

# PYTHON / PANDAS FOR DATA ANALYSIS



Sander van den Oord - Data Scientist

# Wat we gaan doen de komende 2 dagen

- Introductie:
  - waar werk je nu,
  - met welke software,
  - gebruik je nu Python,
  - en: wat wil je vandaag leren
- Hihover uitleg Python, Pandas en belangrijkste Python data libraries
- Intro Jupyter Notebook + Python refresher + opstarten VM
- Data inlezen `---> pd.read_csv()`
- Data inspectie `---> df.info() df.head() df.describe()`
- Data selectie `---> df[df.column == 'value'] df.loc[df.column == 'value', :]`
- Data wrangling `---> df['column'].fillna() of df.drop_duplicates()`
- Data joinen `---> df.merge(df2)`
- Data visualisatie `---> px.scatter(df, x, y)`
- Eigen functies toepassen `---> df.apply(my_own_function)`
- Installeren Jupyter + Python + Pandas `---> pip install pandas`

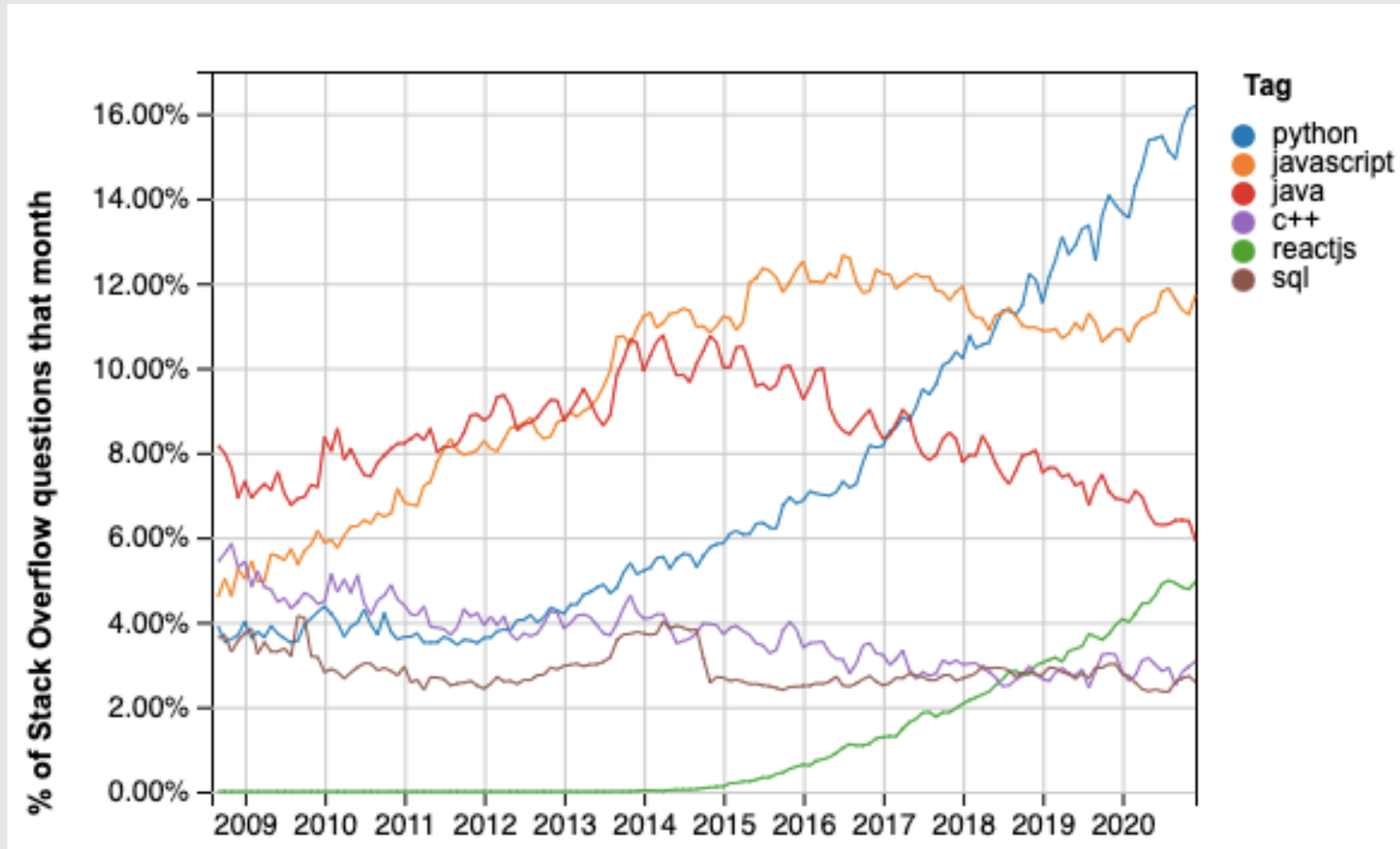
# Introductie

- Introductie:
  - waar werk je nu,
  - met welke software,
  - gebruik je nu Python,
  - en: wat wil je vandaag leren

# Highover uitleg

- Highover uitleg:
  - Python
  - Pandas
  - en belangrijkste Python data libraries

# Python is eating the world



# Why is it handy to know some python / pandas?

- veel gebruikt voor data analysis en machine learning
- steeds vaker ingezet voor data engineering: pandas en python zijn zeer flexibel voor data wrangling
- als je de pandas API enigszins kent, dan helpt dat bijv. ook weer met databricks / pyspark
- steeds groter Python data ecosystem, bijv. scheduling met Airflow
- makkelijk om met python API's aan te spreken
- websites scrapen
- zeer leesbare code (relatief)
- en hopelijk: leuk om wat nieuws te leren

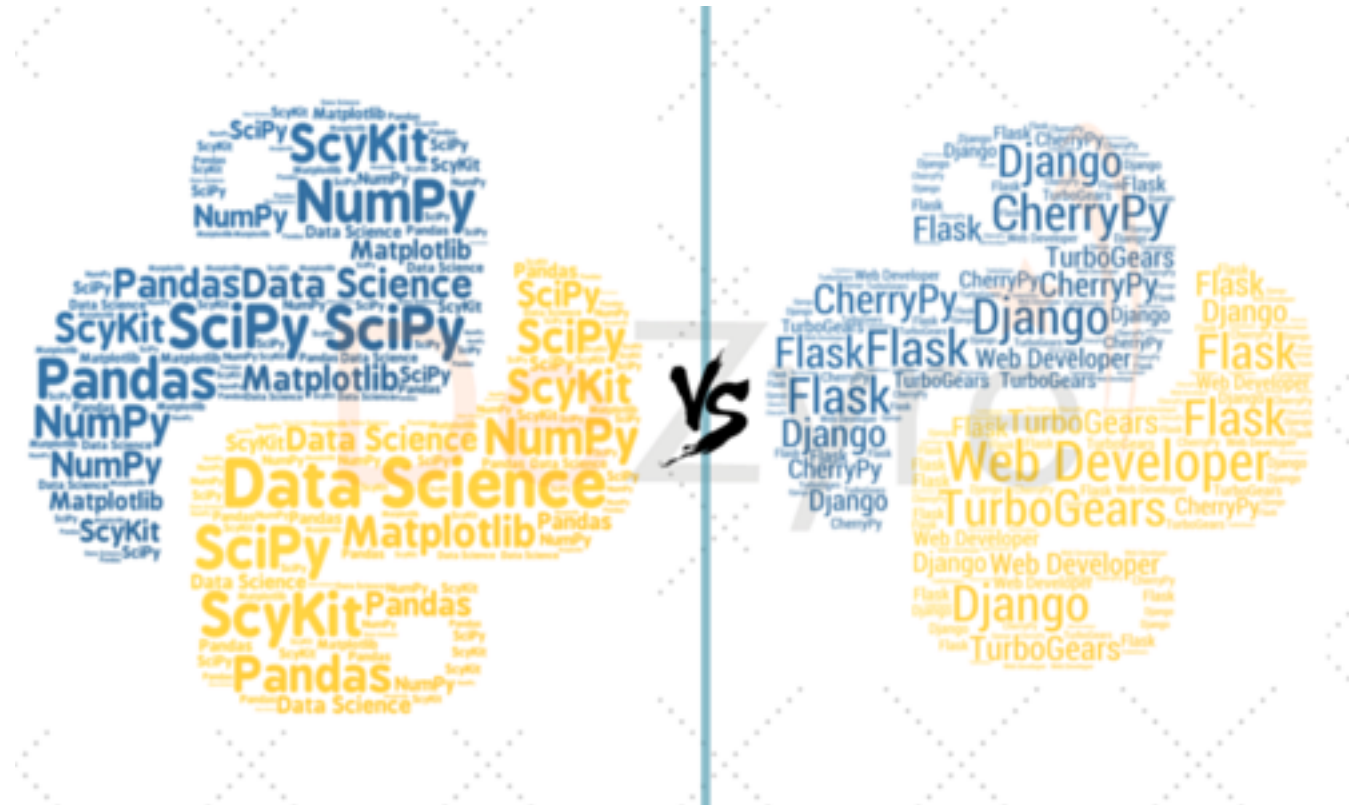
# Main disadvantages of python / pandas

- pandas doet alles in-memory, dus voor gedistribueerde oplossingen heb je bijv. Spark of Dask nodig
- higher-level language: gebouwd bovenop C. Omdat het higher-level is, is het ook langzamer
- niet geschikt voor mobile development

Twee wegen  
die naar  
Python  
leiden

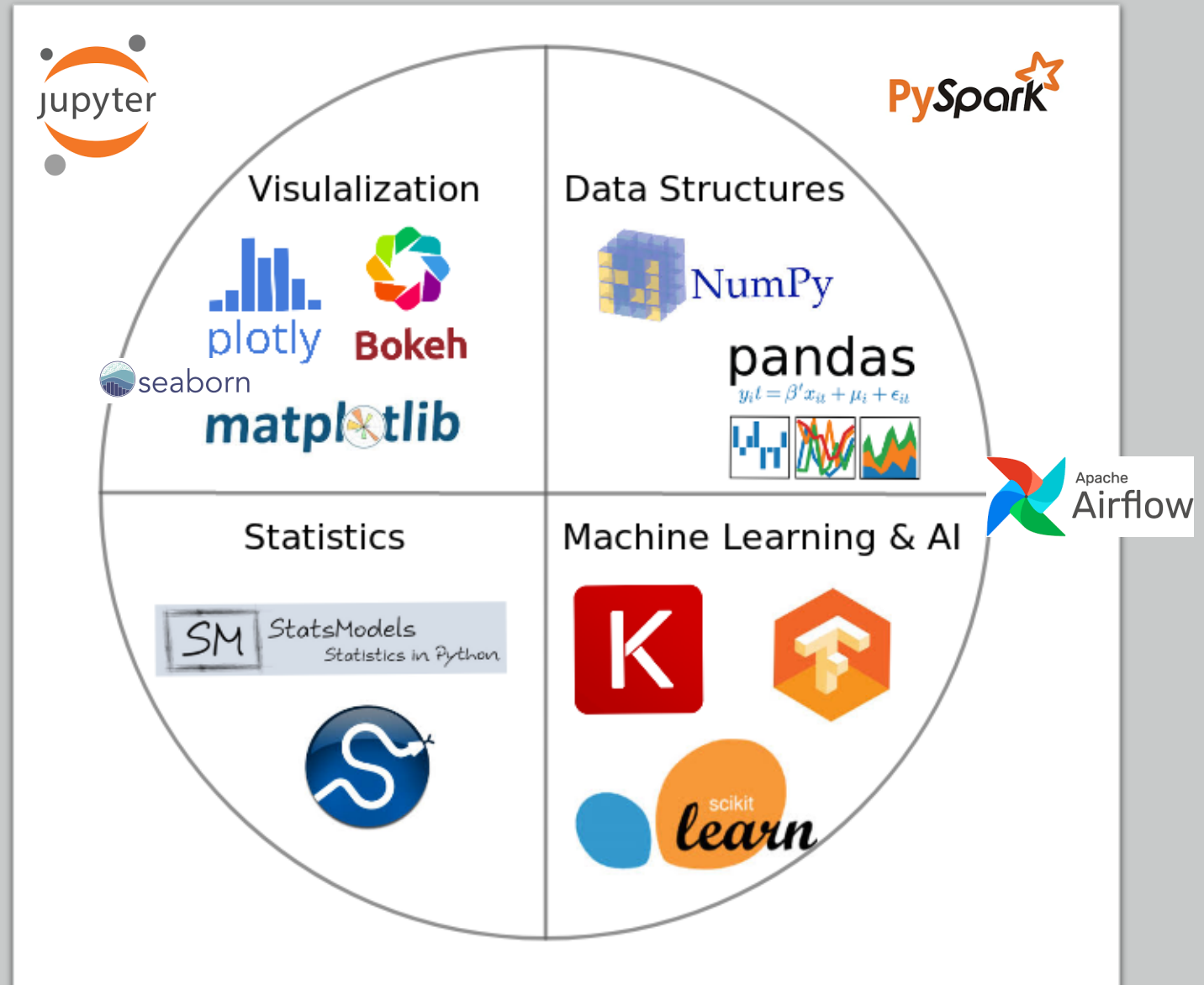
Data Science /  
Data Engineering

Backend Web  
Development





# Python Data Science Libraries



# Jupyter Notebooks + Python

- Intro Jupyter Notebook
- Python refresher
- opstarten VM

# Python data types

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey": "value", "name": "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

# Jupyter Notebooks

vs

# Visual Studio Code

The screenshot displays the Jupyter Notebook environment. The top menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar shows a file explorer with a list of notebooks and files, including 'Lorenz.ipynb' which is currently selected. The main area shows the notebook content, which includes a text description of the Lorenz system, its equations, and a code cell. The code cell contains a function definition for 'solve\_lorenz' and a call to it. Below the code cell, the 'Output View' shows a 3D plot of the Lorenz attractor, which is a complex, swirling shape. To the right of the plot, there are sliders for parameters 'sigma', 'beta', and 'rho', with values 10.00, 2.67, and 28.00 respectively.

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around called attractors.

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

sigma: 10.00  
beta: 2.67  
rho: 28.00

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

The screenshot displays the Visual Studio Code editor. The top menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar shows a file explorer with a list of files, including 'machineLearning.py' which is currently selected. The main area shows the code editor with a Python script. The script imports pandas, numpy, sklearn, and matplotlib. It reads data from a CSV file and plots it. The code is as follows:

```
1 # Import pandas
2 import pandas as pd
3 import numpy as np
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.model_selection import train_test_split
6 import matplotlib as plt
7 %matplotlib inline
8
9 # Read in white wine data
10 white = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data")
11
12 # Read in red wine data
13 red = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data")
14
15 import matplotlib.pyplot as plt
16 import numpy as np
17
18 np.random.seed(570)
19
20 redlabels = np.unique(red['quality'])
21 whitelabels = np.unique(white['quality'])
22
23 import matplotlib.pyplot as plt
24 fig, ax = plt.subplots(1, 2, figsize=(8, 8))
25 redcolors = np.random.rand(6,4)
26 whitecolors = np.append(redcolors, np.random.rand(6,4))
```

The right sidebar shows the 'Output pane' with a search bar and a list of imports. Below the imports, there is a 'graph' section showing two scatter plots: 'Red Wine' and 'White Wine'. The 'Red Wine' plot shows a scatter of points with a color scale from 3 to 9. The 'White Wine' plot shows a scatter of points with a color scale from 3 to 9.

# Jupyter Notebooks tips and tricks

- Shift + Enter to run code
- Tab completion
- Nieuwe cell: Escape gevolgd door a (above) of b (below) of dd (delete)
- Shift Tab to see arguments and information about methods, functions or classes
- Magic commands, such as ls
- ? or ?? to get extra help and info

# Pandas Introductie

- Data inlezen `---> pd.read_csv()`
- Data inspectie `---> df.info() df.head() df.describe()`
- Data selectie `---> df[df.column == 'value'] df.loc[df.column == 'value', :]`
- Data wrangling `---> df['column'].fillna() df.drop_duplicates()`
- Data joinen `---> df.merge(df2, how='inner', on='column_name')`
- Data visualisatie `---> px.scatter(df, x, y)`

# Wat is Pandas?



**pandas**

**pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the [Python](#) programming language.

# Most important concept is a DataFrame

The diagram illustrates a pandas DataFrame with the following structure and annotations:

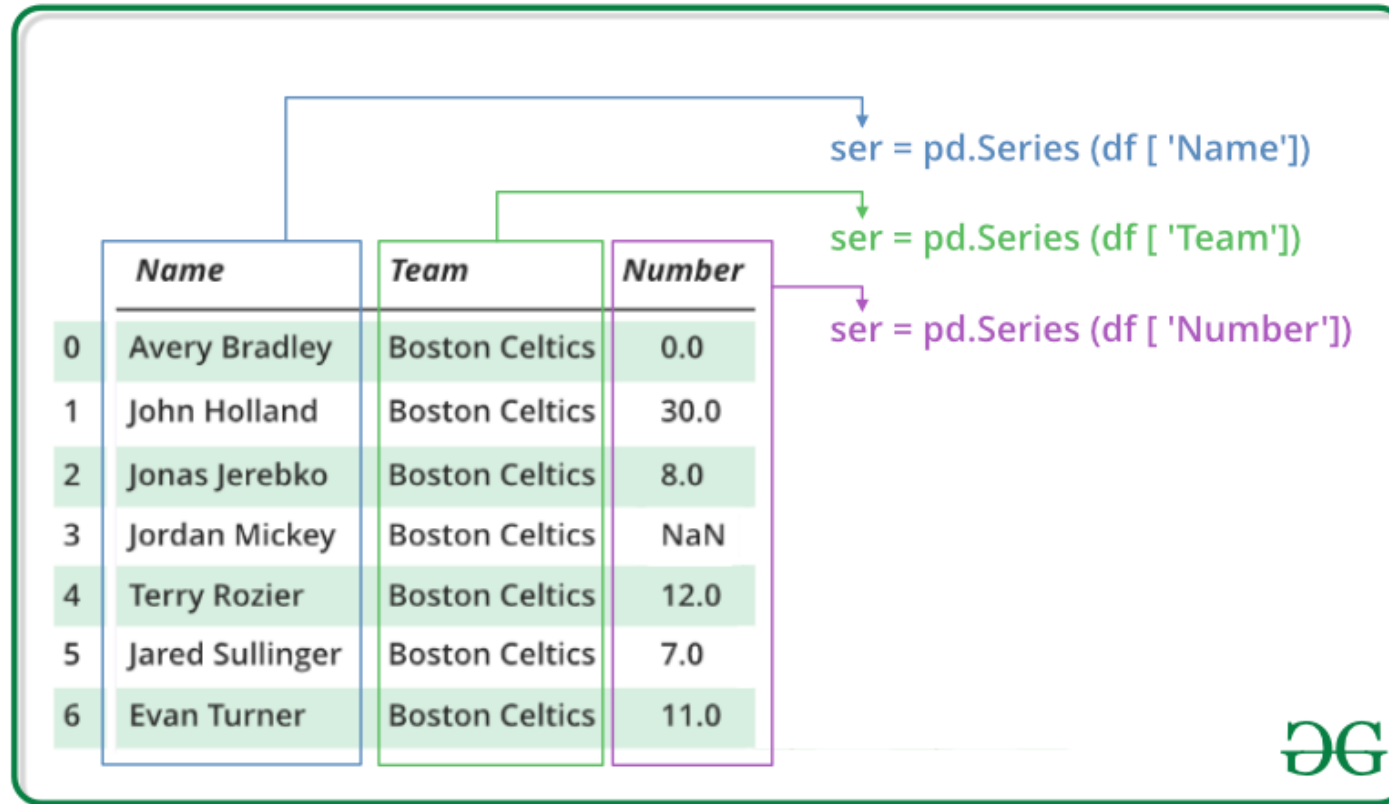
- Column names:** Name, Team, Number, Position, Age, Height, Weight, College, Salary.
- Index labels:** 0, 1, 2, 3, 4, 5, 6.
- Annotations:**
  - Columns axis=1:** Points to the column headers.
  - Index label:** Points to the index values (0-6).
  - Index axis=0:** Points to the index values (0-6).
  - Missing value:** Points to the 'NaN' value in the 'Number' column for index 3.
  - Data:** Points to the numerical values in the 'Age', 'Height', 'Weight', and 'Salary' columns for index 3.

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston Uniersity	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

All calculations are done vectorized.  
Don't use for loops with pandas!!!



# Example of a Pandas Series



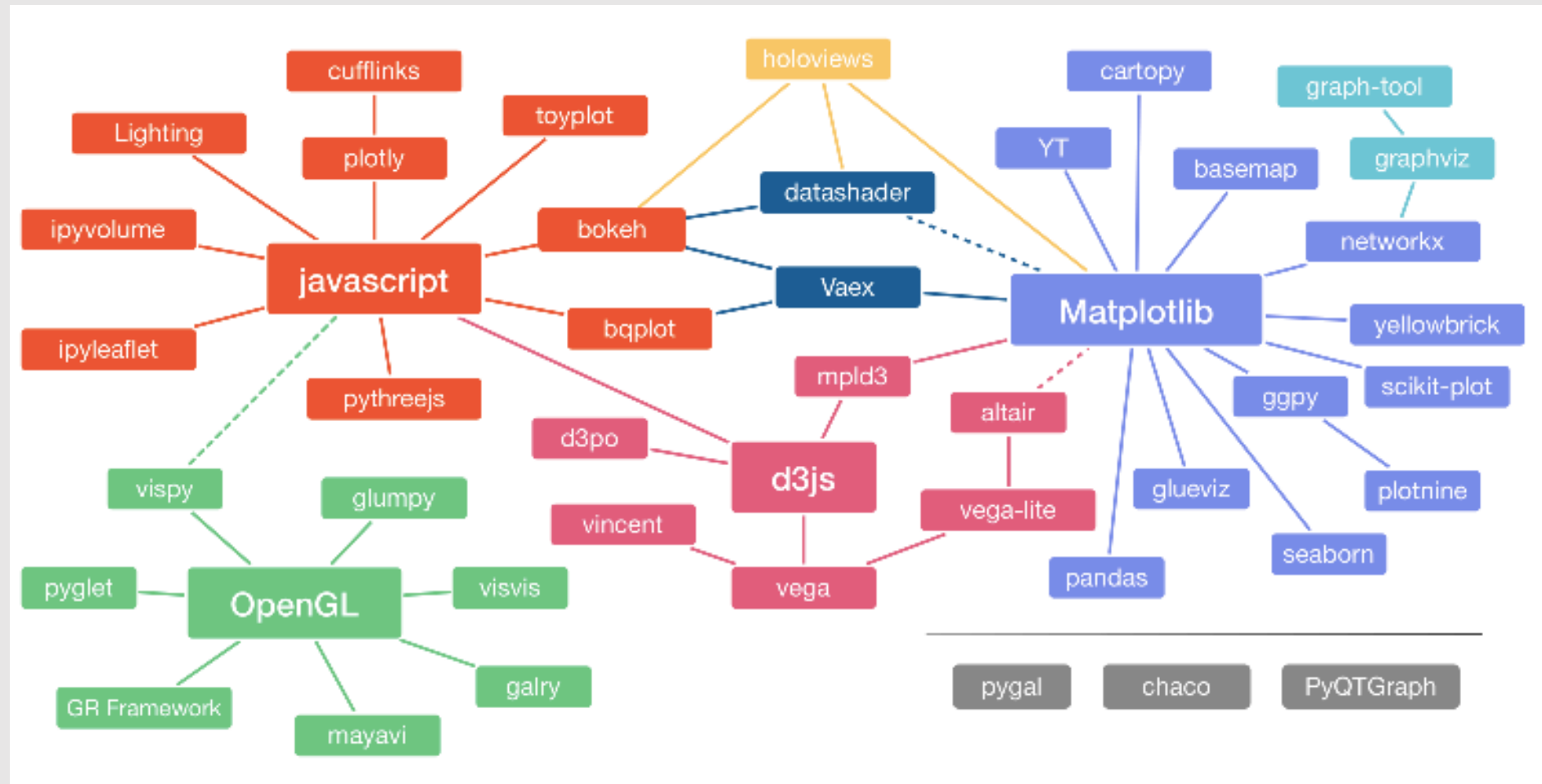
# Plotting

- Data visualisatie

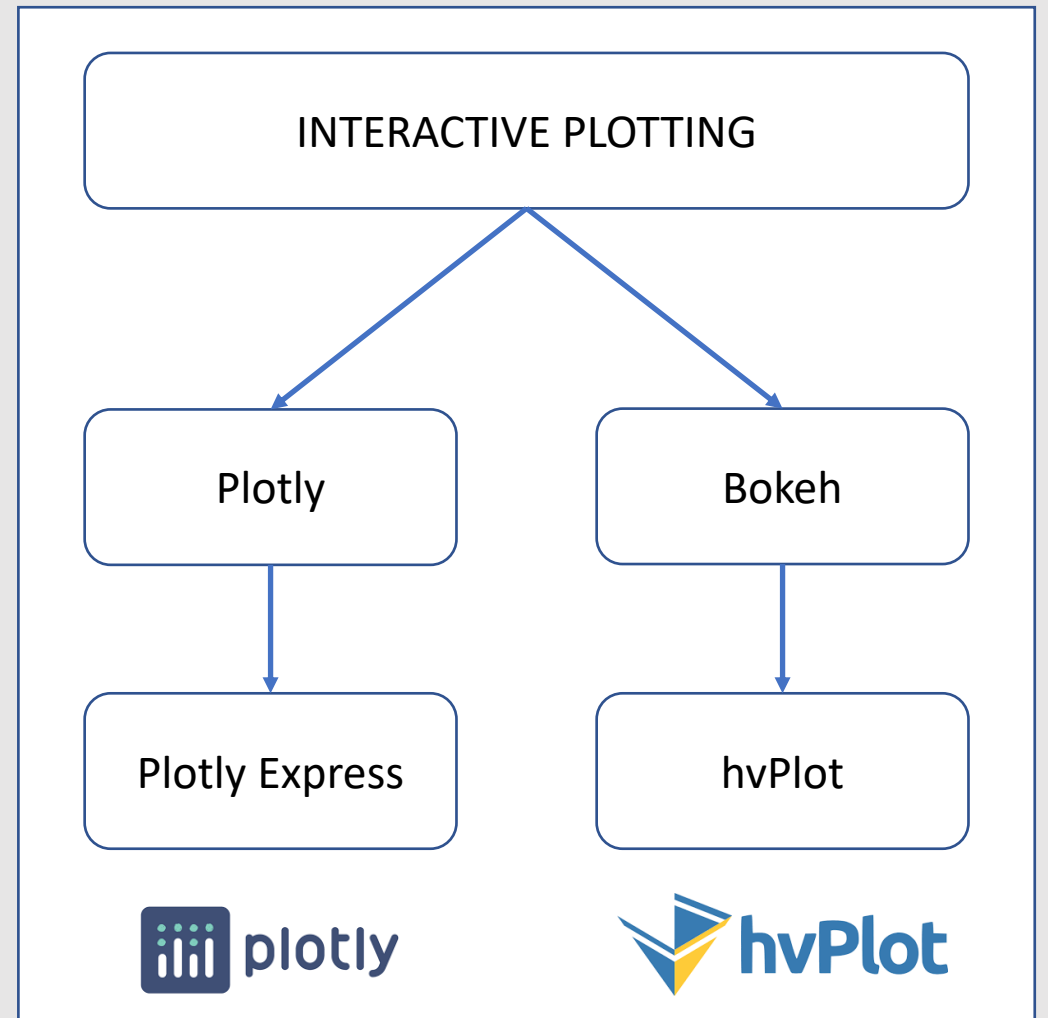
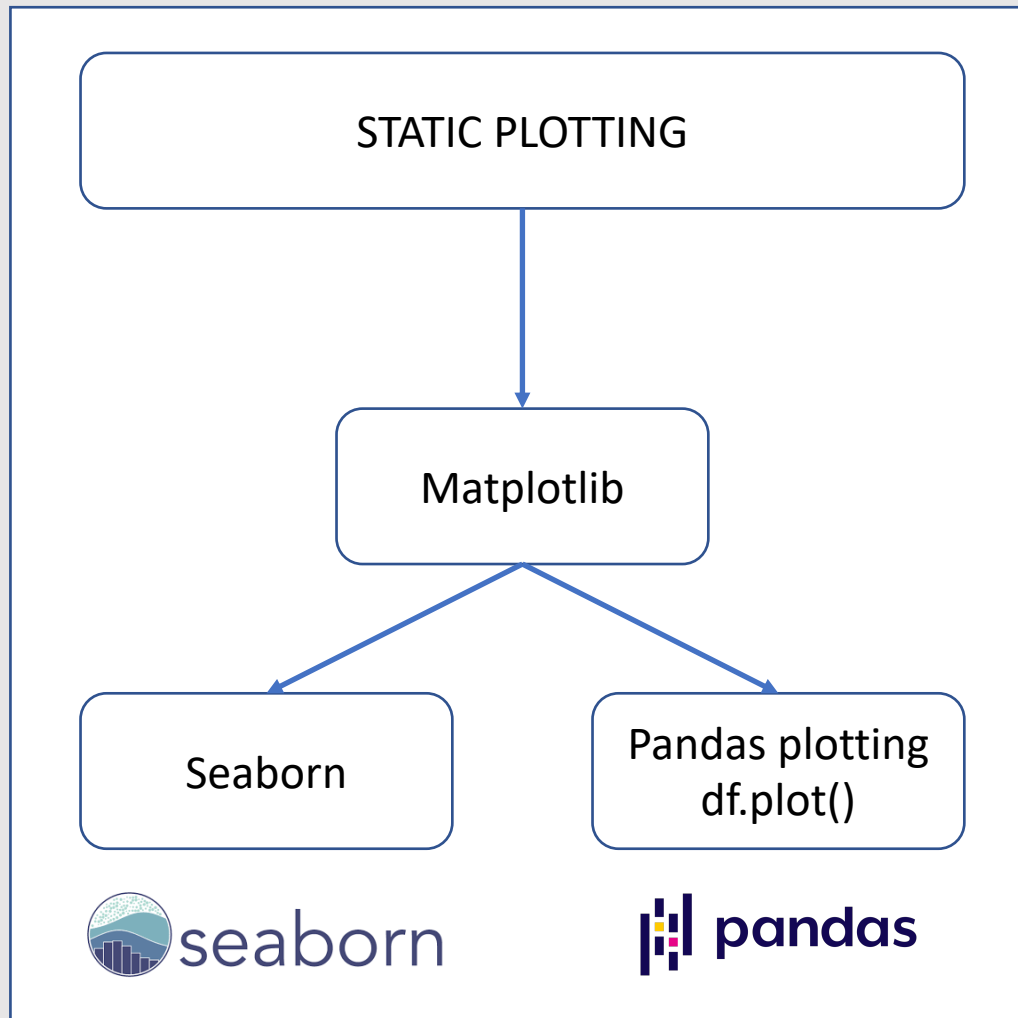
---> `px.scatter(df, x, y)`

---> `sns.scatterplot(df, x, y)`

# There are too many plotting packages



# So my advice is: only focus on these



# Data aggregation

- Using groupby to aggregate:

```
---> df.groupby(['startYear'])[['averageRating']].mean()
```

# How groupby() works

GroupBy: split-apply-combine

split

apply

combine

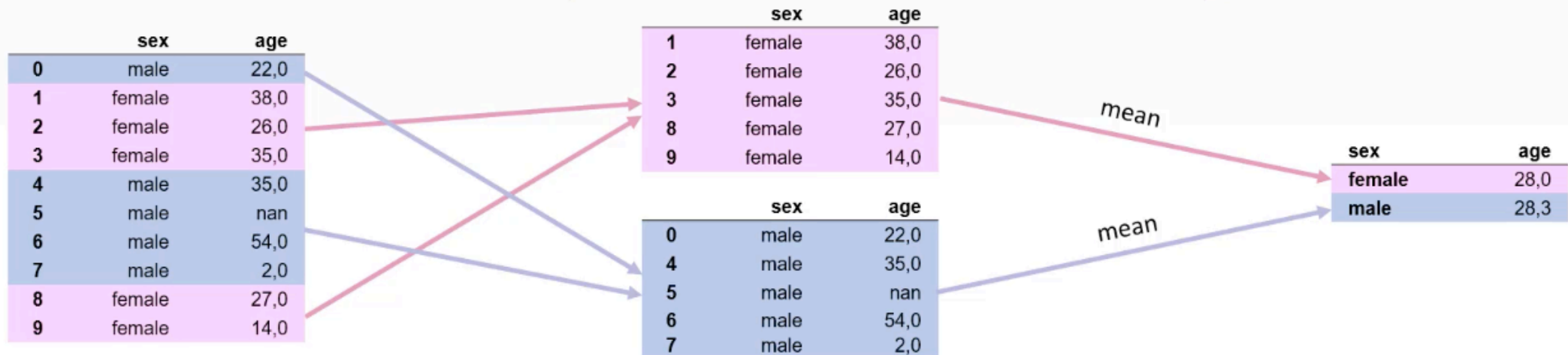
titanic-DataFrame

titanic.groupby("sex")

GroupBy object

.mean()

new DataFrame



# Joining tables

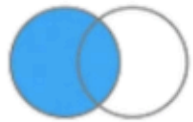
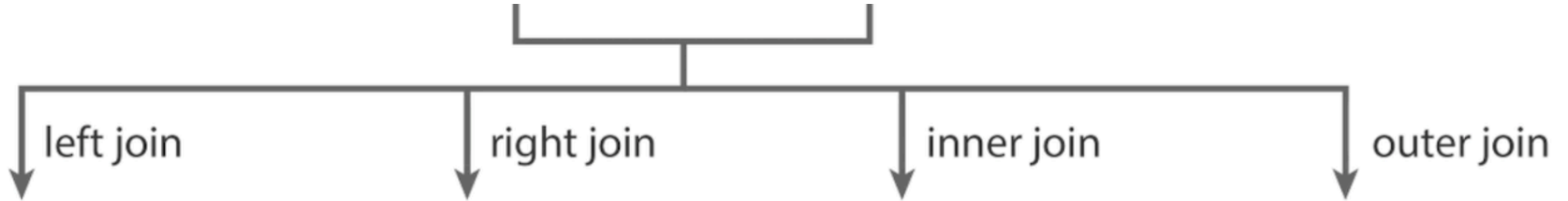
- Using `df.merg()` to join tables:

---> `df.merge(df2, how='inner', on='mutual_column')`

# Joining tables with df.merge()

	title	runtime
0	Pulp Fiction	90
1	James Bond	120
2	Titanic	115

	title	name
0	Pulp Fiction	John Travolta
1	Pulp Fiction	Samuel L. Jackson
2	James Bond	Sean Connery
3	Terminator	Arnold Schwarzenegger



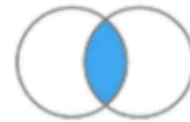
	title	runtime	name
0	Pulp Fiction	90	John Travolta
1	Pulp Fiction	90	Samuel L. Jackson
2	James Bond	120	Sean Connery
3	Titanic	115	NaN

```
movies.merge(actors, how='left', on='title')
```



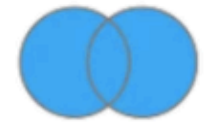
	title	runtime	name
0	Pulp Fiction	90.000	John Travolta
1	Pulp Fiction	90.000	Samuel L. Jackson
2	James Bond	120.000	Sean Connery
3	Terminator	NaN	Arnold Schwarzenegger

```
movies.merge(actors, how='right', on='title')
```



	title	runtime	name
0	Pulp Fiction	90	John Travolta
1	Pulp Fiction	90	Samuel L. Jackson
2	James Bond	120	Sean Connery

```
movies.merge(actors, how='inner', on='title')
```



	title	runtime	name
0	Pulp Fiction	90.000	John Travolta
1	Pulp Fiction	90.000	Samuel L. Jackson
2	James Bond	120.000	Sean Connery
3	Titanic	115.000	NaN
4	Terminator	NaN	Arnold Schwarzenegger

```
movies.merge(actors, how='outer', on='title')
```