Pandas/Python: Set value of one column based on value in another column

Ask Question

22

5

I need to set the value of one column based on the value of another in a Pandas dataframe. This is the logic:

```
if df['c1'] == 'Value':
    df['c2'] = 10
else:
    df['c2'] = df['c3']
```

I am unable to get this to do what I want, which is to simply create a column with new values (or change the value of an existing column: either one works for me).

If I try to run the code above or if I write it as a function and use the apply method, I get the following:

```
ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(),
a.item(), a.any() or a.all().
```

python pandas conditional

edited Mar 7 '18 at 22:18

DJK
4,148 2 13 33

asked Mar 7 '18 at 21:01

NLR
184 1 1 14

4 Answers

25

one way to do this would be to use indexing with `.loc`.

Example

In the absence of an example dataframe, I'll make one up here:

```
df = pd.DataFrame({'c1': list('ab
df.loc[5, 'c1'] = 'Value'

>>> df
      c1
0      a
1      b
2      c
3      d
4      e
5  Value
6      g
```

Assuming you wanted to **create a new column** `c2`, equivalent to `c1` except where `c1` is `Value`, in which case, you would like to assign it to 10:

First, you could create a new column `c2`, and set it to equivalent as `c1`, using one of the following two lines (they essentially do the same thing):

```
df = df.assign(c2 = df['c1'])
# OR:
df['c2'] = df['c1']
```

Then, find all the indices where `c1` is equal to `'Value'` using `.loc`, and assign your desired value in `c2` at those indices:

```
df.loc[df['c1'] == 'Value', 'c2']
```

And you end up with this:

```
>>> df
      c1  c2
0      a   a
1      b   b
2      c   c
3      d   d
4      e   e
5  Value  10
6      g   g
```

If, as you suggested in your question, you would perhaps sometimes just want to **replace the values in the column you already have**, rather than create a new column, then just skip the column creation, and do the following:

```
df['c1'].loc[df['c1'] == 'Value']
```

Giving you:

```
>>> df
      c1
0      a
1      b
```

```
5     10
6      g
```

edited Sep 12 '18 at 18:49

Jacob Miller
27 9

answered Mar 7 '18 at 21:15

sacuL
30.9k 4 21 44

1 The second solution nailed it for me. I didn't realize you could use .loc like a WHERE statement in SQL. Makes sense. Thanks! – NLR Mar 7 '18 at 22:12

5

try:

```
df['c2'] = df['c1'].apply(lambda x: 10 if x == 'Value' else x)
```

answered Mar 7 '18 at 21:06

AlexanderHughes
72 7

Thanks @AlexanderHughes. My original post had a typo: there are actually three columns to consider, so this solution wouldn't work. – NLR Mar 7 '18 at 22:06

1 should be `df.apply(lambda x: 10 if x['c1'] == 'Value' else x['c3'],axis=1)` – DJK Mar 7 '18 at 22:32

1 This might have performance issues with large datasets. df.apply() is slower. – ErnestScribbler Nov 1 '18 at 10:21

4

you can use `np.where()` to set values based on a codition

```
#df
   c1  c2  c3
0   4   2   1
1   8   7   9
2   1   5   8
3   3   3   5
4   3   6   8
```

```
df['c2'] = np.where(df.c1 == 8,'X'
```

```
  c1  c3 c4
0   4   1  1
1   8   9  X
2   1   8  8
3   3   5  5
4   3   8  8
```

answered Mar 7 '18 at 22:28

DJK
4,148 2 13 33

1

I suggest doing it in two steps:

```
# set fixed value to 'c2' where th
df.loc[df['c1'] == 'Value', 'c2']

# copy value from 'c3' to 'c2' whe
df.loc[df['c1'] != 'Value', 'c2']
```

answered Mar 7 '18 at 22:29

Ralf
7,339 4 14 38
