

Lesson #03

Introduction to Pandas

Feb. 2019
Real Python

Introduction to Pandas

Exploring Data with Pandas

Data Cleaning Basics

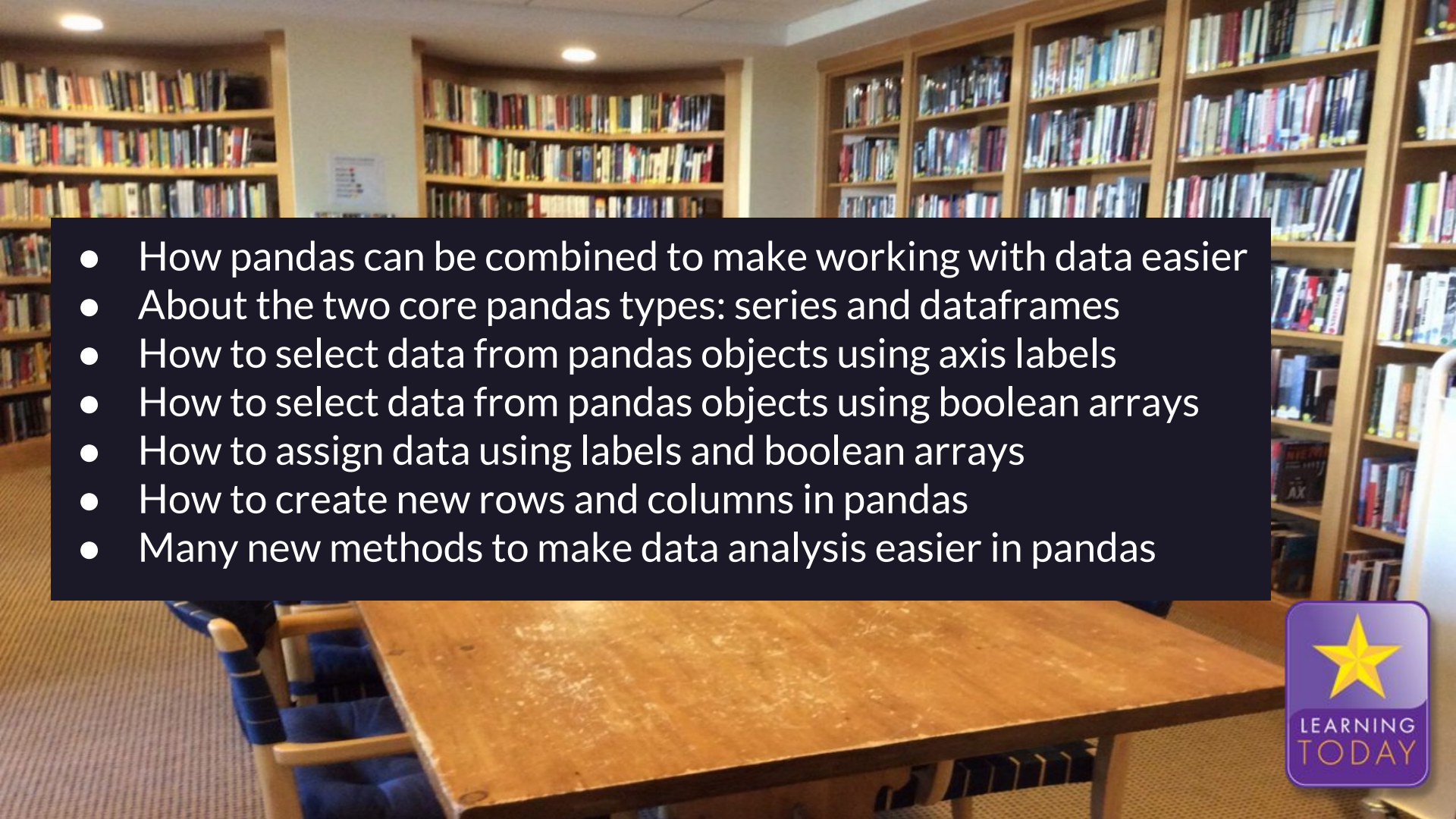
Data Aggregation

Combining Data with Pandas

Transforming Data with Pandas

Working with String in Pandas

Working with missing and duplicate data

- 
- How pandas can be combined to make working with data easier
 - About the two core pandas types: series and dataframes
 - How to select data from pandas objects using axis labels
 - How to select data from pandas objects using boolean arrays
 - How to assign data using labels and boolean arrays
 - How to create new rows and columns in pandas
 - Many new methods to make data analysis easier in pandas



Update from repository

```
git clone https://github.com/ivanovitchm/datascience_one_2019_1
```

Or

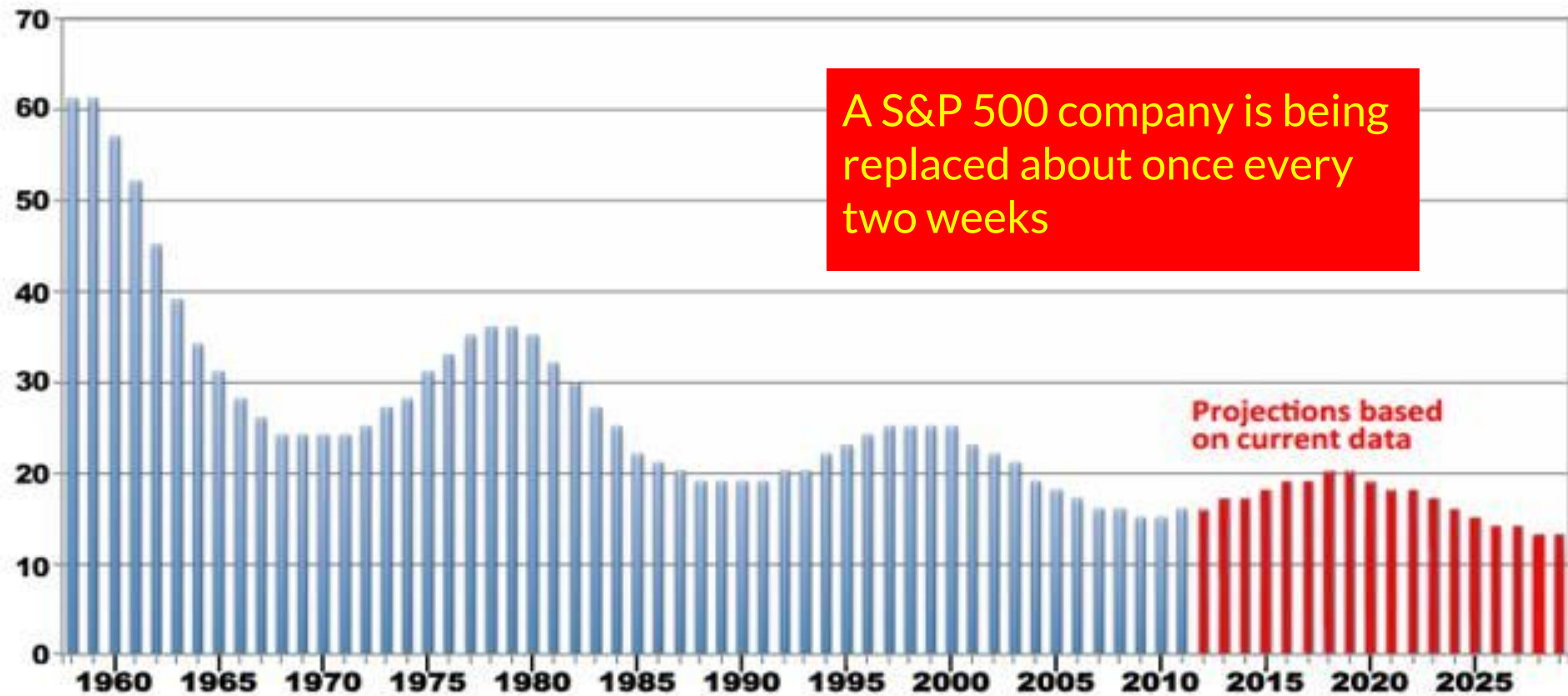
```
git pull
```





2017

Average company lifespan on S&P 500 Index (in years)



Year (each data point represents a rolling 7-year average of average lifespan)

DATA: INNOSIGHT/Richard N. Foster/Standard & Poor's

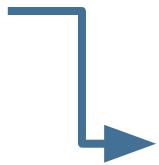


f500.csv

company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry
Walmart	1	485873	0.8	13643.0	198825	-7.2	C. Douglas McMillon	General Merchandisers
State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	Utilities
Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Petroleum Refining
China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Petroleum Refining
Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Akio Toyoda	Motor Vehicles and Parts

```
import pandas as pd
```

```
f500 = pd.read_csv("f500.csv")
```



	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo
0	Walmart	1	485873		0.8	13643.0	198825	C. Douglas McMillon
1	State Grid	2	315199		-4.4	9571.3	489838	Kou Wei
2	Sinopec Group	3	267518		-9.1	1257.9	310726	Wang Yupu
3	China National Petroleum	4	262573		-12.3	1867.5	585619	Zhang Jianhua
4	Toyota Motor	5	254694		7.7	16899.3	437575	Akio Toyoda

Pandas



shape, dtype, head(), tail(), info(), describe()



Dataframes

The diagram illustrates a DataFrame structure with the following components:

- Column Labels:** rank, revenues, profits, country
- Index Axis:** Indicated by a red arrow pointing downwards.
- Row Labels:** Walmart, State Grid, Sinopec Group, China Natural Petroleum, Toyota Motor
- Column Axis:** Indicated by a red arrow pointing to the right.
- Data Types:**
 - Integer Type:** Indicated by green arrows pointing to the 'rank' and 'revenues' columns.
 - Float Type:** Indicated by a green arrow pointing to the 'profits' column.
 - String Type:** Indicated by green arrows pointing to the 'country' column.

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

Selecting a column by label

```
f500_selection
```

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

```
f500_selection.loc[:, "rank"]
```

Walmart	1
State Grid	2
Sinopec Group	3
China Natural Petroleum	4
Toyota Motor	5

Selecting columns by label

```
f500_selection
```

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

```
f500_selection.loc[:, ["country", "rank"]]
```

	country	rank
Walmart	USA	1
State Grid	China	2
Sinopec Group	China	3
China Natural Petroleum	China	4
Toyota Motor	Japan	5

Selecting columns by label

```
f500_selection
```

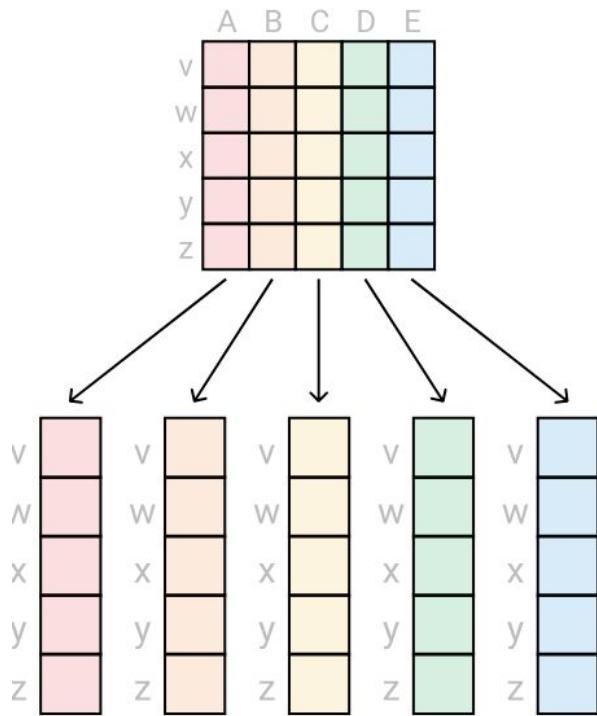
	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

```
f500_selection.loc[:, "rank": "profits"]
```

	rank	revenues	profits
Walmart	1	485873	13643.0
State Grid	2	315199	9571.3
Sinopec Group	3	267518	1257.9
China Natural Petroleum	4	262573	1867.5
Toyota Motor	5	254694	16899.3

Column selection shortcuts

Select by Label	Explicit Syntax	Common Shorthand	Other Shorthand
Single column	<code>df.loc[:, "col1"]</code>	<code>df["col1"]</code>	<code>df.col1</code>
List of columns	<code>df.loc[:, ["col1", "col7"]]</code>	<code>df[["col1", "col7"]]</code>	
Slice of columns	<code>df.loc[:, "col1":"col4"]</code>		



Series vs Dataframe

Original
Dataframe

	A	B	C	D	E
V					
W					
X					
Y					
Z					

Code

```
single_col = df["D"]
```

Result

V	
W	
X	
Y	
Z	

single_col is a
series object

	A	B	C	D	E
V					
W					
X					
Y					
Z					

```
single_row = df.head(1)
```

A	
B	
C	
D	
E	

single_row is a
series object

Original
Dataframe

	A	B	C	D	E
V					
W					
X					
Y					
Z					

Code

```
multi_cols = df[["A", "C", "D"]]
```

Result

	A	C	D
V			
W			
X			
Y			
Z			

multi_cols is a
dataframe object

	A	B	C	D	E
V					
W					
X					
Y					
Z					

```
multi_rows = df.head(3)
```

	A	B	C	D	E
V					
W					
X					

multi_rows is a
dataframe object

Dataframe vs Series

	Series	DataFrame
Dimensions	One	Two
Has 'index' axis	Yes	Yes
Has 'columns' axis	No	Yes
Number of dtypes	One	Many (one per column)

Series and Dataframe Describe Methods

```
revs = f500["revenues"]  
print(revs.describe())
```

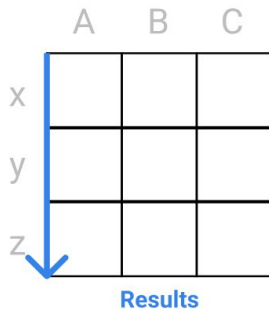
```
count      500.000000  
mean      55416.358000  
std       45725.478963  
min       21609.000000  
25%       29003.000000  
50%       40236.000000  
75%       63926.750000  
max       485873.000000  
Name: revenues, dtype: float64
```

```
print(f500["assets"].describe())
```

```
count      5.000000e+02  
mean      2.436323e+05  
std       4.851937e+05  
min       3.717000e+03  
25%       3.658850e+04  
50%       7.326150e+04  
75%       1.805640e+05  
max       3.473238e+06  
Name: assets, dtype: float64
```


`DataFrame.method(axis=0)`
or
`DataFrame.method(axis="index")`

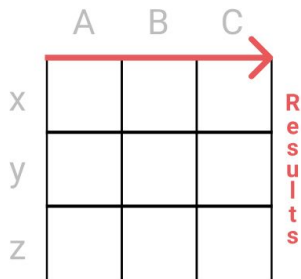
Calculates along the **row** axis



Calculates result for
each **column**.

`DataFrame.method(axis=1)`
or
`DataFrame.method(axis="column")`

Calculates along the **column** axis



Calculates result for
each **row**.

More data exploration methods

```
medians = f500[["revenues", "profits"]].median(axis=0)
# we could also use .median(axis="index")
print(medians)
```

```
revenues    40236.0
profits      1761.6
dtype: float64
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues
Walmart	1	485873
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573
Toyota Motor	5	254694

```
>>> top5_rank_revenue["revenues"] = 0
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues
Walmart	1	0
State Grid	2	0
Sinopec Group	3	0
China National Petroleum	4	0
Toyota Motor	5	0

Assignment with Pandas

Assignment with Pandas

```
>>> top5_rank_revenue.loc["Sinopec Group", "revenues"] = 999
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues
Walmart	1	0
State Grid	2	0
Sinopec Group	3	999
China National Petroleum	4	0
Toyota Motor	5	0

Add a new column

```
>>> top5_rank_revenue["year_founded"] = 0
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0

Add a new row

```
>>> top5_rank_revenue.loc["My New Company"] = 555
```

```
>>> print(top5_rank_revenue)
```

	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0
My New Company	555	555	555

Using boolean indexing with pandas objects

w	2
x	4
y	6
z	8

pandas series



w	2	< 5
x	4	< 5
y	6	< 5
z	8	< 5

Vectorized
boolean operation



w	True
x	True
y	False
z	False

Boolean
pandas series

	A	B
w	2	3
x	4	6
y	6	9
z	8	12

pandas DataFrame



	A	B	
w	2	3	< 5
x	4	6	< 5
y	6	9	< 5
z	8	12	< 5

Vectorized
boolean operation



	A	B
w	True	True
x	True	False
y	False	False
z	False	False

Boolean
pandas DataFrame

Using boolean indexing with pandas objects

```
result = df.loc[num_bool, "name"]
```

		name		
False	w	Kylie	12	
True	→ x	Rahul	8	→ x
False	y	Michael	5	z
True	→ z	Sarah	8	↗
				result
				Rahul
				Sarah

```
result = df[num_bool]
```

		name	num	
False	w	Kylie	12	
True	→ x	Rahul	8	→ x
False	y	Michael	5	z
True	→ z	Sarah	8	↗
				result
				Rahul
				Sarah

Using boolean arrays to assign values

```
f500.loc[f500["sector"] == "Motor Vehicles & Parts", "sector"] = "Motor Vehicles and Parts"
```

Challenge

Finding top performers by country

```
>>> top_3_countries = f500["country"].value_counts().head(3)
```

```
>>> print(top_3_countries)
```

```
USA      132
```

```
China    109
```

```
Japan     51
```

```
Name: country, dtype: int64
```