

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:
 - IEX and Yahoo! Finance (including derivatives)
 - Federal Reserve
 - World Bank, OECD, Eurostat
 - OANDA

Stock prices: Yahoo! 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 = 'yahoo'
stock_data = DataReader(ticker, data_source, start, end)
```

Stock prices: Yahoo! Finance

```
stock_data.info()
```

```
DatetimeIndex: 504 entries, 2015-01-02 to 2016-12-30
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
--  --
0   High         504 non-null    float64 # First price
1   Low          504 non-null    float64 # Highest price
2   Open         504 non-null    float64 # Lowest price
3   Close        504 non-null    float64 # Last price
4   Volume       504 non-null    float64 # No shares traded
5   Adj Close    504 non-null    float64 # Adj. price
dtypes: float64(6)
```

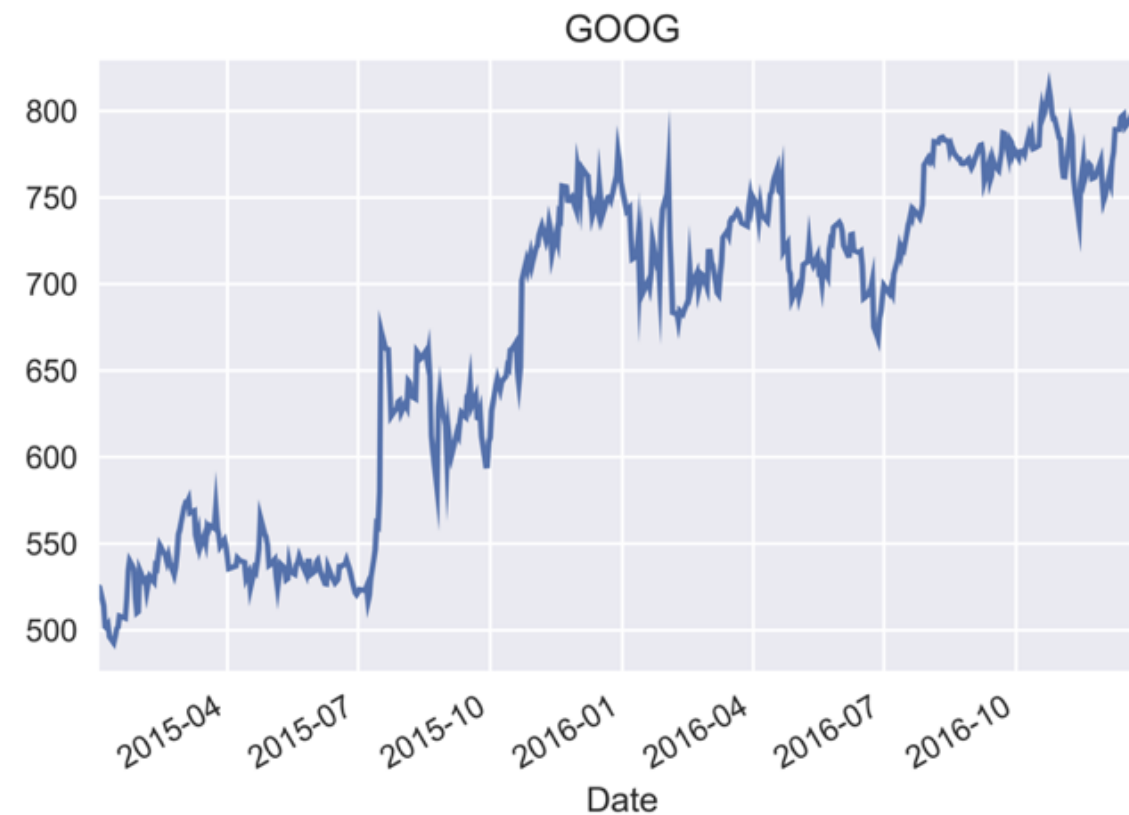
Stock prices: Yahoo! Finance

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

	High	Low	Open	Close	Volume	Adj Close
Date						
2015-01-02	26.49	26.13	26.38	26.17	28951268	26.17
2015-01-05	26.14	25.58	26.09	25.62	41196796	25.62
2015-01-06	25.74	24.98	25.68	25.03	57998800	25.03
2016-12-28	39.71	39.16	39.69	39.25	23076000	39.25
2016-12-29	39.30	38.95	39.17	39.14	14886000	39.14
2016-12-30	39.14	38.52	39.14	38.59	35400000	38.59

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 banner states "Download, graph, and track 470,000 US and international time series from 84 sources." and provides browsing options. The main content area features a "FRED News" section with links to "FRED Adds House Price Data" and "Moody's Corporate Bond Yields Restored in FRED", a "FRED Blog" section with "The house price tumble in pictures", and a "Research News" section with "Diverging Forecasts". To the right, a "PAGE ONE Economics" sidebar promotes a newsletter. At the bottom, a navigation bar includes links for "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?

¹ <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)

¹ <https://fred.stlouisfed.org/>

Get data from FRED



¹ <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: 15754 entries, 1962-01-02 to 2022-05-20
Data columns (total 1 columns):
#   Column  Non-Null Count  Dtype
--  --
0   DGS10    15083 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, 'yanoo', start)
data = pd.concat([stock[['Close']], oil], axis=1)
data.info()
```

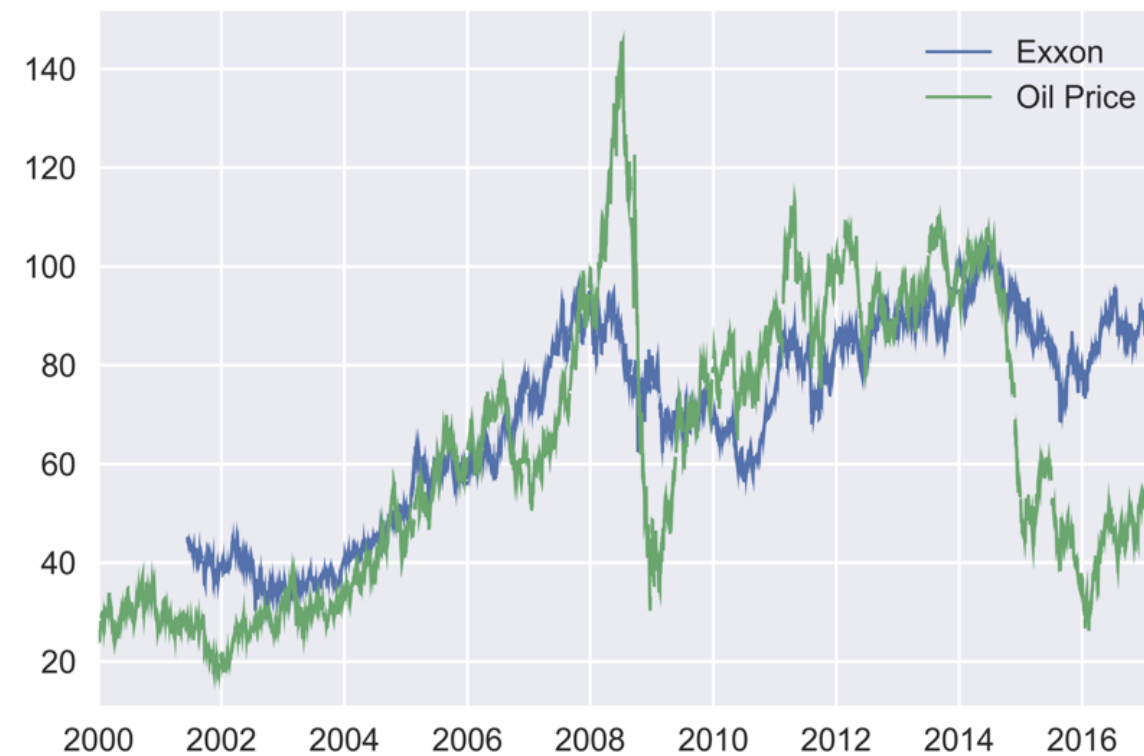
```
DatetimeIndex: 5841 entries, 2000-01-03 to 2022-05-23
```

```
Data columns (total 2 columns):
```

#	Column	Non-Null Count	Dtype
0	Close	5634 non-null	float64
1	DCOILWTICO	5615 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 Yahoo! 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', sheet_name='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):
```

#	Column	Non-Null Count	Dtype
--	-----	-----	-----
0	Company Name	3147 non-null	object
1	Last Sale	3079 non-null	float64
2	Market Capitalization	3147 non-null	float64
...			

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

```
'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()
```

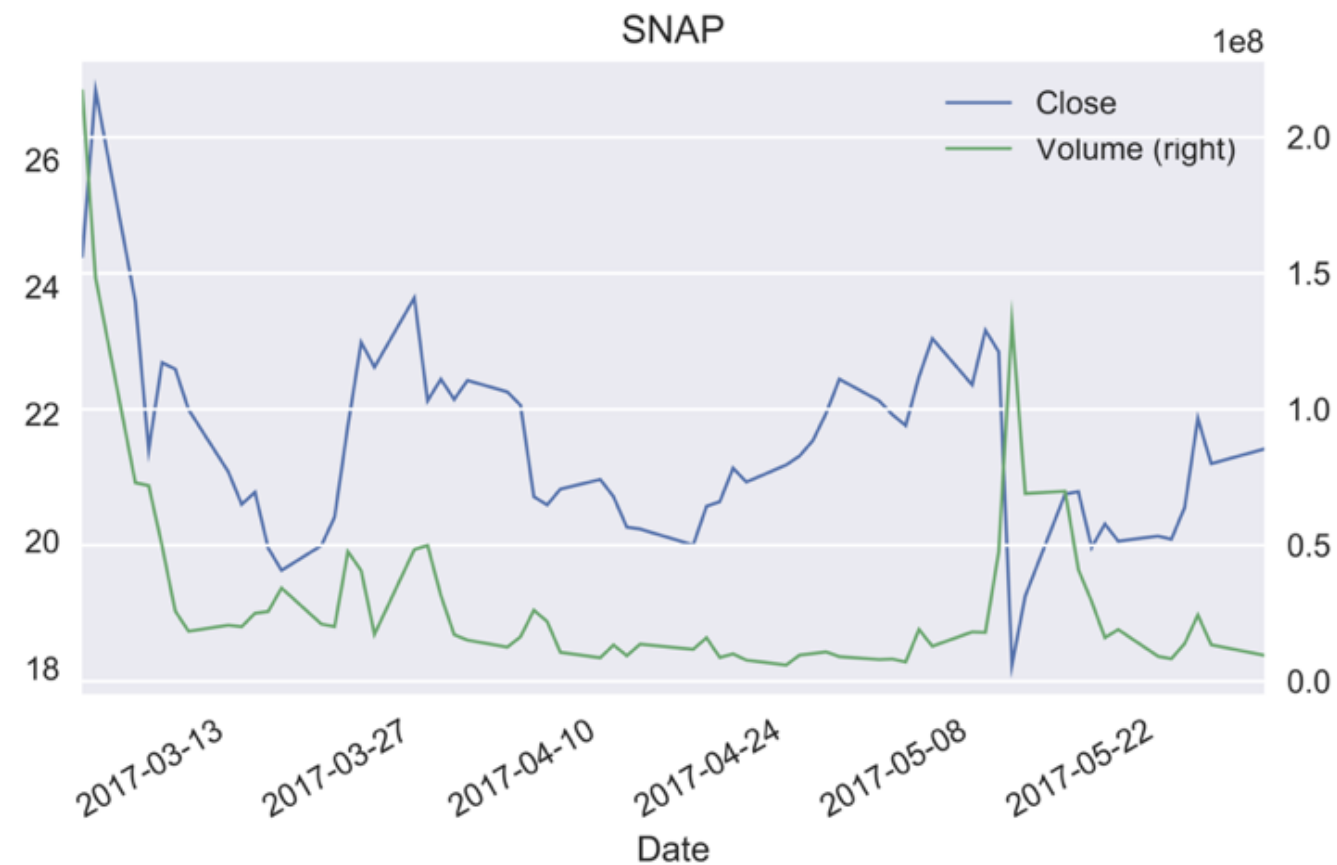
```
'ORCL'
```

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, 'yahoo') # 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()
```

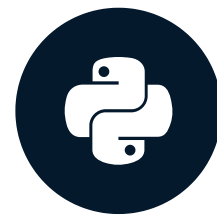


Let's practice!

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 Yahoo! 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', sheet_name='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

```
df = DataReader(tickers, 'yahoo', start=date(2020, 1, 1))
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 712 entries, 2020-01-02 to 2022-10-27
Data columns (total 30 columns):
#   Column                Non-Null Count  Dtype
--  --
0   (Adj Close, AAPL)      712 non-null   float64
1   (Adj Close, GOOG)      712 non-null   float64
2   (Adj Close, MSFT)      712 non-null   float64
...
28  (Volume, AMZN)         712 non-null   float64
29  (Volume, FB)           253 non-null   float64
dtypes: float64(30)
memory usage: 172.4 KB
```

```
df = df.stack()
```

Load prices for top 5 companies

```
df.info()
```

```
MultiIndex: 3101 entries, (Timestamp('2020-01-02 00:00:00'), 'AAPL') to (Timestamp('2020-01-02 00:00:00'), 'MSFT')
```

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

#	Column	Non-Null Count	Dtype
--	-----	-----	-----
0	Adj Close	3101 non-null	float64
...			

Reshape your data: .unstack()

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

```
DatetimeIndex: 712 entries, 2020-01-02 to 2022-10-27
```

```
Data columns (total 5 columns):
```

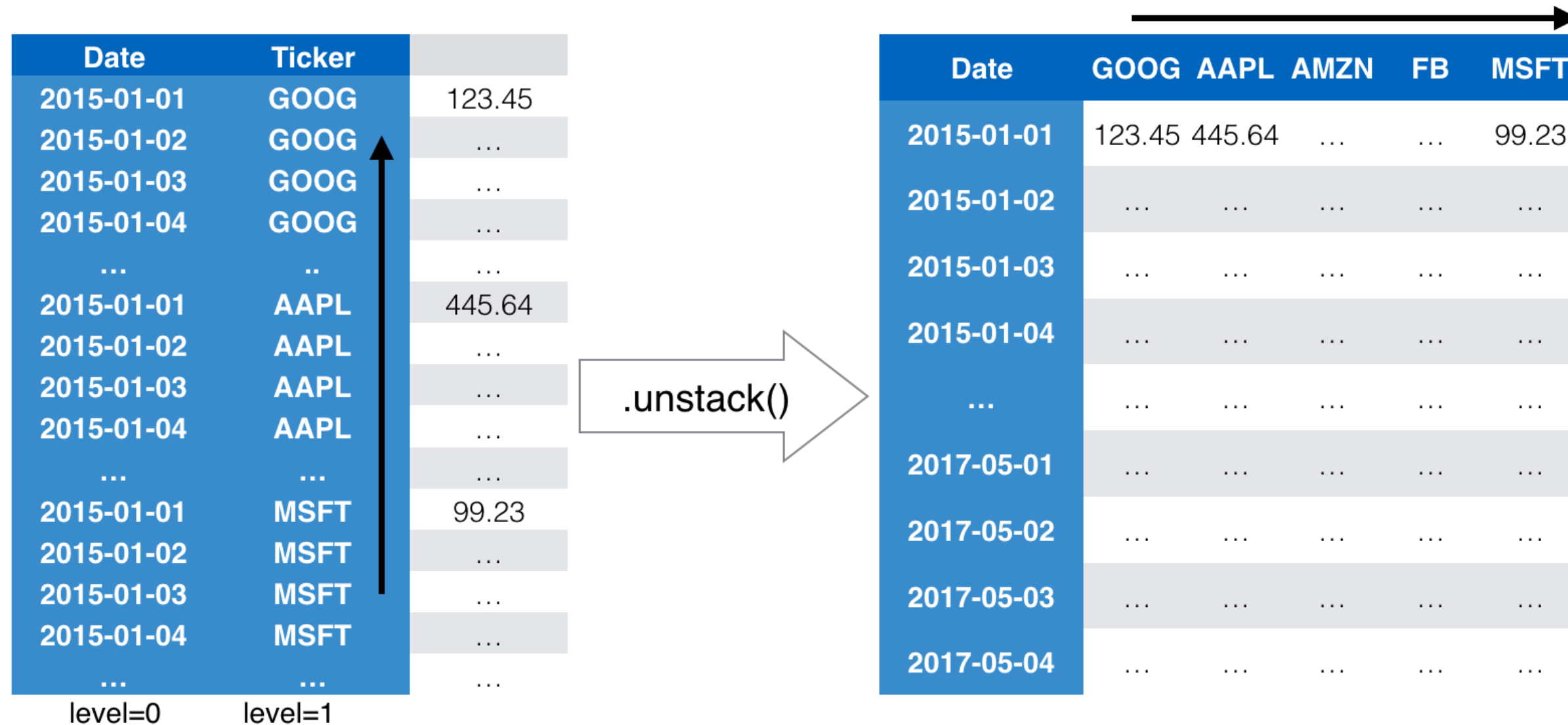
#	Column	Non-Null Count	Dtype
--	-----	-----	-----
0	AAPL	712 non-null	float64
1	GOOG	712 non-null	float64
2	MSFT	712 non-null	float64
3	AMZN	712 non-null	float64
4	FB	253 non-null	float64

```
dtypes: float64(5)
```

```
memory usage: 33.4 KB
```

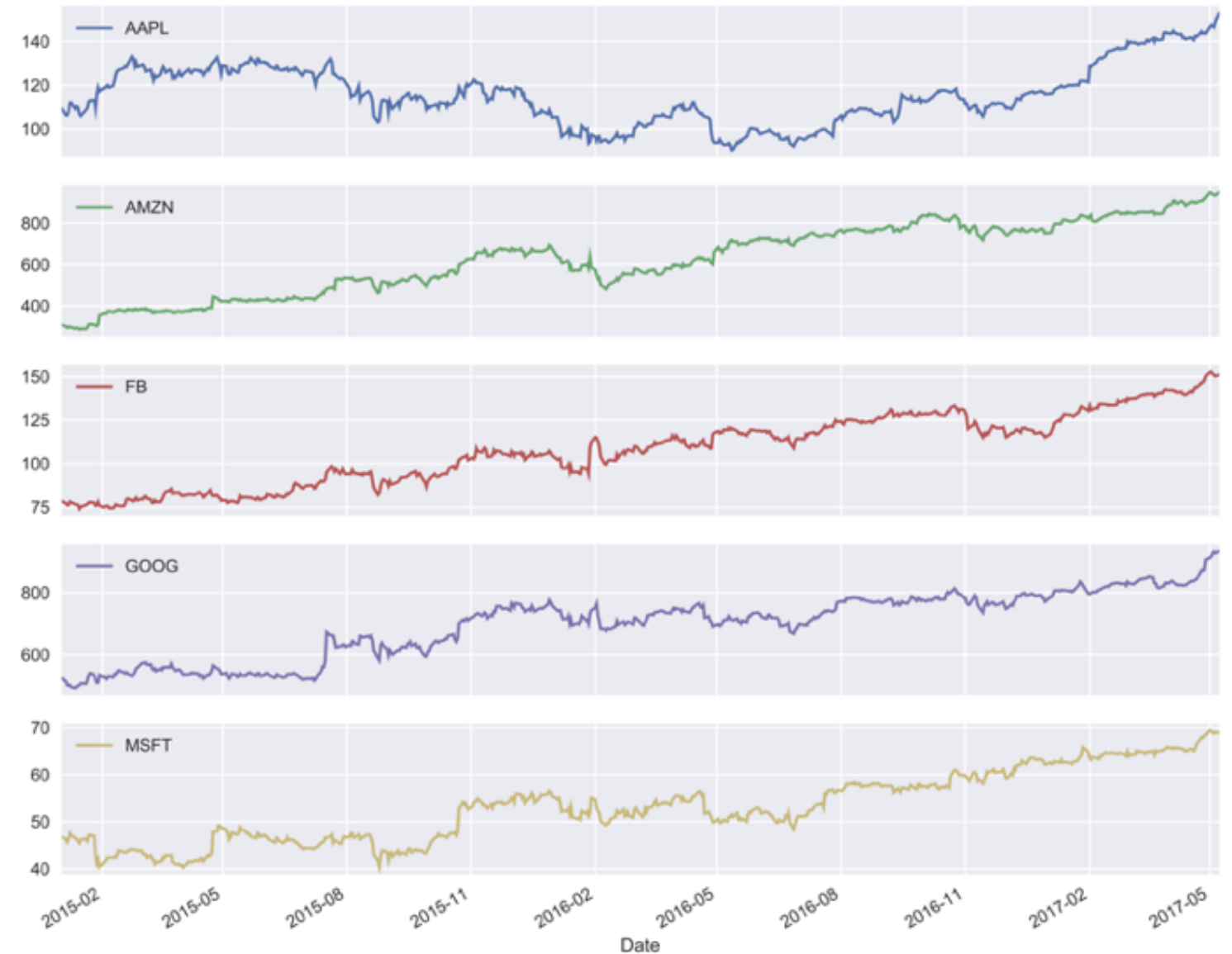
From long to wide format

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



Stock prices: Visualization

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



Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON