pandas.DataFrame.groupby

`DataFrame.groupby(by=None, axis=<no_default>, level=None, as_index=True, sort=True, group_keys=True, observed=<no_default>, dropna=True)` [source]

Group DataFrame using a mapper or by a Series of columns.

A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

Parameters:

by : *mapping, function, label, pd.Grouper or list of such*

Used to determine the groups for the groupby. If `by` is a function, it's called on each value of the object's index. If a dict or Series is passed, the Series or dict VALUES will be used to determine the groups (the Series' values are first aligned; see `.align()` method). If a list or ndarray of length equal to the selected axis is passed (see the groupby user guide), the values are used as-is to determine the groups. A label or list of labels may be passed to group by the columns in `self`. Notice that a tuple is interpreted as a (single) key.

axis : *{0 or 'index', 1 or 'columns'}, default 0*

Split along rows (0) or columns (1). For *Series* this parameter is unused and defaults to 0.

> **Deprecated since version 2.1.0:** Will be removed and behave like axis=0 in a future version. For `axis=1`, do `frame.T.groupby(...)` instead.

level : *int, level name, or sequence of such, default None*

If the axis is a MultiIndex (hierarchical), group by a particular level or levels. Do not specify both `by` and `level`.

as_index : *bool, default True*

Return object with group labels as the index. Only relevant for DataFrame input. as_index=False is effectively “SQL-style” grouped output. This argument has no effect on filtrations (see the filtrations in the user guide), such as `head()`, `tail()`, `nth()` and in transformations (see the transformations in the user guide).

sort : *bool, default True*

Sort group keys. Get better performance by turning this off. Note this does not influence the order of observations within each group. Groupby preserves the order of rows within each group. If False, the groups will appear in the same order as they did in the original DataFrame. This argument has no effect on filtrations (see the filtrations in the user guide), such as `head()`, `tail()`, `nth()` and in transformations (see the transformations in the user guide).

> ***Changed in version 2.0.0:*** Specifying `sort=False` with an ordered categorical grouper will no longer sort the values.

group_keys : *bool, default True*

When calling apply and the `by` argument produces a like-indexed (i.e. a transform) result, add group keys to index to identify pieces. By

> ***Changed in version 1.5.0:*** Warns that `group_keys` will no longer be ignored when the result from `apply` is a like-indexed Series or DataFrame. Specify `group_keys` explicitly to include the group keys or not.

> **Changed in version 2.0.0:** `group_keys` now defaults to `True`.

observed : *bool, default False*

This only applies if any of the groupers are Categoricals. If True: only show observed values for categorical groupers. If False: show all values for categorical groupers.

> **Deprecated since version 2.1.0:** The default value will change to True in a future version of pandas.

dropna : *bool, default True*

If True, and if group keys contain NA values, NA values together with row/column will be dropped. If False, NA values will also be treated as the key in groups.

Returns:

pandas.api.typing.DataFrameGroupBy

Returns a groupby object that contains information about the groups.

See also

`resample`

Convenience method for frequency conversion and resampling of time series.

Notes

See the user guide for more detailed usage and examples, including splitting an object into groups, iterating through groups, selecting a group, aggregation, and more.

Examples

```
>>> df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
...                               'Parrot', 'Parrot'],
...                    'Max Speed': [380., 370., 24., 26.]})
>>> df
   Animal  Max Speed
0  Falcon      380.0
1  Falcon      370.0
2  Parrot       24.0
3  Parrot       26.0
>>> df.groupby(['Animal']).mean()
        Max Speed
Animal
Falcon      375.0
Parrot       25.0
```

Hierarchical Indexes

We can groupby different levels of a hierarchical index using the *level* parameter:

```
>>> arrays = [['Falcon', 'Falcon', 'Parrot', 'Parrot'],
...           ['Captive', 'Wild', 'Captive', 'Wild']]
>>> index = pd.MultiIndex.from_arrays(arrays, names=('Animal', 'Type'))
>>> df = pd.DataFrame({'Max Speed': [390., 350., 30., 20.]},
...                   index=index)
>>> df
                Max Speed
Animal Type
Falcon Captive      390.0
       Wild         350.0
Parrot Captive       30.0
       Wild          20.0
>>> df.groupby(level=0).mean()
        Max Speed
Animal
Falcon      370.0
Parrot       25.0
>>> df.groupby(level="Type").mean()
         Max Speed
Type
Captive      210.0
Wild         185.0
```

We can also choose to include NA in group keys or not by setting *dropna* parameter, the default setting is *True*.

```
>>> l = [[1, 2, 3], [1, None, 4], [2, 1, 3], [1, 2, 2]]
>>> df = pd.DataFrame(l, columns=["a", "b", "c"])
```

```
>>> df.groupby(by=["b"]).sum()
    a   c
b
1.0 2   3
2.0 2   5
```

```
>>> df.groupby(by=["b"], dropna=False).sum()
    a   c
b
1.0 2   3
2.0 2   5
NaN 1   4
```

```
>>> l = [["a", 12, 12], [None, 12.3, 33.], ["b", 12.3, 123], ["a", 1, 1]]
>>> df = pd.DataFrame(l, columns=["a", "b", "c"])
```

```
>>> df.groupby(by="a").sum()
    b     c
a
a   13.0   13.0
b   12.3  123.0
```

```
>>> df.groupby(by="a", dropna=False).sum()
    b     c
a
a   13.0   13.0
b   12.3  123.0
NaN 12.3   33.0
```

When using `.apply()`, use `group_keys` to include or exclude the group keys. The `group_keys` argument defaults to `True` (include).

```
>>> df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
...                               'Parrot', 'Parrot'],
...                    'Max Speed': [380., 370., 24., 26.]})
>>> df.groupby("Animal", group_keys=True)[['Max Speed']].apply(lambda x:
```

```
           Max Speed
Animal
Falcon 0      380.0
       1      370.0
Parrot 2       24.0
       3       26.0
```

```
>>> df.groupby("Animal", group_keys=False)[['Max Speed']].apply(lambda x
   Max Speed
0      380.0
1      370.0
2       24.0
3       26.0
```

Previous

Next

© 2025, pandas via NumFOCUS, Inc. Hosted by OVHcloud.

Built with the PyData Sphinx Theme 0.14.4.

Created using Sphinx 8.1.3.