Fachhochschule Kiel
University of Applied Sciences

Tools and Programming Languages for Data Science
Pandas: Data Preparation

Study Program Data Science
Prof. Dr. Tillmann Schwörer

Our course agenda

- **Introduction and overview**
- **NumPy**: Basic data handling with Numpy arrays
- **Pandas**
  - Exploratory data analysis
  - Data consolidation
  - Data cleaning
- **Data visualization using Matplotlib and Seaborn**
- **Interacting with APIs**
- **Interacting with SQL databases**
- **Version Control with Git and GitHub**
- **Advanced Python**

Python foundations

Data types

Operators

Functions

Control flow and iterators

Programming concepts & paradigms

See also Precourse Programming

Tooling

Installation

Visual Studio Code

Jupyter Notebooks

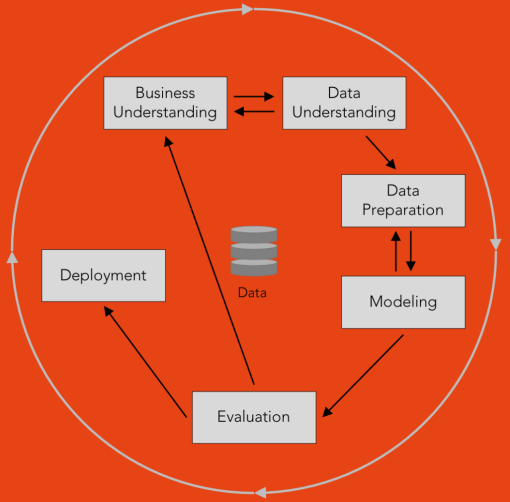
Packages

Virtual Environments

Git and Github

Python

Data Science Workflow



NumPy pandas matplotlib

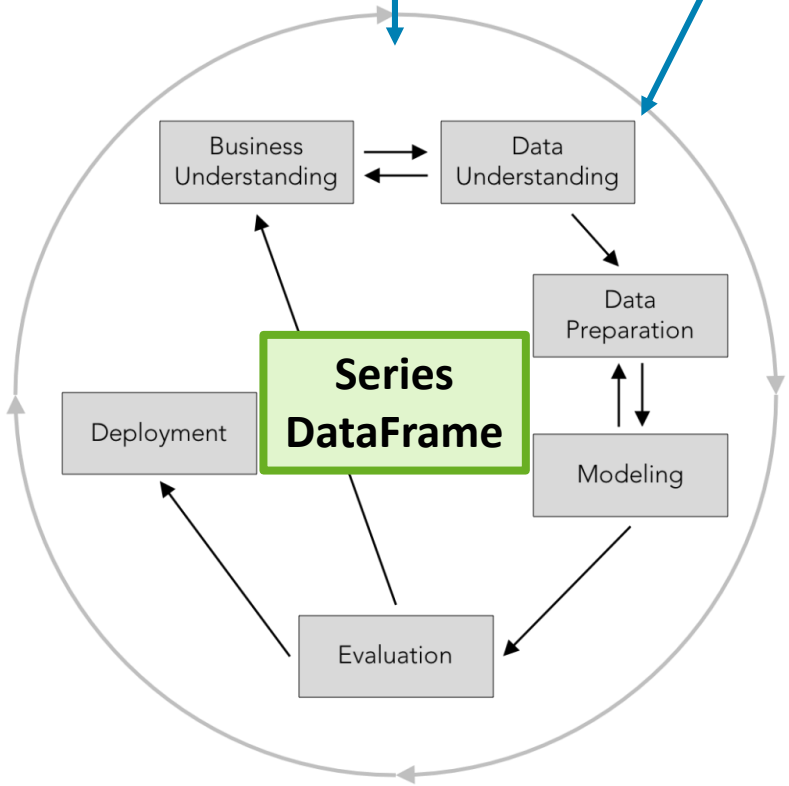
Pandas in the Data Science Process

Data Input and Output

- Python objects
- Files
- Databases

Data Exploration

- Overview
- Selecting data subsets
- Sorting
- Aggregating
- Visualizing



Data Preparation

Data Cleaning

- Identify missing values, duplicates, outliers, …
- Remove, replace, rename, …

Data Enhancing

- Combining multiple Series / DataFrames
- Reshaping
- Feature Engineering
- Operations on datetime, string, and categorical data

Be prepared

> *80 percent of a data scientist's valuable time is spent simply finding, cleansing, and organizing data, leaving only 20 percent to actually perform analysis…*
>
> IBM Data Analytics

Messy data

> "Happy families are all alike; every unhappy family is unhappy in its own way."
> — Leo Tolstoy

> "Tidy datasets are all alike, but every messy dataset is messy in its own way."
> — Hadley Wickham

Combining datasets

Merge

Merge or join two datasets based on a common set of key columns

Key

	fruit	amount
0	"apples"	3
1	"bananas"	2
2	"cherries	5

+

Key

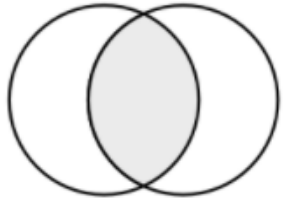
	fruit	price
0	"cherries	3.49
1	"bananas"	1.99
2	"apples"	2.99

=

	fruit	amount	price
0	"apples"	3	2.99
1	"bananas"	2	1.99
2	"cherries	5	3.49

```
pd.merge(left=amounts,
         right=prices,
         on="fruit")
```

Inner Merge



- Use only intersection of keys from the left and right table

	fruit	amount
0	"apples"	3
1	"bananas"	2
2	"cherries	5

+

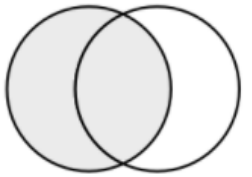
	fruit	price
0	"apples"	2.99
1	"bananas"	1.99
2	"dates"	4.99

=

```
pd.merge(left=amounts,
         right=prices,
         on="fruit",
         how = "inner")
```

	fruit	amount	price
0	"apples"	3	2.99
1	"bananas"	2	1.99

Left Merge



- Only keys from the left table

	fruit	amount
0	"apples"	3
1	"bananas"	2
2	"cherries	5

+

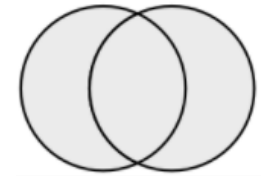
	fruit	price
0	"apples"	2.99
1	"bananas"	1.99
2	"dates"	4.99

=

```
pd.merge(left=amounts,
         right=prices,
         on="fruit",
         how = "left")
```

	fruit	amount	price
0	"apples"	3	2.99
1	"bananas"	2	1.99
2	"cherries	5	NaN

Outer Merge



- Keys from the left and right table (union)

	fruit	amount
0	"apples"	3
1	"bananas"	2
2	"cherries	5

+

	fruit	price
0	"apples"	2.99
1	"bananas"	1.99
2	"dates"	4.99

=

```
pd.merge(left=amounts,
         right=prices,
         on="fruit",
         how = "outer")
```

	fruit	amount	price
0	"apples"	3	2.99
1	"bananas"	2	1.99
2	"cherries	5	NaN
3	"dates"	NaN	4.99

Pandas Merge Method

Useful if key columns have different names

Merge can be performed via index

If left and right table contain identical column names, append specified suffixes

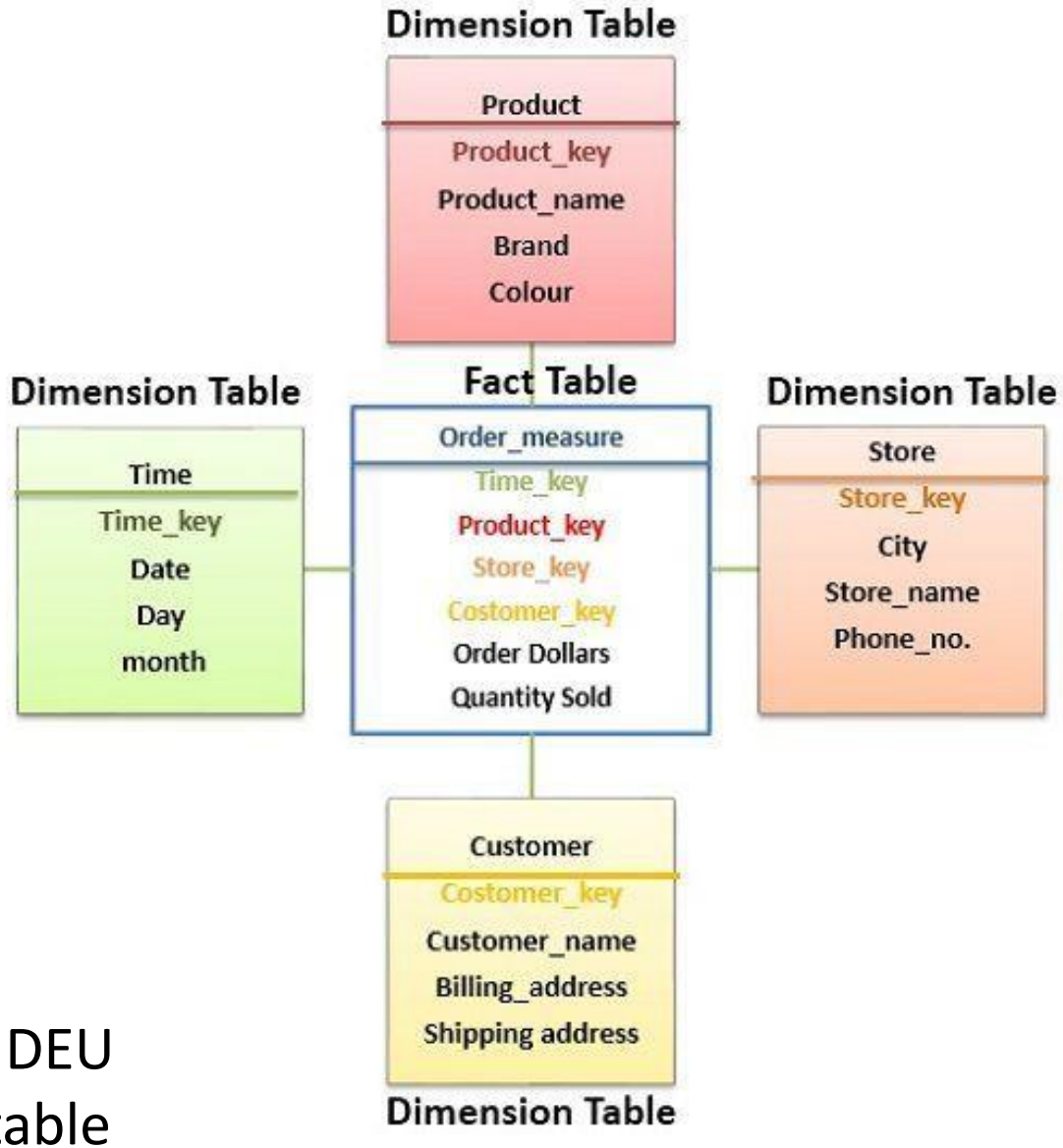
Check merge success via _merge indicator

```
pd.merge(
    left: 'DataFrame | Series',
    right: 'DataFrame | Series',
    how: 'MergeHow' = 'inner',
    on: 'IndexLabel | None' = None,
    left_on: 'IndexLabel | None' = None,
    right_on: 'IndexLabel | None' = None,
    left_index: 'bool' = False,
    right_index: 'bool' = False,
    sort: 'bool' = False,
    suffixes: 'Suffixes' = ('_x', '_y'),
    copy: 'bool | None' = None,
    indicator: 'str | bool' = False,
    validate: 'str | None' = None,
) -> 'DataFrame'
```

Merge Type Guidelines

Choice of merge type depends on use case!

- Choice only matters if data is incomplete
- Fact + Dimension Table → often left Join
- Fact + Fact Table → often outer Join



Always check for problems

- Do the keys match?
- If not, what are the reasons?
- Can you improve the merge?
  - e.g. provide consistent spelling in keys: DE vs. DEU
  - e.g. add new products to product dimension table

Pandas Concat Method

- Concatenate one below the other (axis = 0)
- Column names are matched
- Outer join: NaN values are filled in if column name is missing

```
pd.concat(objs=[day1, day2],
          axis=0,
          join='outer',
          ignore_index=True)
```

	fruit	date	amount	price
0	"apples"	2024-03-01	5	2.99
1	"bananas"	2024-03-01	4	1.99
2	"cherries	2024-03-01	3	3.99

	fruit	amount	date
0	"apples"	10	2024-03-02
1	"bananas"	8	2024-03-02
2	"cherries	9	2024-03-02

	fruit	date	amount	price
0	"apples"	2024-03-01	5	2.99
1	"bananas"	2024-03-01	4	1.99
2	"cherries	2024-03-01	3	3.99
3	"apples"	2024-03-02	10	NaN
4	"bananas"	2024-03-02	8	NaN
5	"cherries	2024-03-02	9	NaN

Pandas Concat Method

- Concatenate side by side (axis = 1)
- Row names (indices) are matched
- Outer join: NaN values are filled in if index is missing

```
pd.concat(objs=[day1, day2],
          axis=0,
          join='outer',
          ignore_index=True)
```

→ Equivalent to an outer merge based on index

	amount
"apples"	5
"bananas"	4
"cherries	3

	price
"apples"	2.99
"cherries	3.99
"bananas"	1.99
"dates"	4.99

	amount	price
"apples"	5	2.99
"bananas"	4	1.99
"cherries	3	3.99
"dates"	6	4.99

Reshaping data

Tidy data?

	fruit	year	type	value
0	apples	2023	amount	5
1	apples	2023	price	2.99
2	apples	2024	amount	10
3	apples	2024	price	2.99
4	bananas	2023	amount	4
5	bananas	2023	price	1.99
6	bananas	2024	amount	8
7	bananas	2024	price	2.29
8	cherries	2023	amount	3
9	cherries	2023	price	3.99
10	cherries	2024	amount	9
11	cherries	2024	price	3.89

Tidy data?

	Amount		Price	
	2023	2024	2023	2024
apples	5	10	2.99	2.99
bananas	4	8	1.99	2.29
cherries	3	9	3.99	3.89

Tidy data?

	fruit	year	details
0	apples	2023	{"amount":5, "price": 2.99}
1	bananas	2023	{"amount":4, "price": 1.99}
2	cherries	2023	{"amount":3, "price": 3.99}
3	apples	2024	{"amount":10, "price": 2.99}
4	bananas	2024	{"amount":8, "price": 2.29}
5	cherries	2024	{"amount":9, "price": 3.89}

What defines tidy data?

1. Each variable forms one column

	fruit	year	amount	price
0	apples	2023	5	2.99
1	bananas	2023	4	1.99
2	cherries	2023	3	3.99
3	apples	2024	10	2.99
4	bananas	2024	8	2.29
5	cherries	2024	9	3.89

2. Every observation forms one row

3. Every value has its own cell

Benefits of tidy data

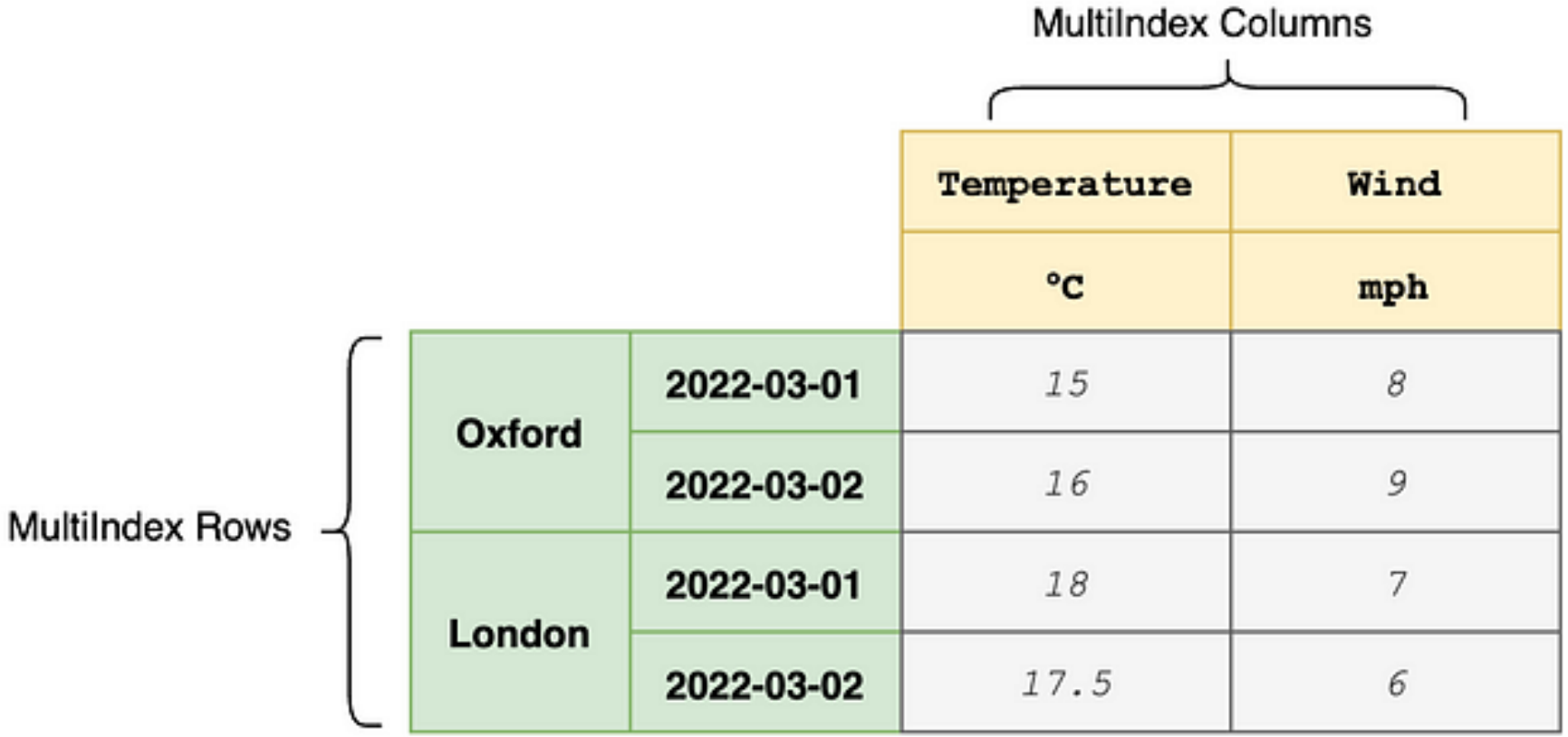
- **Having data in a tidy format often simplisfies data analysis and processing:**
  - Vectorized operations on columns (e.g. aggregations)
  - Data visualization packages (Seaborn, Plotly, Altair) prefer tidy data
  - Facilitates merge operations
  - Often necessary preprocessing step before statistical analysis of machine learning

- **Some other use cases favor other formatting:**
  - Reporting and presentation often favors a “wide” format
  - Heatmap visualizations often favor a “wide” format

→ **Become confident with reshaping of data**

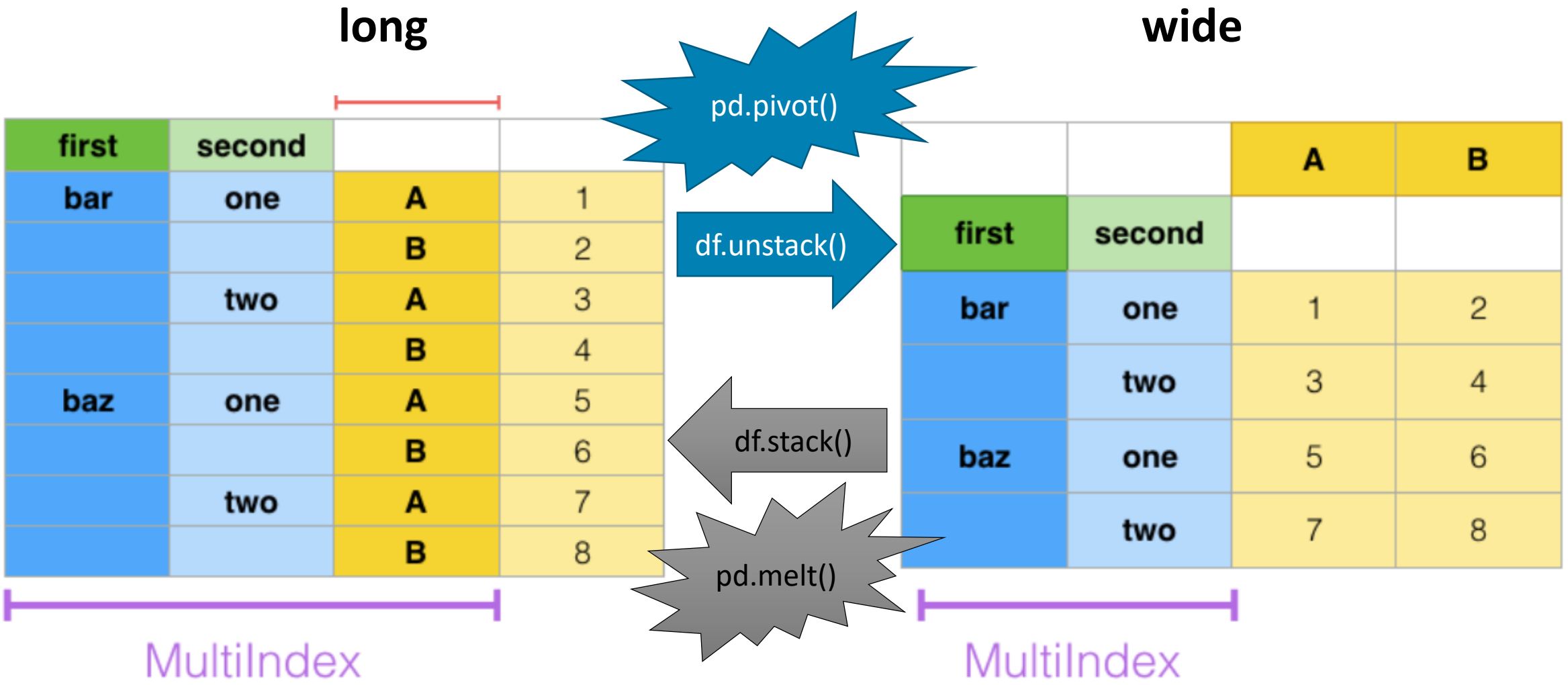
Set and reset index

- Index methods
  - **`set_index`**: columns → indices
  - **`reset_index`**: indices → column
  - **`swaplevels`**



		Temperature	Wind
		°C	mph
Oxford	2022-03-01	15	8
	2022-03-02	16	9
London	2022-03-01	18	7
	2022-03-02	17.5	6

Stack and unstack



Shortcuts: pivot and melt

- **pivot**: set_index + unstack
- **melt**: stack + reset_index

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

Other data preparation and cleaning

Missing values

Pandas methods:

- **isna** or **notna:** identify missing values
- **dropna**: dropping rows (axis=0) or columns (axis=1) with missing values
- **fillna**: replace missing values by specific other value
- **fillna(method=„pad“)**: forward propagation
- **interpolate**: e.g. linear interpolation between existing values in time series

Imputation via other Python packages:

- sklearn
- fancyimpute
- missingno (visualization of NaN)

NaN

Types

Pandas dtype	Usage
object	Text or mixed numeric and non-numeric
int64	Integer numbers
float64	Floating point numbers
bool	True/False values
datetime64	Date and time values
timedelta[ns]	Differences between two datetimes
category	Finite list of text values

- **Get Data Types:**
  - df.dtypes
  - df.info()
- **Convert Type:**
  - astype(type)
  - to_numeric()
  - to_datetime()
- **Dedicated functions/attributes:**
  - pd.Series.str.count()
  - pd.Series.dt.year
  - …

Many more useful functions

- **drop()**: Dropping columns or rows
- **rename()**: Rename columns or row index
- **transform():** e.g. add aggregates at the group level to original DataFrame
- **apply():** apply some function to rows or columns of the DataFrame
- **…**