

Pandas – concat()

Concatenate pandas objects along a particular axis with optional set logic along the other axes

pandas.concat(objs, axis=0, join='outer', ignore_index=False, keys=None, levels=None, names=None, verify_integrity=False, sort=False, copy=True)

- **objs** : a sequence or mapping of Series or DataFrame objects. If a dict is passed, the sorted keys will be used as the keys argument, unless it is passed, in which case the values will be selected (see below). Any None objects will be dropped silently unless they are all None in which case a ValueError will be raised.
- **axis** : {0, 1, ...}, default 0. The axis to concatenate along.
- **join** : {'inner', 'outer'}, default 'outer'. How to handle indexes on other axis(es). Outer for union and inner for intersection.
- **ignore_index** : boolean, default False. If True, do not use the index values on the concatenation axis. The resulting axis will be labeled 0, ..., n - 1. This is useful if you are concatenating objects where the concatenation axis does not have meaningful indexing information. Note the index values on the other axes are still respected in the join.
- **keys** : sequence, default None. Construct hierarchical index using the passed keys as the outermost level. If multiple levels passed, should contain tuples.
- **levels** : list of sequences, default None. Specific levels (unique values) to use for constructing a MultiIndex. Otherwise they will be inferred from the keys.
- **names** : list, default None. Names for the levels in the resulting hierarchical index.
- **verify_integrity** : boolean, default False. Check whether the new concatenated axis contains duplicates. This can be very expensive relative to the actual data concatenation.
- **copy** : boolean, default True. If False, do not copy data unnecessarily.

Pandas – concat()

```
df1 = pd.DataFrame(
    {
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    },
    index=[0, 1, 2, 3],
)

df2 = pd.DataFrame(
    {
        "A": ["A4", "A5", "A6", "A7"],
        "B": ["B4", "B5", "B6", "B7"],
        "C": ["C4", "C5", "C6", "C7"],
        "D": ["D4", "D5", "D6", "D7"],
    },
    index=[4, 5, 6, 7],
)

df3 = pd.DataFrame(
    {
        "A": ["A8", "A9", "A10", "A11"],
        "B": ["B8", "B9", "B10", "B11"],
        "C": ["C8", "C9", "C10", "C11"],
        "D": ["D8", "D9", "D10", "D11"],
    },
    index=[8, 9, 10, 11],
)

frames = [df1, df2, df3]

result = pd.concat(frames)
```

df1					Result				
	A	B	C	D		A	B	C	D
0	A0	B0	C0	D0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	A3	B3	C3	D3
df2					4	A4	B4	C4	D4
	A	B	C	D	5	A5	B5	C5	D5
4	A4	B4	C4	D4	6	A6	B6	C6	D6
5	A5	B5	C5	D5	7	A7	B7	C7	D7
6	A6	B6	C6	D6	8	A8	B8	C8	D8
7	A7	B7	C7	D7	9	A9	B9	C9	D9
df3					10	A10	B10	C10	D10
	A	B	C	D	11	A11	B11	C11	D11
8	A8	B8	C8	D8					
9	A9	B9	C9	D9					
10	A10	B10	C10	D10					
11	A11	B11	C11	D11					

Pandas – concat()

```
df4 = pd.DataFrame(
    {
        "B": ["B2", "B3", "B6", "B7"],
        "D": ["D2", "D3", "D6", "D7"],
        "F": ["F2", "F3", "F6", "F7"],
    },
    index=[2, 3, 6, 7],
)

result = pd.concat([df1, df4], axis=1)
```

join = 'outer'

df1					df4				Result							
	A	B	C	D		B	D	F		A	B	C	D	B	D	F
0	A0	B0	C0	D0	2	B2	D2	F2	0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	3	B3	D3	F3	1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	6	B6	D6	F6	2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	7	B7	D7	F7	3	A3	B3	C3	D3	B3	D3	F3
									6	NaN	NaN	NaN	NaN	B6	D6	F6
									7	NaN	NaN	NaN	NaN	B7	D7	F7

join = 'inner'

df1					df4				Result							
	A	B	C	D		B	D	F		A	B	C	D	B	D	F
0	A0	B0	C0	D0	2	B2	D2	F2	2	A2	B2	C2	D2	B2	D2	F2
1	A1	B1	C1	D1	3	B3	D3	F3	3	A3	B3	C3	D3	B3	D3	F3
2	A2	B2	C2	D2	6	B6	D6	F6								
3	A3	B3	C3	D3	7	B7	D7	F7								

```
result = pd.concat([df1, df4], axis=1, join="inner")
```

Pandas – concat() - examples

```
In [1]: import pandas as pd
```

```
In [2]: df1 = pd.read_csv('gym_1.csv', sep=';', index_col='id', parse_dates=['date'])
df1
```

Out[2]:

	name	date	height	weight	age
id					
1373913	Marisa Martins	2013-02-05	155	48	45
1109818	Rita Fonseca	2018-08-28	166	54	45

```
In [3]: df2 = pd.read_csv('gym_2.csv', sep=';', index_col='id', parse_dates=['date'])
df2
```

Out[3]:

	name	date	height	weight	age
id					
1767703	Manuel Martins	2003-01-25	179	85	24
1071208	Florbela Freitas	2008-09-26	166	53	28

```
In [4]: df3 = pd.read_csv('gym_3.csv', sep=';', index_col='id')
df3
```

Out[4]:

	name	hours	status	children	sex
id					
1767703	Manuel Martins	6	single	0	M
1373913	Marisa Martins	3	married	2	F
1158813	Joana Freitas	3	widow	1	F

Pandas – concat() - example

Concat along the axis=0, add rows

```
In [5]: df_concat=pd.concat([df1,df2,df3])
df_concat
```

Out[5]:

	name	date	height	weight	age	hours	status	children	sex
id									
1373913	Marisa Martins	2013-02-05	155.0	48.0	45.0	NaN	NaN	NaN	NaN
1109818	Rita Fonseca	2018-08-28	166.0	54.0	45.0	NaN	NaN	NaN	NaN
1767703	Manuel Martins	2003-01-25	179.0	85.0	24.0	NaN	NaN	NaN	NaN
1071208	Florbela Freitas	2008-09-26	166.0	53.0	28.0	NaN	NaN	NaN	NaN
1767703	Manuel Martins	NaT	NaN	NaN	NaN	6.0	single	0.0	M
1373913	Marisa Martins	NaT	NaN	NaN	NaN	3.0	married	2.0	F
1158813	Joana Freitas	NaT	NaN	NaN	NaN	3.0	widow	1.0	F

Pandas – concat() - example

Concat along the axis=1, add columns

```
In [6]: df_concat_cols_outter = pd.concat([df1,df3], axis=1)
df_concat_cols_outter
```

Out[6]:

	name	date	height	weight	age	name	hours	status	children	sex
id										
1109818	Rita Fonseca	2018-08-28	166.0	54.0	45.0	NaN	NaN	NaN	NaN	NaN
1158813	NaN	NaT	NaN	NaN	NaN	Joana Freitas	3.0	widow	1.0	F
1373913	Marisa Martins	2013-02-05	155.0	48.0	45.0	Marisa Martins	3.0	married	2.0	F
1767703	NaN	NaT	NaN	NaN	NaN	Manuel Martins	6.0	single	0.0	M

```
In [7]: df_concat_cols_inner = pd.concat([df1,df3], axis=1, join='inner')
df_concat_cols_inner
```

Out[7]:

	name	date	height	weight	age	name	hours	status	children	sex
id										
1373913	Marisa Martins	2013-02-05	155	48	45	Marisa Martins	3	married	2	F

Pandas – append()

Append rows of **other** to the end of **caller**, returning a new object

DataFrame.append(other, ignore_index = False, verify_integrity = False, sort = False)

- **other** : DataFrame or Series/dict-like object, or list of these
The data to append.
- **ignore_index** : bool, default False
If True, the resulting axis will be labeled 0, 1, ..., n - 1.
- **verify_integrity** : bool, default False
If True, raise ValueError on creating index with duplicates.
- **sort** : bool, default False
Sort columns if the columns of self and other are not aligned.

Pandas – append()

```
result = df1.append(df2)
```

df1					Result				
	A	B	C	D		A	B	C	D
0	A0	B0	C0	D0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	A3	B3	C3	D3
df2					4	A4	B4	C4	D4
	A	B	C	D	5	A5	B5	C5	D5
4	A4	B4	C4	D4	6	A6	B6	C6	D6
5	A5	B5	C5	D5	7	A7	B7	C7	D7
6	A6	B6	C6	D6					
7	A7	B7	C7	D7					

Different columns

```
result = df1.append(df4, sort=False)
```

df1					Result					
	A	B	C	D		A	B	C	D	F
0	A0	B0	C0	D0	0	A0	B0	C0	D0	NaN
1	A1	B1	C1	D1	1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	3	A3	B3	C3	D3	NaN
df4					2	NaN	B2	NaN	D2	F2
	B	D	F		3	NaN	B3	NaN	D3	F3
2	B2	D2	F2		6	NaN	B6	NaN	D6	F6
3	B3	D3	F3		7	NaN	B7	NaN	D7	F7
6	B6	D6	F6							
7	B7	D7	F7							

Pandas – append() - example

Append, add rows

```
In [8]: df_append = df1.append([df2, df3])
df_append
```

Out[8]:

	name	date	height	weight	age	hours	status	children	sex
id									
1373913	Marisa Martins	2013-02-05	155.0	48.0	45.0	NaN	NaN	NaN	NaN
1109818	Rita Fonseca	2018-08-28	166.0	54.0	45.0	NaN	NaN	NaN	NaN
1767703	Manuel Martins	2003-01-25	179.0	85.0	24.0	NaN	NaN	NaN	NaN
1071208	Florbela Freitas	2008-09-26	166.0	53.0	28.0	NaN	NaN	NaN	NaN
1767703	Manuel Martins	NaT	NaN	NaN	NaN	6.0	single	0.0	M
1373913	Marisa Martins	NaT	NaN	NaN	NaN	3.0	married	2.0	F
1158813	Joana Freitas	NaT	NaN	NaN	NaN	3.0	widow	1.0	F