pandas.DataFrame.join

`DataFrame.`**`join`**(*self*, *other*, *on=None*, *how='left'*, *lsuffix=''*, *rsuffix=''*, *sort=False*) [source]

Join columns of another DataFrame.

Join columns with *other* DataFrame either on index or on a key column. Efficiently join multiple DataFrame objects by index at once by passing a list.

Parameters:

other : *DataFrame, Series, or list of DataFrame*
Index should be similar to one of the columns in this one. If a Series is passed, its name attribute must be set, and that will be used as the column name in the resulting joined DataFrame.

on : *str, list of str, or array-like, optional*
Column or index level name(s) in the caller to join on the index in *other*, otherwise joins index-on-index. If multiple values given, the *other* DataFrame must have a MultiIndex. Can pass an array as the join key if it is not already contained in the calling DataFrame. Like an Excel VLOOKUP operation.

how : *{'left', 'right', 'outer', 'inner'}, default 'left'*
How to handle the operation of the two objects.

- left: use calling frame's index (or column if on is specified)
- right: use *other*'s index.
- outer: form union of calling frame's index (or column if on is specified) with *other*'s index, and sort it. lexicographically.
- inner: form intersection of calling frame's index (or column if on is specified) with *other*'s index, preserving the order of the calling's one.

lsuffix : *str, default ''*
Suffix to use from left frame's overlapping columns.

rsuffix : *str, default ''*
Suffix to use from right frame's overlapping columns.

sort : *bool, default False*
Order result DataFrame lexicographically by the join key. If False, the order of the join key depends on the join type (how keyword).

Returns:

DataFrame
A dataframe containing columns from both the caller and *other*.

See also:

`DataFrame.merge`
: For column(s)-on-columns(s) operations.

Notes

Parameters *on*, *lsuffix*, and *rsuffix* are not supported when passing a list of *DataFrame* objects.

Support for specifying index levels as the *on* parameter was added in version 0.23.0.

Examples

```
>>> df = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3', 'K4', 'K5']
...                    'A': ['A0', 'A1', 'A2', 'A3', 'A4', 'A5']})
```

```
>>> df
  key   A
0  K0  A0
1  K1  A1
2  K2  A2
3  K3  A3
4  K4  A4
5  K5  A5
```

```
>>> other = pd.DataFrame({'key': ['K0', 'K1', 'K2'],
...                       'B': ['B0', 'B1', 'B2']})
```

```
>>> other
  key   B
0  K0  B0
1  K1  B1
2  K2  B2
```

Join DataFrames using their indexes.

```
>>> df.join(other, lsuffix='_caller', rsuffix='_other')
  key_caller   A key_other    B
0         K0  A0        K0   B0
1         K1  A1        K1   B1
2         K2  A2        K2   B2
3         K3  A3       NaN  NaN
4         K4  A4       NaN  NaN
5         K5  A5       NaN  NaN
```

If we want to join using the key columns, we need to set key to be the index in both *df* and *other*. The joined DataFrame will have key as its index.

```
>>> df.set_index('key').join(other.set_index('key'))
      A    B
key
K0   A0   B0
K1   A1   B1
K2   A2   B2
K3   A3  NaN
K4   A4  NaN
K5   A5  NaN
```

Another option to join using the key columns is to use the *on* parameter. DataFrame.join always uses *other*'s index but we can use any column in *df*. This method preserves the original DataFrame's index in the result.

```
>>> df.join(other.set_index('key'), on='key')
  key   A    B
0  K0  A0   B0
1  K1  A1   B1
2  K2  A2   B2
3  K3  A3  NaN
4  K4  A4  NaN
5  K5  A5  NaN
```