Append a NumPy array to a NumPy array

I have a numpy_array. Something like `[ a b c ]`.

And then I want to append it into another NumPy array (just like we create a list of lists). How do we create an array of NumPy arrays containing NumPy arrays?

I tried to do the following without any luck

```
>>> M = np.array([])
>>> M
array([], dtype=float64)
>>> M.append(a,axis=0)
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
AttributeError: 'numpy.ndarray' object has no attribute 'append'
>>> a
array([1, 2, 3])
```

python numpy

edited Mar 22 '16 at 18:57
Eric Leschinski
60.6k 28 255 221

asked Mar 19 '12 at 17:55
Fraz
7,266 39 98 162

2 You can create an "array of arrays" (you use an object array), but you almost definitely don't want to. What are you trying to do? Do you just want a 2d array? – Joe Kington Mar 19 '12 at 18:00

2 yeah.. I am trying to get a 2D array – Fraz Mar 19 '12 at 18:04

@Fraz: *Why* do you want a 2D array? What are you trying to do? – endolith Mar 19 '12 at 18:15

6 Answers

```
In [1]: import numpy as np

In [2]: a = np.array([[1, 2, 3], [4, 5, 6]])

In [3]: b = np.array([[9, 8, 7], [6, 5, 4]])

In [4]: np.concatenate((a, b))
Out[4]:
array([[1, 2, 3],
       [4, 5, 6],
       [9, 8, 7],
       [6, 5, 4]])
```

or this:

```
In [1]: a = np.array([1, 2, 3])

In [2]: b = np.array([4, 5, 6])

In [3]: np.vstack((a, b))
Out[3]:
array([[1, 2, 3],
       [4, 5, 6]])
```

edited Aug 2 '16 at 17:27

answered Mar 19 '12 at 18:01
endolith
7,780 13 67 126

1 Hi when i run this i get this np.concatenate((a,b),axis=1) Output: array([1, 2, 3, 2, 3, 4]) But what I looking for is numpy 2d array?? – Fraz Mar 19 '12 at 18:05

2 @Fraz: I've added Sven's `vstack()` idea. You know you can create the array with `array([[1,2,3], [2,3,4]])`, right? – endolith Mar 19 '12 at 18:14

concatenate() is the one I needed. – kakyo Feb 19 '15 at 0:17

`numpy.vstack` can accept more than 2 arrays in the sequence argument. Thus if you need to combine more than 2 arrays, vstack is more handy. – oneleggedmule Oct 22 '15 at 12:57

1 @oneleggedmule `concatenate` can also take multiple arrays – endolith Oct 22 '15 at 13:28

Well, the error message says it all: NumPy arrays do not have an `append()` method. There's a free function `numpy.append()` however:

```
numpy.append(M, a)
```

This will create a new array instead of mutating `M` in place. Note that using `numpy.append()` involves copying both arrays. You will get better performing code if you use fixed-sized NumPy arrays.

edited Jan 17 '14 at 18:24
Uli Köhler
7,537 7 31 67

answered Mar 19 '12 at 17:59
Sven Marnach
262k 51 638 632

Hi.. when i try this.. I get this >>> np.append(M,a) array([1., 2., 3.]) >>> np.append(M,b) array([2., 3., 4.]) >>> M array([], dtype=float64) I was hoping M to be a 2D array?? – Fraz Mar 19 '12 at 18:06

5 @Fraz: Have a look at `numpy.vstack()` . – Sven Marnach Mar 19 '12 at 18:08

Sven said it all, just be very cautious because of automatic type adjustments when append is called.

```
In [2]: import numpy as np

In [3]: a = np.array([1,2,3])

In [4]: b = np.array([1.,2.,3.])

In [5]: c = np.array(['a','b','c'])

In [6]: np.append(a,b)
Out[6]: array([ 1.,  2.,  3.,  1.,  2.,  3.])

In [7]: a.dtype
Out[7]: dtype('int64')

In [8]: np.append(a,c)
Out[8]:
array(['1', '2', '3', 'a', 'b', 'c'],
      dtype='|S1')
```

As you see based on the contents the dtype went from int64 to float32, and then to S1

answered Mar 19 '12 at 18:03
lukecampbell
5,238 2 20 24

You may use `numpy.append()` ...

```
import numpy

B = numpy.array([3])
A = numpy.array([1, 2, 2])
B = numpy.append( B , A )

print B

> [3 1 2 2]
```

This will not create two separate arrays but will append two arrays into a single dimensional array.

edited Apr 1 at 15:09
Bolboa
1,833 2 16 47

answered Jul 13 '16 at 12:05
Zhai Zhiwei
21 2

If I understand your question, here's one way. Say you have:

```
a = [4.1, 6.21, 1.0]
```

so here's some code...

```
def array_in_array(scalarlist):
    return [(x,) for x in scalarlist]
```

Which leads to:

```
In [72]: a = [4.1, 6.21, 1.0]

In [73]: a
Out[73]: [4.1, 6.21, 1.0]

In [74]: def array_in_array(scalarlist):
   ....:     return [(x,) for x in scalarlist]
   ....:

In [75]: b = array_in_array(a)

In [76]: b
Out[76]: [(4.1,), (6.21,), (1.0,)]
```

answered Sep 22 '14 at 12:10

linhares
315 2 11

Actually one can always create an ordinary list of numpy arrays and convert it later.

```
In [1]: import numpy as np

In [2]: a = np.array([[1,2],[3,4]])

In [3]: b = np.array([[1,2],[3,4]])

In [4]: l = [a]

In [5]: l.append(b)

In [6]: l = np.array(l)

In [7]: l.shape
Out[7]: (2, 2, 2)

In [8]: l
Out[8]:
array([[[1, 2],
        [3, 4]],

       [[1, 2],
        [3, 4]]])
```

answered Mar 16 at 9:14

Michael Ma
32 9
