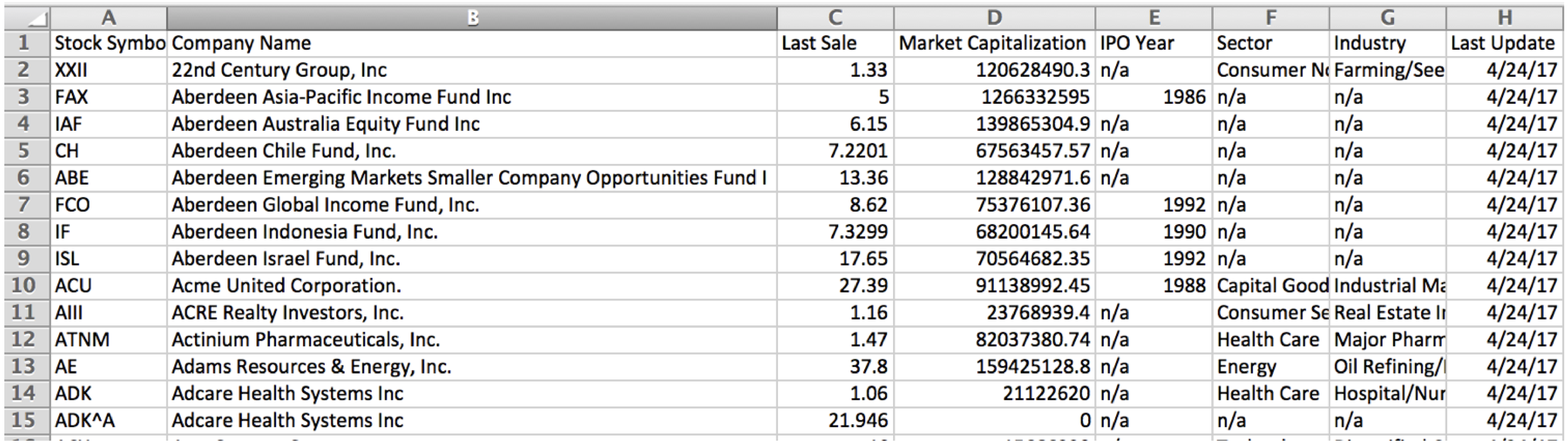
Reading, inspecting, and cleaning data from CSV

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

Stefan Jansen
Instructor

Import and clean data

- Ensure that `pd.DataFrame()` is same as CSV source file
- Stock exchange listings: `amex-listings.csv`



	A	B	C	D	E	F	G	H
1	Stock Symbo	Company Name	Last Sale	Market Capitalization	IPO Year	Sector	Industry	Last Update
2	XXII	22nd Century Group, Inc	1.33	120628490.3	n/a	Consumer No	Farming/See	4/24/17
3	FAX	Aberdeen Asia-Pacific Income Fund Inc	5	1266332595	1986	n/a	n/a	4/24/17
4	IAF	Aberdeen Australia Equity Fund Inc	6.15	139865304.9	n/a	n/a	n/a	4/24/17
5	CH	Aberdeen Chile Fund, Inc.	7.2201	67563457.57	n/a	n/a	n/a	4/24/17
6	ABE	Aberdeen Emerging Markets Smaller Company Opportunities Fund I	13.36	128842971.6	n/a	n/a	n/a	4/24/17
7	FCO	Aberdeen Global Income Fund, Inc.	8.62	75376107.36	1992	n/a	n/a	4/24/17
8	IF	Aberdeen Indonesia Fund, Inc.	7.3299	68200145.64	1990	n/a	n/a	4/24/17
9	ISL	Aberdeen Israel Fund, Inc.	17.65	70564682.35	1992	n/a	n/a	4/24/17
10	ACU	Acme United Corporation.	27.39	91138992.45	1988	Capital Good	Industrial Ma	4/24/17
11	AIII	ACRE Realty Investors, Inc.	1.16	23768939.4	n/a	Consumer Se	Real Estate Ir	4/24/17
12	ATNM	Actinium Pharmaceuticals, Inc.	1.47	82037380.74	n/a	Health Care	Major Pharm	4/24/17
13	AE	Adams Resources & Energy, Inc.	37.8	159425128.8	n/a	Energy	Oil Refining/I	4/24/17
14	ADK	Adcare Health Systems Inc	1.06	21122620	n/a	Health Care	Hospital/Nur	4/24/17
15	ADK^A	Adcare Health Systems Inc	21.946	0	n/a	n/a	n/a	4/24/17

How pandas stores data

- Each column has its own data format (`dtype`)
- `dtype` affects your calculation and visualization

pandas **dtype**	**Column characteristics**
`object`	Text, or a mix of text and numeric data
`int64`	Numeric: whole numbers - 64 bits ($\leq 2^{64}$)
`float64`	Numeric: Decimals, or whole numbers with missing values
`datetime64`	Date and time information

Import & inspect

```python
import pandas as pd
amex = pd.read_csv('amex-listings.csv')
amex.info() # To inspect table structure & data types
```

```
RangeIndex: 360 entries, 0 to 359
Data columns (total 8 columns):
Stock Symbol             360 non-null object
Company Name             360 non-null object
Last Sale                360 non-null object
Market Capitalization    360 non-null float64
IPO Year                 360 non-null object
Sector                   360 non-null object
Industry                 360 non-null object
Last Update              360 non-null object
dtypes: float64(1), object(7)
```

Dealing with missing values

```
# Replace 'n/a' with np.nan
amex = pd.read_csv('amex-listings.csv', na_values='n/a')
amex.info()
```

Pandas will replace them with the numpy value np.nan, which stands for not-a-number.

```
RangeIndex: 360 entries, 0 to 359
Data columns (total 8 columns):
Stock Symbol             360 non-null object
Company Name             360 non-null object
Last Sale                346 non-null float64
Market Capitalization    360 non-null float64
IPO Year                 105 non-null float64
Sector                   238 non-null object
Industry                 238 non-null object
Last Update              360 non-null object
dtypes: float64(3), object(5)
```

Properly parsing dates

```
amex = pd.read_csv('amex-listings.csv',
                   na_values='n/a',
                   parse_dates=['Last Update'])
amex.info()
```

Pass a list with the names of one or several columns with date information.

```
RangeIndex: 360 entries, 0 to 359
Data columns (total 8 columns):
Stock Symbol             360 non-null object
Company Name             360 non-null object
Last Sale                346 non-null float64
Market Capitalization    360 non-null float64
IPO Year                 105 non-null float64
Sector                   238 non-null object
Industry                 238 non-null object
Last Update              360 non-null datetime64[ns]
dtypes: datetime64[ns](1) float64(3), object(4)
```

Showing off the result

```
amex.head(2) # Show first n rows (default: 5)
```

```
  Stock Symbol   Company Name
0         XXII   22nd Century Group, Inc
1          FAX   Aberdeen Asia-Pacific Income Fund Inc

   Last Sale  Market Capitalization  IPO Year
0     1.3300           1.206285e+08       NaN
1     5.0000           1.266333e+09    1986.0

   Sector         Industry              Last Update
0  Non-Durables   Farming/Seeds/Milling  2017-04-26
1  NaN            NaN                    2017-04-25
```

Let's practice!

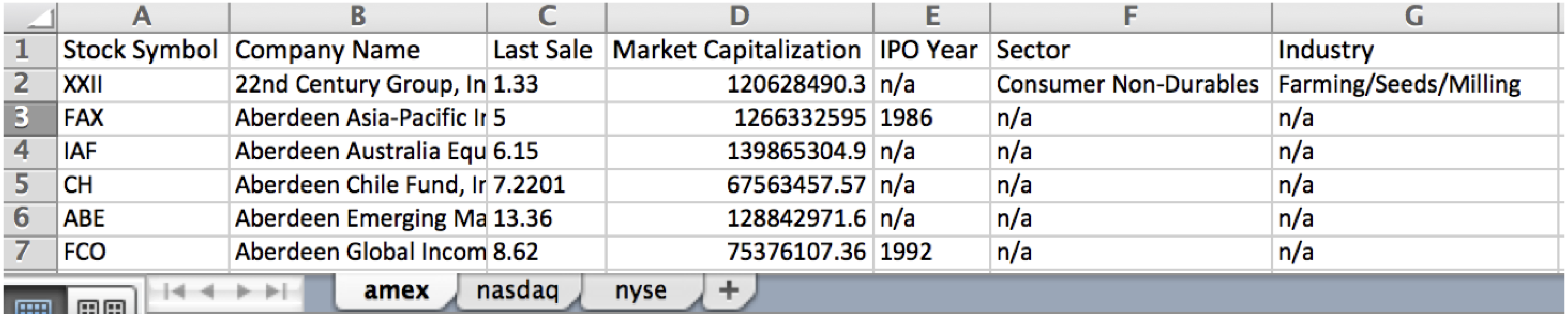
IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

Read data from Excel worksheets

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

Stefan Jansen
Instructor

Import data from Excel



	A	B	C	D	E	F	G
1	Stock Symbol	Company Name	Last Sale	Market Capitalization	IPO Year	Sector	Industry
2	XXII	22nd Century Group, In	1.33	120628490.3	n/a	Consumer Non-Durables	Farming/Seeds/Milling
3	FAX	Aberdeen Asia-Pacific Ir	5	1266332595	1986	n/a	n/a
4	IAF	Aberdeen Australia Equ	6.15	139865304.9	n/a	n/a	n/a
5	CH	Aberdeen Chile Fund, Ir	7.2201	67563457.57	n/a	n/a	n/a
6	ABE	Aberdeen Emerging Ma	13.36	128842971.6	n/a	n/a	n/a
7	FCO	Aberdeen Global Incom	8.62	75376107.36	1992	n/a	n/a

- `pd.read_excel(file, sheetname=0)`
  - Select first sheet by default with `sheetname=0`
  - Select by name with `sheetname='amex'`
  - Import several sheets with list such as `sheetname=['amex', 'nasdaq']`

Import data from one sheet

```
amex = pd.read_excel('listings.xlsx',
                     sheetname='amex',
                     na_values='n/a')

amex.info()
```

```
RangeIndex: 360 entries, 0 to 359
Data columns (total 8 columns):
Stock Symbol             360 non-null object
Company Name             360 non-null object
Last Sale                346 non-null float64
Market Capitalization    360 non-null float64
IPO Year                 105 non-null float64
...
```

Import data from two sheets

```
listings = pd.read_excel('listings.xlsx',
                         sheetname=['amex', 'nasdaq'],    # keys = sheet name
                         na_values='n/a')                 # values = DataFrame

listings['nasdaq'].info()
```

The result contained in the variable 'listings' is a dictionary that contains two key-value pairs.

```
RangeIndex: 3167 entries, 0 to 3166
Data columns (total 7 columns):
Stock Symbol             3167 non-null object
Company Name             3167 non-null object
Last Sale                3165 non-null float64
Market Capitalization    3167 non-null float64
IPO Year                 1386 non-null float64
...
```

Get sheet names

```
xls = pd.ExcelFile('listings.xlsx') # pd.ExcelFile object

exchanges = xls.sheet_names

exchanges
```

```
['amex', 'nasdaq', 'nyse']
```

```
nyse = pd.read_excel(xls,
                     sheetname=exchanges[2],
                     na_values='n/a')
```

Get sheet names

```
nyse.info()
```

```
RangeIndex: 3147 entries, 0 to 3146
Data columns (total 7 columns):
Stock Symbol            3147 non-null object
Company Name            3147 non-null object
...                     ...
Industry                2177 non-null object
dtypes: float64(3), object(4)
memory usage: 172.2+ KB
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

Combine data from multiple worksheets

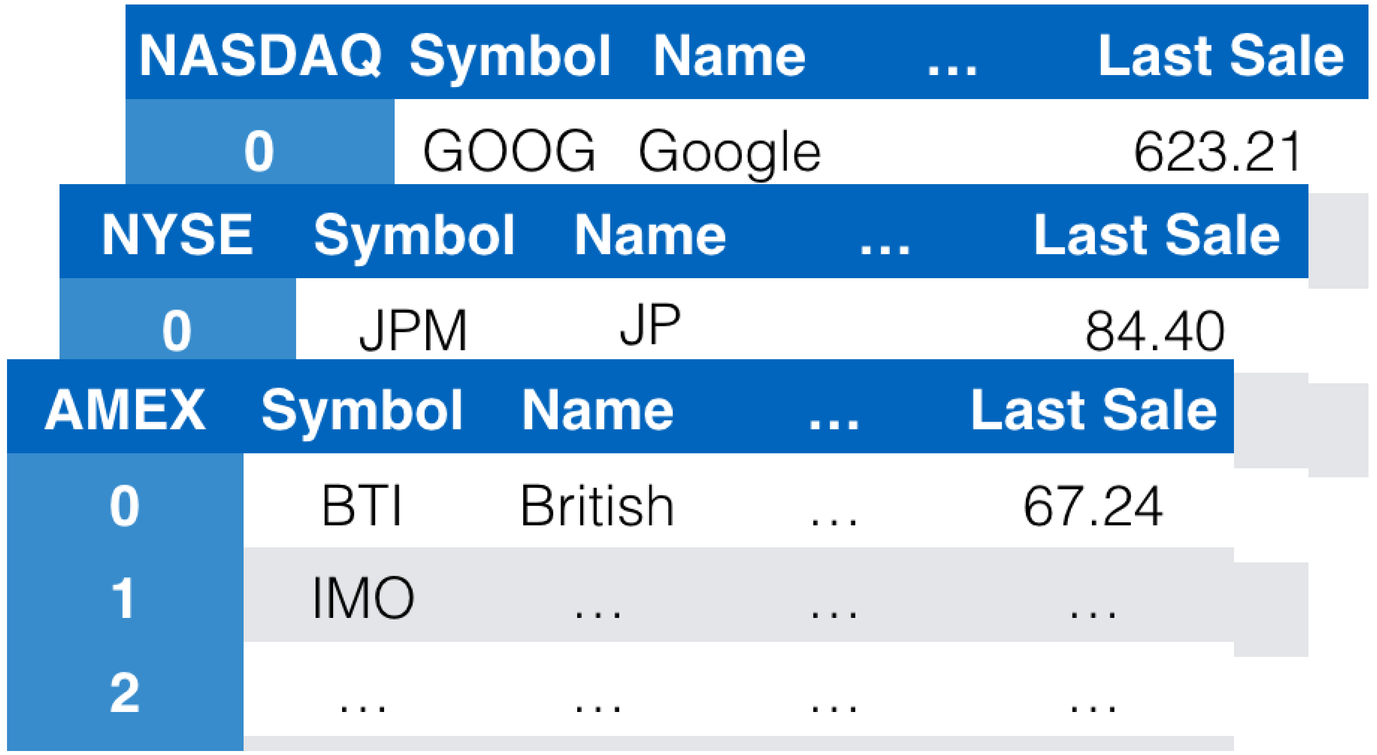
IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON

Stefan Jansen
Instructor

Combine DataFrames

- Concatenate or "stack" a list of `pd.DataFrame` s
- Syntax: `pd.concat([amex, nasdaq, nyse])`

It combines DataFrames vertically.



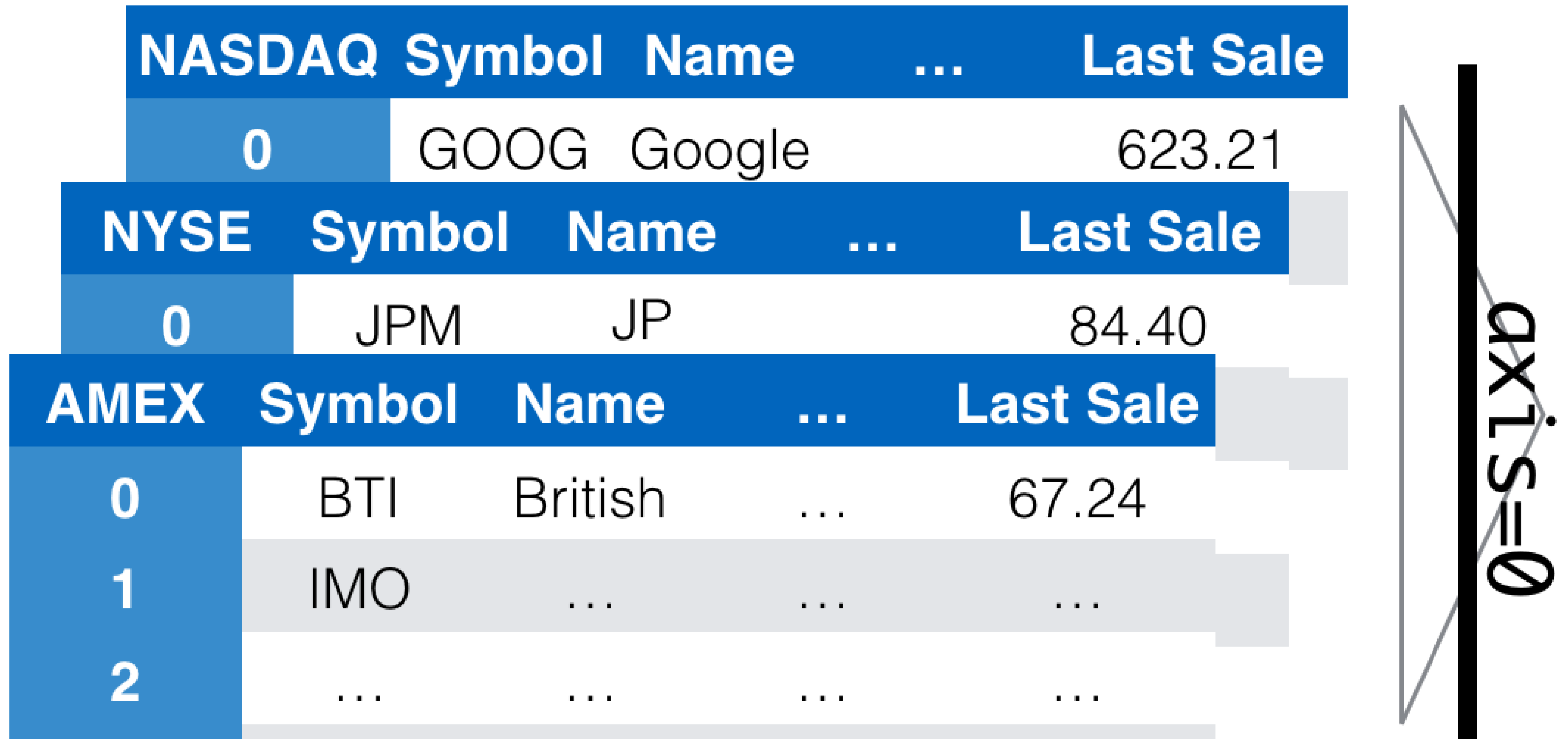
NASDAQ	Symbol	Name	…	Last Sale
0	GOOG	Google		623.21

NYSE	Symbol	Name	…	Last Sale
0	JPM	JP		84.40

AMEX	Symbol	Name	…	Last Sale
0	BTI	British	…	67.24
1	IMO	…	…	…
2	…	…	…	…

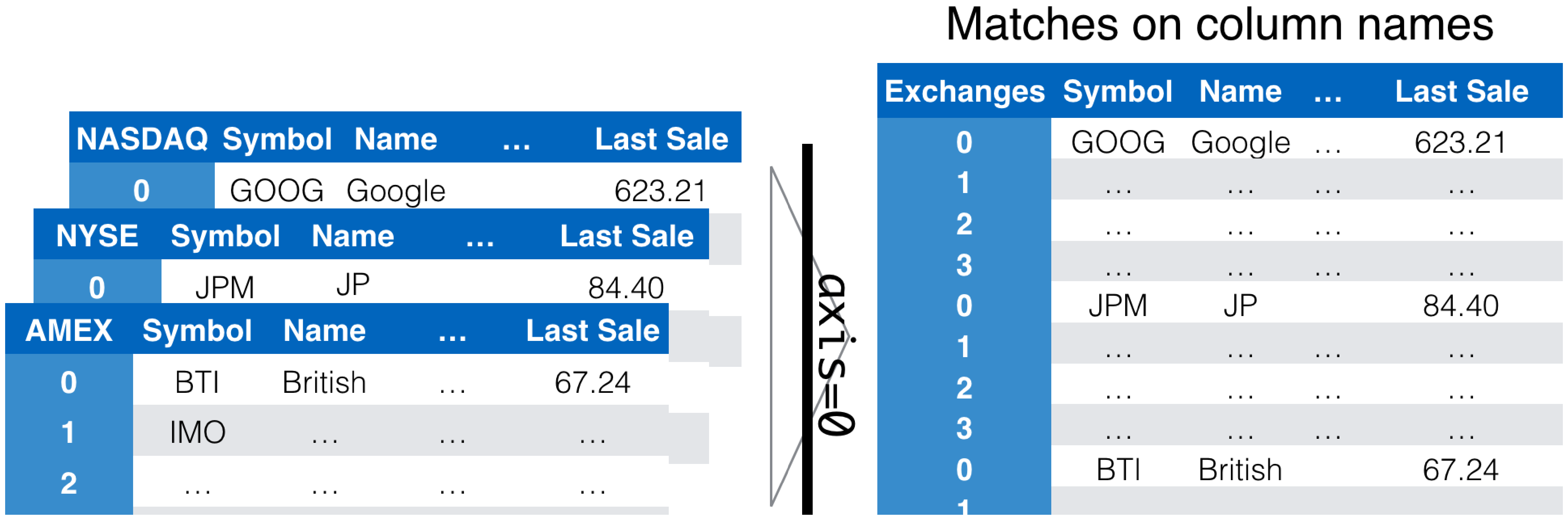
Combine DataFrames

- Concatenate or "stack" a list of `pd.DataFrame` s
- Syntax: `pd.concat([amex, nasdaq, nyse])`



Combine DataFrames

- Concatenate or "stack" a list of `pd.DataFrame` s
- Syntax: `pd.concat([amex, nasdaq, nyse])`



Concatenate two DataFrames

```
amex = pd.read_excel('listings.xlsx',
                     sheetname='amex',
                     na_values=['n/a'])

nyse = pd.read_excel('listings.xlsx',
                     sheetname='nyse',
                     na_values=['n/a'])

pd.concat([amex, nyse]).info()
```

```
Int64Index: 3507 entries, 0 to 3146
Stock Symbol             3507 non-null object
...
```

Add a reference column

```
amex['Exchange'] = 'AMEX' # Add column to reference source
nyse['Exchange'] = 'NYSE'

listings = pd.concat([amex, nyse])

listings.head(2)
```

Pandas will make sure that this value propagates across all the rows.
This feature is also called broadcasting.

```
  Stock Symbol    ...      Exchange
0         XXII    ...          AMEX
1          FAX    ...          AMEX
```

Combine three DataFrames

```
xls = pd.ExcelFile('listings.xlsx')

exchanges = xls.sheet_names

# Create empty list to collect DataFrames
listings = []

for exchange in exchanges:
  listing = pd.read_excel(xls, sheetname=exchange)
  # Add reference col
  listing['Exchange'] = exchanges
  # Add DataFrame to list
  listings.append(listing)

# List of DataFrames
combined_listings = pd.concat(listings)
```

Once the loop is completed, the variable 'listings' contains all three DataFrames.

Combine three DataFrames

```
combined_listings.info()
```

```
Int64Index: 6674 entries, 0 to 359
Data columns (total 8 columns):
Stock Symbol             6674 non-null object
Company Name             6674 non-null object
Last Sale                6590 non-null float64
Market Capitalization    6674 non-null float64
IPO Year                 2852 non-null float64
Sector                   5182 non-null object
Industry                 5182 non-null object
Exchange                 6674 non-null object
dtypes: float64(3), object(5)
```

Let's practice!

IMPORTING AND MANAGING FINANCIAL DATA IN PYTHON