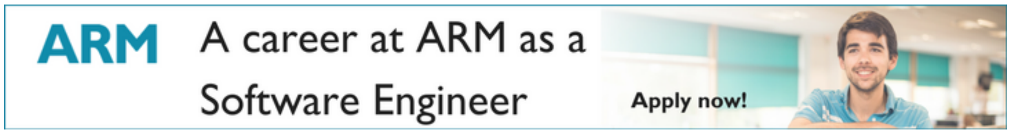
python nested list comprehension



I have a this list:

```
l = [['40', '20', '10', '30'], ['20', '20', '20', '20', '20', '30', '20'], ['30', '20',
'30', '50', '10', '30', '20', '20', '20'], ['100', '100'], ['100', '100', '100', '100',
'100'], ['100', '100', '100', '100']]
```

Now, what I want to do is convert each element in a list to float. My solution is this:

```
newList = []
for x in l:
  for y in x:
    newList.append(float(y))
```

But can this be done using nested list comprehension, right?

what I've done is:

```
[float(y) for y in x for x in l]
```

But then the result is bunch of 100's with the sum of 2400.

any solution, an explanation would be much appreciated. Thanks!

python list list-comprehension

edited Feb 21 '16 at 1:28
falsetru
194k 25 268 305

asked Aug 6 '13 at 6:02
Boy Pasmo
2,023 7 19 33

8 Do you *also* want to flatten your list? – Greg Hewgill Aug 6 '13 at 6:05

7 Answers

Here is how you would do this with a nested list comprehension:

```
[[float(y) for y in x] for x in l]
```

This would give you a list of lists, similar to what you started with except with floats instead of strings. If you want one flat list then you would use `[float(y) for x in l for y in x]` .

answered Aug 6 '13 at 6:05
Andrew Clark
114k 12 152 214

```
>>> l = [['40', '20', '10', '30'], ['20', '20', '20', '20', '20', '30', '20'], ['30',
'20', '30', '50', '10', '30', '20', '20', '20'], ['100', '100'], ['100', '100', '100',
'100', '100'], ['100', '100', '100', '100']]
>>> new_list = [float(x) for xs in l for x in xs]
>>> new_list
[40.0, 20.0, 10.0, 30.0, 20.0, 20.0, 20.0, 20.0, 20.0, 30.0, 20.0, 30.0, 20.0, 30.0, 50.0,
10.0, 30.0, 20.0, 20.0, 20.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0,
100.0, 100.0, 100.0]
```

answered Aug 6 '13 at 6:04
falsetru
194k 25 268 305

Not sure what your desired output is, but if you're using list comprehension, the order follows the order of nested loops, which you have backwards. So I got the what I think you want with:

```
[float(y) for x in l for y in x]
```

The principle is: use the same order you'd use in writing it out as nested for loops.

answered Mar 26 at 2:49
Harry Binswanger
126 1 7

this should be the answer, as some times we don't want to square bracket the iteratool – zinking Apr 19 at 1:48

Yes, you can do it with such a code:

```
l = [[float(y) for y in x] for x in l]
```

answered Aug 6 '13 at 6:06
Victor
50 1 6

`[float(y) for y in x for x in l]` this would result to a bunch of 100's with a sum of 2400. – Boy Pasmo Aug 6 '13 at 6:09

If you don't like nested list comprehensions, you can make use of the **map** function as well,

```
>>> from pprint import pprint

>>> l = l = [['40', '20', '10', '30'], ['20', '20', '20', '20', '20', '30', '20'], ['30',
'20', '30', '50', '10', '30', '20', '20', '20'], ['100', '100'], ['100', '100', '100',
'100', '100'], ['100', '100', '100', '100']]

>>> pprint(l)
[['40', '20', '10', '30'],
['20', '20', '20', '20', '20', '30', '20'],
['30', '20', '30', '50', '10', '30', '20', '20', '20'],
['100', '100'],
['100', '100', '100', '100', '100'],
['100', '100', '100', '100']]

>>> float_l = [map(float, nested_list) for nested_list in l]

>>> pprint(float_l)
[[40.0, 20.0, 10.0, 30.0],
[20.0, 20.0, 20.0, 20.0, 20.0, 30.0, 20.0],
[30.0, 20.0, 30.0, 50.0, 10.0, 30.0, 20.0, 20.0, 20.0],
[100.0, 100.0],
[100.0, 100.0, 100.0, 100.0, 100.0],
[100.0, 100.0, 100.0, 100.0]]
```

edited Jan 17 '16 at 7:10
Kevin Guan
11.3k 9 29 53

answered Nov 27 '15 at 5:55
narayan
351 2 8

Your code generates map objects instead of lists: `>>> float_l = [map(float, nested_list) for nested_list in l]` `[[<map at 0x47be9b0>], [<map at 0x47be2e8>], [<map at 0x47be4a8>], [<map at 0x47beeb8>], [<map at 0x484b048>], [<map at 0x484b0b8>]]` but adding an additional call to list it works as expected: `>>> float_l = [list(map(float, nested_list)) for nested_list in l]` – pixelperfect Mar 17 at 15:49

The best way to do this in my opinion is to use python's `itertools` package.

```
>>>import itertools
>>>l1 = [1,2,3]
>>>l2 = [10,20,30]
>>>[l*2 for l in itertools.chain(*[l1,l2])]
[2, 4, 6, 20, 40, 60]
```

answered Jul 12 '16 at 10:46

Thomasillo
6 1

This Problem can be solved without using for loop.Single line code will be sufficient for this. Using Nested Map with lambda function will also works here.

l = [['40', '20', '10', '30'], ['20', '20', '20', '20', '20', '30', '20'], ['30', '20', '30', '50', '10', '30', '20', '20', '20'], ['100', '100'], ['100', '100', '100', '100', '100'], ['100', '100', '100', '100']]

```
map(lambda x:map(lambda y:float(y),x),l)
```

And Output List would be as follows:

[[40.0, 20.0, 10.0, 30.0], [20.0, 20.0, 20.0, 20.0, 20.0, 30.0, 20.0], [30.0, 20.0, 30.0, 50.0, 10.0, 30.0, 20.0, 20.0, 20.0], [100.0, 100.0], [100.0, 100.0, 100.0, 100.0, 100.0], [100.0, 100.0, 100.0, 100.0]]

answered Apr 27 at 6:52

Aakash Goel
83 6