Numpy array, how to select indices satisfying multiple conditions?

Suppose I have a numpy array `x = [5, 2, 3, 1, 4, 5]`, `y = ['f', 'o', 'o', 'b', 'a', 'r']`. I want to select the elements in `y` corresponding to elements in `x` that are greater than 1 and less than 5.

I tried

```
x = array([5, 2, 3, 1, 4, 5])
y = array(['f','o','o','b','a','r'])
output = y[x > 1 & x < 5] # desired output is ['o','o','a']
```

but this doesn't work. How would I do this?

python numpy

edited Feb 28 '12 at 22:14
Hooked
29.5k 12 95 155

asked Jun 12 '10 at 23:28
Bob
300 1 4 6

3 Answers

Your expression works if you add parentheses:

```
>>> y[(1 < x) & (x < 5)]
array(['o', 'o', 'a'],
      dtype='|S1')
```

answered Jun 13 '10 at 0:50
J.F. Sebastian
184k 43 343 504

That is nice.. vecMask=1<x generates a vector mask like vecMask=(False, True, ...), which can be just combined with other vector masks. Each element is the condition for taking the elements of a source vector (True) or not (False). This can be used also with the full version numpy.extract(vecMask, vecSrc), or numpy.where(vecMask, vecSrc, vecSrc2). – Ralf Nov 3 at 17:08

IMO OP does not actually want `np.bitwise_and()` (aka `&`) but actually wants `np.logical_and()` because they are comparing logical values such as `True` and `False` - see this SO post on logical vs. bitwise to see the difference.

```
>>> x = array([5, 2, 3, 1, 4, 5])
>>> y = array(['f','o','o','b','a','r'])
>>> output = y[np.logical_and(x > 1, x < 5)] # desired output is ['o','o','a']
>>> output
array(['o', 'o', 'a'],
      dtype='|S1')
```

And equivalent way to do this is with `np.all()` by setting the `axis` argument appropriately.

```
>>> output = y[np.all([x > 1, x < 5], axis=0)] # desired output is ['o','o','a']
>>> output
array(['o', 'o', 'a'],
      dtype='|S1')
```

edited Oct 16 '13 at 18:10

answered Sep 5 '13 at 19:23

Mark Mikofski
8,114 1 24 44

5 You need to be a little careful about how you speak about what's evaluated. For example, in `output = y[np.logical_and(x > 1, x < 5)]`, `x < 5` *is* evaluated (possibly creating an enormous array), even though it's the second argument, because that evaluation happens outside of the function. IOW, `logical_and` gets passed two already-evaluated arguments. This is different from the usual case of `a and b`, in which `b` isn't evaluated if `a` is truelike. – DSM Sep 5 '13 at 19:29

6 there is no difference between bitwise_and() and logical_and() for boolean arrays – J.F. Sebastian Apr 13 '14 at 20:07

Add one detail to @J.F. Sebastian's and @Mark Mikofski's answers:
If one wants to get the corresponding indices (rather than the actual values of array), the following

code will do:

For satisfying multiple (all) conditions:

```
select_indices = np.where( np.logical_and( x > 1, x < 5) ) #   1 < x <5
```

For satisfying multiple (or) conditions:

```
select_indices = np.where( np.logical_or( x < 1, x > 5 ) ) # x <1 or x >5
```

edited Nov 18 '14 at 16:27

fredtantini
6,968 5 16 33

answered Nov 18 '14 at 16:03

Good Will
151 2 4