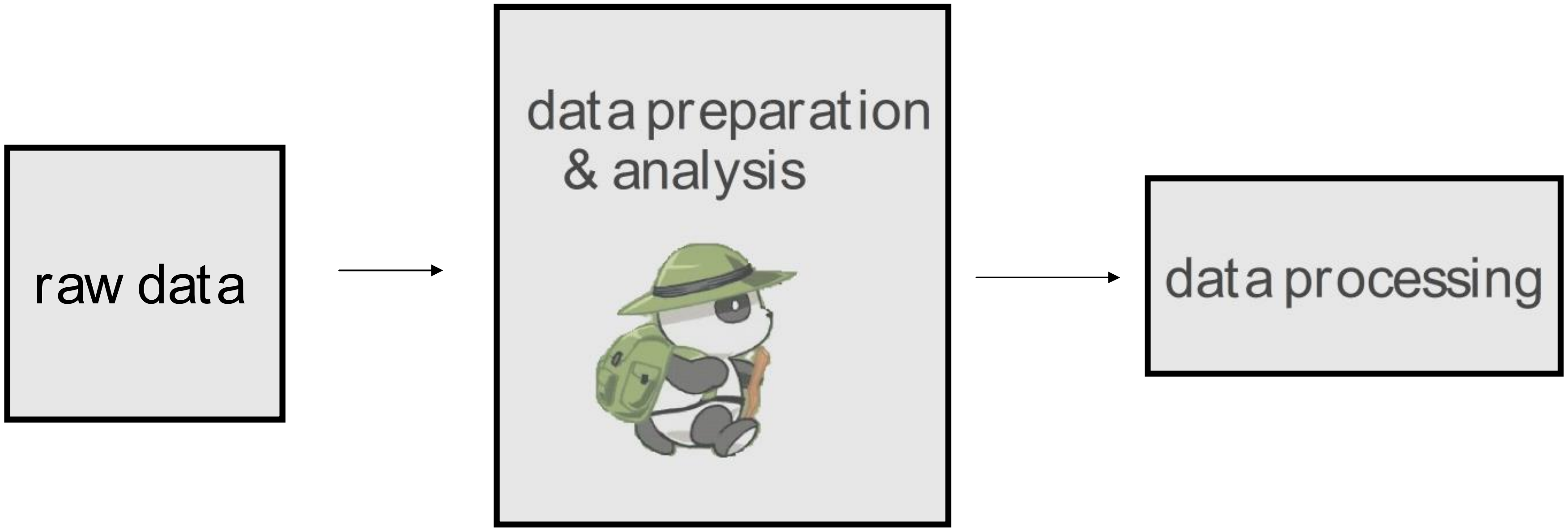
Pandas: now available for data analysis

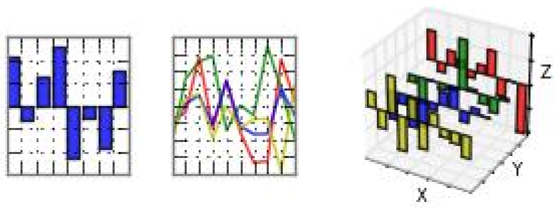
Simplify data wrangling

- data munging / preparation / cleaning / integration process is slow, error prone and time consuming



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Panel Data System: a python data analysis library

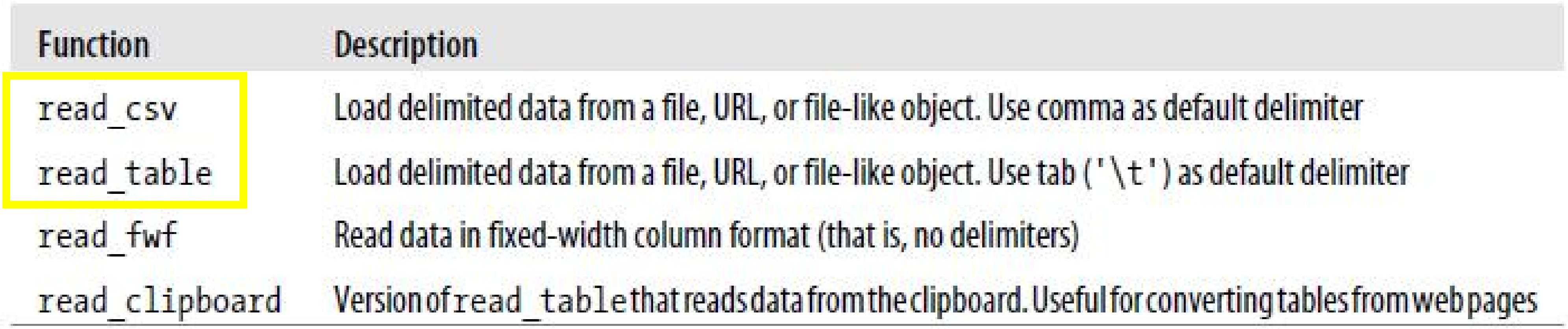
- rich data manipulation tool for working with relational / labeled data
- Fast, intuitive data structures
- great for analyzing data interactively
- Very active language
- fills the gap between python and domain-specific languages like R

Important features

1. Basics Data Structures:
   - Series & DataFame objects
2. Label-based indexing
   - Selecting / filtering data
   - Hierarchical indexing
3. Missing Data
4. Descriptive statistics
5. Many SQL-like operations (in-memory)
   - groupby: aggregating & transforming data
   - Join, merge, concatenate
6. Pivot tables & reshaping

- Time series operations (not covered)
  - Indexing conveniences
  - Resample time series
  - Moving window fn & stats
  - Date range generation
  - Frequency conversion
  - Date shifting and lagging

What can be a panda?



Function	Description
`read_csv`	Load delimited data from a file, URL, or file-like object. Use comma as default delimiter
`read_table`	Load delimited data from a file, URL, or file-like object. Use tab (`'\t'`) as default delimiter
`read_fwf`	Read data in fixed-width column format (that is, no delimiters)
`read_clipboard`	Version of `read_table` that reads data from the clipboard. Useful for converting tables from web pages

Interacting with databases

- SQL-based databases are in wide use
    - SQL Server, PostgreSQL, and MySQL
- pandas now has one of the fastest in-memory database join operators out there

- loading data from SQL into a DataFrame is fairly straightforward
- example: an in-memory SQLite database using Python's built-in sqlite3 driver

```
>>> import sqlite3
```

Data Structures in Pandas

Series & DataFrames

Series

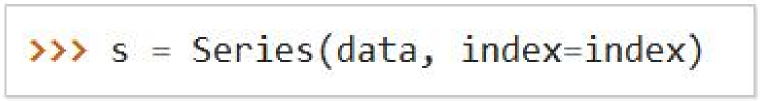
- A 1D labeled NumPy arrary

Data: any type (python Dict, ndarray, scalar value, string)

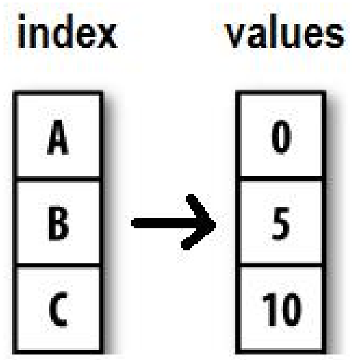
Index: must be a valid key in dict object, do not need to be ordered

- **Duplicates** are possible (but result in reduced functionality

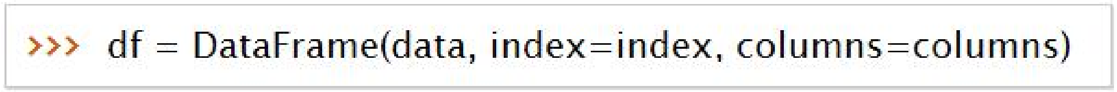
- Supports vectorized operations



```
>>> s = Series(data, index=index)
```

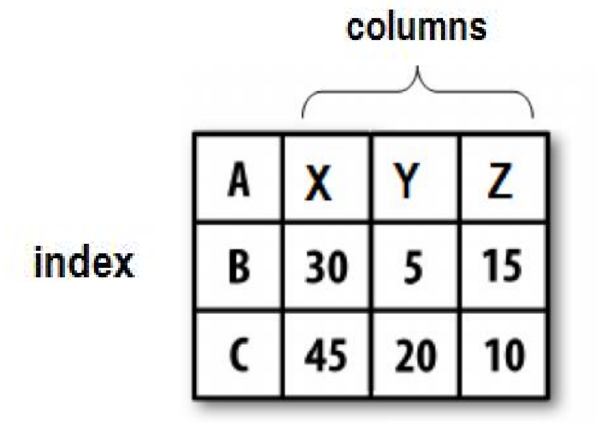


DataFrame



```
>>> df = DataFrame(data, index=index, columns=columns)
```

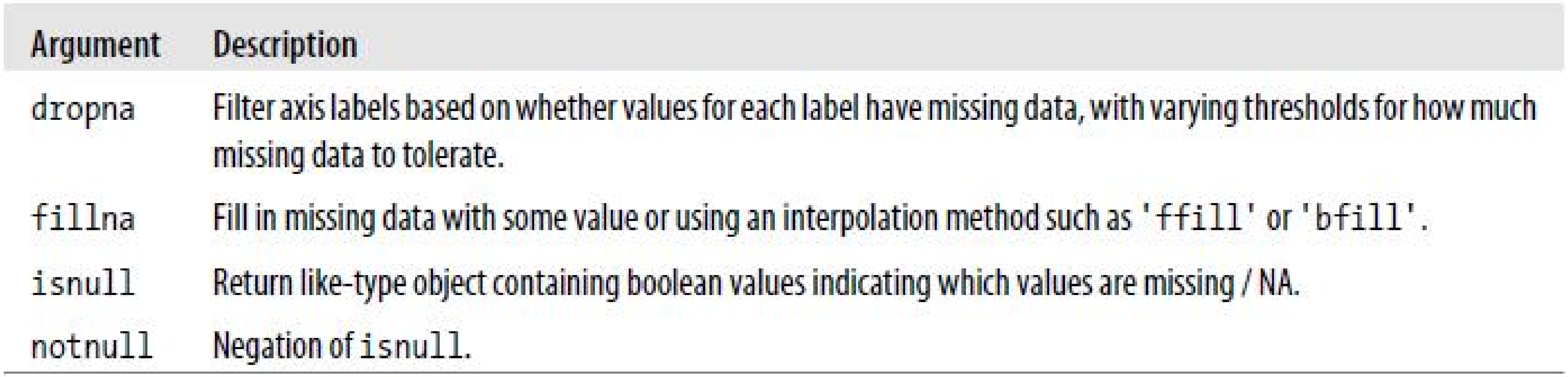
- R users: “data.frame” on steriods!
- **Data**: any type
- Each column can be of different type
- **Size Mutable**: insert & delete columns



A	X	Y	Z
B	30	5	15
C	45	20	10

Missing data

- NaN is the standard missing data marker used in pandas



Argument	Description
`dropna`	Filter axis labels based on whether values for each label have missing data, with varying thresholds for how much missing data to tolerate.
`fillna`	Fill in missing data with some value or using an interpolation method such as `'ffill'` or `'bfill'`.
`isnull`	Return like-type object containing boolean values indicating which values are missing / NA.
`notnull`	Negation of `isnull`.

Index

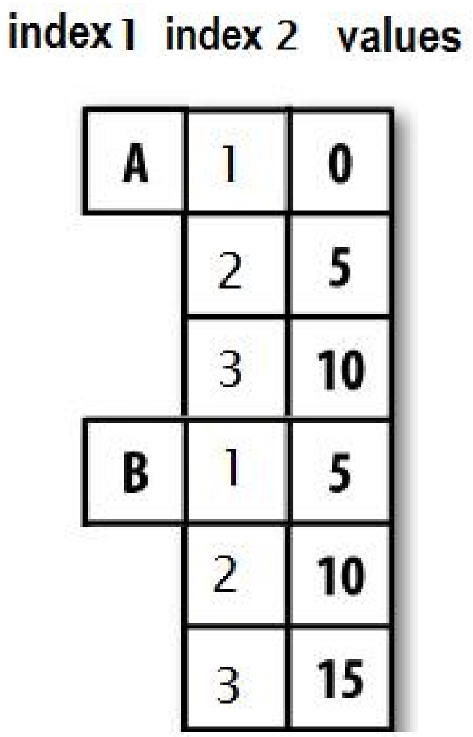
- Indexes enable:
  - Enables 'alignment-free' fee programming
  - Automatic and explicit data alignment
  - Fast lookups: (O(l) or O(log n)) selecting data
  - Identifies data (i.e. provides metadata)
  - Used for join/merge/reshape /pivot table operations & interactive console display
- May not always be meaningful but it's important to know it exists
- Can be ignored if needed (set `ignore_index=True`)

Hierarchical indexes

- provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure

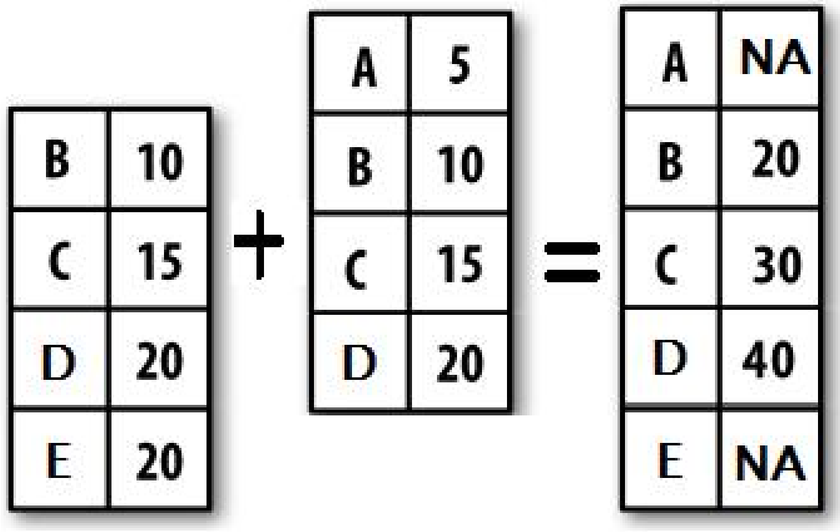
- Enables easy group selection
- Multiple levels
- hierarchical indexing not found in R

- A tuple at each index



Join

- Binary operations are joins
- When indexes match, operation
  is performed, otherwise put “NA”
- Automatic join data alignment functionality
- DataFrame objects joins & aligns on both axes
- Doesn’t discard any info



B	10
C	15
D	20
E	20

+

A	5
B	10
C	15
D	20

=

A	NA
B	20
C	30
D	40
E	NA

DataFrame operations

Join, Merge, Concatenate, Combine_First

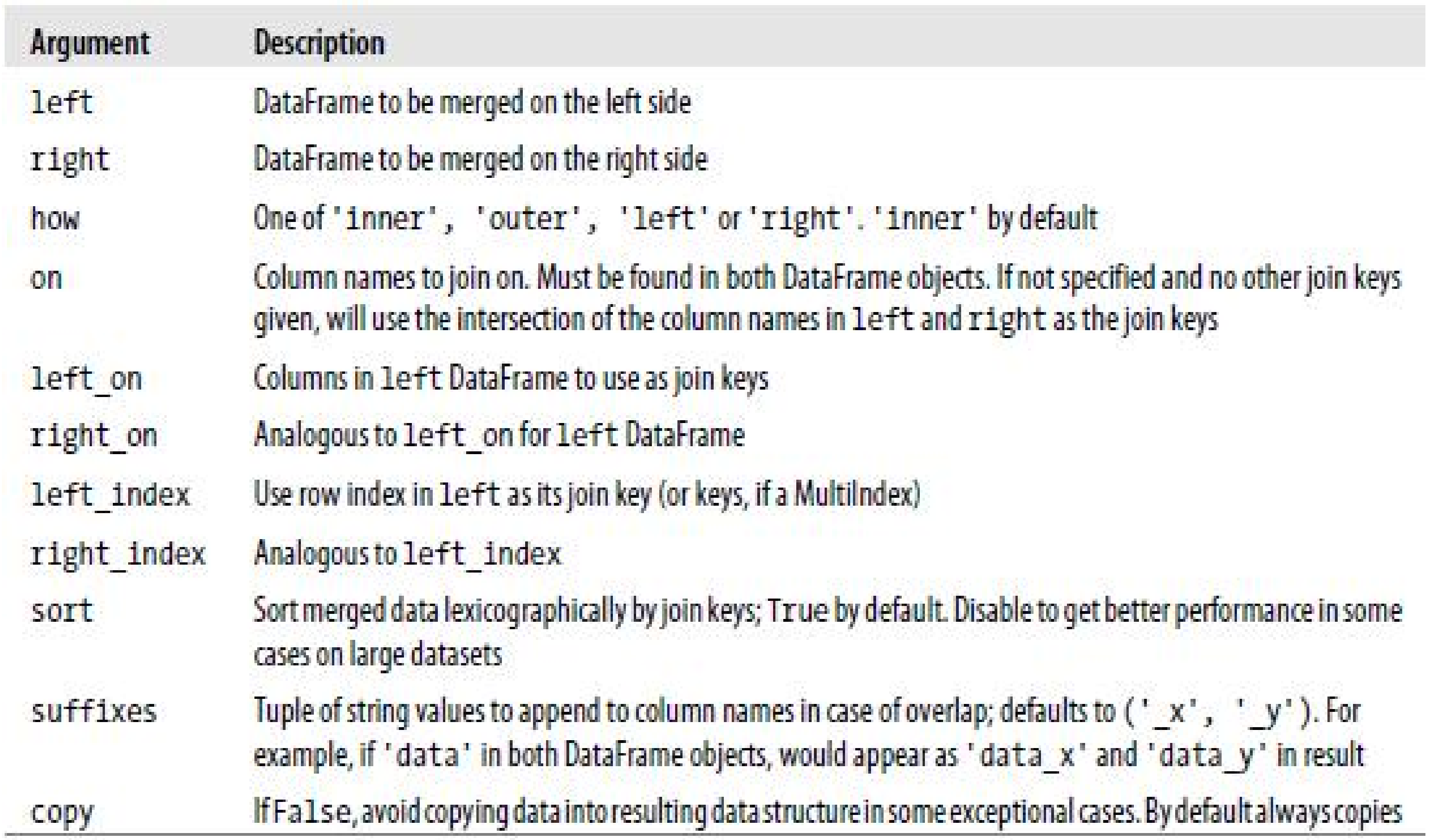
- SQL-style join of both dataframes using a non-simplistic criteria

- *Merge* or *join* operations combine data sets by linking rows using one or more *keys*

DataFrames: merging & combining

- **pandas.merge**
  - connects rows in DataFrames based on one or more keys (like database *join* operations in SQL or other relational databases)
- **pandas.concat**
  - glues or stacks together objects along an axis
- **combine_first**
  - instance method enables splicing together overlapping data to fill in missing values in one object with values from another
  - similar to NumPy's where function, with data alignment

merge function arguments



Argument	Description
left	DataFrame to be merged on the left side
right	DataFrame to be merged on the right side
how	One of 'inner', 'outer', 'left' or 'right'. 'inner' by default
on	Column names to join on. Must be found in both DataFrame objects. If not specified and no other join keys given, will use the intersection of the column names in left and right as the join keys
left_on	Columns in left DataFrame to use as join keys
right_on	Analogous to left_on for left DataFrame
left_index	Use row index in left as its join key (or keys, if a MultiIndex)
right_index	Analogous to left_index
sort	Sort merged data lexicographically by join keys; True by default. Disable to get better performance in some cases on large datasets
suffixes	Tuple of string values to append to column names in case of overlap; defaults to ('_x', '_y'). For example, if 'data' in both DataFrame objects, would appear as 'data_x' and 'data_y' in result
copy	If False, avoid copying data into resulting data structure in some exceptional cases. By default always copies

Data transformation

- Removing duplicates (`.duplicated(), .drop_duplicates()`)
- Transforming data using a function or mapping `(.map, lambda fns)`
- Replacing values (`.replace`)
- Computing indicator/dummy variables (`.get_dummies`)
- Detecting and filtering outliers
- Discretization and binning (`.cut`)

Descriptive stats

- No statistical modeling outside of linear and panel regression
- use statsmodels or scikit learn
- `DataFrame.describe()` gives descriptive statistics

Differences with NumPy:

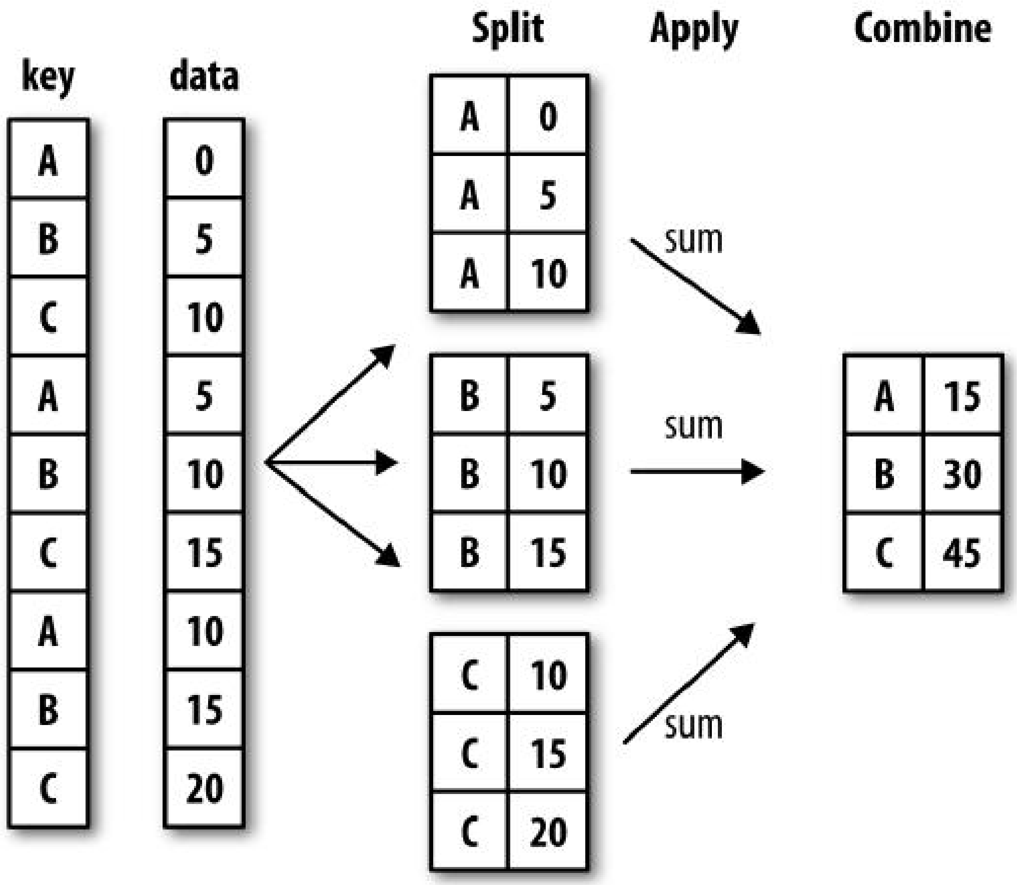
- For Series and DataFrame objects, `var` normalizes by N-1 to produce unbiased estimates of the sample variance, while NumPy's var normalizes by N, which measures the variance of the sample
- `cov` normalizes by N-1 in both pandas and NumPy

Data Aggregation and Group Operations

groupby, pivot Tables

GroupBy

- The GroupBy method let's you perform SQL-like grouping operations
- Similar to SQL-based tool, itertools
- groupby is a "pythonic" way to transform, pass to a function, and get summary statistics



groupby

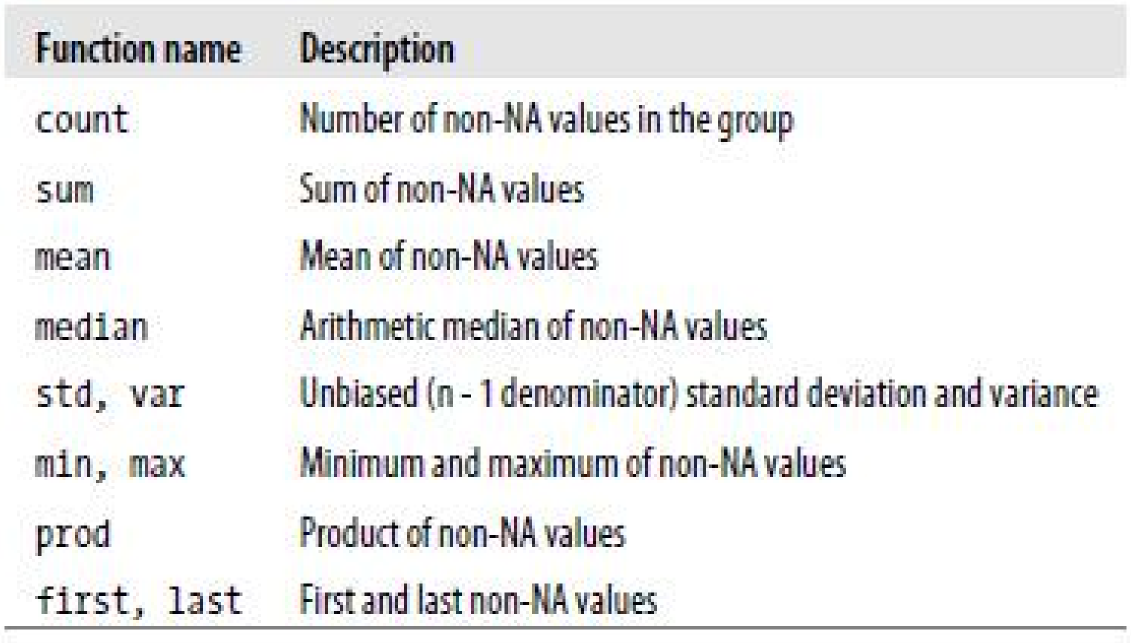
Has 3 built-in groups of functions:

- **agg/transform**: specialized, faster
  - agg: any data transformation that produces scalar values from arrays (eg. mean, count, min and sum)
  - transform: applies a function to each group, alter values, not their size (eg. standardizing data)

- **apply**: completely generic, slower
  - splits the object being manipulated into pieces, invokes the passed function on each piece, then attempts to concatenate the pieces together

Optimized groupby methods

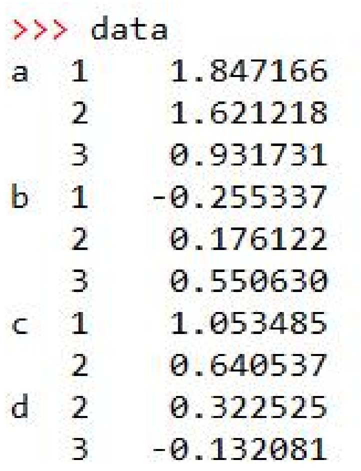
- custom aggregation functions are much slower than the optimized functions



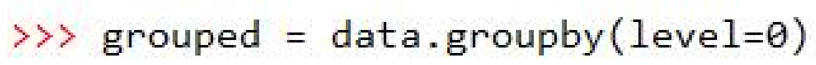
Function name	Description
`count`	Number of non-NA values in the group
`sum`	Sum of non-NA values
`mean`	Mean of non-NA values
`median`	Arithmetic median of non-NA values
`std, var`	Unbiased (n - 1 denominator) standard deviation and variance
`min, max`	Minimum and maximum of non-NA values
`prod`	Product of non-NA values
`first, last`	First and last non-NA values

agg

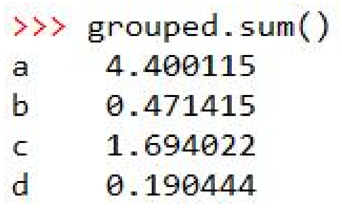
Return a scalar for each group



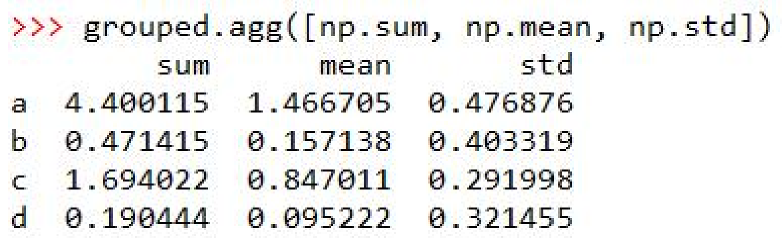
```
>>> data
a  1     1.847166
   2     1.621218
   3     0.931731
b  1    -0.255337
   2     0.176122
   3     0.550630
c  1     1.053485
   2     0.640537
d  2     0.322525
   3    -0.132081
```



```
>>> grouped = data.groupby(level=0)
```



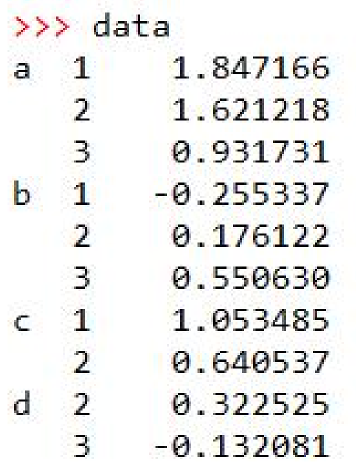
```
>>> grouped.sum()
a    4.400115
b    0.471415
c    1.694022
d    0.190444
```



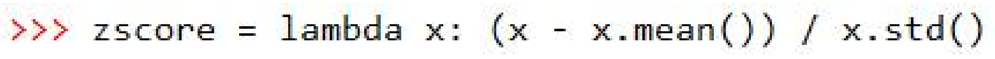
```
>>> grouped.agg([np.sum, np.mean, np.std])
        sum      mean       std
a  4.400115  1.466705  0.476876
b  0.471415  0.157138  0.403319
c  1.694022  0.847011  0.291998
d  0.190444  0.095222  0.321455
```

transform

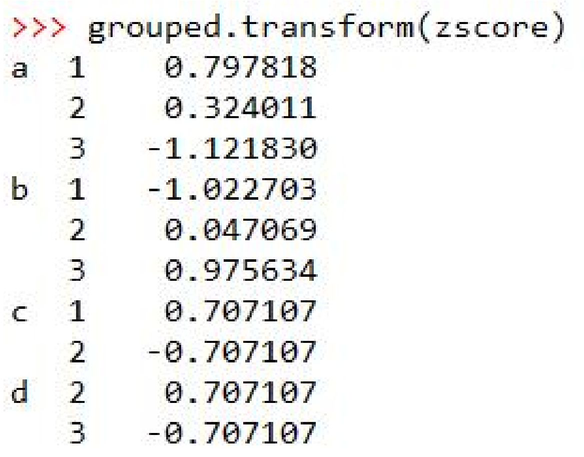
Return same sized object



```
>>> data
a  1    1.847166
   2    1.621218
   3    0.931731
b  1   -0.255337
   2    0.176122
   3    0.550630
c  1    1.053485
   2    0.640537
d  2    0.322525
   3   -0.132081
```



```
>>> zscore = lambda x: (x - x.mean()) / x.std()
```



```
>>> grouped.transform(zscore)
a  1    0.797818
   2    0.324011
   3   -1.121830
b  1   -1.022703
   2    0.047069
   3    0.975634
c  1    0.707107
   2   -0.707107
d  2    0.707107
   3   -0.707107
```

Pivot tables

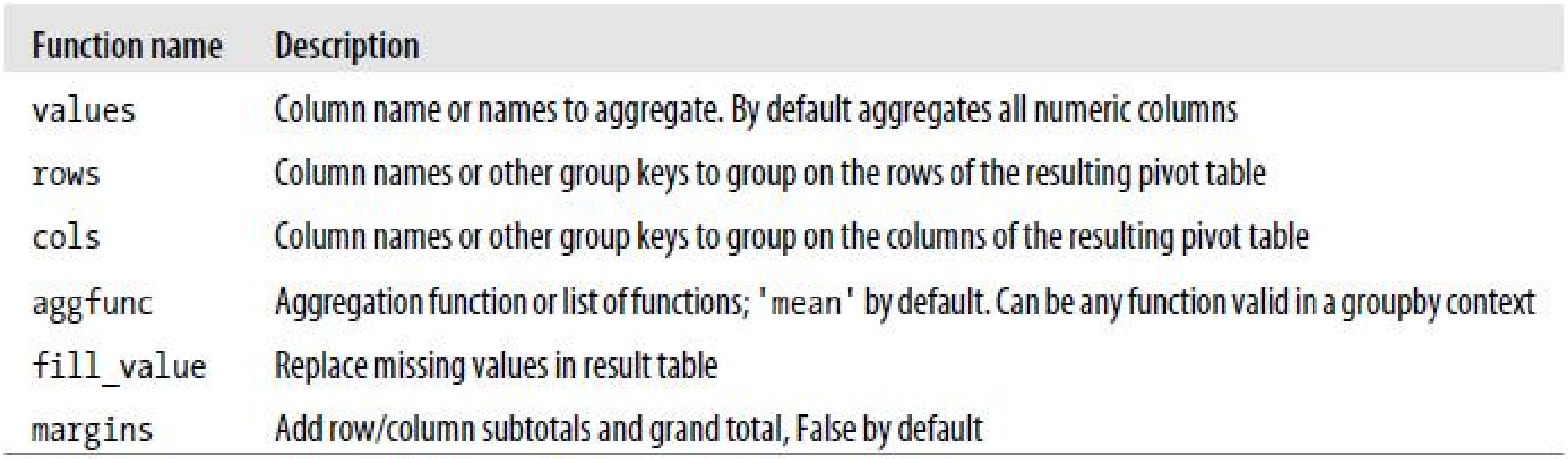
- rearranging tabular data
- aggregating a data set on some number of keys, then reshaping it to form a 2D table with the stratified aggregated values

- Pivot tables in Python with pandas are made possible using the
- groupby facility described in this chapter combined with reshape operations utilizing hierarchical indexing
- DataFrame has a pivot_table method, and additionally there is a top-level pandas.pivot_table function

Two basic actions:

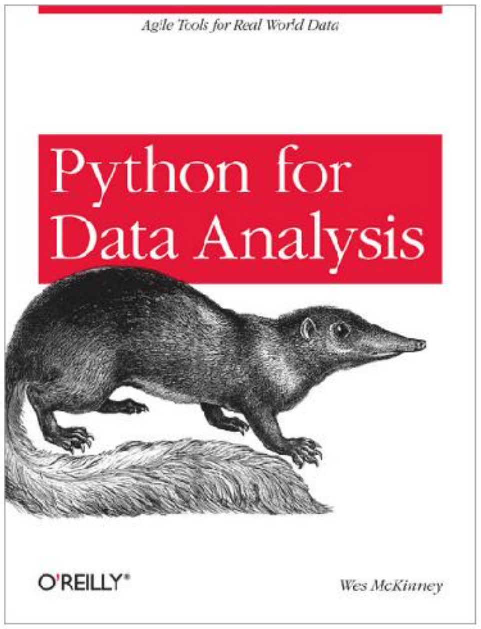
- stack: “rotates” or pivots the columns into the rows, producing a Series
- unstack: pivots the rows into the columns

pivot_table options



Function name	Description
`values`	Column name or names to aggregate. By default aggregates all numeric columns
`rows`	Column names or other group keys to group on the rows of the resulting pivot table
`cols`	Column names or other group keys to group on the columns of the resulting pivot table
`aggfunc`	Aggregation function or list of functions; `'mean'` by default. Can be any function valid in a groupby context
`fill_value`	Replace missing values in result table
`margins`	Add row/column subtotals and grand total, False by default

Resources



Related Library: PyTables

- a popular database which optimizes writing, reading and analyzing large data sets out-of-memory, i.e. on disk

www.pandas.pydata.org

SQL → Pandas

- http://www.gregreda.com/2013/01/23/translating-sql-to-pandas-part1/

WesMcKinney's blog:

http://wesmckinney.com/blog/?p=315