

# The DataReader: Access financial data online

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



Stefan Jansen  
Instructor

# pandas\_datareader

- Easy access to various financial internet data sources
- Little code needed to import into a `pandas` `DataFrame`
- Available sources include:
  - Yahoo! and Google Finance (including derivatives)
  - Federal Reserve
  - World Bank, OECD, Eurostat
  - OANDA

# Stock prices: Google Finance

```
from pandas_datareader.data import DataReader
from datetime import date # Date & time functionality
```

```
start = date(2015, 1, 1) # Default: Jan 1, 2010
end = date(2016, 12, 31) # Default: today

ticker = 'GOOG'
data_source = 'google'

stock_data = DataReader(ticker, data_source, start, end)
```

# Stock prices: Google Finance

```
stock_data.info()
```

```
DatetimeIndex: 504 entries, 2015-01-02 to 2016-12-30
Data columns (total 6 columns):
Open           504 non-null float64   # First price
High           504 non-null float64   # Highest price
Low            504 non-null float64   # Lowest price
Close          504 non-null float64   # Last price
Volume         504 non-null int64      # Nb shares traded
dtypes: float64(6), int64(1)
memory usage: 32.3 KB
```

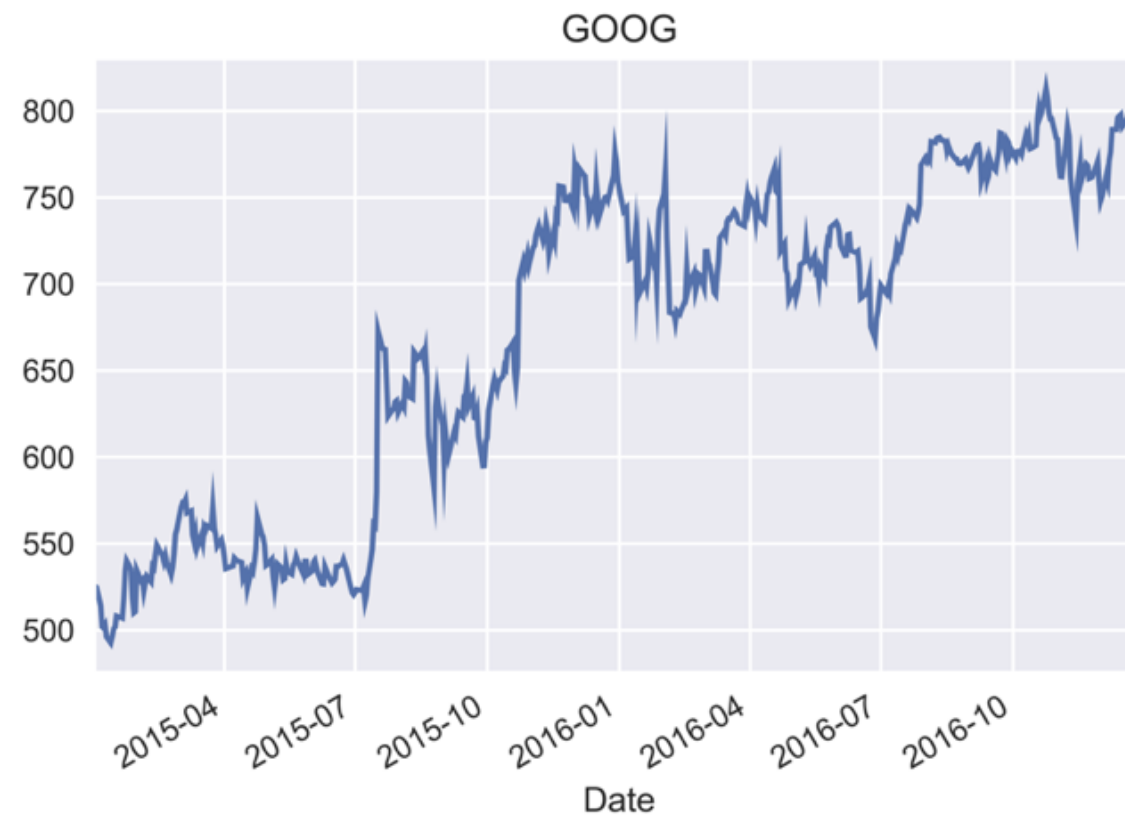
# Stock prices: Google Finance

```
pd.concat([stock_data.head(3), stock_data.tail(3)])
```

	Open	High	Low	Close	Volume
Date					
2015-01-02	529.01	531.27	524.10	524.81	1446662
2015-01-05	523.26	524.33	513.06	513.87	2054238
2015-01-06	515.00	516.18	501.05	501.96	2891950
2016-12-28	793.70	794.23	783.20	785.05	1153824
2016-12-29	783.33	785.93	778.92	782.79	744272
2016-12-30	782.75	782.78	770.41	771.82	1769950

# Stock prices: Visualization

```
import matplotlib.pyplot as plt  
stock_data['Close'].plot(title=ticker)  
plt.show()
```



# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

# Economic data from the Federal Reserve

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



Stefan Jansen  
Instructor



# Economic data from FRED



- Federal Reserve Economic Data
- 500,000 series covering a range of categories:
  - Economic growth & employment
  - Monetary & fiscal policy
  - Demographics, industries, commodity prices
  - Daily, monthly, annual frequencies

# Get data from FRED

The screenshot shows the FRED website header with the logo, 'ECONOMIC RESEARCH' title, and navigation links. A search bar contains the text 'interest rate'. Below the search bar, a message states 'Download, graph, and track 470,000 US and international time series from 84 sources.' A secondary navigation bar lists categories like 'Tag', 'Category', 'Release', etc. The main content area features a 'FRED News' section with links to recent updates and a 'PAGE ONE Economics' newsletter promotion. At the bottom, a horizontal menu includes 'AT A GLANCE', 'POPULAR SERIES', 'LATEST RELEASES', 'TOOLS', and 'NEED HELP?'.

FRED ECONOMIC DATA | ST. LOUIS FED | ECONOMIC RESEARCH | FEDERAL RESERVE BANK OF ST. LOUIS

MY ACCOUNT | SIGN OUT

Search FRED

FRED Economic Data | Information Services | Publications | Working Papers | Economists | About | St. Louis Fed Home

Download, graph, and track **470,000** US and international time series from **84** sources.

interest rate

Browse data by Tag, Category, Release, Source, Release Calendar or Get Help

**FRED News**

FRED Adds House Price Data

Moody's Corporate Bond Yields Restored in FRED

**FRED Blog**

The house price tumble in pictures

**Research News**

Diverging Forecasts

**PAGE ONE Economics**

Read the newsletter with class room application, search the glossary, and browse a data "starter set".

THE BACK STORY ON FRONT PAGE ECONOMICS

AT A GLANCE | POPULAR SERIES | LATEST RELEASES | TOOLS | NEED HELP?

<sup>1</sup> <https://fred.stlouisfed.org/>

# Get data from FRED



ECONOMIC RESEARCH

FEDERAL RESERVE BANK OF ST. LOUIS

MY ACCOUNT

SIGN OUT

interest rate

FRED Economic Data

Information Services

Publications

Working Papers

Economists

About

St. Louis Fed Home

Filter Series by Tags

With Tag:

Interest Rate

Clear All Tags

Search Tags

Sources

Releases

Seasonal Adjustments

Frequencies

Geography Types

Geographies

Concepts

Yield (466)

Bonds (451)

Corporate (276)

Discontinued (270)

Treasury (255)

Interbank (240)

Home

Search Results

1,423 series

Showing results for: interest rate

Add to Data List

Add to Graph

Sort by Search Rank

Effective Federal Funds Rate

Percent, Not Seasonally Adjusted

Monthly

Daily

Weekly

Jul 1954 to Apr 2017 (May 1)

1954-07-01 to 2017-05-05 (3 hours ago)

1954-07-07 to 2017-05-03 (4 days ago)

10-Year Treasury Constant Maturity Rate

Percent, Not Seasonally Adjusted

Daily

Monthly

Weekly

1962-01-02 to 2017-05-05 (3 hours ago)

Apr 1953 to Apr 2017 (May 1)

1962-01-05 to 2017-05-05 (3 hours ago)

<sup>1</sup> <https://fred.stlouisfed.org/>

# Get data from FRED



<sup>1</sup> <https://fred.stlouisfed.org/>

# Interest rates

```
from pandas_datareader.data import DataReader
from datetime import date

series_code = 'DGS10' # 10-year Treasury Rate
data_source = 'fred' # FED Economic Data Service
start = date(1962, 1, 1)

data = DataReader(series_code, data_source, start)

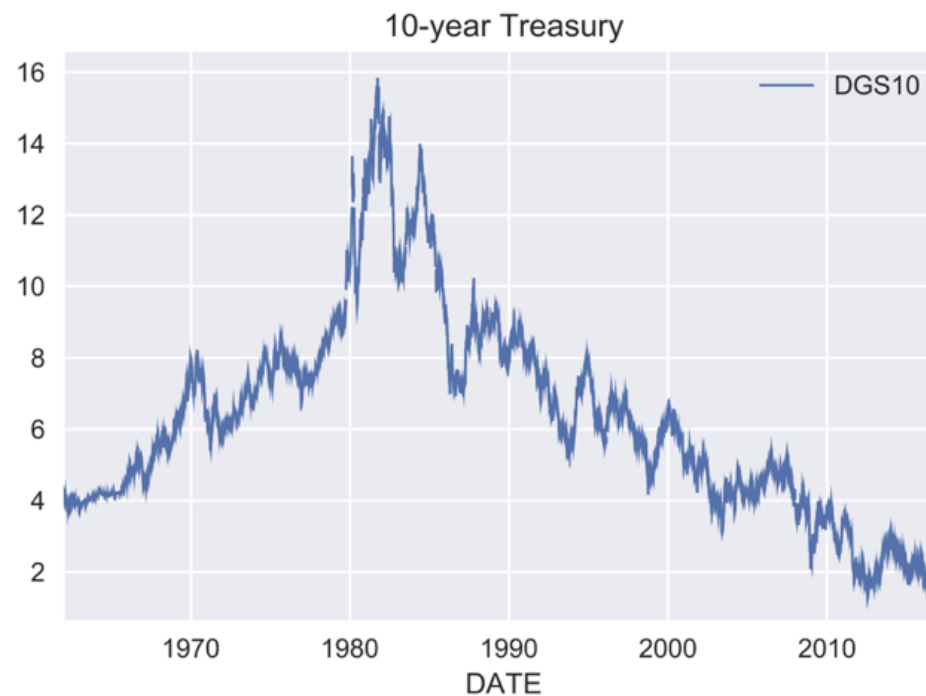
data.info()
```

```
DatetimeIndex: 14439 entries, 1962-01-02 to 2017-05-05
Data columns (total 1 columns):
DGS10      13821 non-null float64
dtypes: float64(1)
```

# Stock prices: Visualization

- `.rename(columns={old_name: new_name})`

```
series_name = '10-year Treasury'  
data = data.rename(columns={series_code: series_name})  
data.plot(title=series_name); plt.show()
```



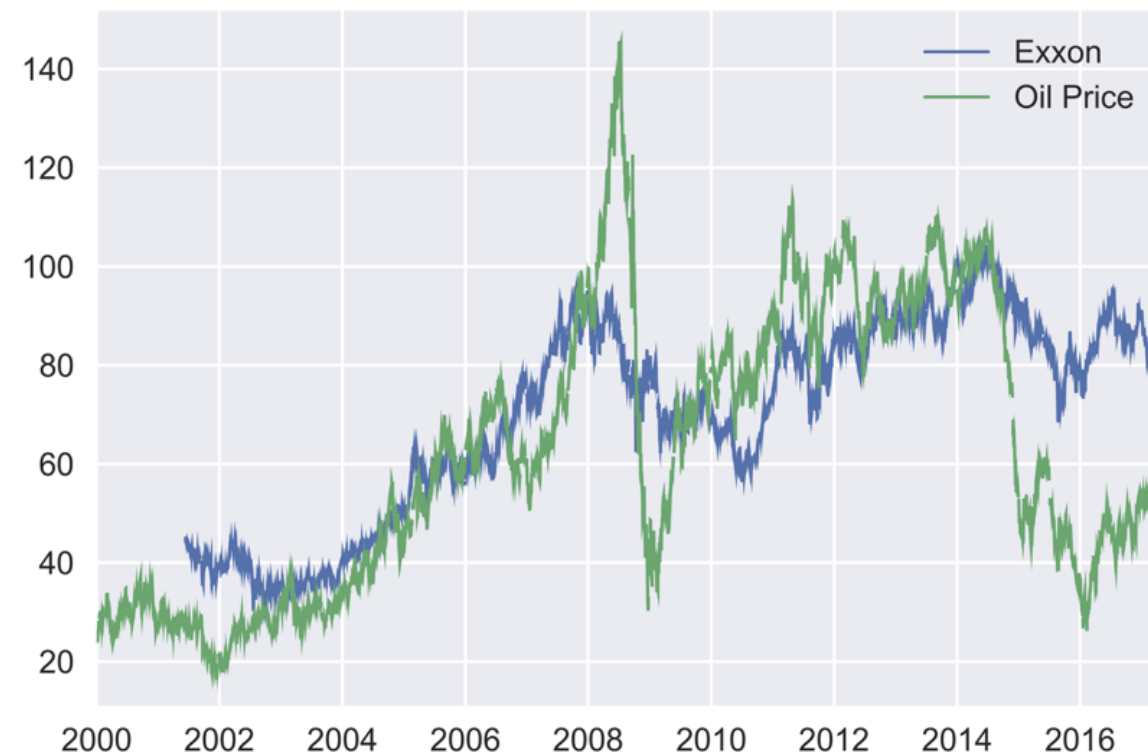
# Combine stock and economic data

```
start = date(2000, 1, 1)
series = 'DCOILWTICO' # West Texas Intermediate Oil Price
oil = DataReader(series, 'fred', start)
ticker = 'XOM' # Exxon Mobile Corporation
stock = DataReader(ticker, 'google', start)
data = pd.concat([stock[['Close']], oil], axis=1)
data.info()
```

```
DatetimeIndex: 4526 entries, 2000-01-03 to 2017-05-08
Data columns (total 2 columns):
Close          4364 non-null float64
DCOILWTICO     4352 non-null float64
```

# Combine stock and economic data

```
data.columns = ['Exxon', 'Oil Price']  
data.plot()  
plt.show()
```





# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

# Select stocks and get data from Google Finance

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



**Stefan Jansen**  
Instructor

# Select stocks based on criteria

- Use the listing information to select specific stocks
- As criteria:
  - Stock Exchange
  - Sector or Industry
  - IPO Year
  - Market Capitalization

# Get ticker for largest company

```
nyse = pd.read_excel('listings.xlsx', sheetname='nyse', na_values='n/a')
nyse = nyse.sort_values('Market Capitalization', ascending=False)
nyse[['Stock Symbol', 'Company Name']].head(3)
```

	Stock Symbol	Company Name
1586	JNJ	Johnson & Johnson
1125	XOM	Exxon Mobil Corporation
1548	JPM	J P Morgan Chase & Co

```
largest_by_market_cap = nyse.iloc[0] # 1st row
largest_by_market_cap['Stock Symbol'] # Select row label
```

```
'JNJ'
```

# Get ticker for largest company

```
nyse = nyse.set_index('Stock Symbol') # Stock ticker as index
nyse.info()
```

```
Index: 3147 entries, JNJ to EAE
```

```
Data columns (total 6 columns):
```

```
Company Name      3147 non-null object
Last Sale         3147 non-null object
Market Capitalization 3147 non-null float64
...
```

```
nyse['Market Capitalization'].idxmax() # Index of max value
```

The method returns the index value that corresponds to the largest market capitalization.

```
'JNJ'
```

# Get ticker for largest tech company

```
nyse['Sector'].unique() # Unique values as numpy array
```

```
array(['Technology', 'Health Care', ...], dtype=object)
```

```
tech = nyse.loc[nyse.Sector == 'Technology']  
tech['Company Name'].head(2)
```

Stock Symbol	Company Name
ORCL	Oracle Corporation
TSM	Taiwan Semiconductor Manufacturing

```
nyse.loc[nyse.Sector=='Technology', 'Market Capitalization'].idxmax()
```

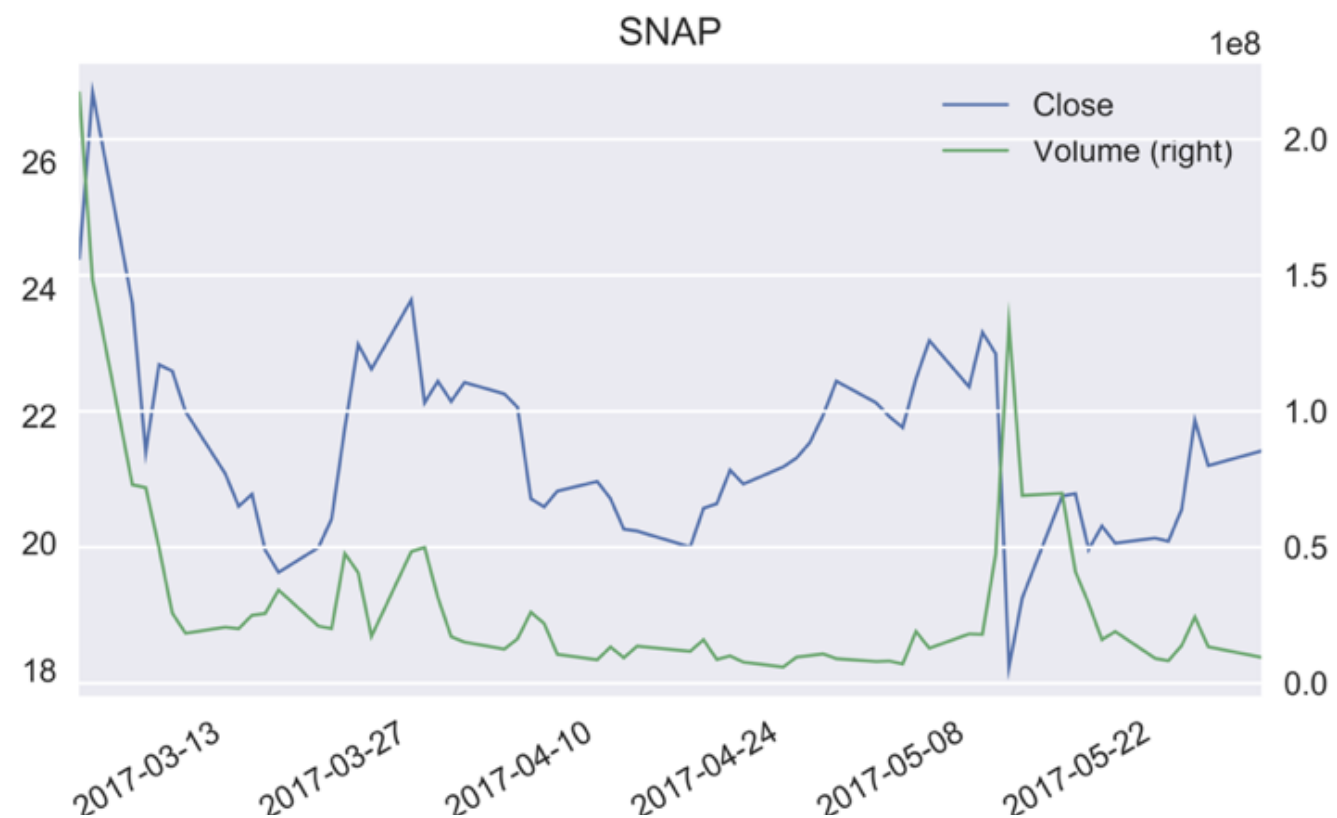
```
'ORCL'
```

# Get data for largest tech company with 2017 IPO

```
ticker = nyse.loc[(nyse.Sector=='Technology') & (nyse['IPO Year']==2017),  
                  'Market Capitalization'].idxmax()  
  
data = DataReader(ticker, 'google') # Start: 2010/1/1  
data = data.loc[:, ['Close', 'Volume']]
```

# Visualize price and volume on two axes

```
import matplotlib.pyplot as plt
data.plot(title=ticker, secondary_y='Volume')
plt.tight_layout(); plt.show()
    improve the layout by reducing whitespace
```





# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

# Get several stocks & manage a MultiIndex

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON



Stefan Jansen  
Instructor

# Get data for several stocks

- Use the listing information to select multiple stocks
  - E.g. largest 3 stocks per sector
- Use Google Finance to retrieve data for several stocks
- Learn how to manage a `pandas MultiIndex`, a powerful tool to deal with more complex data sets

# Load prices for top 5 companies

```
nasdaq = pd.read_excel('listings.xlsx', sheetname='nasdaq', na_values='n/a')
nasdaq.set_index('Stock Symbol', inplace=True)
top_5 = nasdaq['Market Capitalization'].nlargest(n=5) # Top 5
top_5.div(1000000) # Market Cap in million USD
```

```
AAPL      740024.467000
GOOG      569426.124504
...
Name: Market Capitalization, dtype: float64
```

```
tickers = top_5.index.tolist() # Convert index to list
```

```
['AAPL', 'GOOG', 'MSFT', 'AMZN', 'FB']
```

# Load prices for top 5 companies

```
panel = DataReader(tickers, 'google', start=date(2015, 1, 1))
```

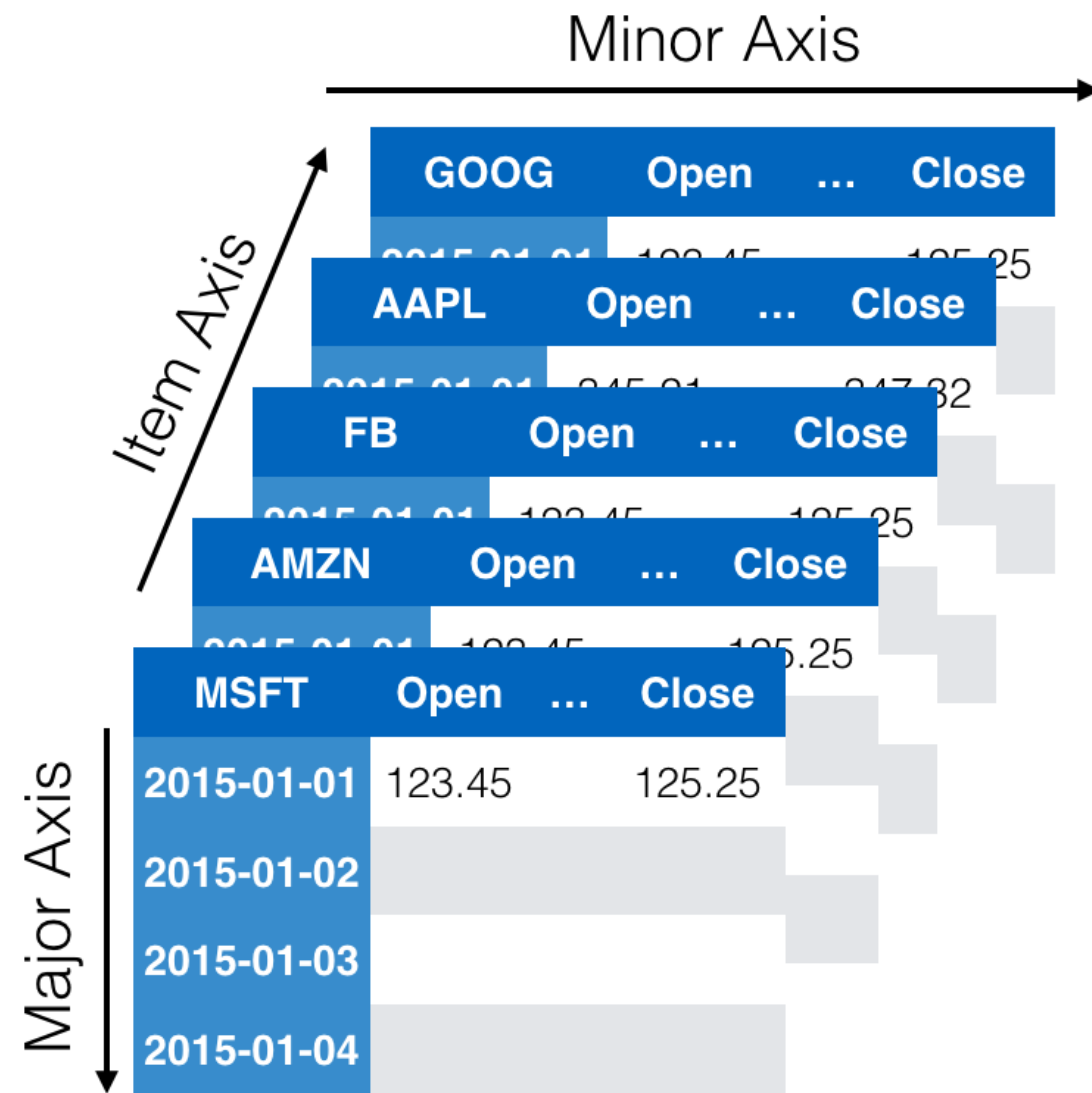
```
<class 'pandas.core.panel.Panel'>  
Dimensions: 5 (items) x 591 (major_axis) x 5 (minor_axis)  
Items axis: Open to Volume  
Major_axis axis: 2015-01-02 to 2017-05-08  
Minor_axis axis: AAPL to MSFT
```

```
data = panel.to_frame()  
data.info()
```

```
MultiIndex: 2955 entries, (2015-01-02, AAPL) to (2017-05-08, MSFT)  
Data columns (total 5 columns):  
Open      2955 non-null float64  
...
```

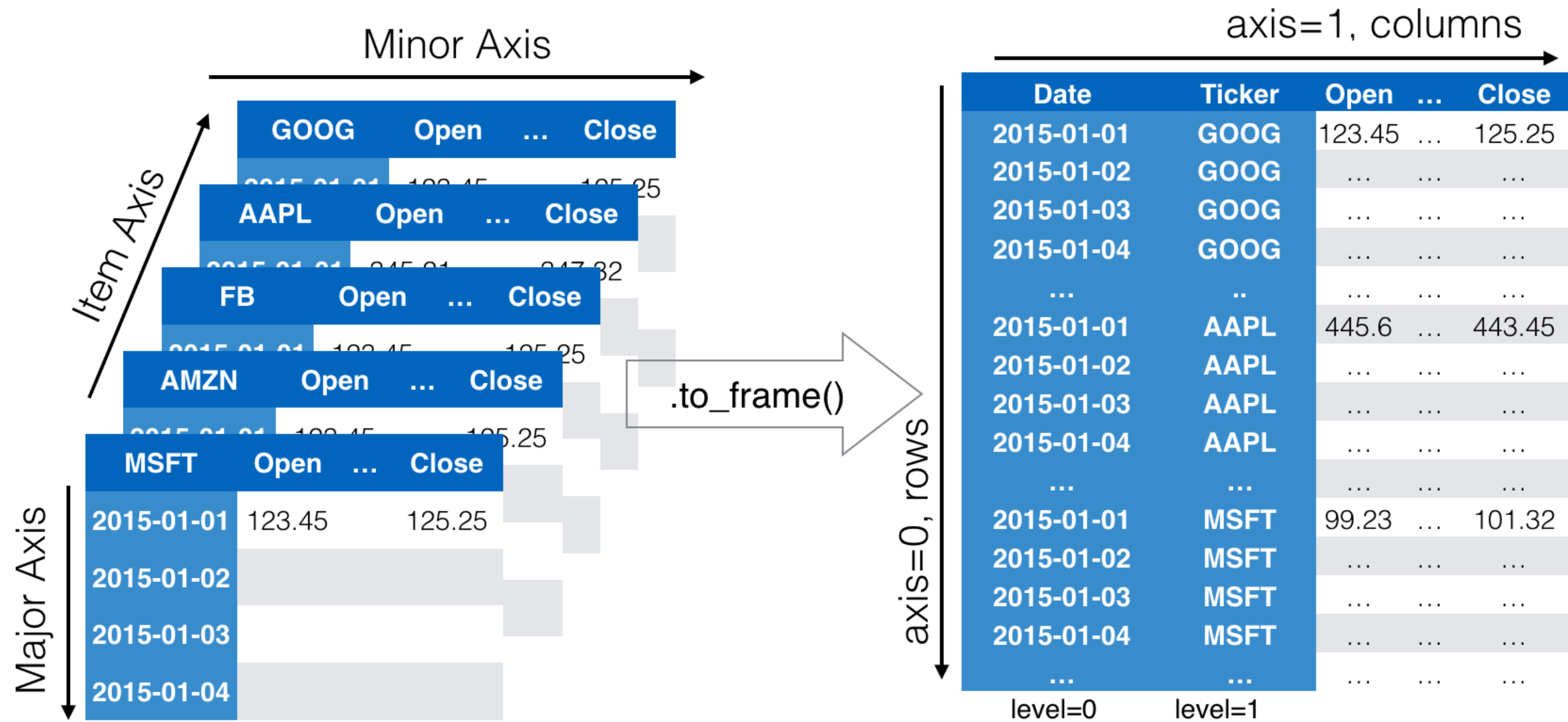
# Into higher dimensions: MultiIndex

- `.to_frame()` : from `pd.Panel()` to `pd.DataFrame()`



# Into higher dimensions: MultiIndex

- `.to_frame()` : from `pd.Panel()` to `pd.DataFrame()`



# Reshape your data: .unstack()

```
unstacked = data['Close'].unstack()  
unstacked.info()
```

```
DatetimeIndex: 591 entries, 2015-01-02 to 2017-05-08  
Data columns (total 5 columns):  
AAPL      591 non-null float64  
AMZN      591 non-null float64  
FB         591 non-null float64  
GOOG      591 non-null float64  
MSFT      591 non-null float64  
dtypes: float64(5)  
memory usage: 27.7 KB
```



# From long to wide format

```
unstacked = data['Close'].unstack() # Results in DataFrame
```

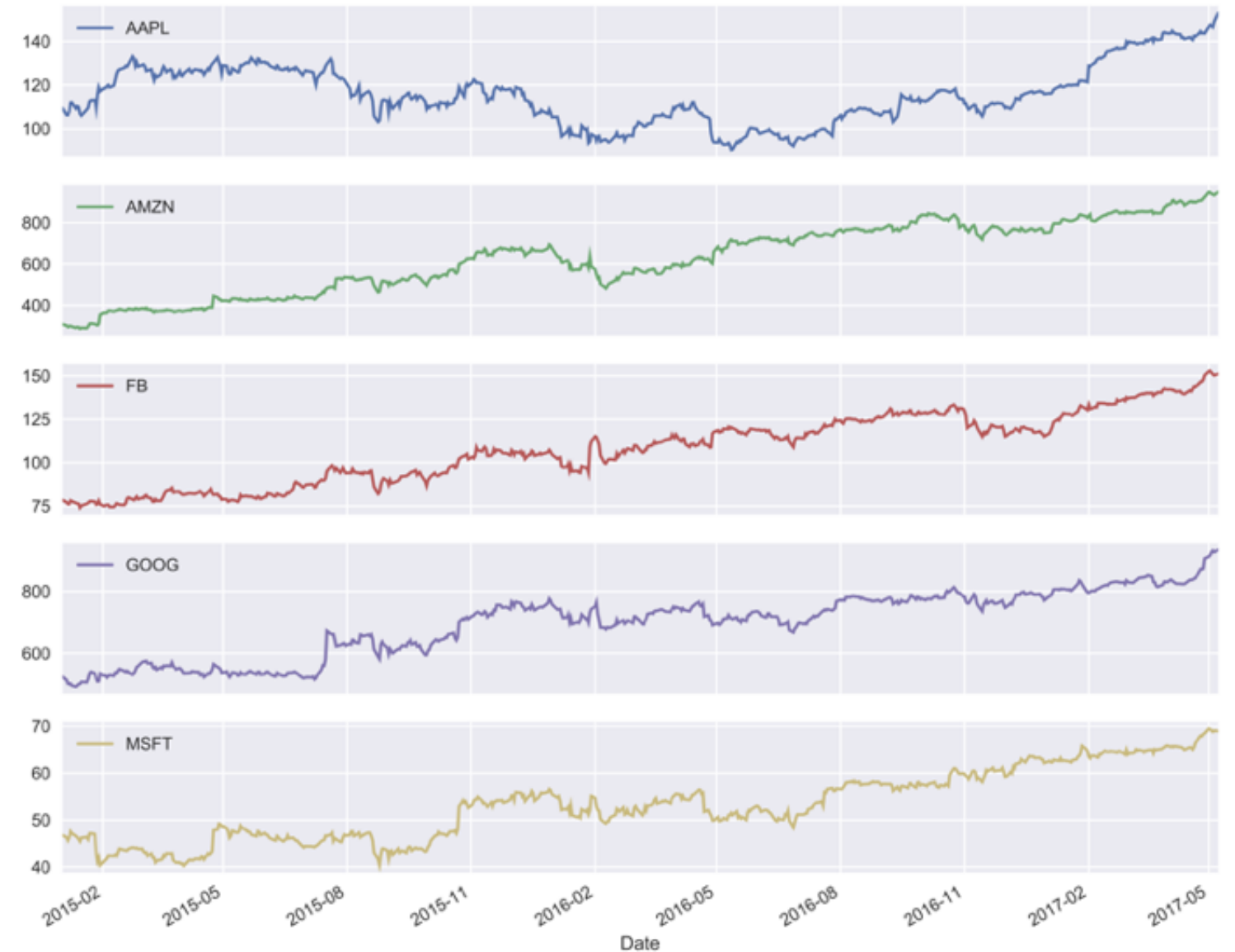
Date	Ticker	
2015-01-01	GOOG	123.45
2015-01-02	GOOG	...
2015-01-03	GOOG	...
2015-01-04	GOOG	...
...	..	...
2015-01-01	AAPL	445.64
2015-01-02	AAPL	...
2015-01-03	AAPL	...
2015-01-04	AAPL	...
...	...	...
2015-01-01	MSFT	99.23
2015-01-02	MSFT	...
2015-01-03	MSFT	...
2015-01-04	MSFT	...

.unstack()

Date	GOOG	AAPL	AMZN	FB	MSFT
2015-01-01	123.45	445.64	...	...	99.23
2015-01-02	...	...	...	...	...
2015-01-03	...	...	...	...	...
2015-01-04	...	...	...	...	...
...	...	...	...	...	...
2017-05-01	...	...	...	...	...
2017-05-02	...	...	...	...	...
2017-05-03	...	...	...	...	...
2017-05-04	...	...	...	...	...

# Stock prices: Visualization

```
unstacked.plot(subplots=True)  
plt.tight_layout(); plt.show()
```



# Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON