

## Re: What about an EXPLICIT naming scheme for built-ins?

**Source:** <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-09/0654.html>

---

**From:** Carlos Ribeiro ([carribeiro\\_at\\_gmail.com](mailto:carribeiro_at_gmail.com))

**Date:** 09/03/04

Date: Fri, 3 Sep 2004 12:05:38 -0300  
To: Andrew Durdin <[adurdin@gmail.com](mailto:adurdin@gmail.com)>

On Sat, 4 Sep 2004 00:15:41 +1000, Andrew Durdin <[adurdin@gmail.com](mailto:adurdin@gmail.com)> wrote:

- > *However, I think it'd be hard to make an iterator implementation of*
- > *sorted() without either copying the sequence or resorting to  $O(n^2)$*
- > *performance, which is not good.*
- > *This may just be an example where inconsistency is a practical necessity.*

You may be right. At the least, this reasoning could possibly be included in the FAQ, as it will probably turn out to be a frequent question in the near future. But having `sorted()` to return a iterator still sounds like a logical thing to do, so I'm not sure about it.

So we now have a different question --- does it make sense to have `sort()` to return a iterator? What would be the advantages, and what does it take to implement it?

Today, `sorted()` returns a brand new sorted sequence. So it already takes the full memory and time penalty while doing so. However, **\*\*if `sorted()` was implemented as a generator\*\*, it would have some advantage in special cases:**

--- sorting long sequences to use only a few of the first elements. For example, if you need to sort a long list to get the top 10 elements. A generator would yield only those elements, saving a lot of time that would go into sorting everything else.

--- overall responsiveness in applications where `sort` is frequently called. Instead of waiting for the full sort at each `sorted()` call, the running time of the sorting method would be divided between the calls to the generator. Interactive and multithreaded applications can benefit of this approach (I'm not sure if `sorted()` grabs the GIL or not, but if it does, then it's definitely something to look at). (but then, again, it may be a poor design choice to use `sorted()` in these scenarios --- a heap or some other similar structure would be better suited. But I digress here).

comp.lang.python: Re: What about an EXPLICIT naming scheme for built-ins?

Modifying the sorting algorithm to work as a generator is not as hard as it seems. For example, Quicksort can be easily adapted. It's just a matter to yield the results as soon as the "left" partition is sorted. Actually, a similar technique can be used to yield the reversed sort (which is not the same as reversed(), incidentally). Other sorting algorithms can also be adapted.

p.s. I opted to send a copy of this answer to the list, I think it's good for the discussion. I hope you don't mind.

--

Carlos Ribeiro  
Consultoria em Projetos  
blog: <http://rascunhosrotos.blogspot.com>  
blog: <http://pythonnotes.blogspot.com>  
mail: [carribeiro@gmail.com](mailto:carribeiro@gmail.com)  
mail: [carribeiro@yahoo.com](mailto:carribeiro@yahoo.com)