Ruby master - Bug #10475

Array#flatten should not accept a nil argument

11/04/2014 02:37 PM - dubek (Dov Murik)

Status:	Open		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	ruby 2.1.4p265 (2014-10-27 revision 48166) [x86_64-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

Currently Array#flatten accepts an explicit nil argument which has the same meaning as -1 (or no arguemnt) - meaning endless recursive flattening. I'd expect flatten to accept an integer argument only (or not at all, with a default of -1).

Moreover, the behaviour of arr.flatten(-1) is not described in the method's documentation.

Current behaviour (Ruby 2.1.4):

```
2.1.4 :001 > arr = [1, [2, [3, [4, 5]]]]
 => [1, [2, [3, [4, 5]]]]
2.1.4 :002 > arr.flatten
 => [1, 2, 3, 4, 5]
2.1.4 :003 > arr.flatten(-1)
 => [1, 2, 3, 4, 5]
2.1.4 :004 > arr.flatten(nil)
 => [1, 2, 3, 4, 5]
```

Expected behaviour:

- The last call (arr.flatten(nil)) should raise ArgumentError exception.
- The -1 (as "indefinite") should be documented.

Note that this suggestion will break applications/gems that rely on the fact that arr.flatten(nil) is identical to arr.flatten.

I'd like to hear your opinions on this. If accepted, I'm willing to try and write a patch.

History

#1 - 11/16/2014 02:22 PM - jaredbeck (Jared Beck)

flatten also accepts Float values, apparently using to_i (aka. truncate) to convert to Integer.

```
arr.flatten(0.9)
=> [1, [2, [3, [4, 5]]]]
arr.flatten(1.1)
=> [1, 2, [3, [4, 5]]]
arr.flatten(-0.9)
=> [1, [2, [3, [4, 5]]]]
arr.flatten(-1.1)
=> [1, 2, 3, 4, 5]
```

Given that to_i (or something like it) is used on Float values, it's odd that nil is treated as -1, rather than 0.

```
nil.to_i
=> 0
```

#2 - 11/23/2014 09:13 AM - jwmittag (Jörg W Mittag)

Dov Murik wrote:

> Currently Array#flatten accepts an explicit nil argument which has the same meaning as -1 (or no arguemnt) - meaning endless recursive flattening. I'd expect flatten to accept an integer argument only (or not at all, with a default of -1).

Jared Beck wrote:

> flatten also accepts Float values, apparently using to_i (aka. truncate) to convert to Integer.

IMO, the correct behavior would be to use to_int. Requiring Integer is too restrictive (an Integer-like object is perfectly fine), but to_i is too lax. The special casing of nil seems weird.

#3 - 11/24/2014 01:54 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to matz (Yukihiro Matsumoto)

Jörg W Mittag wrote:

> IMO, the correct behavior would be to use to_int.

It *is* using to_int, except for nil which is treated specially.

> The special casing of nil seems weird.

Right. The documentation lists two interfaces (one with no argument, one with a single argument), so it can be read to imply that nil is not an acceptable argument. Still, the interface to fill (which explicitly accepts nil) does a similar thing, probably to be more readable given its complexity.

FWIW, this behavior dates from 2007.

A couple of methods accept an explicit nil, like cycle, or fill

Matz, should we disallow nil as an argument, or accept it and change the documentation?

I'm guessing you will choose the latter, in which case, the doc could be changed like:

```
diff --git a/array.c b/array.c
index 04a8286..7820a13 100644
--- a/array.c
+++ b/array.c
@@ -4373,15 +4373,16 @@ flatten(VALUE ary, int level, int *modified)

 /*
  *  call-seq:
-*     ary.flatten!        -> ary or nil
-*     ary.flatten!(level) -> ary or nil
+*     ary.flatten!(recursion_limit=nil) -> ary or nil
  *
  *  Flattens +self+ in place.
  *
  *  Returns +nil+ if no modifications were made (i.e., the array contains no
  *  subarrays.)
  *
-*  The optional +level+ argument determines the level of recursion to flatten.
+*  The optional +recursion_limit+ argument determines maximum level of recursi
+*  to flatten. A value of 0 means no flattening will happen, while +nil+ or a
+*  negative number stand for no recursion limit.
```

I note in passing that the explanation for a returned value of nil is not completely accurate (i.e. the array contains no subarrays.), in the corner case when 0 is passed.

#4 - 11/25/2014 01:03 AM - nobu (Nobuyoshi Nakada)

I agree that this is a documentation issue.
But afraid that the negative value is worth to mention.