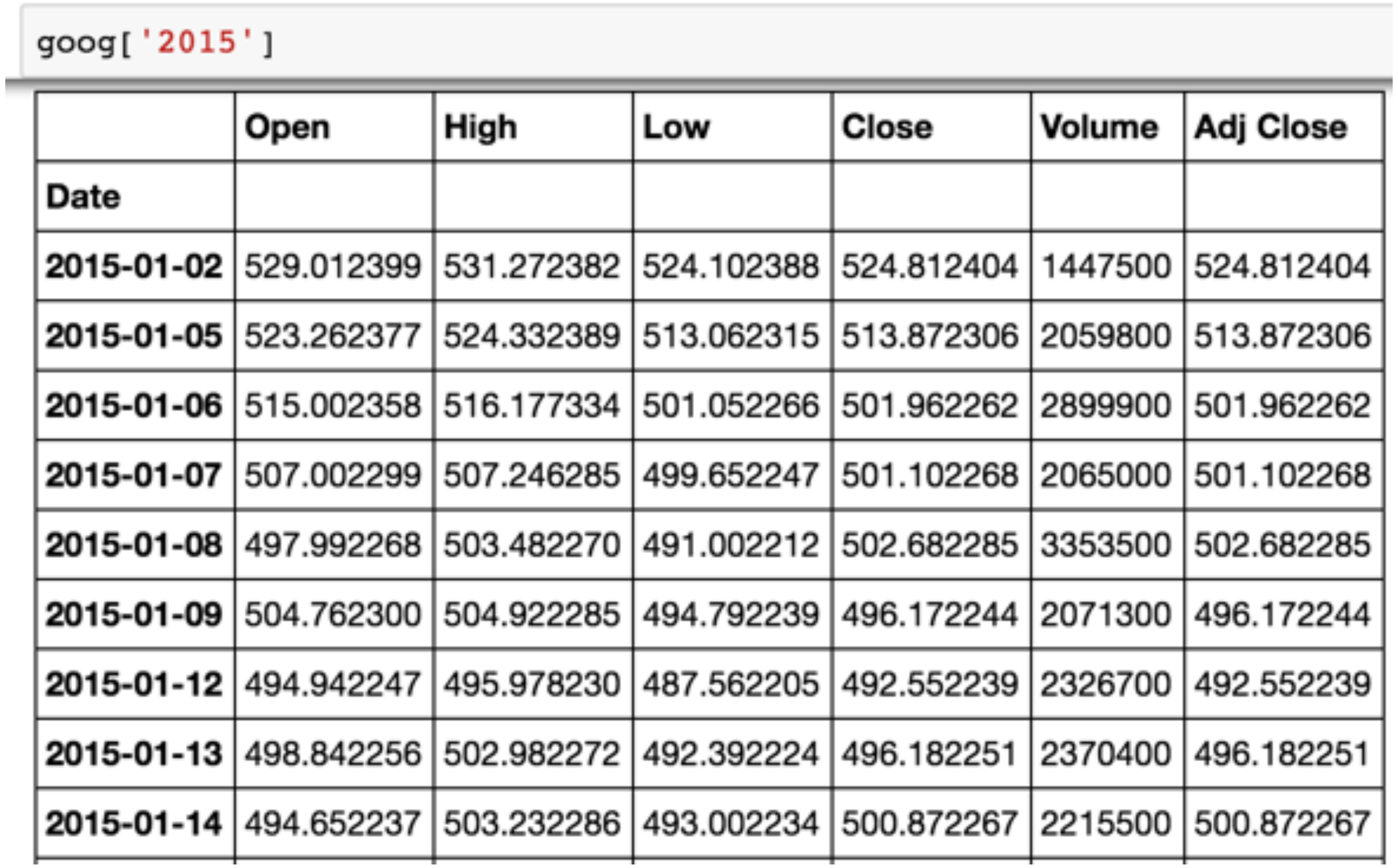
‣ The DatetimeIndex:



```
goog.index
```

```
DatetimeIndex(['2015-01-02', '2015-01-05', '2015-01-06', '2015-01-07',
               '2015-01-08', '2015-01-09', '2015-01-12', '2015-01-13',
               '2015-01-14', '2015-01-15',
               ...
               '2016-07-18', '2016-07-19', '2016-07-20', '2016-07-21',
               '2016-07-22', '2016-07-25', '2016-07-26', '2016-07-27',
               '2016-07-28', '2016-07-29'],
              dtype='datetime64[ns]', name=u'Date', length=397, freq=None)
```

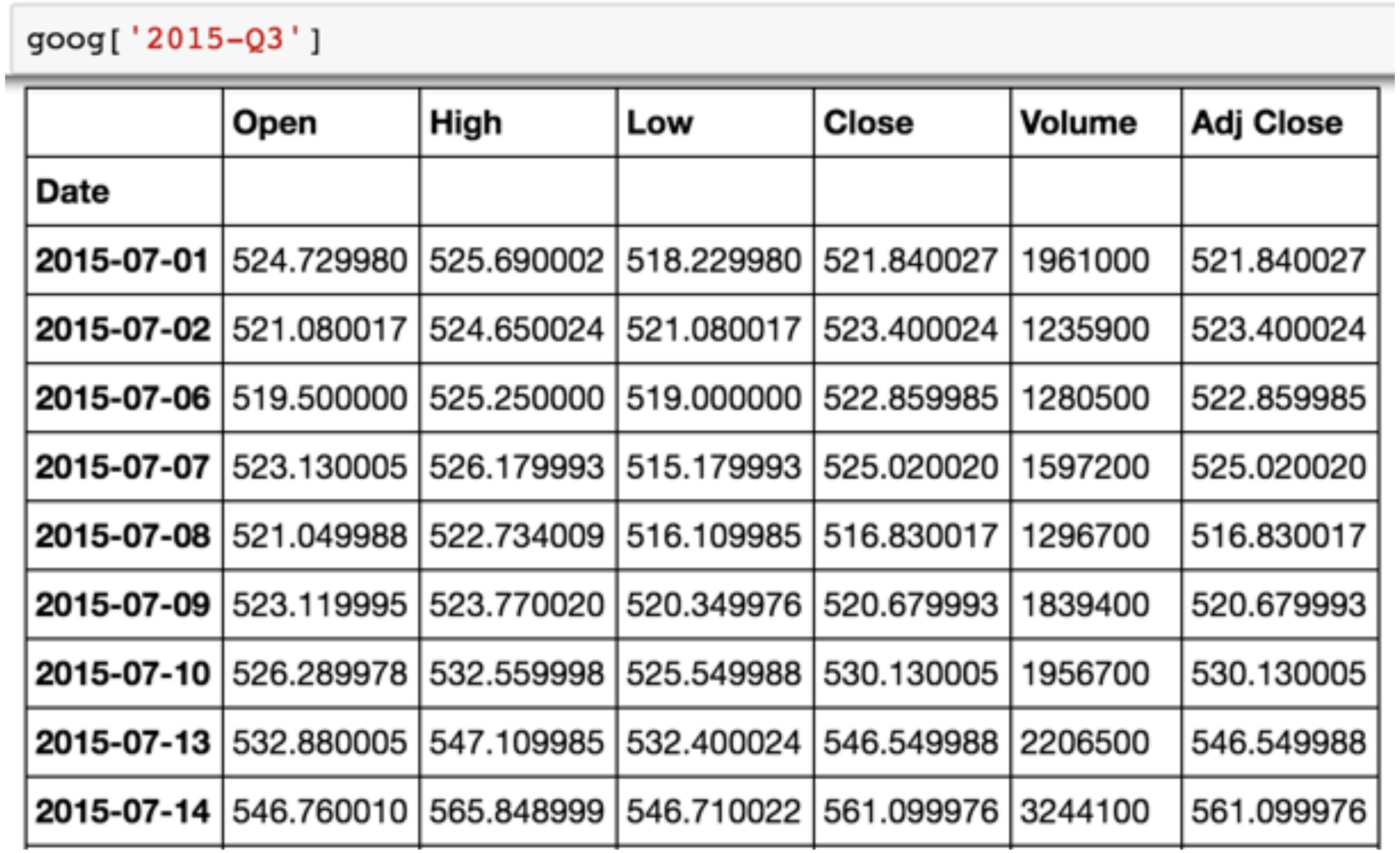
‣ DatetimeIndex Indexing & Slicing:



```
goog['2015']
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-01-02	529.012399	531.272382	524.102388	524.812404	1447500	524.812404
2015-01-05	523.262377	524.332389	513.062315	513.872306	2059800	513.872306
2015-01-06	515.002358	516.177334	501.052266	501.962262	2899900	501.962262
2015-01-07	507.002299	507.246285	499.652247	501.102268	2065000	501.102268
2015-01-08	497.992268	503.482270	491.002212	502.682285	3353500	502.682285
2015-01-09	504.762300	504.922285	494.792239	496.172244	2071300	496.172244
2015-01-12	494.942247	495.978230	487.562205	492.552239	2326700	492.552239
2015-01-13	498.842256	502.982272	492.392224	496.182251	2370400	496.182251
2015-01-14	494.652237	503.232286	493.002234	500.872267	2215500	500.872267

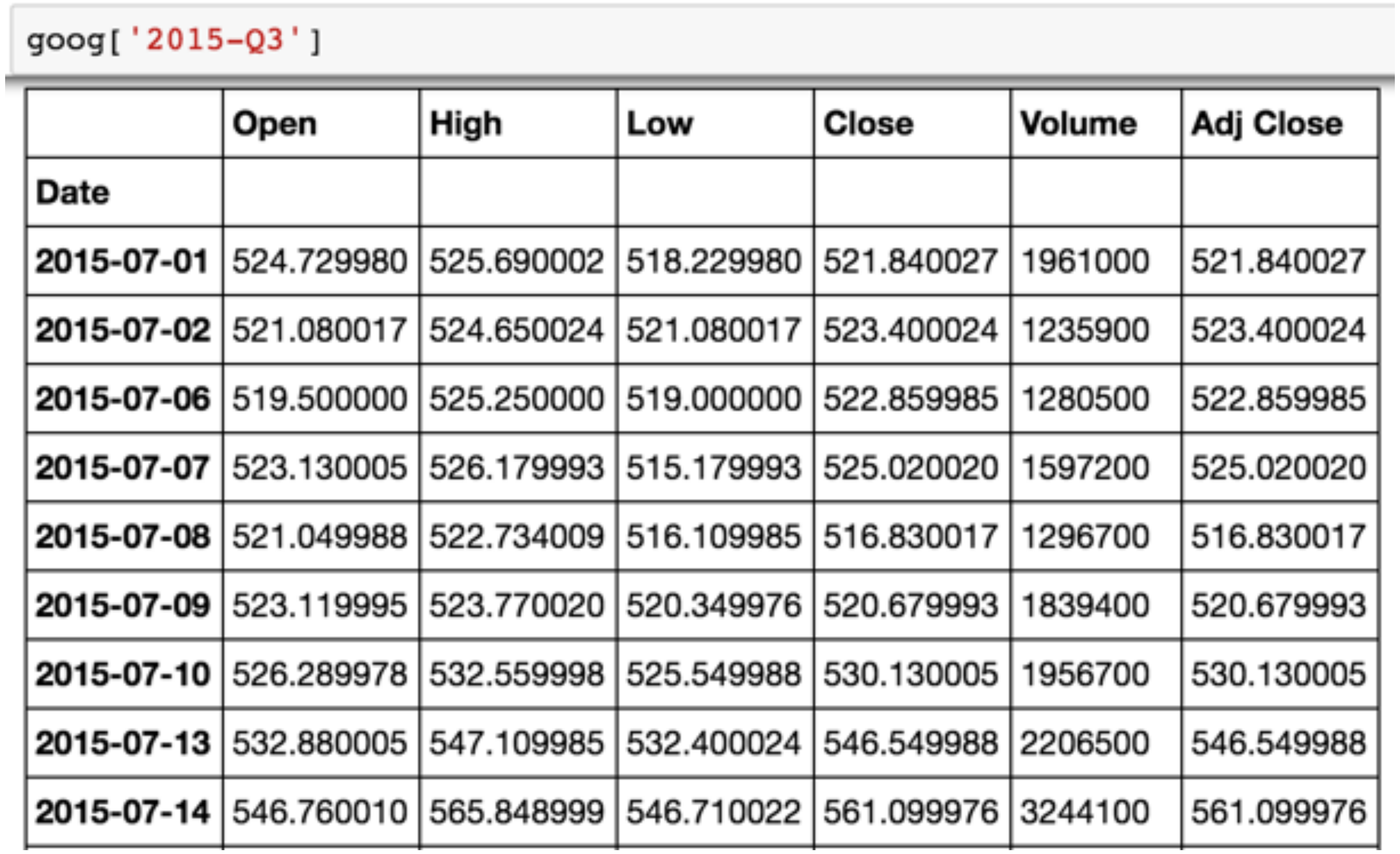
‣ DatetimeIndex Indexing & Slicing:



```
goog['2015-Q3']
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-07-01	524.729980	525.690002	518.229980	521.840027	1961000	521.840027
2015-07-02	521.080017	524.650024	521.080017	523.400024	1235900	523.400024
2015-07-06	519.500000	525.250000	519.000000	522.859985	1280500	522.859985
2015-07-07	523.130005	526.179993	515.179993	525.020020	1597200	525.020020
2015-07-08	521.049988	522.734009	516.109985	516.830017	1296700	516.830017
2015-07-09	523.119995	523.770020	520.349976	520.679993	1839400	520.679993
2015-07-10	526.289978	532.559998	525.549988	530.130005	1956700	530.130005
2015-07-13	532.880005	547.109985	532.400024	546.549988	2206500	546.549988
2015-07-14	546.760010	565.848999	546.710022	561.099976	3244100	561.099976

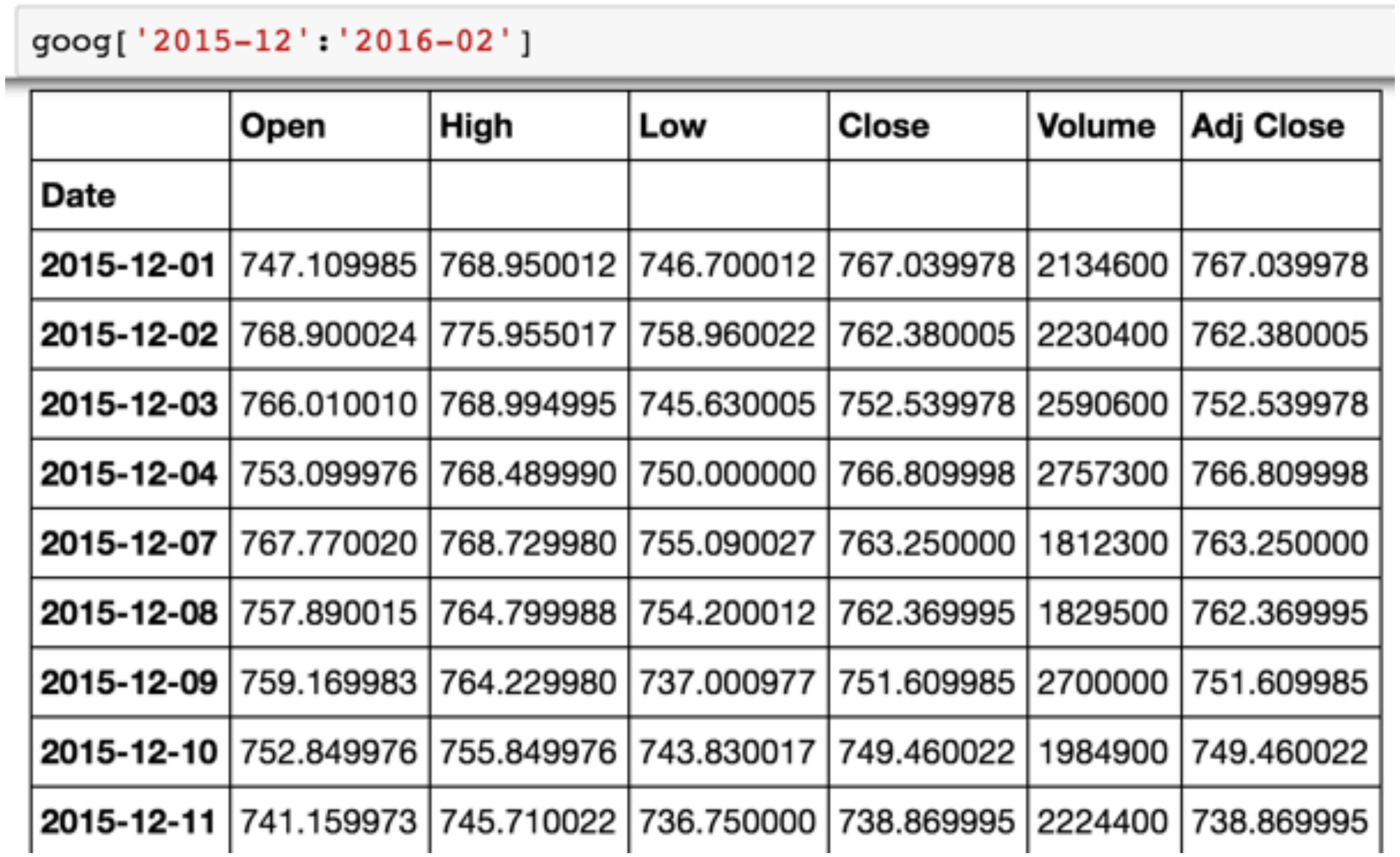
‣ DatetimeIndex Indexing & Slicing:



```
goog['2015-Q3']
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-07-01	524.729980	525.690002	518.229980	521.840027	1961000	521.840027
2015-07-02	521.080017	524.650024	521.080017	523.400024	1235900	523.400024
2015-07-06	519.500000	525.250000	519.000000	522.859985	1280500	522.859985
2015-07-07	523.130005	526.179993	515.179993	525.020020	1597200	525.020020
2015-07-08	521.049988	522.734009	516.109985	516.830017	1296700	516.830017
2015-07-09	523.119995	523.770020	520.349976	520.679993	1839400	520.679993
2015-07-10	526.289978	532.559998	525.549988	530.130005	1956700	530.130005
2015-07-13	532.880005	547.109985	532.400024	546.549988	2206500	546.549988
2015-07-14	546.760010	565.848999	546.710022	561.099976	3244100	561.099976

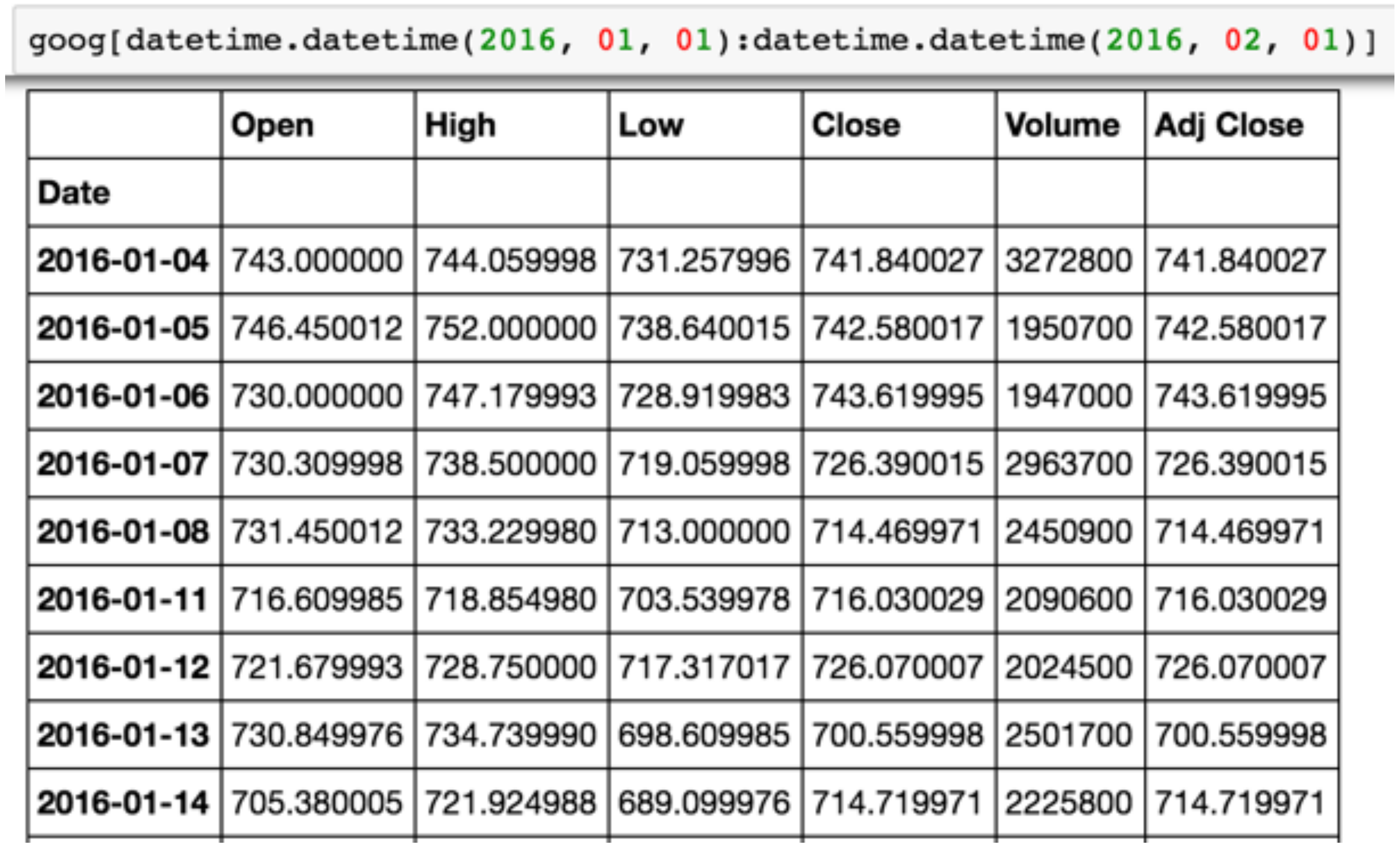
DatetimeIndex Indexing & Slicing:



```
goog['2015-12':'2016-02']
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-12-01	747.109985	768.950012	746.700012	767.039978	2134600	767.039978
2015-12-02	768.900024	775.955017	758.960022	762.380005	2230400	762.380005
2015-12-03	766.010010	768.994995	745.630005	752.539978	2590600	752.539978
2015-12-04	753.099976	768.489990	750.000000	766.809998	2757300	766.809998
2015-12-07	767.770020	768.729980	755.090027	763.250000	1812300	763.250000
2015-12-08	757.890015	764.799988	754.200012	762.369995	1829500	762.369995
2015-12-09	759.169983	764.229980	737.000977	751.609985	2700000	751.609985
2015-12-10	752.849976	755.849976	743.830017	749.460022	1984900	749.460022
2015-12-11	741.159973	745.710022	736.750000	738.869995	2224400	738.869995

‣ DatetimeIndex Indexing & Slicing:



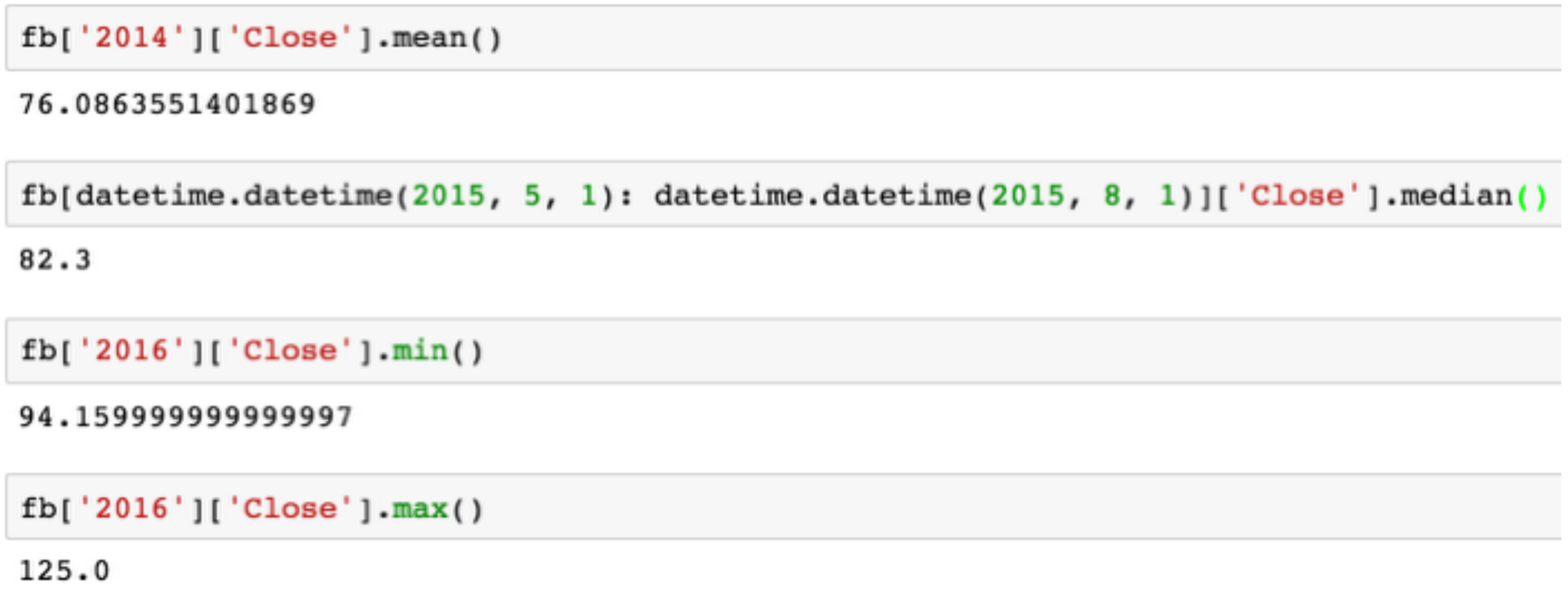
```
goog[datetime.datetime(2016, 01, 01):datetime.datetime(2016, 02, 01)]
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-01-04	743.000000	744.059998	731.257996	741.840027	3272800	741.840027
2016-01-05	746.450012	752.000000	738.640015	742.580017	1950700	742.580017
2016-01-06	730.000000	747.179993	728.919983	743.619995	1947000	743.619995
2016-01-07	730.309998	738.500000	719.059998	726.390015	2963700	726.390015
2016-01-08	731.450012	733.229980	713.000000	714.469971	2450900	714.469971
2016-01-11	716.609985	718.854980	703.539978	716.030029	2090600	716.030029
2016-01-12	721.679993	728.750000	717.317017	726.070007	2024500	726.070007
2016-01-13	730.849976	734.739990	698.609985	700.559998	2501700	700.559998
2016-01-14	705.380005	721.924988	689.099976	714.719971	2225800	714.719971

Time Series Analysis

- Exercise:
  - Using you fb data perform the following operations:
    - Select the 2014 data for the closing price - what is the mean?
    - What is the median close between May 1, 2015 and Aug 1, 2015?
    - For 2016 - what is the min and the max?

Exercise:



```
fb['2014']['Close'].mean()
```

```
76.0863551401869
```

```
fb[datetime.datetime(2015, 5, 1): datetime.datetime(2015, 8, 1)]['Close'].median()
```

```
82.3
```

```
fb['2016']['Close'].min()
```

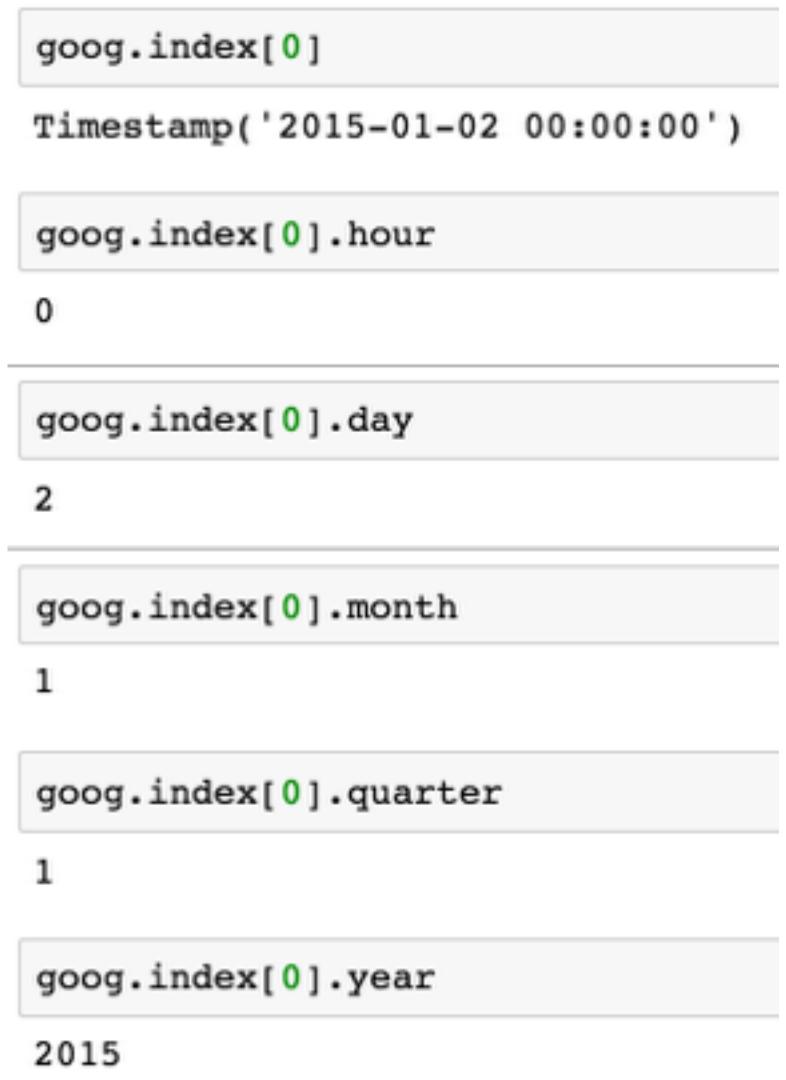
```
94.15999999999997
```

```
fb['2016']['Close'].max()
```

```
125.0
```

Time Series Analysis

The Datetime Object:



```
goog.index[0]
```

```
Timestamp('2015-01-02 00:00:00')
```

```
goog.index[0].hour
```

```
0
```

```
goog.index[0].day
```

```
2
```

```
goog.index[0].month
```

```
1
```

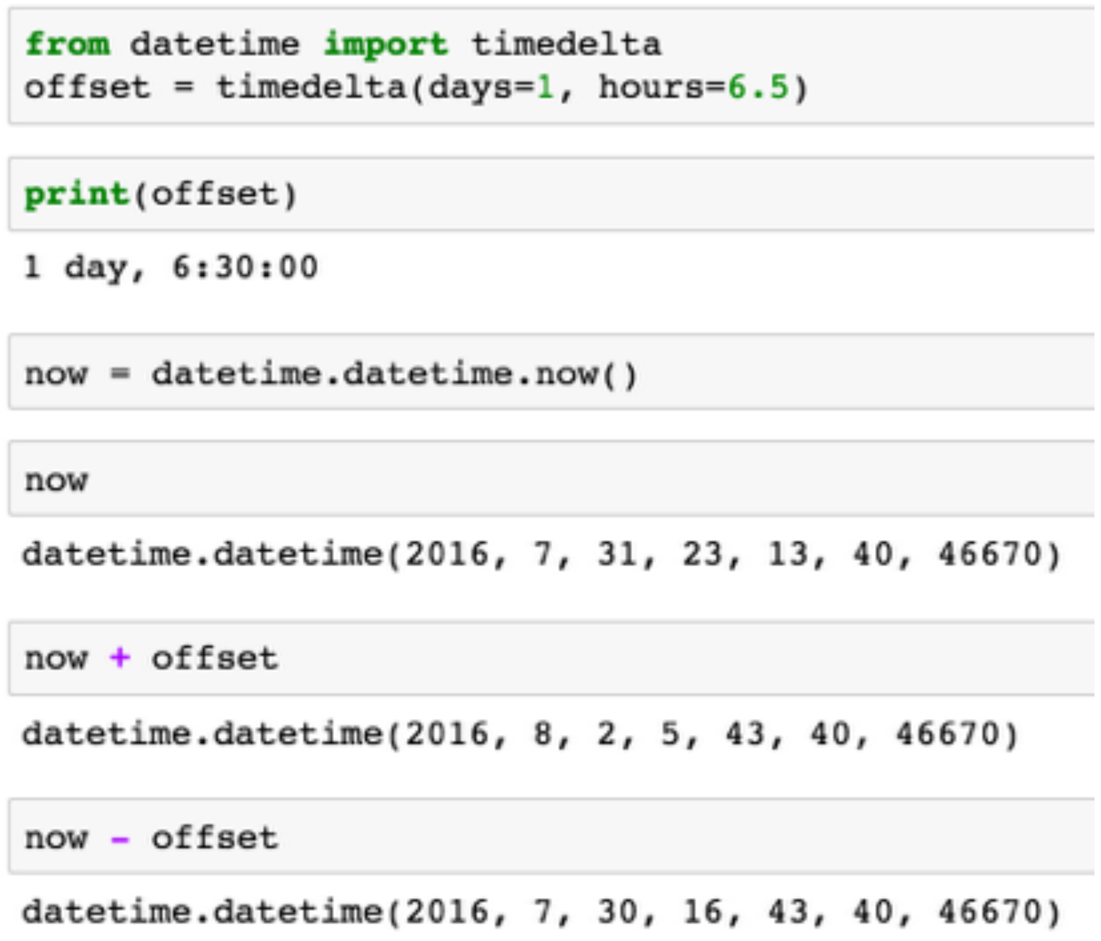
```
goog.index[0].quarter
```

```
1
```

```
goog.index[0].year
```

```
2015
```

‣ TimeDelta Operations:



```
from datetime import timedelta
offset = timedelta(days=1, hours=6.5)
```

```
print(offset)
```

```
1 day, 6:30:00
```

```
now = datetime.datetime.now()
```

```
now
```

```
datetime.datetime(2016, 7, 31, 23, 13, 40, 46670)
```

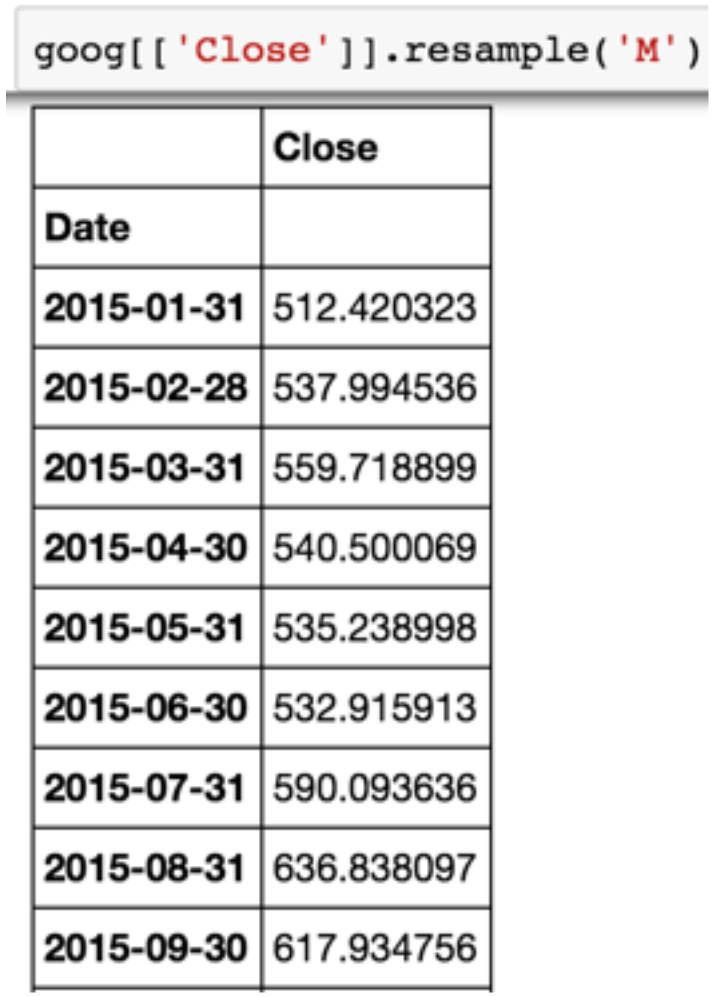
```
now + offset
```

```
datetime.datetime(2016, 8, 2, 5, 43, 40, 46670)
```

```
now - offset
```

```
datetime.datetime(2016, 7, 30, 16, 43, 40, 46670)
```

Changing Time Frequencies - .resample():

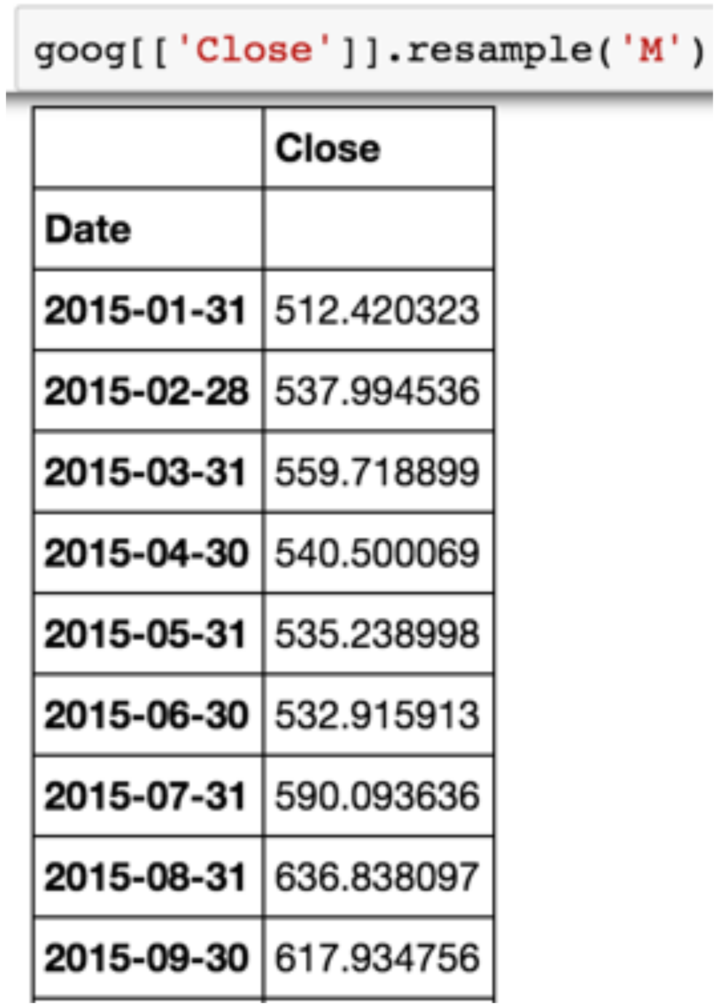


```
goog[['Close']].resample('M')
```

	Close
Date	
2015-01-31	512.420323
2015-02-28	537.994536
2015-03-31	559.718899
2015-04-30	540.500069
2015-05-31	535.238998
2015-06-30	532.915913
2015-07-31	590.093636
2015-08-31	636.838097
2015-09-30	617.934756

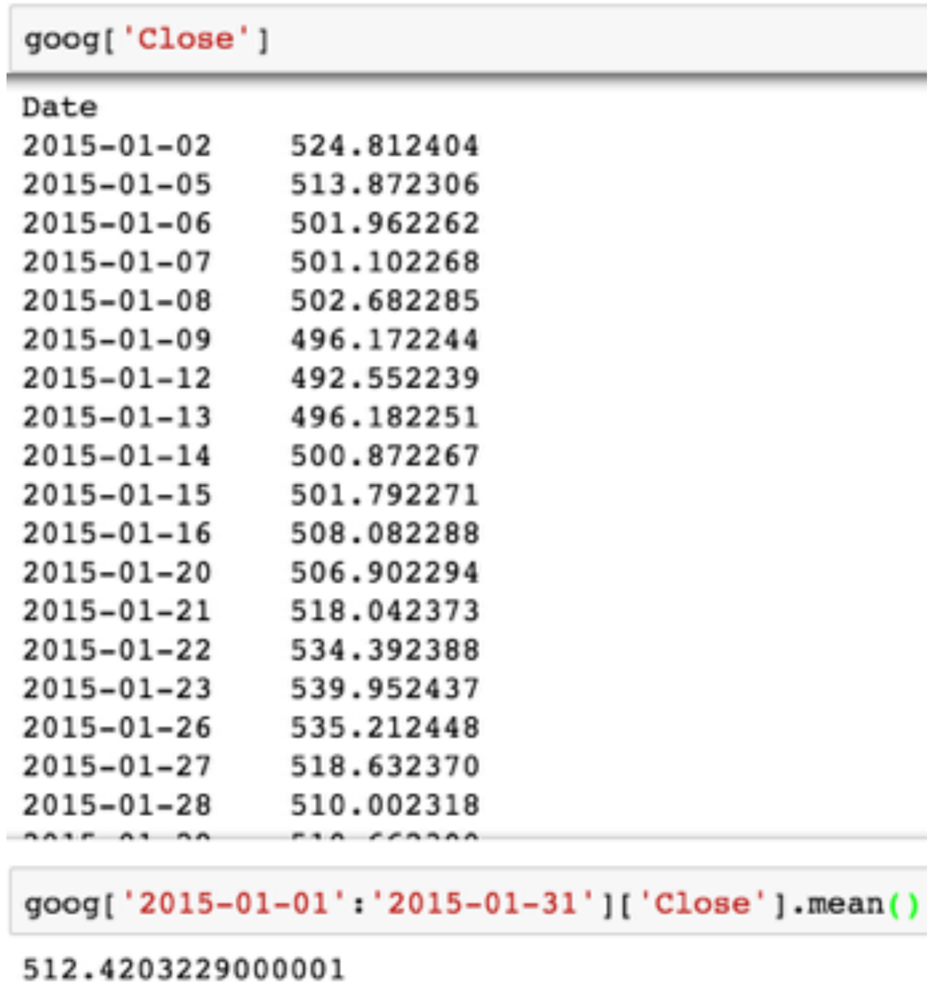
Time Series Analysis

‣ Changing Time Frequencies:



```
goog[['Close']].resample('M')
```

	Close
Date	
2015-01-31	512.420323
2015-02-28	537.994536
2015-03-31	559.718899
2015-04-30	540.500069
2015-05-31	535.238998
2015-06-30	532.915913
2015-07-31	590.093636
2015-08-31	636.838097
2015-09-30	617.934756



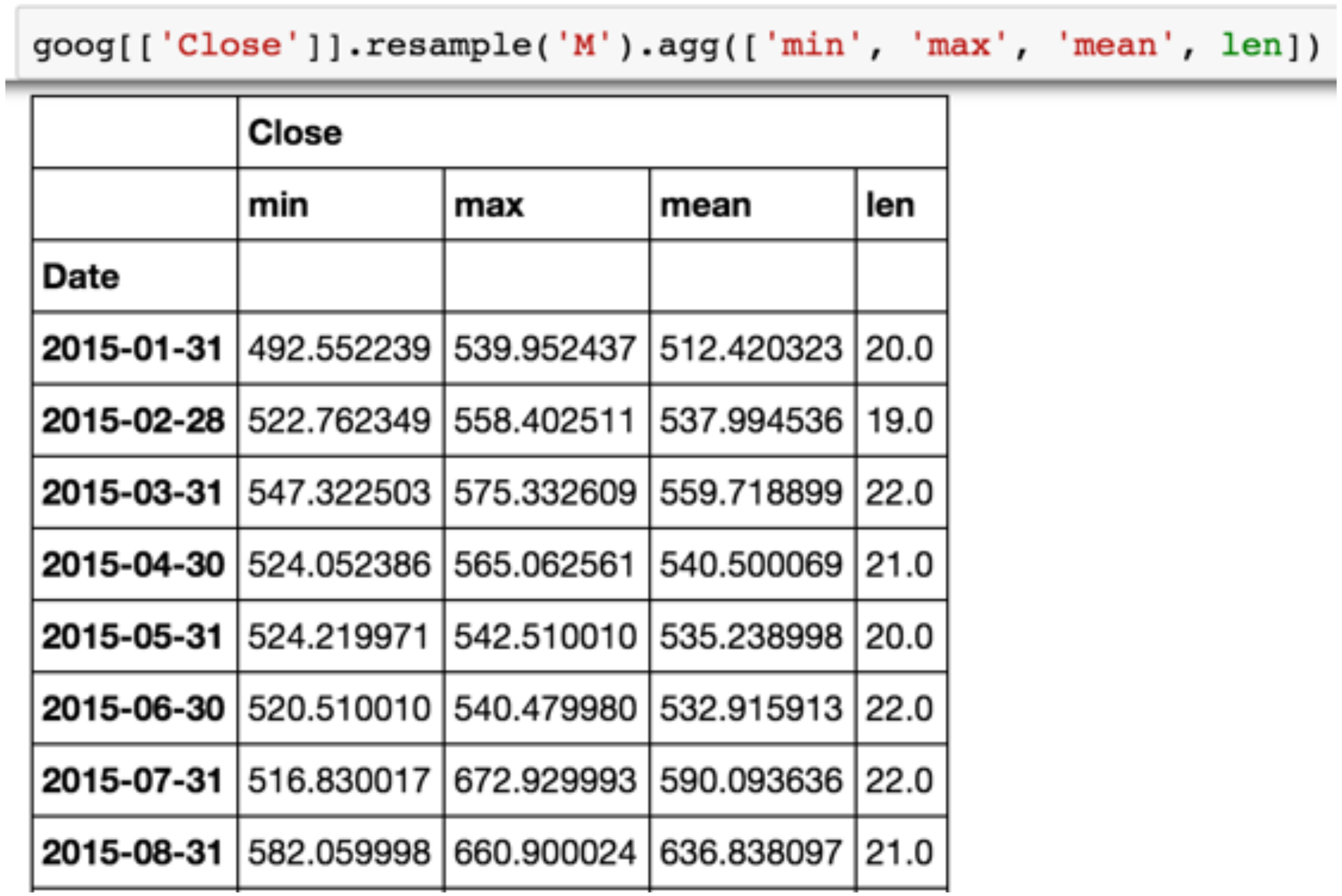
```
goog['Close']
```

```
Date
2015-01-02    524.812404
2015-01-05    513.872306
2015-01-06    501.962262
2015-01-07    501.102268
2015-01-08    502.682285
2015-01-09    496.172244
2015-01-12    492.552239
2015-01-13    496.182251
2015-01-14    500.872267
2015-01-15    501.792271
2015-01-16    508.082288
2015-01-20    506.902294
2015-01-21    518.042373
2015-01-22    534.392388
2015-01-23    539.952437
2015-01-26    535.212448
2015-01-27    518.632370
2015-01-28    510.002318
```

```
goog['2015-01-01':'2015-01-31']['Close'].mean()
```

```
512.4203229000001
```

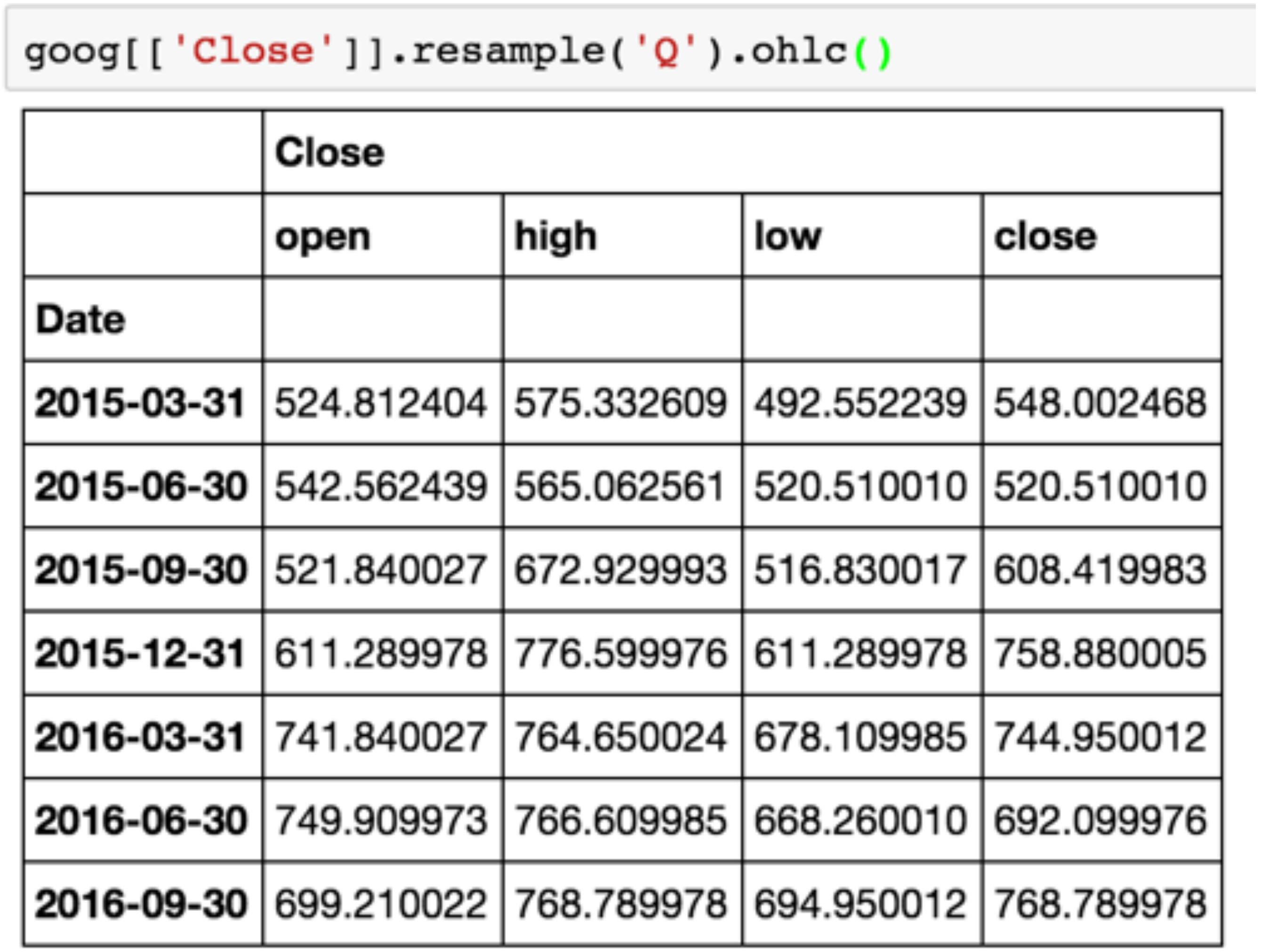
‣ Changing Time Frequencies:



```
goog[['Close']].resample('M').agg(['min', 'max', 'mean', len])
```

	Close			
	min	**max**	**mean**	**len**
Date				
2015-01-31	492.552239	539.952437	512.420323	20.0
2015-02-28	522.762349	558.402511	537.994536	19.0
2015-03-31	547.322503	575.332609	559.718899	22.0
2015-04-30	524.052386	565.062561	540.500069	21.0
2015-05-31	524.219971	542.510010	535.238998	20.0
2015-06-30	520.510010	540.479980	532.915913	22.0
2015-07-31	516.830017	672.929993	590.093636	22.0
2015-08-31	582.059998	660.900024	636.838097	21.0

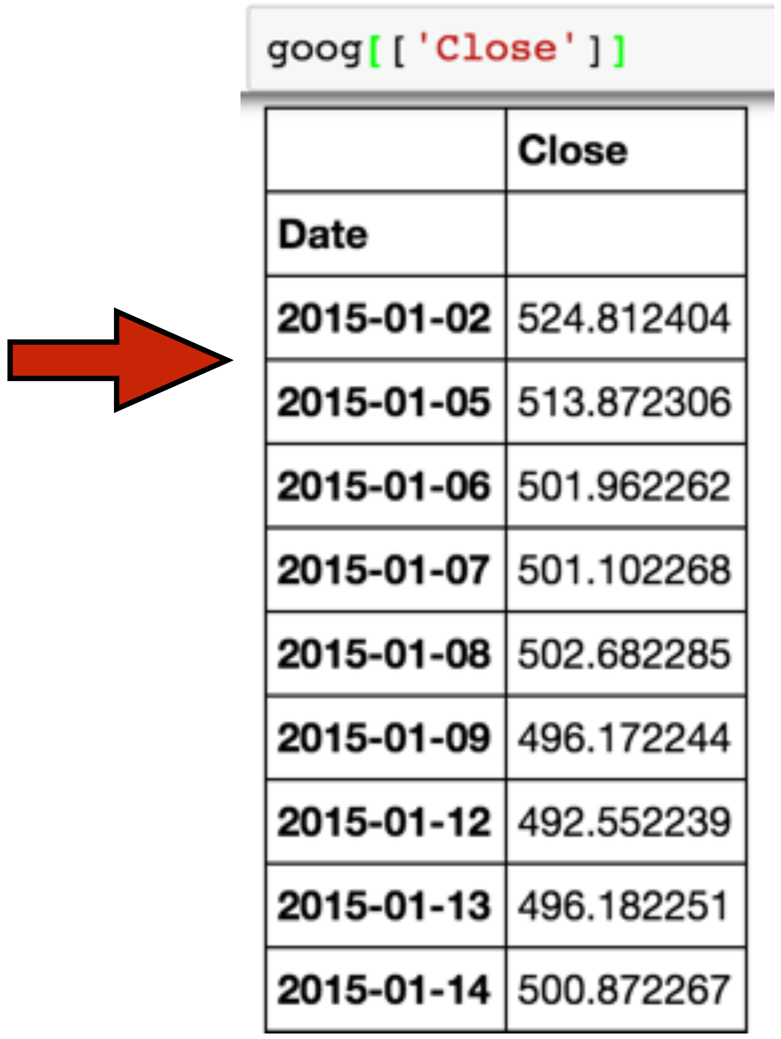
‣ Changing Time Frequencies:



```
goog[['Close']].resample('Q').ohlc()
```

	Close			
	open	**high**	**low**	**close**
Date				
2015-03-31	524.812404	575.332609	492.552239	548.002468
2015-06-30	542.562439	565.062561	520.510010	520.510010
2015-09-30	521.840027	672.929993	516.830017	608.419983
2015-12-31	611.289978	776.599976	611.289978	758.880005
2016-03-31	741.840027	764.650024	678.109985	744.950012
2016-06-30	749.909973	766.609985	668.260010	692.099976
2016-09-30	699.210022	768.789978	694.950012	768.789978

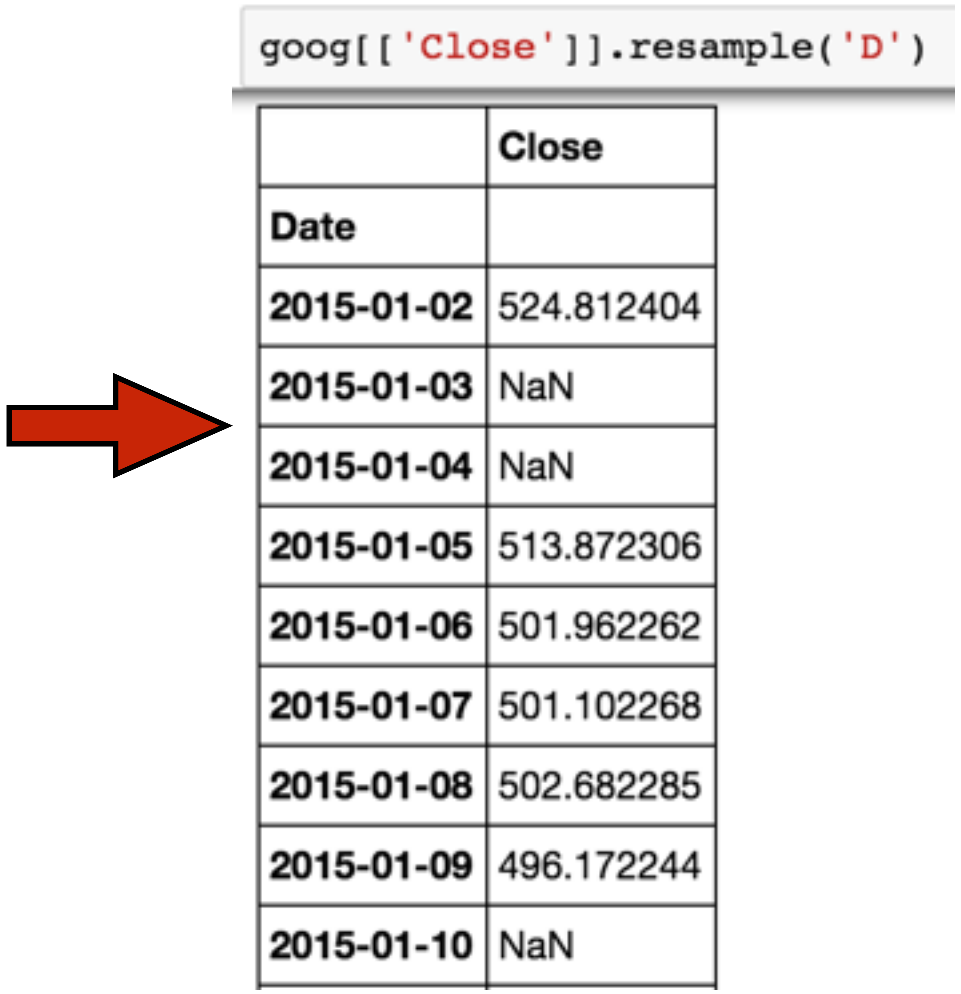
‣ Changing Time Frequencies - .asfreq():



```
goog[['Close']]
```

	Close
Date	
2015-01-02	524.812404
2015-01-05	513.872306
2015-01-06	501.962262
2015-01-07	501.102268
2015-01-08	502.682285
2015-01-09	496.172244
2015-01-12	492.552239
2015-01-13	496.182251
2015-01-14	500.872267

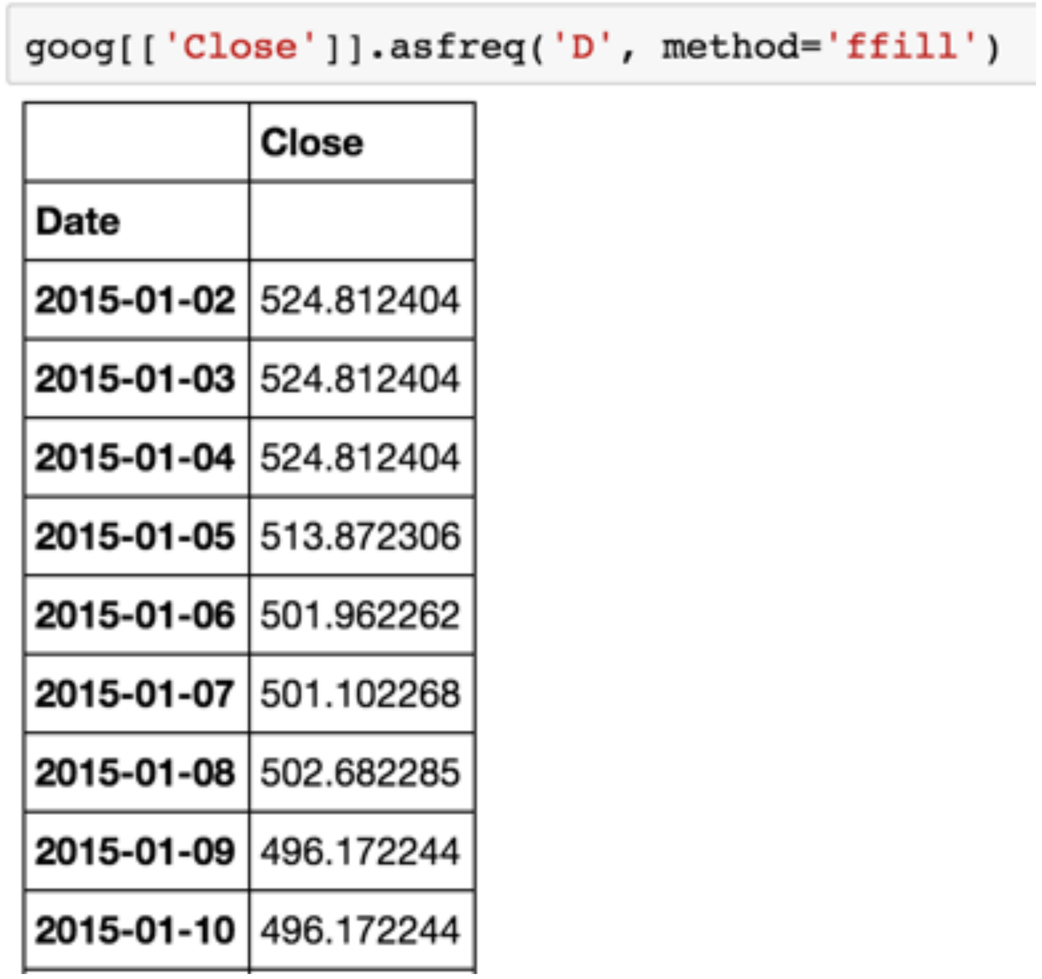
‣ Changing Time Frequencies - .asfreq():



```
goog[['Close']].resample('D')
```

	Close
Date	
2015-01-02	524.812404
2015-01-03	NaN
2015-01-04	NaN
2015-01-05	513.872306
2015-01-06	501.962262
2015-01-07	501.102268
2015-01-08	502.682285
2015-01-09	496.172244
2015-01-10	NaN

Changing Time Frequencies - .asfreq():

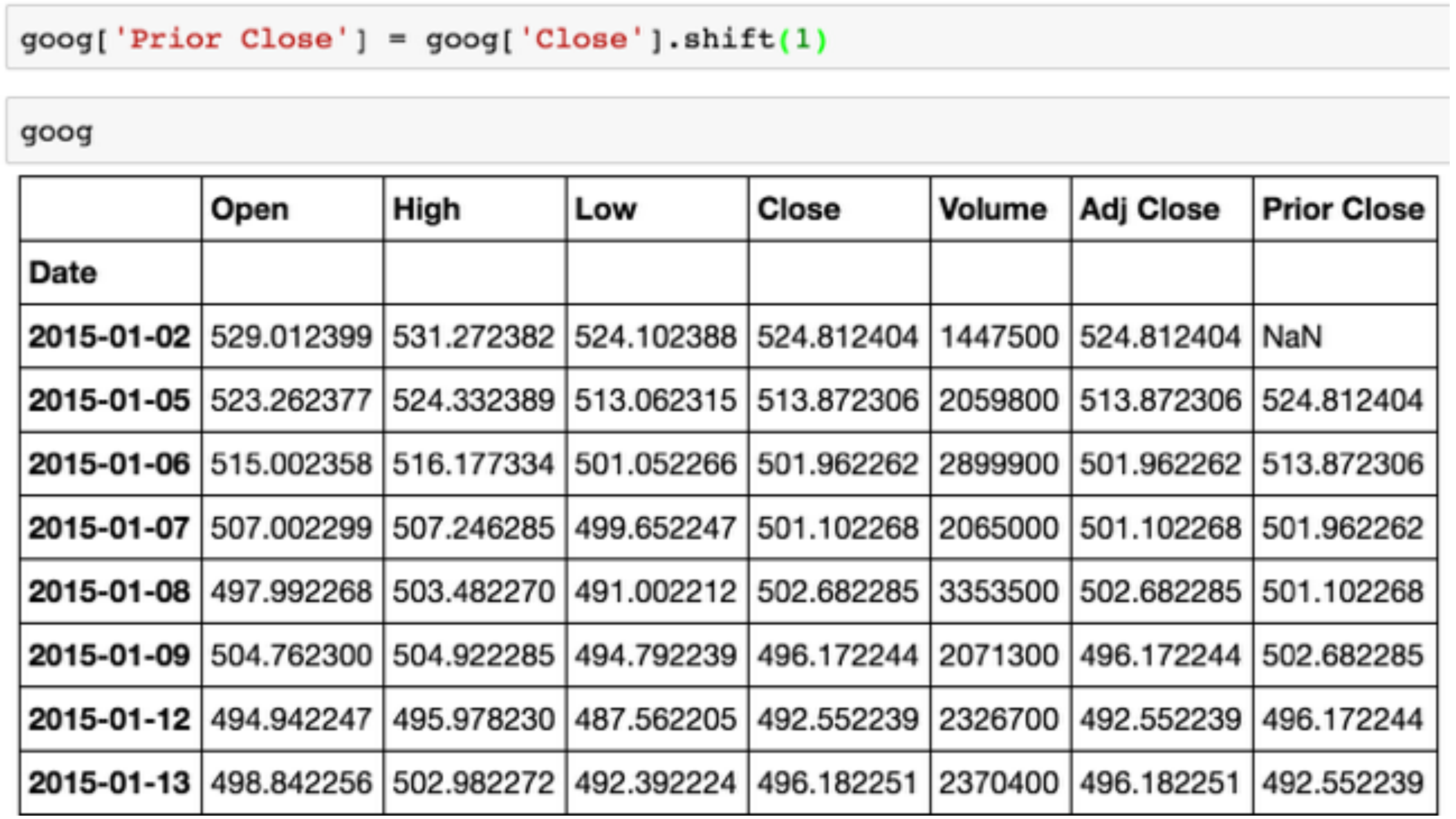


```
goog[['Close']].asfreq('D', method='ffill')
```

	Close
Date	
2015-01-02	524.812404
2015-01-03	524.812404
2015-01-04	524.812404
2015-01-05	513.872306
2015-01-06	501.962262
2015-01-07	501.102268
2015-01-08	502.682285
2015-01-09	496.172244
2015-01-10	496.172244

- Exercise:
  - Using your FB data try the following:
    - Resample the data to weekly - use the max weekly close
    - Resample the same data to daily - notice the dates vs. the weekly which day of the week does the weekly label indicate?
    - Change the weekly resampled data to start at the beginning of the week - this may require looking at the documentation for .resample()
    - What might be the consequences of starting your data mid-week?

‣ Shifting Time Series:

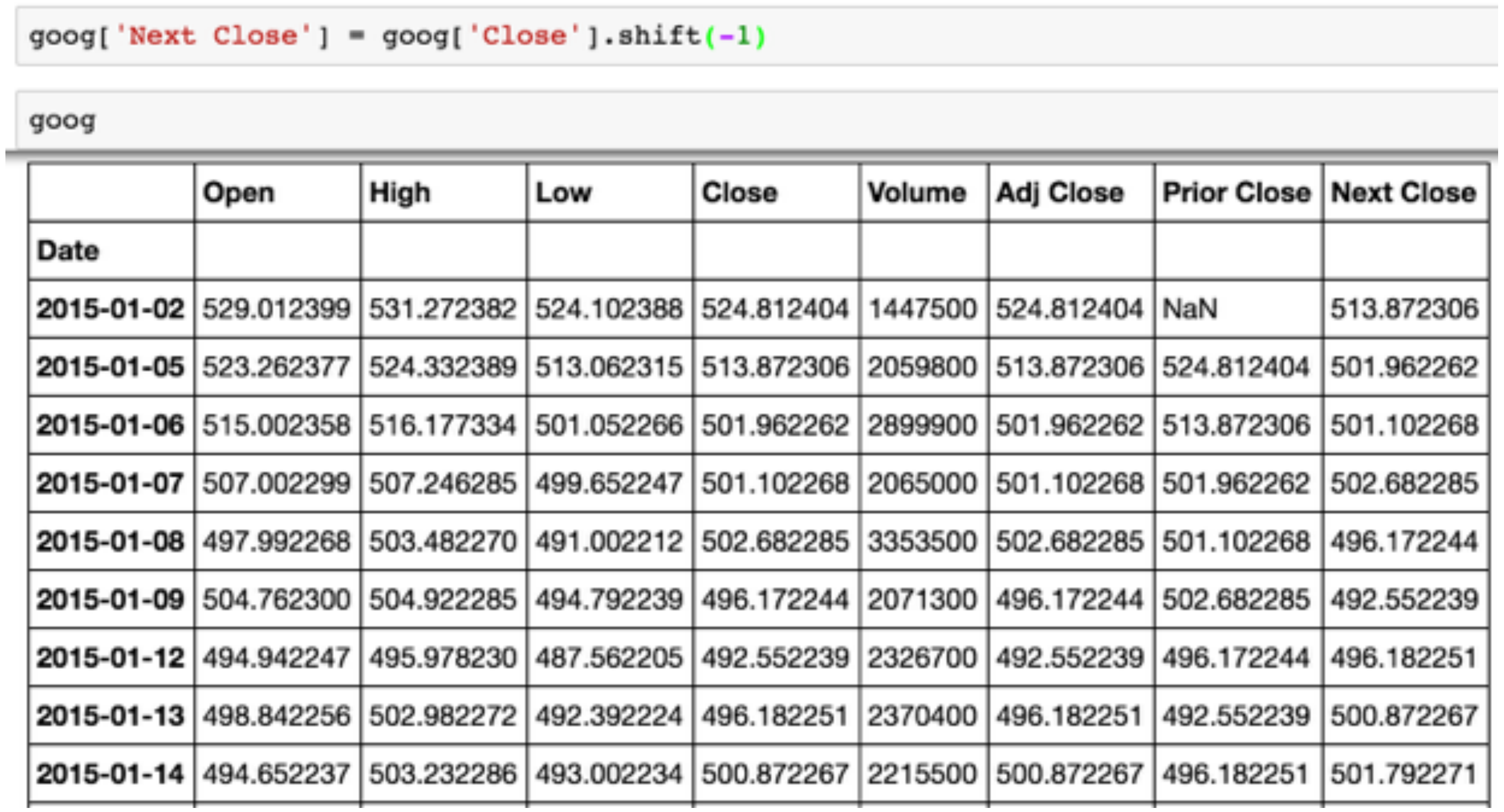


```
goog['Prior Close'] = goog['Close'].shift(1)
```

```
goog
```

	Open	High	Low	Close	Volume	Adj Close	Prior Close
Date							
2015-01-02	529.012399	531.272382	524.102388	524.812404	1447500	524.812404	NaN
2015-01-05	523.262377	524.332389	513.062315	513.872306	2059800	513.872306	524.812404
2015-01-06	515.002358	516.177334	501.052266	501.962262	2899900	501.962262	513.872306
2015-01-07	507.002299	507.246285	499.652247	501.102268	2065000	501.102268	501.962262
2015-01-08	497.992268	503.482270	491.002212	502.682285	3353500	502.682285	501.102268
2015-01-09	504.762300	504.922285	494.792239	496.172244	2071300	496.172244	502.682285
2015-01-12	494.942247	495.978230	487.562205	492.552239	2326700	492.552239	496.172244
2015-01-13	498.842256	502.982272	492.392224	496.182251	2370400	496.182251	492.552239

Shifting Time Series:

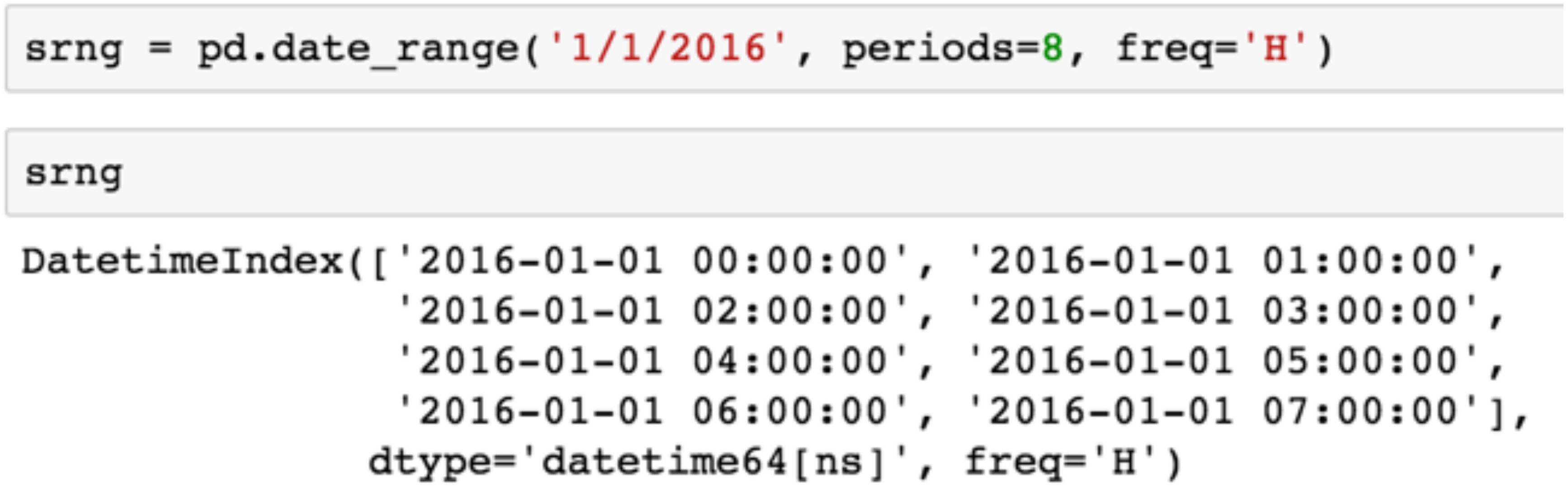


```
goog['Next Close'] = goog['Close'].shift(-1)
```

```
goog
```

	Open	High	Low	Close	Volume	Adj Close	Prior Close	Next Close
Date								
2015-01-02	529.012399	531.272382	524.102388	524.812404	1447500	524.812404	NaN	513.872306
2015-01-05	523.262377	524.332389	513.062315	513.872306	2059800	513.872306	524.812404	501.962262
2015-01-06	515.002358	516.177334	501.052266	501.962262	2899900	501.962262	513.872306	501.102268
2015-01-07	507.002299	507.246285	499.652247	501.102268	2065000	501.102268	501.962262	502.682285
2015-01-08	497.992268	503.482270	491.002212	502.682285	3353500	502.682285	501.102268	496.172244
2015-01-09	504.762300	504.922285	494.792239	496.172244	2071300	496.172244	502.682285	492.552239
2015-01-12	494.942247	495.978230	487.562205	492.552239	2326700	492.552239	496.172244	496.182251
2015-01-13	498.842256	502.982272	492.392224	496.182251	2370400	496.182251	492.552239	500.872267
2015-01-14	494.652237	503.232286	493.002234	500.872267	2215500	500.872267	496.182251	501.792271

‣ Creating a DatetimeIndex from a Range:

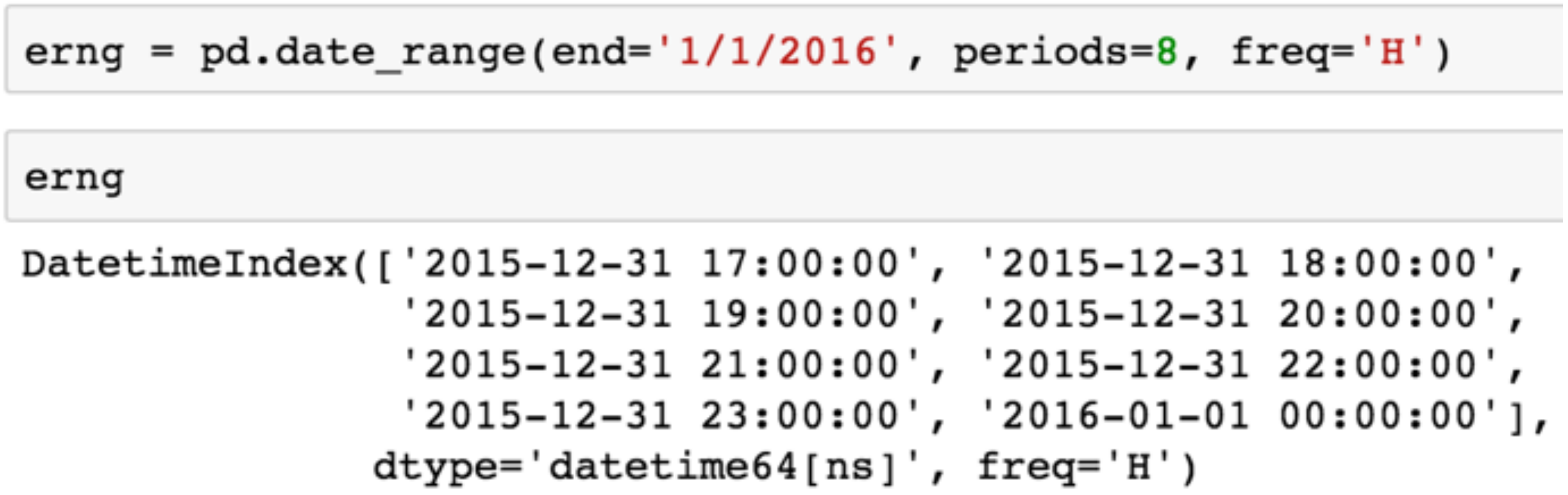


```
srng = pd.date_range('1/1/2016', periods=8, freq='H')
```

```
srng
```

```
DatetimeIndex(['2016-01-01 00:00:00', '2016-01-01 01:00:00',
               '2016-01-01 02:00:00', '2016-01-01 03:00:00',
               '2016-01-01 04:00:00', '2016-01-01 05:00:00',
               '2016-01-01 06:00:00', '2016-01-01 07:00:00'],
              dtype='datetime64[ns]', freq='H')
```

‣ Creating a DatetimeIndex from a Range:

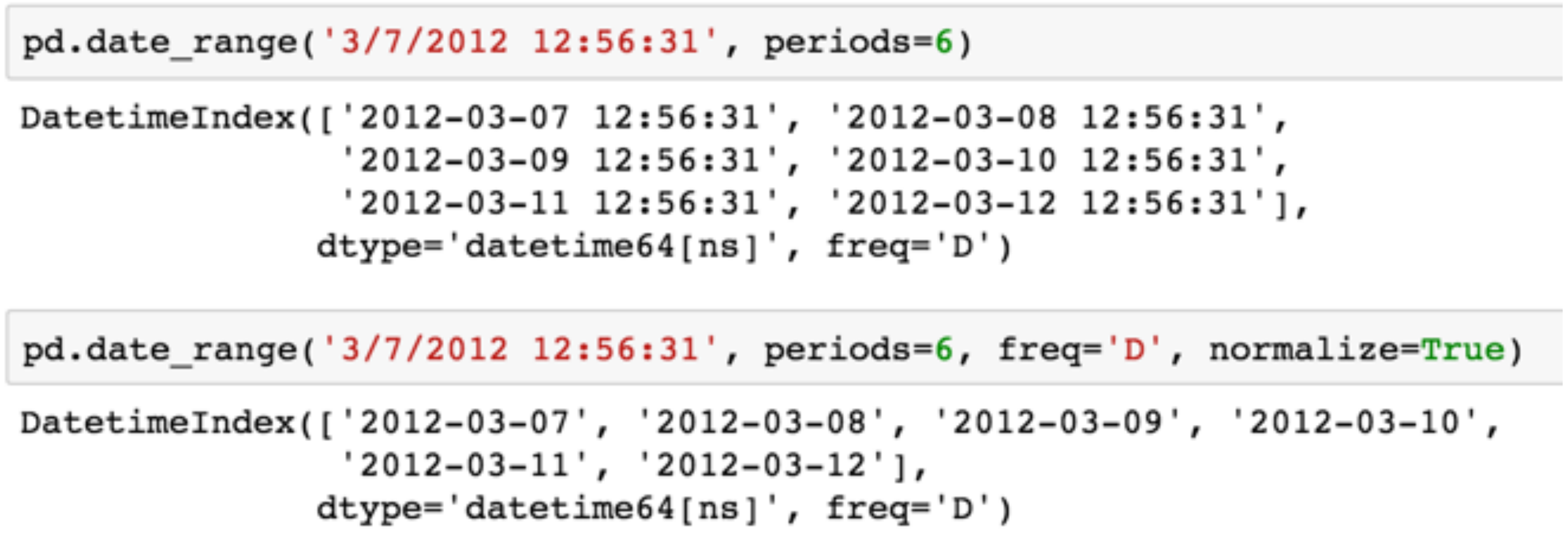


```
erng = pd.date_range(end='1/1/2016', periods=8, freq='H')
```

```
erng
```

```
DatetimeIndex(['2015-12-31 17:00:00', '2015-12-31 18:00:00',
               '2015-12-31 19:00:00', '2015-12-31 20:00:00',
               '2015-12-31 21:00:00', '2015-12-31 22:00:00',
               '2015-12-31 23:00:00', '2016-01-01 00:00:00'],
              dtype='datetime64[ns]', freq='H')
```

‣ Creating a DatetimeIndex from a Range:



```
pd.date_range('3/7/2012 12:56:31', periods=6)
```

```
DatetimeIndex(['2012-03-07 12:56:31', '2012-03-08 12:56:31',
               '2012-03-09 12:56:31', '2012-03-10 12:56:31',
               '2012-03-11 12:56:31', '2012-03-12 12:56:31'],
              dtype='datetime64[ns]', freq='D')
```

```
pd.date_range('3/7/2012 12:56:31', periods=6, freq='D', normalize=True)
```

```
DatetimeIndex(['2012-03-07', '2012-03-08', '2012-03-09', '2012-03-10',
               '2012-03-11', '2012-03-12'],
              dtype='datetime64[ns]', freq='D')
```

- Independent Exercise:
  - Build a model to forecast the S&P 500 using either SVR or RT
  - Use the ticker SPY with data since 2000
  - Your features should include the following at a minimum:
    - Close yesterday
    - Close 2 days ago
    - Close 3 days ago
  - Your target is tomorrow's close
  - You can buy at the next open and sell at the next close
  - Calculate your win ratio and total profits

- Conclusion:
  - Time series data is simply data that is ordered by date
  - Care must be used when testing the data - usually done in sliding windows (train first 3 months, test next 3 -> train first 3 shifted 1 day forward day, test next 3 plus one day forward, and so on)
  - pandas has extensive built-in functionality for working with time series data: pandas_datareader, resample, asfreq, shift, etc.
  - The DateTimeIndex is the key to this time series functionality