

Re: Feature suggestion: sum() ought to use a compensated summation algorithm

Re: Feature suggestion: sum() ought to use a compensated summation algorithm

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2008-05/msg00318.html>

- *From:* Szabolcs Horvát <szhorvat@xxxxxxxx>
 - *Date:* Sat, 03 May 2008 20:44:19 +0200
-

Arnaud Delobelle wrote:

sum() works for any sequence of objects with an `__add__` method, not just floats! Your algorithm is specific to floats.

This occurred to me also, but then I tried

```
sum(['abc', 'efg'], "")
```

and it did not work. Or is this just a special exception to prevent the misuse of sum to join strings? (As I said, I'm only an occasional user.)

Generally, sum() seems to be most useful for numeric types (i.e. those that form a group with respect to `__add__` and `__neg__`/`__sub__`), which may be either exact (e.g. integers) or inexact (e.g. floating point types). For exact types it does not make sense to use compensated summation (though it wouldn't give an incorrect answer, either), and sum() cannot decide whether a user-defined type is exact or inexact.

But I guess that it would still be possible to make sum() use compensated summation for built-in floating point types (float/complex).

Or, to go further, provide a switch to choose between the two methods, and make use compensated summation for float/complex by default. (But perhaps people would consider this last alternative a bit too messy.)

(Just some thoughts ...)

.