Using conditional to generate new column in pandas dataframe

Ask Question

14

7

I have a pandas dataframe that looks like this:

```
   portion  used
0        1   1.0
1        2   0.3
2        3   0.0
3        4   0.8
```

I'd like to create a new column based on the `used` column, so that the `df` looks like this:

```
   portion  used    alert
0        1   1.0     Full
1        2   0.3  Partial
2        3   0.0    Empty
3        4   0.8  Partial
```

- Create a new `alert` column based on
- If `used` is `1.0`, `alert` should be `Full`.
- If `used` is `0.0`, `alert` should be `Empty`.
- Otherwise, `alert` should be `Partial`.

What's the best way to do that?

python pandas conditional calculated-columns

edited Oct 4 '17 at 17:30

Zero
40.6k 8 73 93

asked Nov 20 '14 at 14:14

user3786999
358 3 4 17

possible duplicate of Pandas conditional creation of a series/dataframe column – chrisb Nov 20 '14 at 14:17

31

You can define a function which returns your different states "Full", "Partial", "Empty", etc and then use `df.apply` to apply the function to each row. Note that you have to pass the keyword argument `axis=1` to ensure that it applies the function to rows.

```
import pandas as pd

def alert(c):
  if c['used'] == 1.0:
    return 'Full'
  elif c['used'] == 0.0:
    return 'Empty'
  elif 0.0 < c['used'] < 1.0:
    return 'Partial'
  else:
    return 'Undefined'

df = pd.DataFrame(data={'portion':

df['alert'] = df.apply(alert, axis

#    portion  used    alert
# 0        1   1.0     Full
# 1        2   0.3  Partial
# 2        3   0.0    Empty
# 3        4   0.8  Partial
```

answered Nov 20 '14 at 14:22

Ffisegydd
29.7k 8 97 94

Very cool - well done – kbball Oct 20 '17 at 19:39

Great example. To make the code a little bit clearer (and since you are using `axis=1`), you could re-name the parameter `c` to `row` , that way is really obvious that you have access to all values of the row in the function. – Onema Mar 22 at 22:55

23

Alternatively you could do:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(data={'portion':
'used':np.random.rand(10000)})

%%timeit
df.loc[df['used'] == 1.0, 'alert']
df.loc[df['used'] == 0.0, 'alert']
df.loc[(df['used'] >0.0) & (df['us
```

```
100 loops, best of 3: 2.91 ms per
```

Then using apply:

```
%timeit df['alert'] = df.apply(ale

1 loops, best of 3: 287 ms per loo
```

I guess the choice depends on how big is your dataframe.

edited Oct 4 '17 at 17:31

Zero
40.6k 8 73 93

answered Nov 20 '14 at 16:52

Primer
6,526 3 20 37

5

Use `np.where` , is usually fast

```
In [845]: df['alert'] = np.where(d
                                 n

In [846]: df
Out[846]:
   portion  used    alert
0        1   1.0     Full
1        2   0.3  Partial
2        3   0.0    Empty
3        4   0.8  Partial
```

Timings

```
In [848]: df.shape
Out[848]: (100000, 3)

In [849]: %timeit df['alert'] = np
== 0, 'Empty', 'Partial'))
100 loops, best of 3: 6.17 ms per

In [850]: %%timeit
     ...: df.loc[df['used'] == 1.0
     ...: df.loc[df['used'] == 0.0
     ...: df.loc[(df['used'] >0.0)
     ...:
10 loops, best of 3: 21.9 ms per l

In [851]: %timeit df['alert'] = df
1 loop, best of 3: 2.79 s per loop
```

answered Oct 4 '17 at 17:20

Zero
40.6k 8 73 93

0

Can't comment so making a new answer: Improving on Ffisegydd's approach, you can use a dictionary

```
import pandas as pd

def alert(c):
    mapping = {1.0: 'Full', 0.0: '
    return mapping.get(c['used'],

df = pd.DataFrame(data={'portion':

df['alert'] = df.apply(alert, axis
```

Depending on the use case, you might like to define the dict outside of the function definition as well.

answered Dec 3 '17 at 6:40

Spcogg the second
154 1 9

0

```
df['TaxStatus'] = np.where(df.Publ
False))
```

This would appear to work, except for the ValueError: either both or neither of x and y should be given

edited Mar 22 at 13:19

Rahul Agarwal
2,400 6 12 30

answered Oct 22 '18 at 1:47

user1857373
31 6