How do I perform secondary sorting in python?

Asked 7 years, 6 months ago Active 4 years, 2 months ago Viewed 22k times

28

8

If i have a list of numbers `[4,2,5,1,3]` I want to sort it first by some function `f` and then for numbers with the same value of `f` i want it to be sorted by the magnitude of the number.

This code does not seem to be working.

```
list5 = sorted(list5)
list5 = sorted(list5, key = lambda vertex: degree(vertex))
```

Secondary sorting first: list5 is sorted based on magnitude. Primary sorting next: list5 is sorted based on some function of the numbers.

python Edit tags

edited Apr 24 '13 at 13:42
Henrik Andersson
35.4k 14 84 84

asked Apr 24 '13 at 13:41
Vinu K S
583 1 7 17

1 btw you can just do `key=degree` , here the `lambda` is redundant – GP89 Apr 24 '13 at 13:44

When you say it "does not seem to be working", what do you observe? – ecatmur Aug 26 '14 at 9:42

6 Answers

Active Oldest Votes

66

Sort it by a (firstkey, secondkey) tuple:

```
sorted(list5, key=lambda vertex: (degree(vertex), vertex))
```

answered Apr 24 '13 at 13:44
Pavel Anossov
51.8k 11 130 116

4 To do ascending on one and descending on the other, two calls: list5.sort(key=lambda vertex: vertext, reverse=True) list5.sort(key=lambda vertex: degree(vertext)) – Brad Dre Nov 29 '17 at 22:07

I see why this is a very readable solution, is a very efficient solution also exists? Without coding it yourself, i.e. not computing the second value when unnecessary. – borgr Jan 8 '18 at 13:41

On a phone, but youcan sort by tuple.

3

```
sorted(list5, lambda x: (degree(x),x))
```

Don't forget the reverse flag if you need it.

edited Aug 19 '16 at 16:34
Razor
17.8k 7 45 72

answered Apr 24 '13 at 13:49
jay3686
41 4

5

From the Python 3 docs on sorting

```
from operator import itemgetter, attrgetter
student_objects = [
    Student('john', 'A', 15),
    Student('jane', 'B', 12),
    Student('dave', 'B', 10),
]
student_tuples = [
    ('john', 'A', 15),
    ('jane', 'B', 12),
    ('dave', 'B', 10),
]

#The operator module functions allow multiple levels of sorting. For example, to sort
by grade then by age:

sorted(student_tuples, key=itemgetter(1,2))
sorted(student_objects, key=attrgetter('grade', 'age'))
```

edited Jun 4 '16 at 21:40
Acey
7,582 4 26 46

answered Apr 12 '16 at 4:04
claudio
1,155 12 25

0

This post is hidden. It was deleted 6 years ago by the post author.

You've got it the wrong way round:

```
list5 = sorted(list5, key = lambda vertex: degree(vertex))
list5 = sorted(list5)
```

Note that this only works because Python `sort` is guaranteed to be *stable*; it guarantees not to change the relative order of elements that compare equal.

answered Apr 24 '13 at 13:46
ecatmur
132k 23 254 335

You've got to sort on your secondary key first. The degree is the primary key, and vertex id is the secondary id. Thus as it stands, you've got it backwards. – conradlee Aug 26 '14 at 0:04

comments disabled on deleted / locked posts / reviews

0

This post is hidden. It was deleted 7 years ago by the post author.

Python's `sorted` is guaranteed to be stable, thus this will work:

```
sorted_list = sorted(sorted(original_list, seconary_key), primary_key)
```

edited May 23 '17 at 12:26
Community ♦
1 1

answered Apr 24 '13 at 13:45
vartec
114k 32 198 236

comments disabled on deleted / locked posts / reviews

-1

This post is hidden. It was deleted 3 years ago by the post author.

You're doing it wrong, you need to do both comparisons at the same time.

If you first sort according to one comparison, and then another, only the latter will "survive", the first is completely ignored. Compare sorting an un-sorted list with any comparison function; of course you'd expect *none* of the randomness to remain.

The way to do it is pretty much as you describe, if the "primary" sort generates a collision, use the secondary sort to resolve it.

Something like:

```
def compare2(x, y):
  w = cmp(f(x), f(y))
  if w == 0:
    return cmp(x, y)
  return w

list5 = sorted(list5, cmp = compare2)
```

answered Apr 24 '13 at 13:44
unwind
355k 59 441 570

The key method is faster than the cmp method (which isn't available in Python 3). Also, as ecatmur points out, Python has a stable sort so you *can* do it in two steps. – Paused until further notice. Jan 14 '14 at 15:51

Inaccurate: Python's sort is stable. – Vivian Dec 7 '16 at 18:53

comments disabled on deleted / locked posts / reviews