Inspecting pandas DataFrames

- **Structure**: `df.info()`
- **Top and bottom**: `df.head()`, `df.tail()`
- **Columns names and dtypes**: `df.columns`, `df.dtypes`
- **Index**: `df.index`
- **Select column**: `df['`*name*`']` or `df.`*name*
- **Select rows with Boolean condition**:
  `df[`*bool array*`]`, e.g., `df[df.year > 1980]` or
  `df.query('`*condition*`')`, e.g., `df.query('year > 1980')`

pandas indexing

- **Set index**:
  `df = df.set_index('name')` or
  `df.set_index(['name1','name2'])` for multiindex
- **Sort index**: `df = df.sort_index()`
- **Select rows by index value:** `df.loc[value]`
- **Select by index slice (inclusive):** `df.loc[startvalue:endvalue]`
- **Select rows and column:** `df.loc[value, column name]`
- **Select row by integer index (depends on ordering):** `df.iloc[int]`
- **Select row and column by integer index**: `df.iloc[row int, col int]`

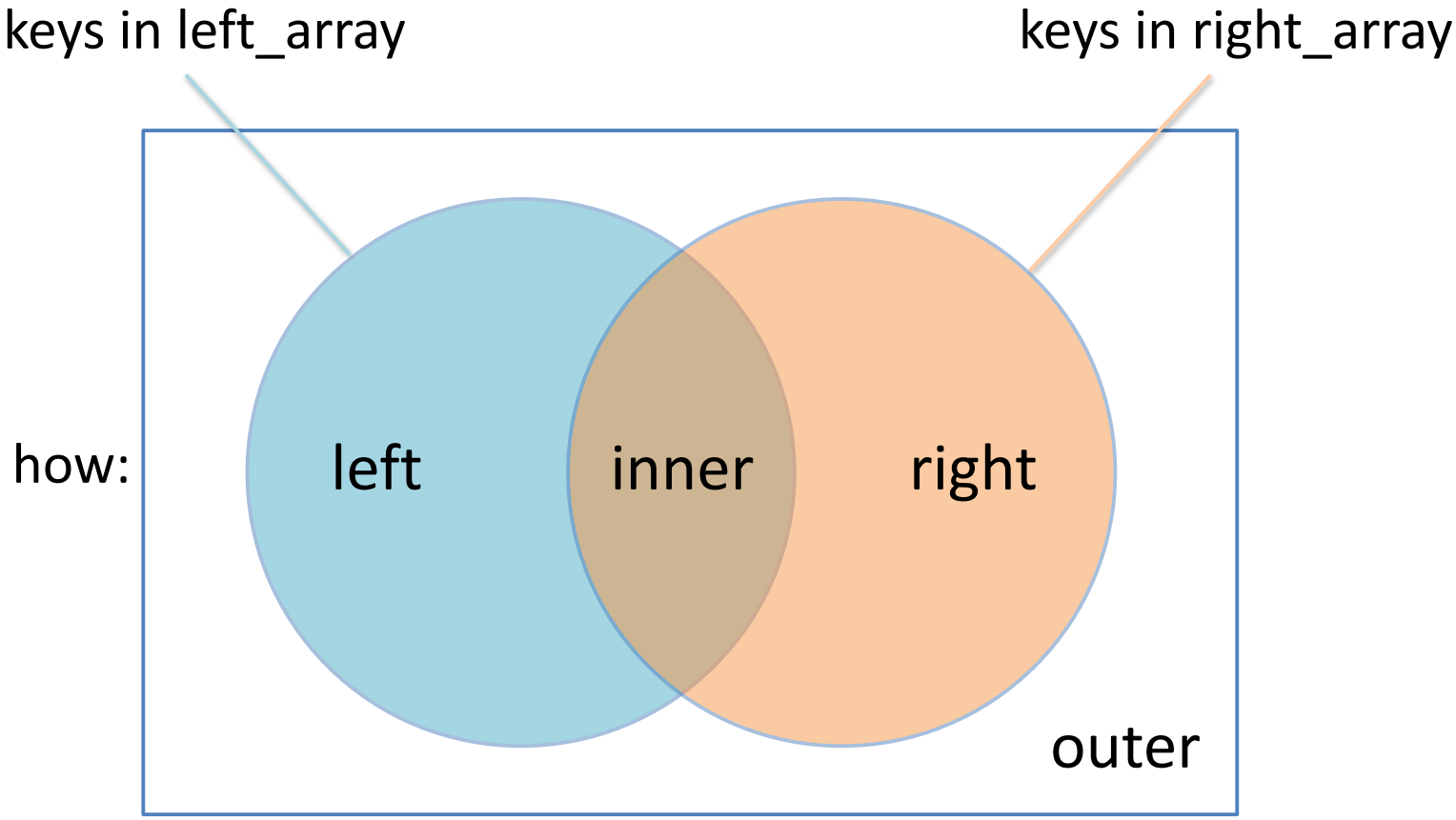
pandas-supported formats

Format	Read with	Write with	Dependency
CSV	`read_csv`	`to_csv`	
JSON	`read_json`	`to_json`	
HTML/XML	`read_html/read_xml`	`to_html/to_xml`	lxml
Excel	`read_excel`	`to_excel`	openpyxl
HDF5	`read_hdf`	`to_hdf`	PyTables
Feather/Parquet (Apache Arrow)	`read_feather/ read_parquet`	`to_feather/ to_parquet`	pyarrow
Stata	`read_stata`	`to_stata`	
SAS/SPSS	`read_sas/read_spss`		pyreadstat
SQL	`read_sql`	`to_sql`	ADBC drivers/ SQLAlchemy
pickle	`read_pickle`	`to_pickle`	

pandas read_csv options

Option	Description	Examples
sep	column separator	`';'`, `'\t'` (tabs), `'\s+'` (any # of spaces)
header	row to provide column names	`None`, `0`, `[0,1]` (multilevel)
skiprows	lines to skip	`2` (from start), `[0,1]` (as list)
skipfooter	lines to skip at end	`10`
nrows	number of rows to read	`100`
index_col	column to use as index	`0`, `[0,1]` (multi-index)
usecols	read subset of columns	`[0,2,3]`
names	column names	`['alpha', 'beta', 'gamma']`
dtype	column data types (dictionary)	`{'age': 'float64'}`
parse_dates	columns to parse as dates	`['`*startdate*`','`*enddate*`']`
date_format	format for date parsing	`'ISO8601'`, `'%d/%m/%Y'` (dict also possible)
converters	Python funcs to parse columns	`{'`*startdate*`':` *dateparser*`}`
thousands/ decimal	thousands and decimal separators	`','`, `'.'`
on_bad_lines	what to do on bad line	`'error'`, `'warn'`, `'skip'`, *Python function*
encoding	file encoding	`'utf-8'`

pandas joins




```
pd.merge(left_array, right_array, how='mode', on='field')
```