

What about an EXPLICIT naming scheme for built-ins?

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-09/0621.html>

From: Marco Aschwanden (*PPNTWIMBXFFC_at_spammotel.com*)

Date: 09/03/04

To: python-list@python.org

Date: Fri, 03 Sep 2004 15:10:48 +0200

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)

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

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

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