How to select all columns, except one column in pandas? [duplicate]

163

44

This question already has an answer here:
Delete column from pandas DataFrame *14 answers*

I have a dataframe look like this:

```
import pandas
import numpy as np
df = DataFrame(np.random.rand(4,4), columns = list('abcd'))
df
      a         b         c         d
0  0.418762  0.042369  0.869203  0.972314
1  0.991058  0.510228  0.594784  0.534366
2  0.407472  0.259811  0.396664  0.894202
3  0.726168  0.139531  0.324932  0.906575
```

How I can get all columns except `column b` ?

python pandas

edited Sep 21 '17 at 9:43
ayhan
39.6k 6 82 117

asked Apr 21 '15 at 5:24
markov zain
2,149 9 19 34

marked as duplicate by cs95 pandas Jan 24 at 11:16

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

@cs95 -- The currently listed duplicate target isn't a duplicate. Despite the original title, the linked question is "Why doesn't this specific syntax work", whereas this question is a more general "What is the best way to do this". -- Add to this the difference between deleting a column from an existing DataFrame versus creating a new DataFrame with all-but-one of the columns of another. – R.M. May 21 at 19:30

@R.M. I'm sorry but I don't agree with the edit you've made to the title on that post, so I've rolled it back. It's true that the intent of the OP was to question the syntax, but the post has grown to address the more broad question of how to delete a column. The answers in this post are carbon copies of the highest upvoted post there. The dupe stays. – cs95 May 21 at 19:46

Note this question is being discussed on Meta. – Heretic Monkey May 21 at 21:24

7 Answers

When you don't have a MultiIndex, `df.columns` is just an array of column names so you can do:

253

```
df.loc[:, df.columns != 'b']

          a         c         d
0  0.561196  0.013768  0.772827
1  0.882641  0.615396  0.075381
2  0.368824  0.651378  0.397203
3  0.788730  0.568099  0.869127
```

edited Sep 21 '17 at 9:42
ayhan
39.6k 6 82 117

answered Apr 21 '15 at 5:27
Marius
35.2k 10 74 80

8 Not bad, but @mike's solution using `drop` is better IMO. A bit more readable and handles multiindexes – travc Jun 30 '17 at 0:24

1 I actually agree that @mike's solution using `drop` is better - I do think it's useful to discover that (single-level) columns are arrays you can work with, but specifically for dropping a column, `drop` is very readable and works well with complex indexes. – Marius Apr 23 at 23:06

152

Don't use `ix` . It's deprecated. The most readable and idiomatic way of doing this is `df.drop()` :

```
>>> df

          a         b         c         d
0  0.175127  0.191051  0.382122  0.869242
1  0.414376  0.300502  0.554819  0.497524
2  0.142878  0.406830  0.314240  0.093132
3  0.337368  0.851783  0.933441  0.949598

>>> df.drop('b', axis=1)

          a         c         d
0  0.175127  0.382122  0.869242
1  0.414376  0.554819  0.497524
2  0.142878  0.314240  0.093132
3  0.337368  0.933441  0.949598
```

Note that by default, `.drop()` does not operate inplace; despite the ominous name, `df` is unharmed by this process. If you want to permanently remove `b` from `df` , do `df.drop('b', inplace=True)` .

`df.drop()` also accepts a list of labels, e.g. `df.drop(['a', 'b'], axis=1)` will drop column `a` and `b` .

edited Jul 1 '17 at 1:17

answered Jun 9 '16 at 5:38
Mike
1,744 1 11 11

1 Also works on a multiindex just like you'd expect it to. `df.drop([('l1name', 'l2name'), 'anotherl1name'], axis=1)` . Seems to use list vs tuple to determine if you want multiple columns (list) or referring to a multiindex (tuple). – travc Jun 30 '17 at 0:20

11 More readable: `df.drop(columns='a')` or `df.drop(columns=['a', 'b'])` . Can also replace `columns=` with `index=` . – BallpointBen May 9 '18 at 13:52

However this is not useful if you happen *not* to know the names of all the columns you want to drop. – yeliabsalohcin Sep 4 '18 at 16:17

91

```
df[df.columns.difference(['b'])]

Out:
          a         c         d
0  0.427809  0.459807  0.333869
1  0.678031  0.668346  0.645951
2  0.996573  0.673730  0.314911
3  0.786942  0.719665  0.330833
```

answered Aug 28 '16 at 14:05
ayhan
39.6k 6 82 117

6 I like this approach as it can be used to omit more than one column. – Nischal Hp Aug 30 '17 at 9:42

@NischalHp df.drop can also omit more than one column df.drop(['a', 'b'], axis=1) – Patrick Li Apr 25 at 13:02

29

You can use `df.columns.isin()`

```
df.loc[:, ~df.columns.isin(['b'])]
```

When you want to drop multiple columns, as simple as:

```
df.loc[:, ~df.columns.isin(['col1', 'col2'])]
```

answered Oct 31 '18 at 1:47
Tom
2,125 1 15 14

11

Here is another way:

```
df[[i for i in list(df.columns) if i != '<your column>']]
```

You just pass all columns to be shown except of the one you do not want.

answered Aug 18 '16 at 3:22
Salvador Dali
124k 89 528 623

4

Another slight modification to @Salvador Dali enables a list of columns to exclude:

```
df[[i for i in list(df.columns) if i not in [list_of_columns_to_exclude]]]
```

or

```
df.loc[:,[i for i in list(df.columns) if i not in [list_of_columns_to_exclude]]]
```

answered Sep 23 '18 at 14:29
user1718097
1,471 3 22 35

2

I think the best way to do is the way mentioned by @Salvador Dali. Not that the others are wrong.

Because when you have a data set where you just want to select one column and put it into one variable and the rest of the columns into another for comparison or computational purposes. Then dropping the column of the data set might not help. Of course there are use cases for that as well.

```
x_cols = [x for x in data.columns if x != 'name of column to be excluded']
```

Then you can put those collection of columns in variable `x_cols` into another variable like `x_cols1` for other computation.

```
ex: x_cols1 = data[x_cols]
```

edited Apr 21 '18 at 14:39
MRizwan33
1,664 5 20 32

answered Apr 21 '18 at 13:19
Sudhi
132 11