Learning goals:

- Understand the Series and Data Frame data structures.
- Learn how to use Google.
- Learn how to read pandas documentation.
- Make progress on A2.

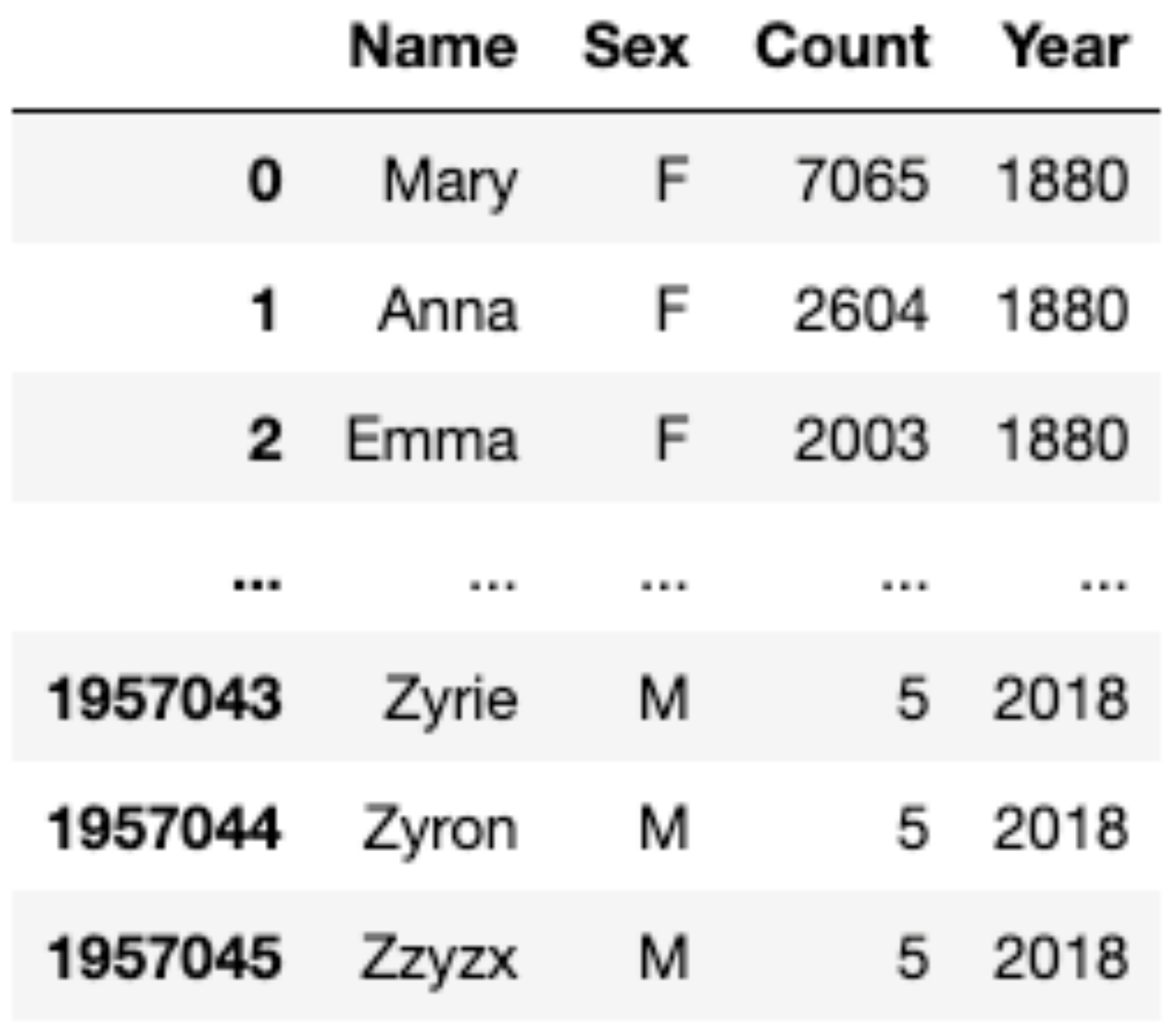
pandas and A2

COGS 108 Spring 2020
Will McCarthy
Discussion 4

wmccarthy@ucsd.edu
OH: Fri 10a-11a on Zoom

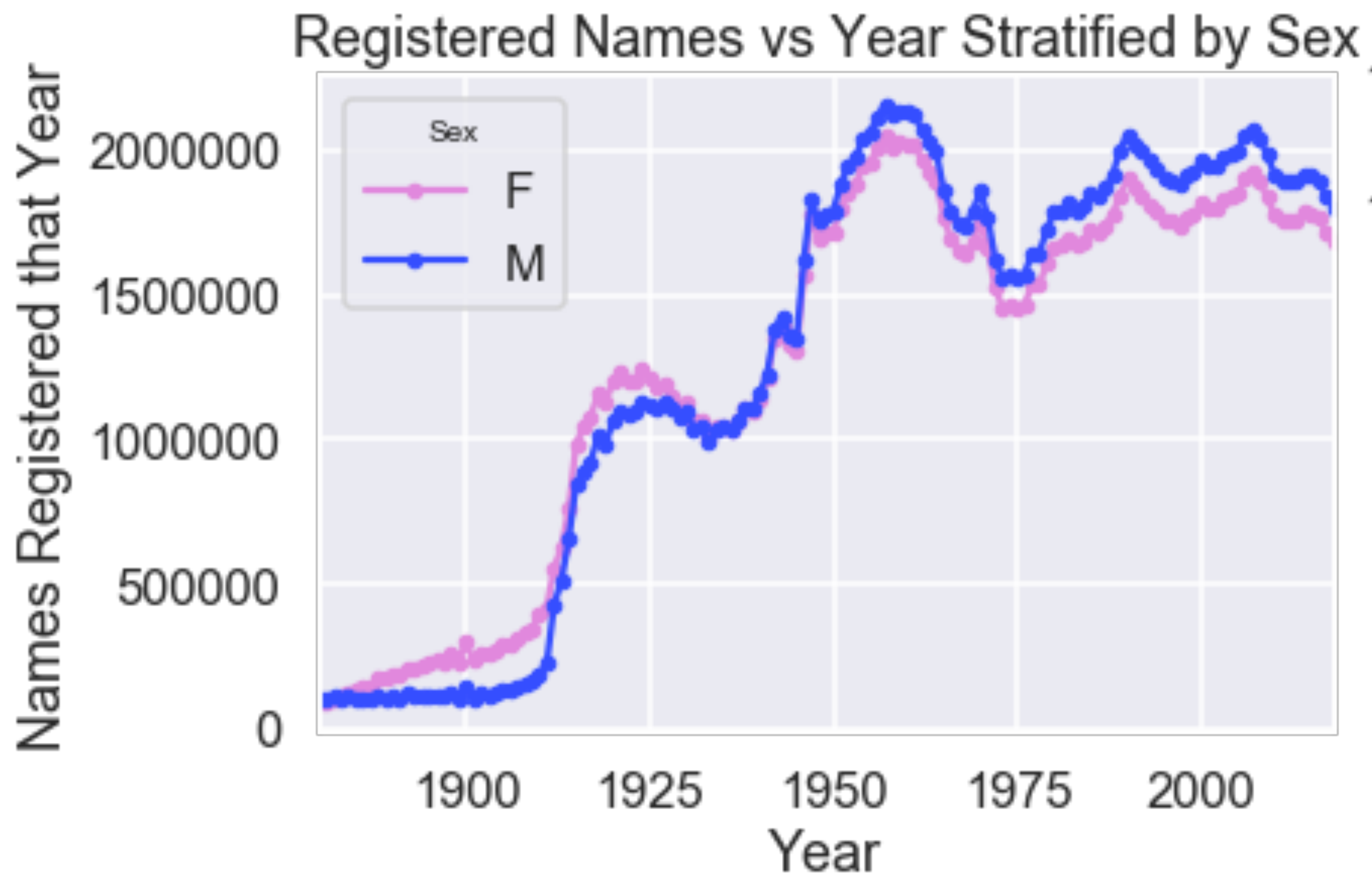
Welcome to the wonderful world of pandas!

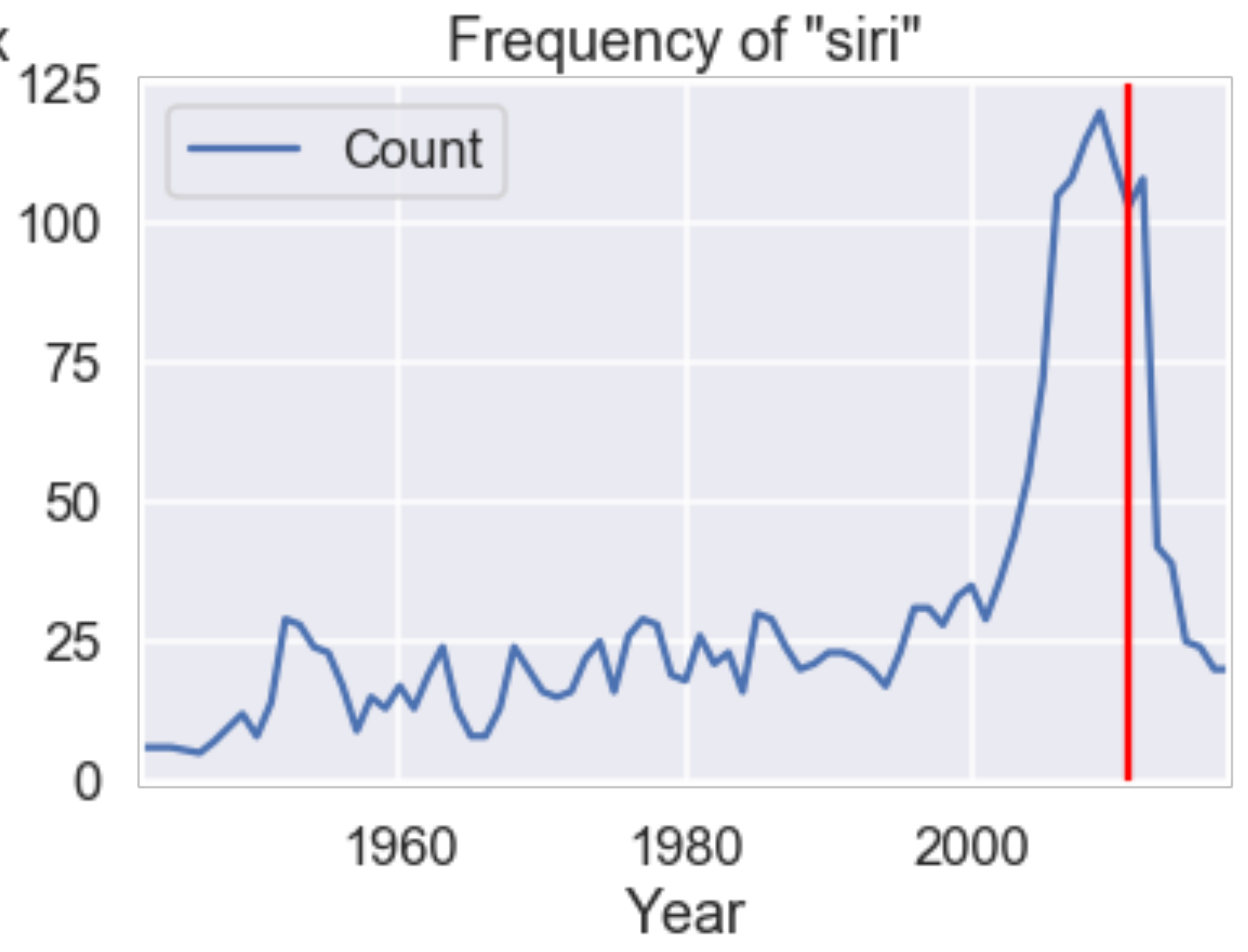
Pandas is really useful!



	Name	Sex	Count	Year
0	Mary	F	7065	1880
1	Anna	F	2604	1880
2	Emma	F	2003	1880
...	...	...	...	...
1957043	Zyrie	M	5	2018
1957044	Zyron	M	5	2018
1957045	Zzyzx	M	5	2018

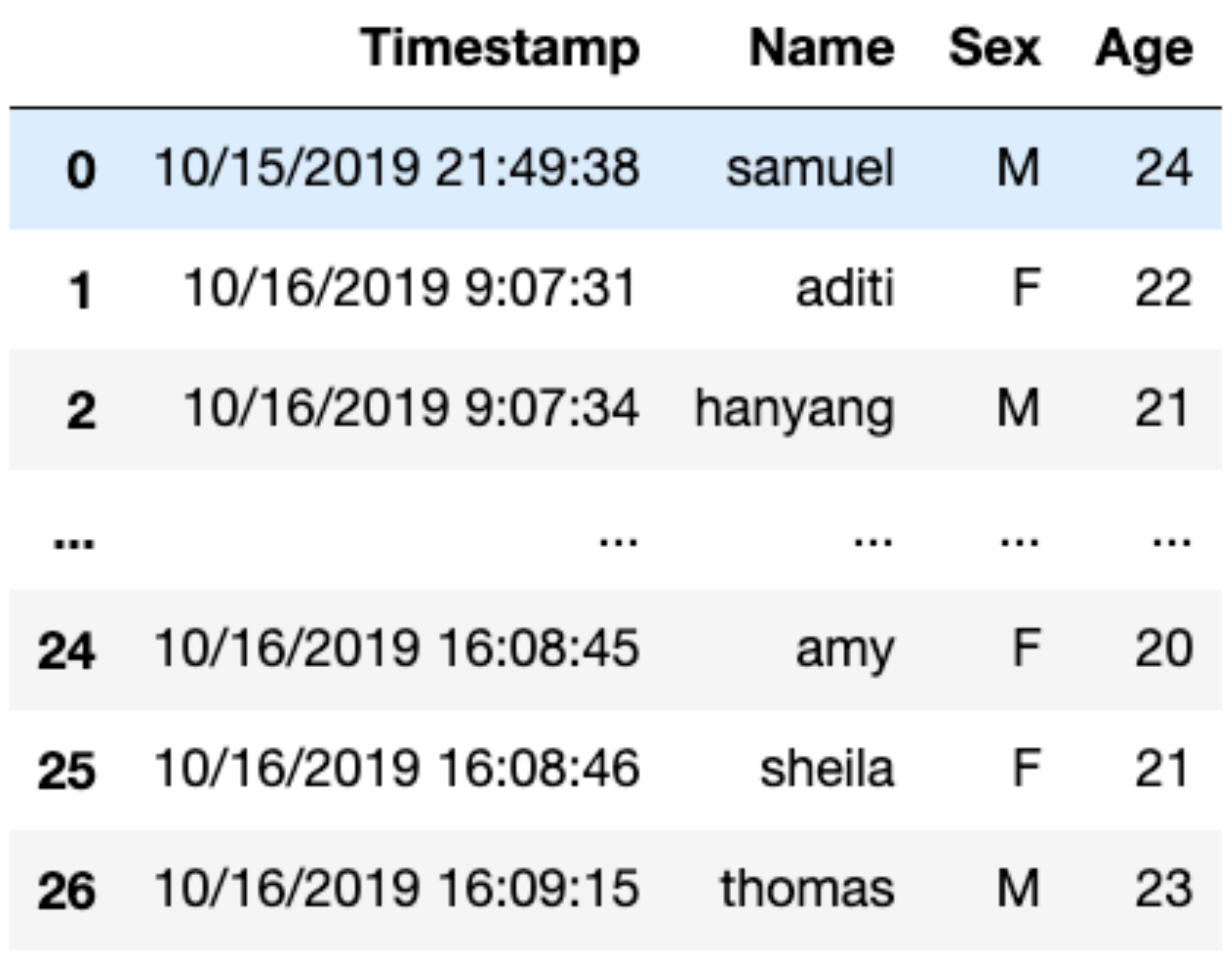
1957046 rows × 4 columns



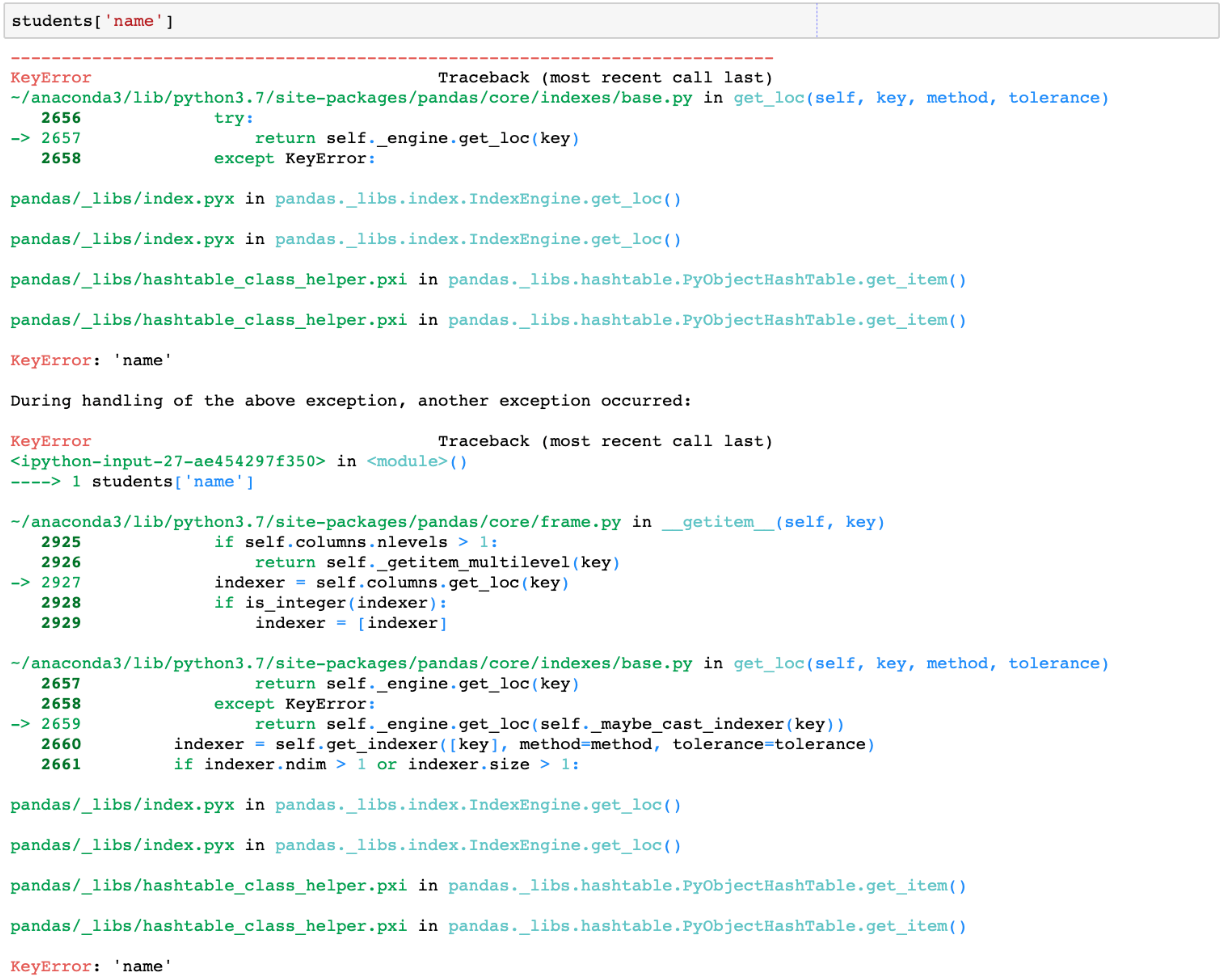


It converts python into a usable (and good!) data analysis tool

Pandas has terrible error messages



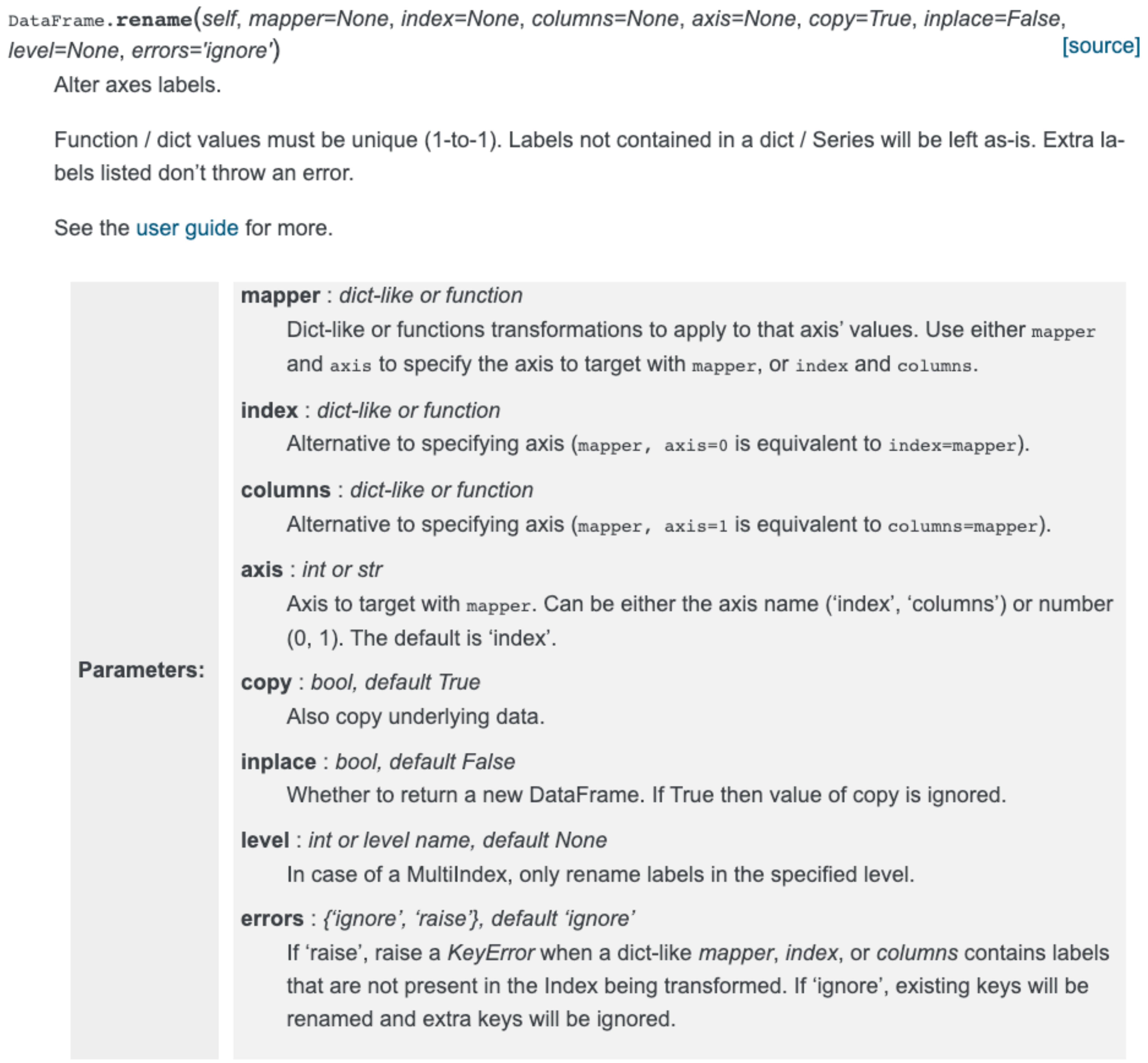
	Timestamp	Name	Sex	Age
0	10/15/2019 21:49:38	samuel	M	24
1	10/16/2019 9:07:31	aditi	F	22
2	10/16/2019 9:07:34	hanyang	M	21
...	...	...	...	...
24	10/16/2019 16:08:45	amy	F	20
25	10/16/2019 16:08:46	sheila	F	21
26	10/16/2019 16:09:15	thomas	M	23



```
students['name']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2656             try:
-> 2657                 return self._engine.get_loc(key)
   2658             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'name'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-27-ae454297f350> in <module>()
----> 1 students['name']

~/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2925             if self.columns.nlevels > 1:
   2926                 return self._getitem_multilevel(key)
-> 2927             indexer = self.columns.get_loc(key)
   2928             if is_integer(indexer):
   2929                 indexer = [indexer]

~/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2657                 return self._engine.get_loc(key)
   2658             except KeyError:
-> 2659                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2660         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2661         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'name'
```

Pandas has unfriendly documentation



`DataFrame.rename`(*self*, *mapper=None*, *index=None*, *columns=None*, *axis=None*, *copy=True*, *inplace=False*, *level=None*, *errors='ignore'*) [source]

Alter axes labels.

Function / dict values must be unique (1-to-1). Labels not contained in a dict / Series will be left as-is. Extra labels listed don't throw an error.

See the user guide for more.

Parameters:	**mapper** : *dict-like or function* Dict-like or functions transformations to apply to that axis' values. Use either `mapper` and `axis` to specify the axis to target with `mapper`, or `index` and `columns`. **index** : *dict-like or function* Alternative to specifying axis (`mapper, axis=0` is equivalent to `index=mapper`). **columns** : *dict-like or function* Alternative to specifying axis (`mapper, axis=1` is equivalent to `columns=mapper`). **axis** : *int or str* Axis to target with `mapper`. Can be either the axis name ('index', 'columns') or number (0, 1). The default is 'index'. **copy** : *bool, default True* Also copy underlying data. **inplace** : *bool, default False* Whether to return a new DataFrame. If True then value of copy is ignored. **level** : *int or level name, default None* In case of a MultiIndex, only rename labels in the specified level. **errors** : *{'ignore', 'raise'}, default 'ignore'* If 'raise', raise a *KeyError* when a dict-like *mapper*, *index*, or *columns* contains labels that are not present in the Index being transformed. If 'ignore', existing keys will be renamed and extra keys will be ignored.

Also, there are typically many ways to do the same thing in pandas.

3 skills that will save you 5+ hours on A2:

- **Knowing the difference between a pandas Series and Data Frame.**
- **Knowing how to use Google effectively.**
- **Knowing how to read the pandas documentation.**

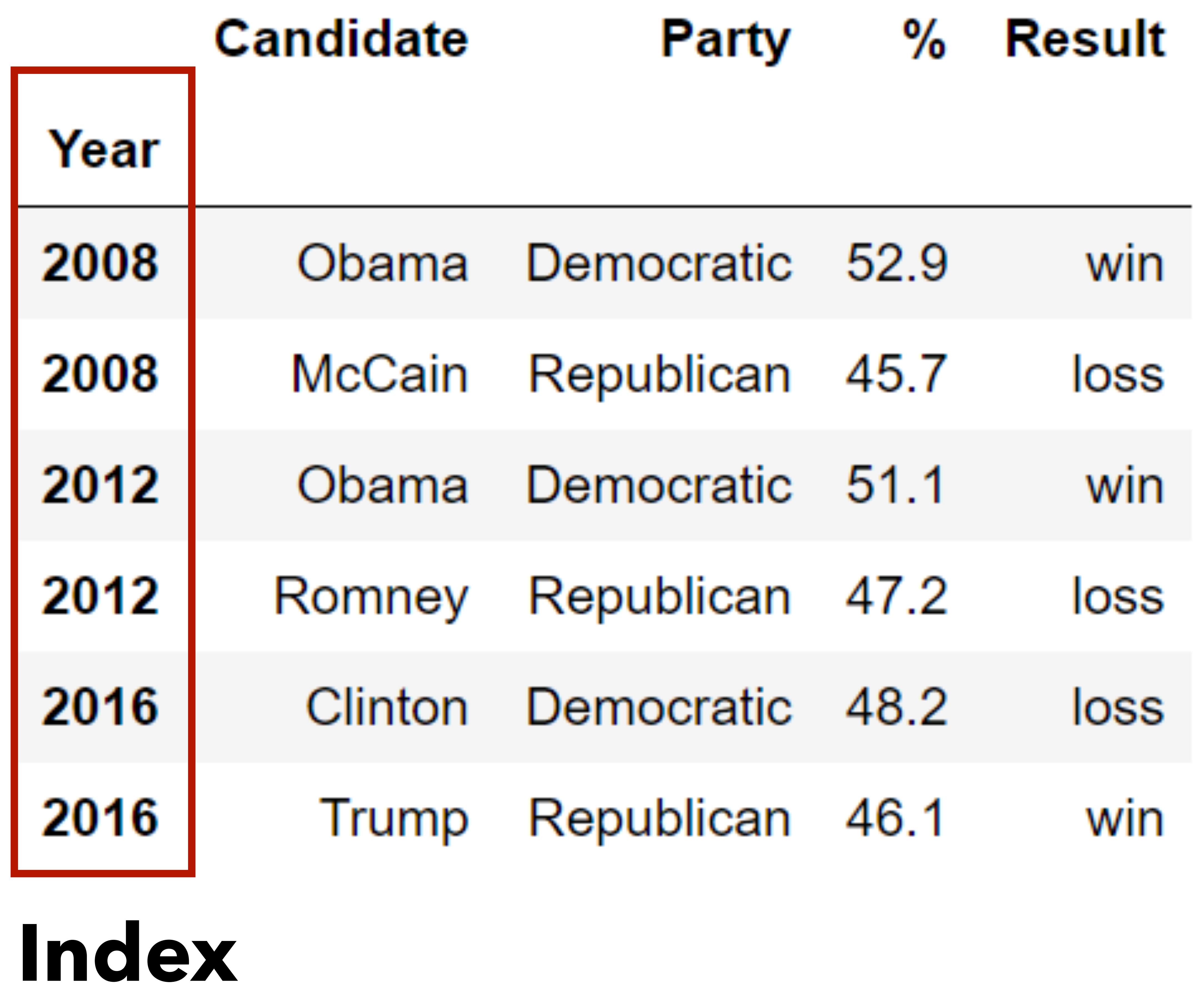
What's a Data Frame?

Data Frame: two-dimensional table of data.

All columns are the same type (but not rows).

Every row and every column has a label.

We call the set of row labels the Index of a DataFrame

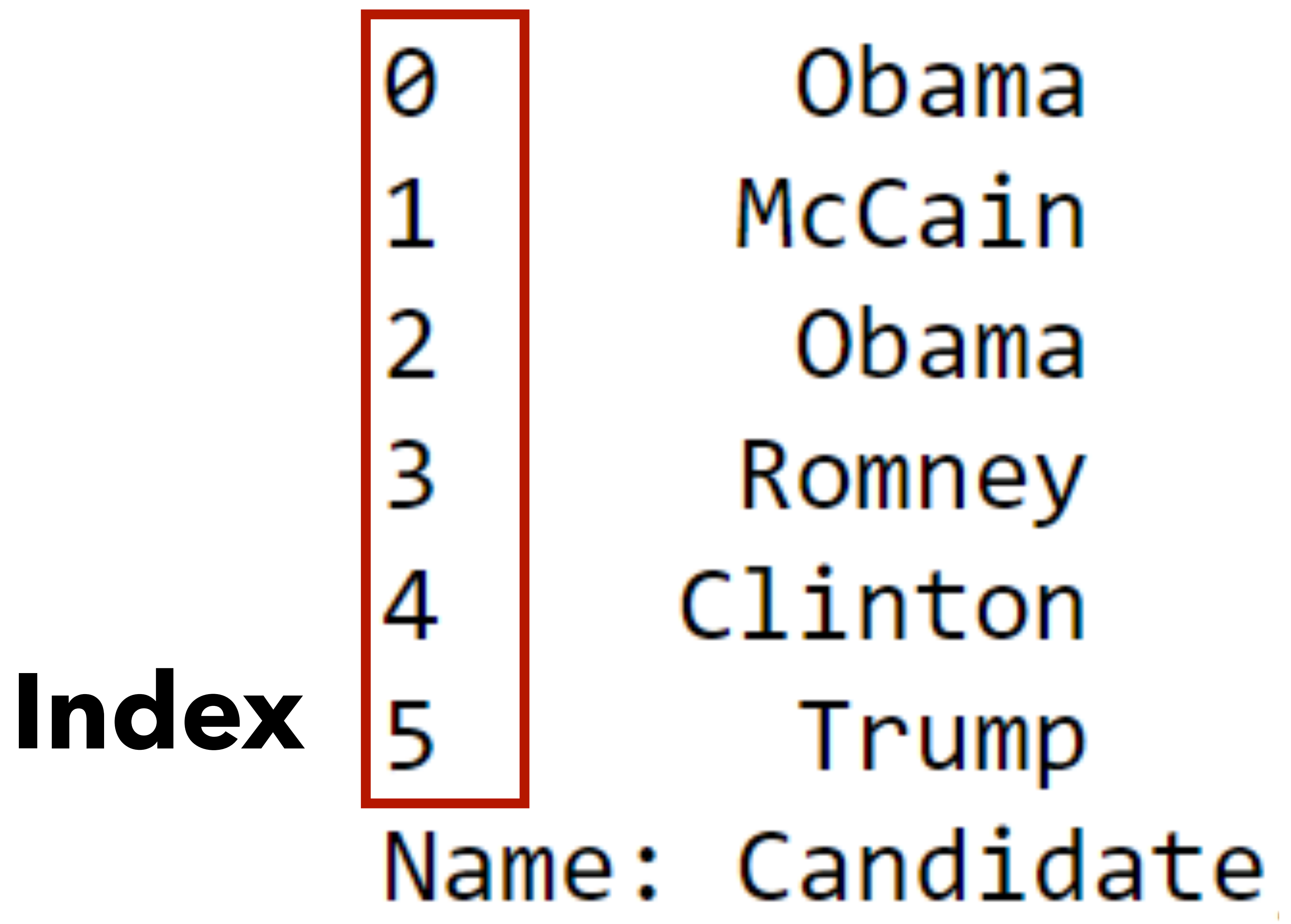


	Candidate	Party	%	Result
Year				
2008	Obama	Democratic	52.9	win
2008	McCain	Republican	45.7	loss
2012	Obama	Democratic	51.1	win
2012	Romney	Republican	47.2	loss
2016	Clinton	Democratic	48.2	loss
2016	Trump	Republican	46.1	win

What's a Series?

Series: one-dimensional sequence of data.

Usually created by taking a single column from a Data Frame.

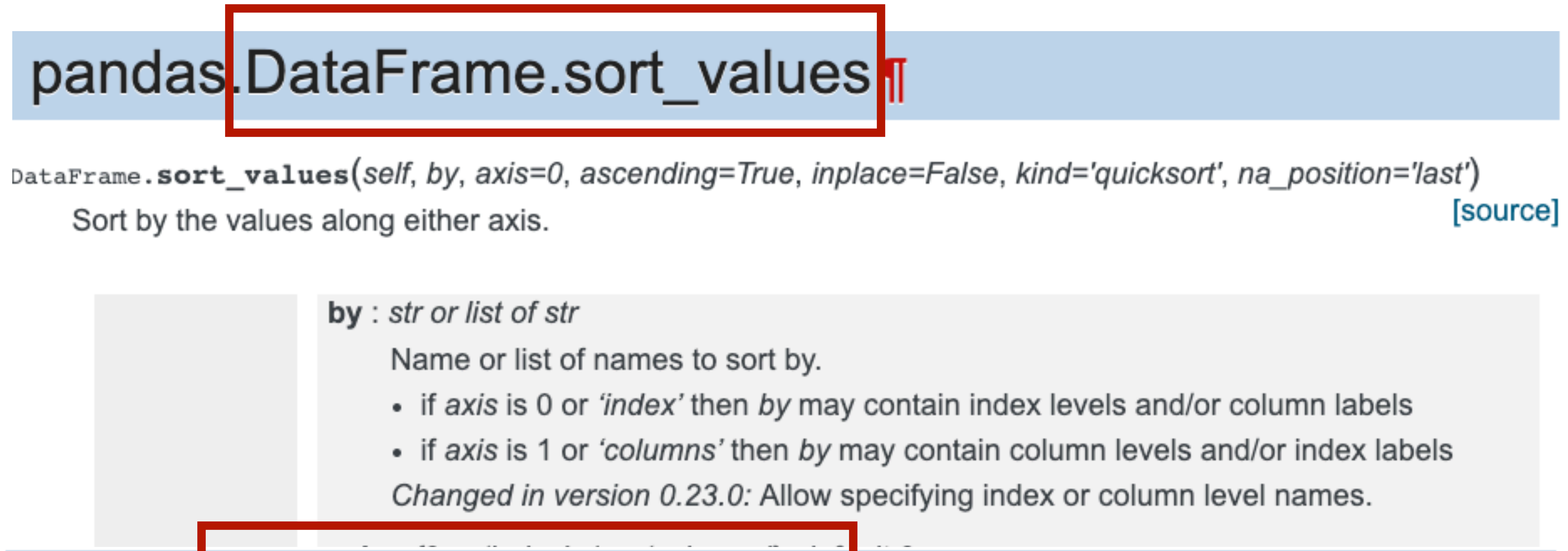


```
0      Obama
1     McCain
2      Obama
3     Romney
4    Clinton
5      Trump
Name: Candidate
```

Why is this important?

Most pandas methods work differently between Data Frames and Series.

The documentation will tell you what type of object the method is for.



pandas.DataFrame.sort_values¶

`DataFrame.sort_values(self, by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')` [source]

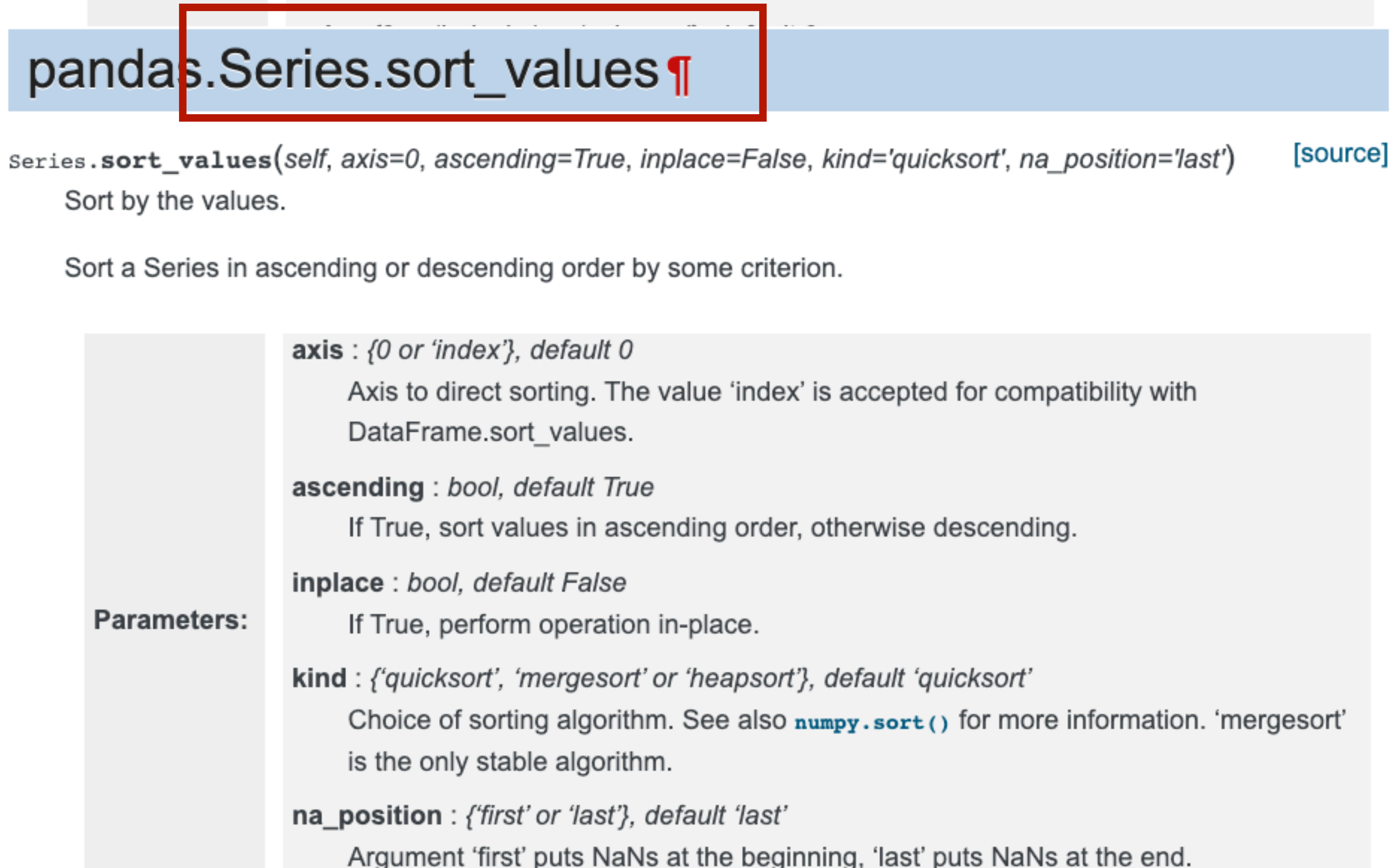
Sort by the values along either axis.

by : *str or list of str*

Name or list of names to sort by.

- if *axis* is 0 or *'index'* then *by* may contain index levels and/or column labels
- if *axis* is 1 or *'columns'* then *by* may contain column levels and/or index labels

Changed in version 0.23.0: Allow specifying index or column level names.



pandas.Series.sort_values¶

`Series.sort_values(self, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')` [source]

Sort by the values.

Sort a Series in ascending or descending order by some criterion.

Parameters:

axis : *{0 or 'index'}, default 0*

Axis to direct sorting. The value 'index' is accepted for compatibility with DataFrame.sort_values.

ascending : *bool, default True*

If True, sort values in ascending order, otherwise descending.

inplace : *bool, default False*

If True, perform operation in-place.

kind : *{'quicksort', 'mergesort' or 'heapsort'}, default 'quicksort'*

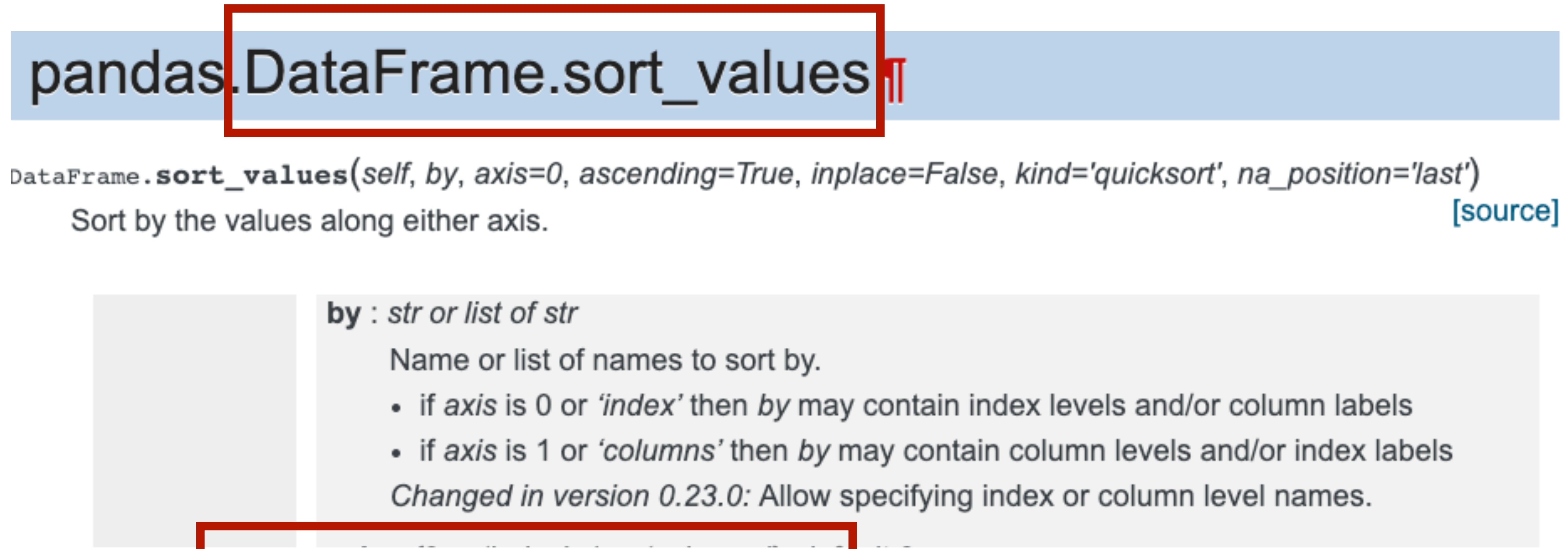
Choice of sorting algorithm. See also `numpy.sort()` for more information. 'mergesort' is the only stable algorithm.

na_position : *{'first' or 'last'}, default 'last'*

Argument 'first' puts NaNs at the beginning, 'last' puts NaNs at the end.

Why is this important?

```
df.sort_values(…)
```



pandas.DataFrame.sort_values¶

`DataFrame.sort_values`(*self, by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last'*) [source]

Sort by the values along either axis.

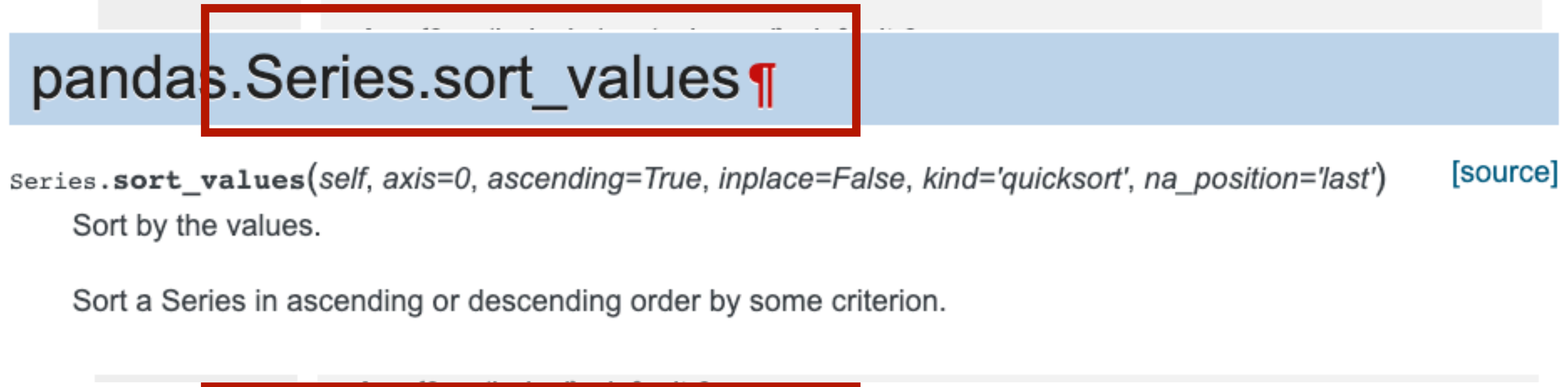
by : *str or list of str*

Name or list of names to sort by.

- if *axis* is 0 or *'index'* then *by* may contain index levels and/or column labels
- if *axis* is 1 or *'columns'* then *by* may contain column levels and/or index labels

Changed in version 0.23.0: Allow specifying index or column level names.

```
df['names'].sort_values(…)
```



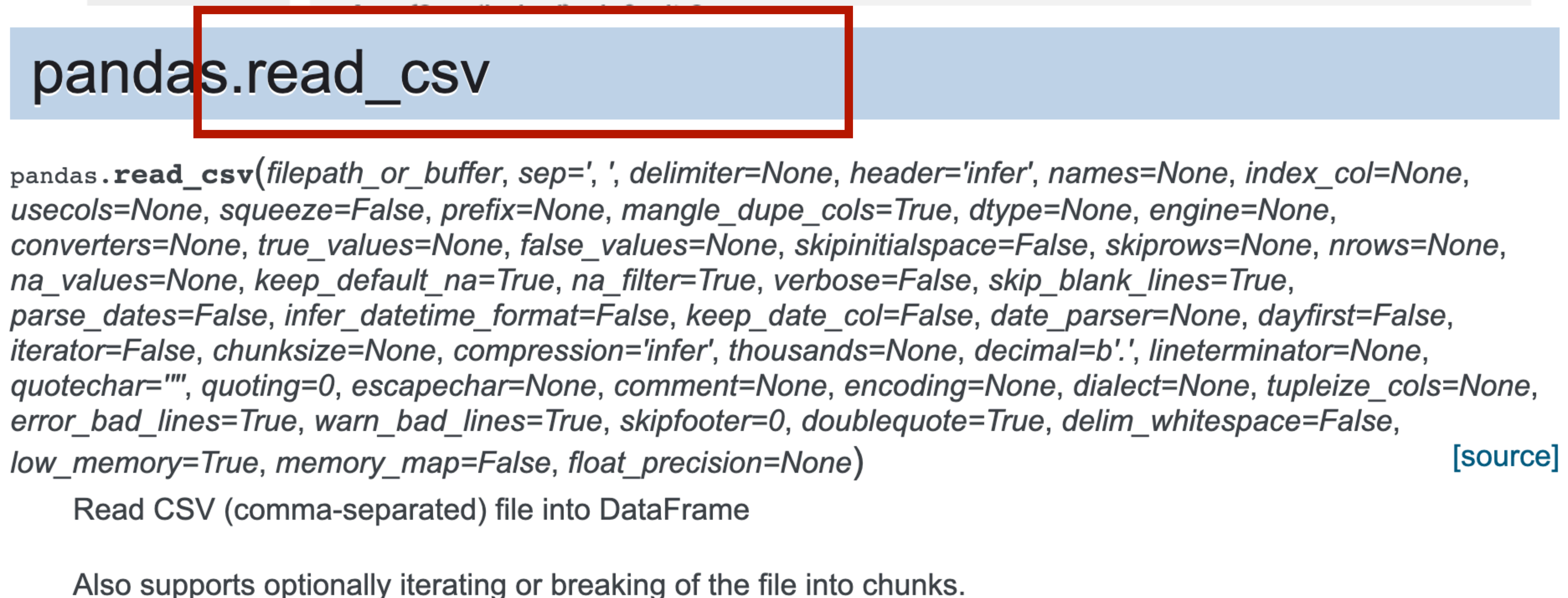
pandas.Series.sort_values¶

`Series.sort_values`(*self, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last'*) [source]

Sort by the values.

Sort a Series in ascending or descending order by some criterion.

```
pd.read_csv(…)
```



pandas.read_csv

`pandas.read_csv`(*filepath_or_buffer, sep=', ', delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=None, error_bad_lines=True, warn_bad_lines=True, skipfooter=0, doublequote=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None*) [source]

Read CSV (comma-separated) file into DataFrame

Also supports optionally iterating or breaking of the file into chunks.

How to use Google properly

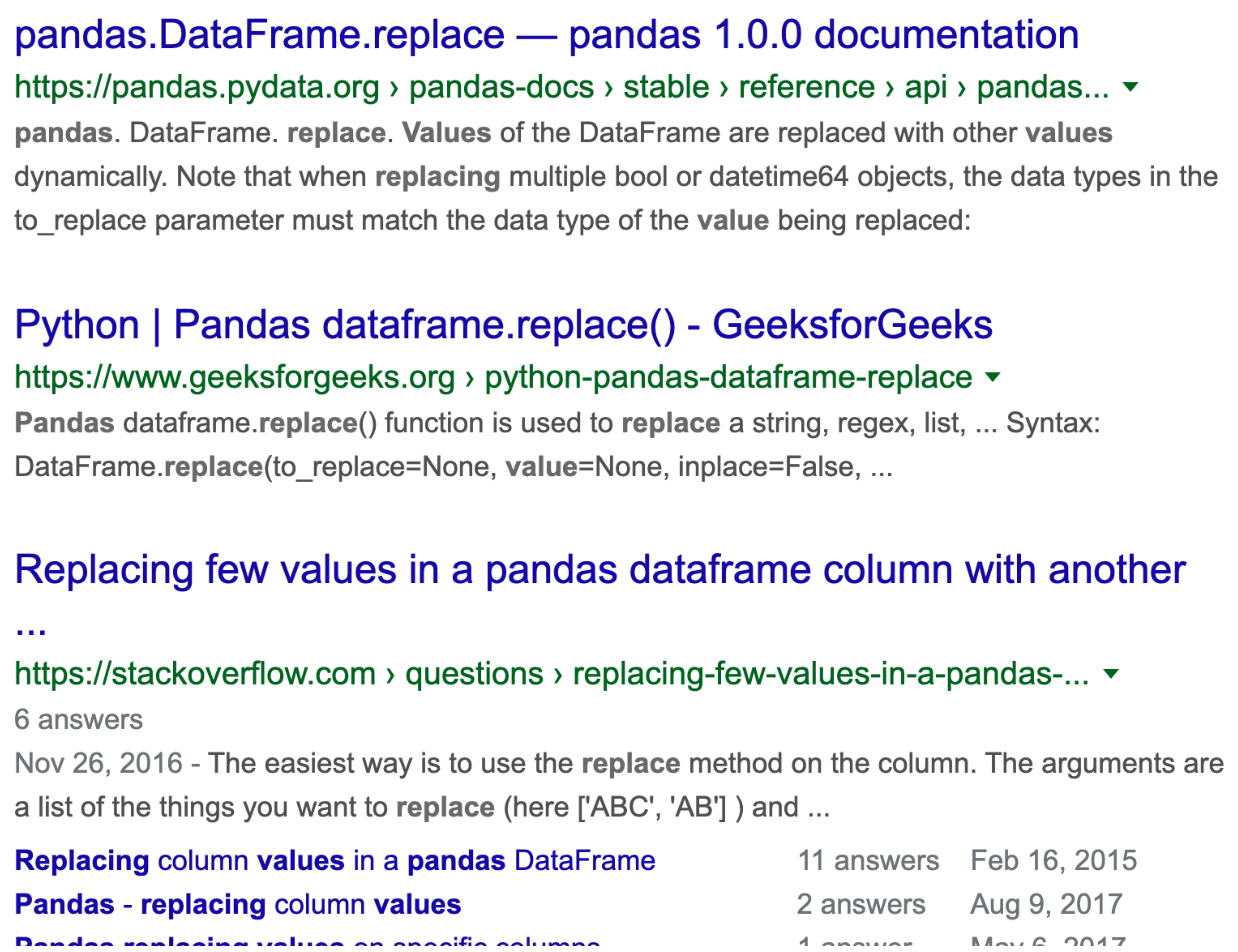
State your task:
"I need to replace 0 with False and 1 with True."

Remove question-specific details:
"replace values"

Add the package name to the front:
"pandas replace values"

If you already know the right method, just google "pandas replace"

Cheatsheets can help you find the right method



pandas.DataFrame.replace — pandas 1.0.0 documentation
https://pandas.pydata.org › pandas-docs › stable › reference › api › pandas... ▾
pandas. DataFrame. **replace**. **Values** of the DataFrame are replaced with other **values** dynamically. Note that when **replacing** multiple bool or datetime64 objects, the data types in the to_replace parameter must match the data type of the **value** being replaced:

Python | Pandas dataframe.replace() - GeeksforGeeks
https://www.geeksforgeeks.org › python-pandas-dataframe-replace ▾
Pandas dataframe.**replace**() function is used to **replace** a string, regex, list, ... Syntax: DataFrame.**replace**(to_replace=None, **value**=None, inplace=False, ...

Replacing few values in a pandas dataframe column with another ...
https://stackoverflow.com › questions › replacing-few-values-in-a-pandas-... ▾
6 answers
Nov 26, 2016 - The easiest way is to use the **replace** method on the column. The arguments are a list of the things you want to **replace** (here ['ABC', 'AB']) and ...

Replacing column **values** in a **pandas** DataFrame	11 answers	Feb 16, 2015
Pandas - **replacing** column **values**	2 answers	Aug 9, 2017
Pandas replacing values on specific columns	1 answer	May 6, 2017

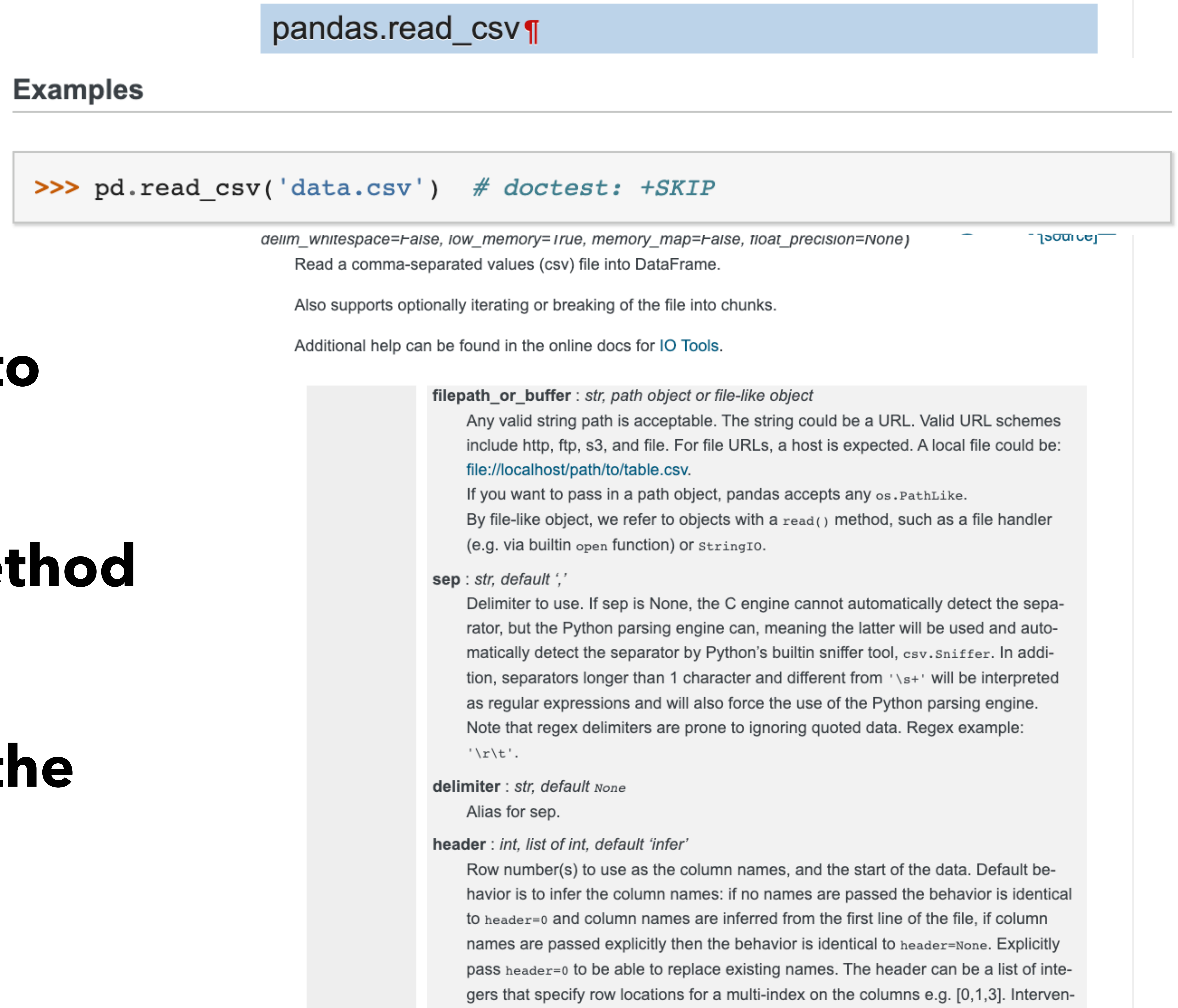
How to read pandas documentation

Skip the table of method parameters and look at the examples.

Copy example, then modify it to work for your notebook.

If needed, refer back to the method parameters for fine-tuning.

(The method in the picture on the right solves Q2.)



pandas.read_csv¶

Examples

```
>>> pd.read_csv('data.csv')  # doctest: +SKIP
```

delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None) [source]

Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file into chunks.

Additional help can be found in the online docs for IO Tools.

filepath_or_buffer : *str, path object or file-like object*

Any valid string path is acceptable. The string could be a URL. Valid URL schemes include http, ftp, s3, and file. For file URLs, a host is expected. A local file could be: file://localhost/path/to/table.csv.

If you want to pass in a path object, pandas accepts any `os.PathLike`.

By file-like object, we refer to objects with a `read()` method, such as a file handler (e.g. via builtin `open` function) or `StringIO`.

sep : *str, default ‘,’*

Delimiter to use. If sep is None, the C engine cannot automatically detect the separator, but the Python parsing engine can, meaning the latter will be used and automatically detect the separator by Python’s builtin sniffer tool, `csv.Sniffer`. In addition, separators longer than 1 character and different from `'\s+'` will be interpreted as regular expressions and will also force the use of the Python parsing engine. Note that regex delimiters are prone to ignoring quoted data. Regex example: `'\r\t'`.

delimiter : *str, default* `None`

Alias for sep.

header : *int, list of int, default ‘infer’*

Row number(s) to use as the column names, and the start of the data. Default behavior is to infer the column names: if no names are passed the behavior is identical to `header=0` and column names are inferred from the first line of the file, if column names are passed explicitly then the behavior is identical to `header=None`. Explicitly pass `header=0` to be able to replace existing names. The header can be a list of integers that specify row locations for a multi-index on the columns e.g. [0,1,3]. Interven-

Finally: don’t use loops

If you find yourself trying to write a for/while loop when working with pandas, you’re almost definitely doing it wrong.

Look for the right pandas method. And ask your friend + staff for help.

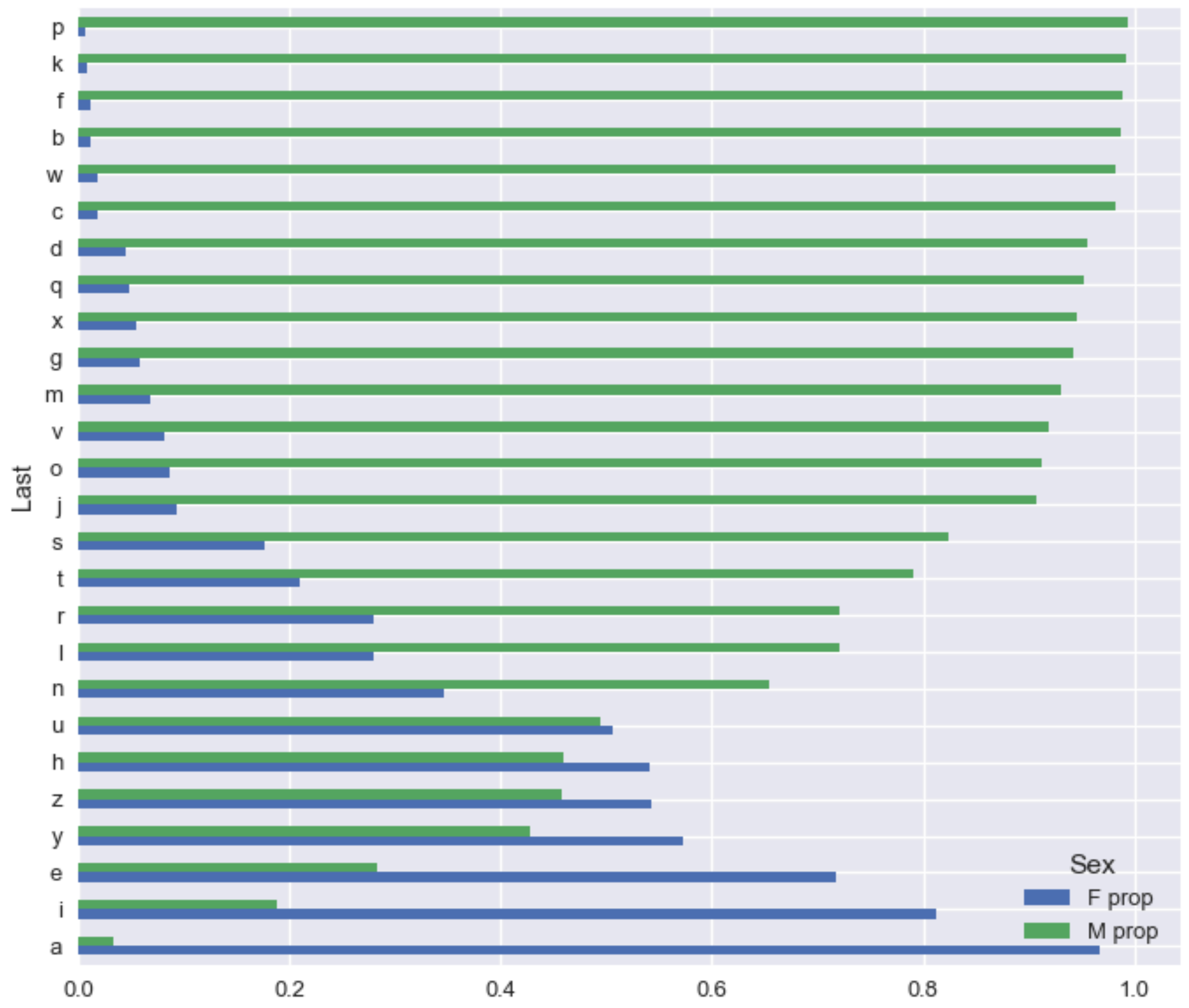
Preview of next week

Why do cells work the first time but not the second?

Why are there so many brackets everywhere?

String methods: how do I work with text?

Demo: using last letter of a person's first name to predict birth sex



Extra resources:

- Ch3 of textbook.ds100.org
- 10 minutes to pandas: pandas.pydata.org/pandas-docs/stable/getting_started/10min.html
- Lecture slides on pandas: bit.ly/sam-pandas-01

A2 tips

- Q3b: The average of a column of 0/1s is the proportion of 1s.
- Q5b: Use a list of dtypes instead of a single string to select multiple dtypes
- Q5d: Adding two Series together sums each element in the two Series. Use df.assign to create a new column.
- Q6a: I'll walk through this one
- Q8b: Use a list of strings in .agg to call multiple aggregation methods