How to add an empty column to a dataframe?

Ask Question

162

43

What's the easiest way to add an empty column to a pandas `DataFrame` object? The best I've stumbled upon is something like

```
df['foo'] = df.apply(lambda _: '', axis=1)
```

Is there a less perverse method?

python pandas

edited Sep 19 '15 at 22:53
Joe Kington
186k 43 450 404

asked May 1 '13 at 21:46
kjo
11.6k 30 97 185

Do you actually want a column containing empty strings or rather `N/A` ? – filmor May 1 '13 at 21:50

6 Answers

270

If I understand correctly, assignment should fill:

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame({"A": [1,2,
>>> df
   A  B
0  1  2
1  2  3
2  3  4
>>> df["C"] = ""
>>> df["D"] = np.nan
>>> df
   A  B C   D
0  1  2    NaN
1  2  3    NaN
2  3  4    NaN
```

answered May 1 '13 at 21:52
DSM
216k 36 412 381

29

To add to DSM's answer and building on [this associated question](), I'd split the approach into two cases:

- Adding a single column: Just assign empty values to the new columns, e.g. `df['C'] = np.nan`
- Adding multiple columns: I'd suggest using the `.reindex(columns=[...])` [method of pandas]() to add the new columns to the dataframe's column index. This also works for adding multiple new rows.

Here is an example adding multiple columns:

```
mydf = mydf.reindex( mydf.columns.
>= 0.20.0
```

or

```
mydf = mydf.reindex( columns = myd
version < 0.20.0
```

You can also always concatenate a new (empty) dataframe to the existing dataframe, but that doesn't feel as pythonic to me :)

edited Nov 6 '17 at 14:31
dreab
308 1 5 15

answered Sep 9 '16 at 6:56
emunsing
4,485 2 15 25

Example for `version >= 0.20.0` deletes the DataFrame and adds the new columns as rows. Example for `version < 0.20.0` works fine on Pandas Version `0.24.1` – Lalo Mar 11 at 14:20

where "header_list" is a list of the headers you want to appear.

any header included in the list that is not found already in the dataframe will be added with blank cells below.

so if

```
header_list = ['a','b','c', 'd']
```

then c and d will be added as columns with blank cells

edited May 16 '17 at 8:29
maazza
3,915 13 44 80

answered May 16 '17 at 8:08
liana
251 3 2

2 More precisely, the columns will be added with NaNs. – broccoli2000 Aug 1 '17 at 14:18

13

Starting with `v0.16.0`, `DF.assign()` could be used to assign new columns (*single/multiple*) to a `DF`. These columns get inserted in alphabetical order at the end of the `DF`.

This becomes advantageous compared to simple assignment in cases wherein you want to perform a series of chained operations directly on the returned dataframe.

Consider the same `DF` sample demonstrated by @DSM:

```
df = pd.DataFrame({"A": [1,2,3], "
df
Out[18]:
   A  B
0  1  2
1  2  3
2  3  4

df.assign(C="",D=np.nan)
Out[21]:
   A  B C   D
0  1  2   NaN
1  2  3   NaN
2  3  4   NaN
```

Note that this returns a copy with all the previous columns along with the newly created ones. Inorder for the original `DF` to be modified

answered Jan 31 '17 at 8:53
Nickil Maveli
18.2k 4 37 49

What is that datatype for C? I am trying to add by looping through a list of strings. But it does not use it. – eleijonmarck Oct 24 '17 at 11:04

3

@emunsing's answer is really cool for adding multiple columns, but I couldn't get it to work for me in python 2.7. Instead, I found this works:

```
mydf = mydf.reindex(columns = np.a
['newcol1','newcol2'])
```

edited May 23 '17 at 12:34
Community ♦
1 1

answered Apr 17 '17 at 13:23
jua-kali
428 4 16

2

if you want to add column name from a list

```
df=pd.DataFrame()
a=['col1','col2','col3','col4']
for i in range(len(a)):
    df[a[i]]=np.nan
```

answered Mar 22 '18 at 4:30
Joy Mazumder
68 1 1 6