Explicitly select items from a Python list or tuple

Ask Question

91

28

I have the following Python list (can also be a tuple):

```
myList = ['foo', 'bar', 'baz', 'quux']
```

I can say

```
>>> myList[0:3]
['foo', 'bar', 'baz']
>>> myList[::2]
['foo', 'baz']
>>> myList[1::2]
['bar', 'quux']
```

How do I explicitly pick out items whose indices have no specific patterns? For example, I want to select `[0,2,3]`. Or from a very big list of 1000 items, I want to select `[87, 342, 217, 998, 500]`. Is there some Python syntax that does that? Something that looks like:

```
>>> myBigList[87, 342, 217, 998, 500]
```

python list select indexing tuples

asked Jul 9 '11 at 1:49

Kit
14.6k 24 83 137

1 This appears to be a duplicate. The other question has more up votes but this seems like it has a better answer with timings. – Annan Jun 4 '17 at 20:06

8 Answers

I compared the answers with python 2.5.2:

- 19.7 usec: `[ myBigList[i] for i in [87, 342, 217, 998, 500] ]`
- 20.6 usec: `map(myBigList.__getitem__, (87, 342, 217, 998, 500))`
- 22.7 usec: `itemgetter(87, 342, 217, 998, 500)(myBigList)`
- 24.6 usec: `list( myBigList[i] for i in [87, 342, 217, 998, 500] )`

Note that in Python 3, the 1st was changed to be the same as the 4th.

Another option would be to start out with a `numpy.array` which allows indexing via a list or a `numpy.array`:

```
>>> import numpy
>>> myBigList = numpy.array(range
>>> myBigList[(87, 342, 217, 998,
Traceback (most recent call last)
  File "<stdin>", line 1, in <modu
IndexError: invalid index
>>> myBigList[[87, 342, 217, 998,
array([ 87, 342, 217, 998, 500])
>>> myBigList[numpy.array([87, 34
array([ 87, 342, 217, 998, 500])
```

The `tuple` doesn't work the same way as those are slices.

edited Nov 25 '15 at 17:32

answered Jul 9 '11 at 1:53

Dan D.
54.8k 10 80 101

2 Preferably as a list comp, `[myBigList[i] for i in [87, 342, 217, 998, 500]]`, but I like this approach the best. – zeekay Jul 9 '11 at 1:57

@MedhatHelmy That's already in the answer. The third option used `from operator import itemgetter` in the initialization part of `python -mtimeit`. – Dan D. Nov 25 '15 at 14:33

I wonder, just from a language design perspective, why `myBigList[(87, 342, 217, 998, 500)]` doesn't work when `myBigList` is a regular python `list`? When I try that I get `TypeError: list indices must`

design/implementation issue involved? – sparc_spread Mar 24 '16 at 10:26

@sparc_spread, this is because `lists` in Python only accept integers or slices. Passing an integer makes sure that only one item is retrieved from an existing list. Passing a slice makes sure a part of it is retrieved, but passing a tuple is like passing a data-type(`tuple`) as an argument to another data-type(`list`) which is syntactically incorrect. – amanb Jun 17 '18 at 21:30

34

What about this:

```
from operator import itemgetter
itemgetter(0,2,3)(myList)
('foo', 'baz', 'quux')
```

answered Jul 9 '11 at 1:52

Marcin
27.9k 2 53 82

This is the sexiest so far. Love that `operator` module! – jathanism Jul 9 '11 at 2:07

9

It isn't built-in, but you can make a subclass of list that takes tuples as "indexes" if you'd like:

```
class MyList(list):

    def __getitem__(self, index):
        if isinstance(index, tuple
            return [self[i] for i
        return super(MyList, self)


seq = MyList("foo bar baaz quux mu
print seq[0]
print seq[2,4]
print seq[1::2]
```

printing

```
foo
['baaz', 'mumble']
['bar', 'quux']
```

1 (+1) Neat solution! With this extension, handling arrays in Python starts to look much R or Matlab. – Assad Ebrahim Feb 11 '14 at 18:30

6

Maybe a list comprehension is in order:

```
L = ['a', 'b', 'c', 'd', 'e', 'f']
print [ L[index] for index in [1,3
```

Produces:

```
['b', 'd', 'f']
```

Is that what you are looking for?

edited Jul 9 '11 at 2:30

answered Jul 9 '11 at 2:00

Dan Witkowski
131 3

5

```
>>> map(myList.__getitem__, (2,2,1
('baz', 'baz', 'bar', 'quux')
```

You can also create your own `List` class which supports tuples as arguments to `__getitem__` if you want to be able to do `myList[(2,2,1,3)]` .

answered Jul 9 '11 at 2:02

ninjagecko
61.7k 18 114 123

While this works it's usually not a good idea to directly invoke magic variables. You're better off using a list comprehension or a helper module like `operator` . – jathanism Jul 9 '11 at 2:08

@jathanism: I have to respectfully disagree. Though if you are concerned about forward compatibility

(as opposed to public/private) I can definitely see where you're coming from. – ninjagecko Jul 9 '11 at 2:13

That is where I'm coming from. :) Following that, it's the same reason why it's better to use `len(myList)` over `myList.__len__()` . – jathanism Jul 11 '11 at 16:25

way based on programming circumstances. – Jacob CUI Mar 25 '15 at 22:57

1

I just want to point out, even syntax of itemgetter looks really neat, but it's kinda slow when perform on large list.

```
import timeit
from operator import itemgetter
start=timeit.default_timer()
for i in range(1000000):
    itemgetter(0,2,3)(myList)
print ("Itemgetter took ", (timeit
```

Itemgetter took 1.065209062149279

```
start=timeit.default_timer()
for i in range(1000000):
    myList[0],myList[2],myList[3]
print ("Multiple slice took ", (ti
```

Multiple slice took 0.6225321444745759

edited Nov 1 '16 at 14:56

answered Nov 1 '16 at 14:50

Wendao Liu
28 5

First snippet, please add `myList = np.array(range(1000000))` otherwise you will get error. – Cloud Cho Jan 23 at 1:31

0

Another possible solution:

```
sek=[]
L=[1,2,3,4,5,6,7,8,9,0]
for i in [2, 4, 7, 0, 3]:
   a=[L[i]]
   sek=sek+a
print (sek)
```

answered Nov 18 '17 at 20:32

fdante
1 1

0

like often when you have a boolean numpy array like `mask`

```
[mylist[i] for i in
np.arange(len(mask), dtype=int)
```

```
 subseq = lambda myseq, mask :
[myseq[i] for i in
np.arange(len(mask), dtype=int)
[mask]]

 newseq = subseq(myseq, mask)
```

edited Jul 16 '18 at 15:37

answered Jul 16 '18 at 15:26

theo olsthoorn
74 1 4