pandas.DataFrame.fillna

`DataFrame.`**`fillna`**(*self*, *value=None*, *method=None*, *axis=None*, *inplace=False*, *limit=None*, *downcast=None*, ***kwargs*) [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’, ‘pad’, ‘ffill’, None}, default None*

Method to use for filling holes in reindexed Series pad / ffill: propagate last valid observation forward to next valid backfill / bfill: use next valid observation to fill gap.

axis : *{0 or ‘index’, 1 or ‘columns’}*

Axis along which to fill missing values.

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 consecutive NaNs, it will only be partially filled. If method is not specified, this is the 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).

Returns: DataFrame

Object with missing values filled.

See also:

`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, 5],
...                    [np.nan, 3, np.nan, 4]],
...                   columns=list('ABCD'))
>>> df
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  NaN NaN  5
3  NaN  3.0 NaN  4
```

Replace all NaN elements with 0s.

```
>>> df.fillna(0)
    A   B   C   D
0   0.0 2.0 0.0 0
1   3.0 4.0 0.0 1
2   0.0 0.0 0.0 5
3   0.0 3.0 0.0 4
```

We can also propagate non-null values forward or backward.

```
>>> df.fillna(method='ffill')
    A   B   C   D
0   NaN 2.0 NaN 0
1   3.0 4.0 NaN 1
2   3.0 4.0 NaN 5
3   3.0 3.0 NaN 4
```

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
1   3.0 4.0 2.0 1
2   0.0 1.0 2.0 5
3   0.0 3.0 2.0 4
```

Only replace the first NaN element.

```
>>> df.fillna(value=values, limit=1)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 NaN 1
2   NaN 1.0 NaN 5
3   NaN 3.0 NaN 4
```