Pandas: How can I use the apply() function for a single column?

Ask Question

149

28

I have a pandas data frame with two columns. I need to change the values of the first column without affecting the second one and get back the whole data frame with just first column values changed. How can I do that using apply in pandas?

python pandas dataframe python-3.5

edited Apr 19 '17 at 12:44

Fabio Lamanna
8,511 9 51 89

asked Jan 23 '16 at 10:04

Amani
3,365 11 45 75

3 Please post some input sample data and desired output. – Fabio Lamanna Jan 23 '16 at 10:12

Operate on the column directly instead. – Ted Petrou Nov 6 '17 at 22:43

As Ted Petrou said, avoid using `apply` as much as possible. If you're not sure you need to use it, you probably don't. I recommend taking a look at When should I ever want to use pandas apply() in my code?. – coldspeed Jan 30 at 10:22

3 Answers

218

Given a sample dataframe `df` as:

```
a,b
1,2
2,3
3,4
4,5
```

what you want is:

```
df['a'] = df['a']
```

that returns:

```
   a  b
0  2  2
1  3  3
2  4  4
3  5  5
```

dited Feb 21 '17 at 21:54

mit
6,159 6 34 60

nswered Jan 23 '16 at 10:15

Fabio Lamanna

never be used in a situation like this – Ted Petrou Nov 6 '17 at 22:41

4 @TedPetrou you're perfectly right, it was just an example on how to apply a general function on one single column, as the OP asked. – Fabio Lamanna Nov 7 '17 at 9:26

6 When I try doing this I get the following warning: "A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer, col_indexer] = value instead" – dagrun Mar 13 '18 at 11:24

9 As a matter of curiosity: why should apply not be used in that situation? What is the situation exactly? – Uncle Ben Ben Mar 28 '18 at 18:08

6 @UncleBenBen in general `apply` uses an internal loop over rows that is far slower than vectorized functions, like e.g. `df.a = df.a / 2` (see Mike Muller answer). – Fabio Lamanna Mar 29 '18 at 9:21

33

You don't need a function at all. You can work on a whole column directly.

Example data:

```
>>> df = pd.DataF
>>> df

      a     b
0   100   200   3
1  1000  2000  30
```

Half all the values in column `a` :

```
>>> df.a = df.a /
>>> df

     a     b
0   50   200   30
1  500  2000  300
```

dited Aug 22 '17 at 16:18

Chrisji
296 2 13

nswered Jan 23 '16 at 10:58

Mike Müller
55.3k 10 89 105

28

For a single column better to use `map()` , like this:

```
df = pd.DataFrame
25, 'b': 30, 'c':

    a   b  c
0  15  15  5
1  20  10  7
2  25  30  9
```

```
0   7.5  15  5
1  10.0  10  7
2  12.5  30  9
```

dited Mar 22 '17 at 14:18

Fabio Lamanna
8,511 9 51 89

nswered Jan 23 '16 at 10:49

George Petrov
1,129 6 17

56 Why is `map()` better than `apply()` for a single column? – ChaimG Feb 5 '17 at 18:21

3 I think it should be `lambda a: a / 2.` instead. – Max Candocia Mar 17 '17 at 4:52

1 This was very useful. I used it to extract file names from paths stored in a column `df['file_name'] = df['Path'].map(lambda a: os.path.basename(a))` – mmann1123 May 1 '18 at 19:10

17 map() is for Series (i.e. single columns) and operates on one cell at a time, while apply() is for DataFrame, and operates on a whole row at a time. – jpcgt Jul 24 '18 at 14:27