pandas.DataFrame.fillna

`DataFrame.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=<no_default>)` [source]

Fill NA/NaN values using the specified method.

Parameters:

value : *scalar, dict, Series, or DataFrame*

Value to use to fill holes (e.g. 0), alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). Values not in the dict/Series/DataFrame will not be filled. This value cannot be a list.

method : *{'backfill', 'bfill', 'ffill', None}, default None*

Method to use for filling holes in reindexed Series:

- ffill: propagate last valid observation forward to next valid.
- backfill / bfill: use next valid observation to fill gap.

> ***Deprecated since version 2.1.0:*** Use ffill or bfill instead.

axis : *{0 or 'index'} for Series, {0 or 'index', 1 or 'columns'} for DataFrame*

Axis along which to fill missing values. For *Series* this parameter is unused and defaults to 0.

inplace : *bool, default False*

If True, fill in-place. Note: this will modify any other views on this object (e.g., a no-copy slice for a column in a DataFrame).

limit : *int, default None*

If method is specified, this is the maximum number of consecutive NaN values to forward/backward fill. In other words, if there is a gap with more than this number of

maximum number of entries along the entire axis where NaNs will be filled. Must be greater than 0 if not None.

downcast : *dict, default is None*

A dict of item->dtype of what to downcast if possible, or the string 'infer' which will try to downcast to an appropriate equal type (e.g. float64 to int64 if possible).

> ***Deprecated since version 2.2.0.***

Returns:

Series/DataFrame or None

Object with missing values filled or None if `inplace=True`.

See also

`ffill`
Fill values by propagating the last valid observation to next valid.

`bfill`
Fill values by using the next valid observation to fill the gap.

`interpolate`
Fill NaN values using interpolation.

`reindex`
Conform object to new index.

`asfreq`
Convert TimeSeries to specified frequency.

Examples

```
>>> df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, np.nan, 1],
...                    [np.nan, np.nan, np.nan, np.nan],
...                    [np.nan, 3, np.nan, 4]],
...                   columns=list("ABCD"))
>>> df
     A    B   C    D
0  NaN  2.0 NaN  0.0
1  3.0  4.0 NaN  1.0
2  NaN  NaN NaN  NaN
```

Replace all NaN elements with 0s.

```
>>> df.fillna(0)
     A    B    C    D
0  0.0  2.0  0.0  0.0
1  3.0  4.0  0.0  1.0
2  0.0  0.0  0.0  0.0
3  0.0  3.0  0.0  4.0
```

Replace all NaN elements in column ‘A’, ‘B’, ‘C’, and ‘D’, with 0, 1, 2, and 3 respectively.

```
>>> values = {"A": 0, "B": 1, "C": 2, "D": 3}
>>> df.fillna(value=values)
     A    B    C    D
0  0.0  2.0  2.0  0.0
1  3.0  4.0  2.0  1.0
2  0.0  1.0  2.0  3.0
3  0.0  3.0  2.0  4.0
```

Only replace the first NaN element.

```
>>> df.fillna(value=values, limit=1)
     A    B    C    D
0  0.0  2.0  2.0  0.0
1  3.0  4.0  NaN  1.0
2  NaN  1.0  NaN  3.0
3  NaN  3.0  NaN  4.0
```

When filling using a DataFrame, replacement happens along the same column names and same indices

```
>>> df2 = pd.DataFrame(np.zeros((4, 4)), columns=list("ABCE"))
>>> df.fillna(df2)
     A    B    C    D
0  0.0  2.0  0.0  0.0
1  3.0  4.0  0.0  1.0
2  0.0  0.0  0.0  NaN
3  0.0  3.0  0.0  4.0
```

Note that column D is not affected since it is not present in df2.

pandas.DataFrame.ffill pandas.DataFrame.interpolate