pandas.DataFrame.sort_values

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

Sort by the values along either axis

Parameters:

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.

axis : *{0 or 'index', 1 or 'columns'}, default 0*
Axis to be sorted

ascending : *bool or list of bool, default True*
Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.

inplace : *bool, default False*
if True, perform operation in-place

kind : *{'quicksort', 'mergesort', 'heapsort'}, default 'quicksort'*
Choice of sorting algorithm. See also ndarray.np.sort for more information. *mergesort* is the only stable algorithm. For DataFrames, this option is only applied when sorting on a single column or label.

na_position : *{'first', 'last'}, default 'last'*
first puts NaNs at the beginning, *last* puts NaNs at the end

Returns:

sorted_obj : *DataFrame*

Examples

```
>>> df = pd.DataFrame({
...     'col1' : ['A', 'A', 'B', np.nan, 'D', 'C'],
...     'col2' : [2, 1, 9, 8, 7, 4],
...     'col3': [0, 1, 9, 4, 2, 3],
... })
>>> df
    col1 col2 col3
0   A    2    0
1   A    1    1
2   B    9    9
3   NaN  8    4
4   D    7    2
5   C    4    3
```

Scroll To Top

Sort by col1

```
>>> df.sort_values(by=['col1'])
    col1 col2 col3
0   A    2    0
1   A    1    1
2   B    9    9
5   C    4    3
4   D    7    2
3   NaN  8    4
```

Sort by multiple columns

```
>>> df.sort_values(by=['col1', 'col2'])
    col1 col2 col3
1   A    1    1
0   A    2    0
2   B    9    9
5   C    4    3
4   D    7    2
3   NaN  8    4
```

Sort Descending

```
>>> df.sort_values(by='col1', ascending=False)
    col1 col2 col3
4   D    7    2
5   C    4    3
2   B    9    9
0   A    2    0
1   A    1    1
3   NaN  8    4
```

Putting NAs first

```
>>> df.sort_values(by='col1', ascending=False, na_position='first')
    col1 col2 col3
3   NaN  8    4
4   D    7    2
5   C    4    3
2   B    9    9
0   A    2    0
1   A    1    1
```

Scroll To Top