How to add a new row to an empty numpy array

Using standard Python arrays, I can do the following:

```
arr = []
arr.append([1,2,3])
arr.append([4,5,6])
# arr is now [[1,2,3],[4,5,6]]
```

However, I cannot do the same thing in numpy. For example:

```
arr = np.array([])
arr = np.append(arr, np.array([1,2,3]))
arr = np.append(arr, np.array([4,5,6]))
# arr is now [1,2,3,4,5,6]
```

I also looked into `vstack` , but when I use `vstack` on an empty array, I get:

```
ValueError: all the input array dimensions except for the concatenation axis must match
exactly
```

So how do I do append a new row to an empty array in numpy?

python numpy scipy

edited Mar 14 '14 at 14:21
Fred Foo
247k 42 486 655

asked Mar 13 '14 at 22:39
Tony Stark
692 5 14 25

If it's empty, why bother? Just start from an array holding only the first row. – jonrsharpe Mar 13 '14 at 22:50

I just want to know whether it is possible to append to an empty numpy array. Sometimes it's cleaner to write code like this since the append operations are in a loop. – Tony Stark Mar 13 '14 at 22:55

3 Given the way numpy arrays work, you are much better building an empty array then putting the data in, e.g. See stackoverflow.com/questions/568962/… – jonrsharpe Mar 13 '14 at 22:56

4 Answers

The way to "start" the array that you want is:

```
arr = np.empty((0,3), int)
```

Which is an empty array but it has the proper dimensionality.

```
>>> arr
array([], shape=(0, 3), dtype=int64)
```

Then be sure to append along axis 0:

```
arr = np.append(arr, np.array([[1,2,3]]), axis=0)
arr = np.append(arr, np.array([[4,5,6]]), axis=0)
```

But, @jonrsharpe is right. In fact, if you're going to be appending in a loop, it would be much faster to append to a list as in your first example, then convert to a numpy array at the end, since you're really not using numpy as intended during the loop:

```
In [210]: %%timeit
   .....: l = []
   .....: for i in xrange(1000):
   .....:     l.append([3*i+1,3*i+2,3*i+3])
   .....: l = np.asarray(l)
   .....:
1000 loops, best of 3: 1.18 ms per loop

In [211]: %%timeit
   .....: a = np.empty((0,3), int)
   .....: for i in xrange(1000):
   .....:     a = np.append(a, 3*i+np.array([[1,2,3]]), 0)
   .....:
100 loops, best of 3: 18.5 ms per loop
```

```
In [214]: np.allclose(a, 1)
Out[214]: True
```

The numpythonic way to do it depends on your application, but it would be more like:

```
In [220]: timeit n = np.arange(1,3001).reshape(1000,3)
100000 loops, best of 3: 5.93 µs per loop

In [221]: np.allclose(a, n)
Out[221]: True
```

answered Mar 14 '14 at 1:03
askewchan
21.7k 6 55 91

what if I have to do this 10^5 or 10^6 times? it seems that neither of these methods will hold. any suggestion? – Roberto Franceschini Dec 5 '16 at 9:00

@Roberto, usually there is some way to determine the size or shape (at the very least, values would be preferable) of the array in advance. Do you think you can do that? Appending should really be a one or two time operation. – askewchan Dec 7 '16 at 16:40

sometimes you cannot guess the dimensions, it's life. However you can allocate a big enough array and give values to its views. I do not like it though, because there are unwanted values that one has to find a way to "mask". This idea of masking really does not fit my taste. – Roberto Franceschini Dec 8 '16 at 17:17

No need to mask, just slice! `a = a[:N]` Though I strongly believe you should find a way to vectorize it (post a new question with your specifics if you need help) or just use lists until the loop is over. – askewchan Dec 8 '16 at 20:15

In this case you might want to use the functions np.hstack and np.vstack

```
arr = np.array([])
arr = np.hstack((arr, np.array([1,2,3])))
# arr is now [1,2,3]

arr = np.vstack((arr, np.array([4,5,6])))
# arr is now [[1,2,3],[4,5,6]]
```

You also can use the np.concatenate function.

Cheers

answered Mar 13 '14 at 22:56
mrcl
1,270 6 19

3 Won't work if the second array has dimension >=2 like ones((2, 2)). It appears to me there is no way to avoid boundary cases if you are building up arrays from empty by concatenation. – Taozi Oct 20 '15 at 21:25

Here is my solution:

```
arr = []
arr.append([1,2,3])
arr.append([4,5,6])
np_arr = np.array(arr)
```

answered May 6 at 10:55
just4fun
11 1

---

using an custom dtype definition, what worked for me was:

```
import numpy

# define custom dtype
type1 = numpy.dtype([('freq', numpy.float64, 1), ('amplitude', numpy.float64, 1)])
# declare empty array, zero rows but one column
arr = numpy.empty([0,1],dtype=type1)
# store row data, maybe inside a loop
row = numpy.array([(0.0001, 0.002)], dtype=type1)
# append row to the main array
arr = numpy.row_stack((arr, row))
# print values stored in the row 0
```

```
print float(arr[0]['freq'])
print float(arr[0]['amplitude'])
```

answered Aug 5 '14 at 4:13

boclodoa
333 2 11