

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

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-09/0629.html>

From: Carlos Ribeiro (carribeiro_at_gmail.com)

Date: 09/03/04

Date: Fri, 3 Sep 2004 10:33:29 -0300

To: ppntwimbxffc@spammotel.com

Mostly agreed. There is an inconsistency, as `sorted()` and `reversed()` should return both the same type of result – be it a sequence or a iterator.

I propose a slightly different approach. First, proposing a generic naming scheme for built-ins is an ambitious goal, to say the least. I suggest to keep the focus on this particular issue, if only to avoid a lot of debate and flaming. So — keeping in mind my own suggestion — I would like to focus on the particular case at hand:

1) `sorted()` and `reversed()` should return sequences. So `sorted()` stays like it, and `reversed()` meaning is changed. Now, that could potentially break a lot of code, but probably this is not going to happen — because in most situations, `reversed()` is getting called

2) add two new builtins, called respectively `xsorted()` and `xreversed()`, as the iterator versions of `sorted()` and `reversed()`. This way we keep the existing naming convention for `range()` and `xrange()`.

p.s. Please bear in mind that `sort()` and `reverse()` are methods, while `sorted()` and `reversed()` are builtin functions — which is a big difference that wasn't accounted for in your initial statement of the problem.

On Fri, 03 Sep 2004 15:10:48 +0200, Marco Aschwanden

ppntwimbxffc@spammotel.com wrote:

> *I just read the changes for 2.4 and while scanning the list the past tense*

> *built-ins got my attention:*

>

> *sorted() – a new builtin sorted() acts like an in-place list.sort() but*

> *can be used in expressions, as it returns a copy of the sequence, sorted.*

>

> *reversed() – a new builtin that takes a sequence and returns an iterator*

> *that loops over the elements of the sequence in reverse order (PEP 322)*

>

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

> *sort()* works in-place.
> *reverse()* works in-place.
>
> *The past tense of sort() indicates that a copy of the sequence is returned.*
> *The past tense of reverse() indicates that an iterator over the original*
> *sequence is returned.*
>
> *To my eyes it lacks a bit of consistency... well both past-tense functions*
> *return something (but different somethings although the concepts behind*
> *sorting and reversing are similar – a possible future trap for the faq).*
>
> *It is a common trap that sort() and reverse() work in-place. Any tutorial*
> *will warn you and still users fall into the trap over and over again...*
> *One should take this as a usability weakness of the language!*
>
> <opinion>
> *I would like to see Python introducing a naming scheme for built-ins. Ruby*
> *for example uses the ! to indicate an in-place function [sort() vs.*
> *sort!()]. I know, that the exclamation mark is out of discussion but I*
> *would appreciate to have a clear and distinct function naming (Explicit is*
> *better than implicit).*
>
> *An example but not very well thought out:*
>
> *sort_inpl() -> in-place returns nothing*
> *sort_copy() --> returns a sorted copy*
> *sort_iter() -> returns an iterator over the original sequence*
> *sort_copy_iter() -> returns an iterator on a copied and sorted sequence*
>
> </opinion>
>
> *What do you think about a naming scheme? Do you have any proposals/ideas?*
>
> *Regards,*
> *Marco*
>
> --
> <http://mail.python.org/mailman/listinfo/python-list>
>

--
Carlos Ribeiro
Consultoria em Projetos
blog: <http://rascunhosrotos.blogspot.com>
blog: <http://pythonnotes.blogspot.com>
mail: carribeiro@gmail.com
mail: carribeiro@yahoo.com