pandas的series 的很多方法和numpy中的ndarray的方法是一样的但是where方法就不一样：

以下是pandas的series 的where 方法实例

pandas.Series.where

**Series.where(*cond*, *other=NoDefault.no\_default*, *inplace=False*, *axis=None*, *level=None*, *errors=NoDefault.no\_default*, *try\_cast=NoDefault.no\_default*)**[**[source]**](https://github.com/pandas-dev/pandas/blob/v1.4.4/pandas/core/series.py#L5556-L5569)

Replace values where the condition is False.

**Parameters**

**cond*bool Series/DataFrame, array-like, or callable***

Where *cond* is True, keep the original value. Where False, replace with corresponding value from *other*. If *cond* is callable, it is computed on the Series/DataFrame and should return boolean Series/DataFrame or array. The callable must not change input Series/DataFrame (though pandas doesn’t check it).

**other*scalar, Series/DataFrame, or callable***

Entries where *cond* is False are replaced with corresponding value from *other*. If other is callable, it is computed on the Series/DataFrame and should return scalar or Series/DataFrame. The callable must not change input Series/DataFrame (though pandas doesn’t check it).

**inplace*bool, default False***

Whether to perform the operation in place on the data.

**axis*int, default None***

Alignment axis if needed.

**level*int, default None***

Alignment level if needed.

**errors*str, {‘raise’, ‘ignore’}, default ‘raise’***

Note that currently this parameter won’t affect the results and will always coerce to a suitable dtype.

* ‘raise’ : allow exceptions to be raised.
* ‘ignore’ : suppress exceptions. On error return original object.

**try\_cast*bool, default None***

Try to cast the result back to the input type (if possible).

***Deprecated since version 1.3.0:***Manually cast back if necessary.

**Returns**

**Same type as caller or None if inplace=True.**

**See also**

[**DataFrame.mask()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.mask.html#pandas.DataFrame.mask)

Return an object of same shape as self.

**Notes**

The where method is an application of the if-then idiom. For each element in the calling DataFrame, if cond is True the element is used; otherwise the corresponding element from the DataFrame other is used.

The signature for [**DataFrame.where()**](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.where.html#pandas.DataFrame.where) differs from [**numpy.where()**](https://numpy.org/doc/stable/reference/generated/numpy.where.html#numpy.where). Roughly df1.where(m, df2) is equivalent to np.where(m, df1, df2).

For further details and examples see the where documentation in [indexing](https://pandas.pydata.org/docs/user_guide/indexing.html#indexing-where-mask).

**Examples**

**>>>** s = pd.Series(range(5))

**>>>** s.where(s > 0)

0 NaN

1 1.0

2 2.0

3 3.0

4 4.0

dtype: float64

**>>>** s.mask(s > 0)

0 0.0

1 NaN

2 NaN

3 NaN

4 NaN

dtype: float64

**>>>** s.where(s > 1, 10)

0 10

1 10

2 2

3 3

4 4

dtype: int64

**>>>** s.mask(s > 1, 10)

0 0

1 1

2 10

3 10

4 10

dtype: int64

**>>>** df = pd.DataFrame(np.arange(10).reshape(-1, 2), columns=['A', 'B'])

**>>>** df

A B

0 0 1

1 2 3

2 4 5

3 6 7

4 8 9

**>>>** m = df % 3 == 0

**>>>** df.where(m, -df)

A B

0 0 -1

1 -2 3

2 -4 -5

3 6 -7

4 -8 9

**>>>** df.where(m, -df) == np.where(m, df, -df)

A B

0 True True

1 True True

2 True True

3 True True

4 True True

**>>>** df.where(m, -df) == df.mask(~m, -df)

A B

0 True True

1 True True

2 True True

3 True True

4 True True