Description

(First of all, I understand that the proposed change can break code, but I expect it not to be a large amount empirically.)

I propose that methods that slice an array (#slice and []) and return a sub-array in the normal case, should **never** return nil. E.g.,

\[
\text{ary} = [1, 2, 3]
\]

- 1. Non-empty slice--how it works currently

\[
\text{ary}[1..2] \rightarrow [2, 3] \\
\text{ary}[1...-1] \rightarrow [2]
\]

- 2. Empty slice--how it works currently

\[
\text{ary}[1...1] \rightarrow [] \\
\text{ary}[3...] \rightarrow [] \\
\text{ary}[-1...-2] \rightarrow []
\]

- 3. Sudden nil--what I am proposing to change

\[
\text{ary}[4..] \rightarrow \text{nil} \\
\text{ary}[-10..-9] \rightarrow \text{nil}
\]

I believe that it would be better because the method would have cleaner "type definition" (If there is nothing in the array at the requested address, you'll have an empty array).

Most of the time, the empty array doesn't require any special handling; thus, ary[start...end].map { ... } will behave as expected if the requested range is outside of the array boundary.

It is especially painful with off-by-one errors; for an array of three elements, if ary[3...] (just outside the boundary) is [] while ary[4...] (one more step outside) is nil, it typically results in some nasty NoMethodError for NilClass.

A similar example is ary[1..].reduce { } (everything except the first element--probably the first element was used to construct the initial value for reducing) with ary being non-empty 99.9% of the times. Then you meet one of the 0.1% cases, and instead of no-op reducing nothing, NoMethodError is fired.

History

**#1 - 04/30/2020 12:07 PM - sawa (Tsuyoshi Sawada)**

- Description updated

**#2 - 04/30/2020 07:42 PM - shevegen (Robert A. Heiler)**

I do not have a strong preference here either way; I guess one can reason in favour for both behaviour types/styles, and I think a primary point in the suggestion is that it refers to startless/endless situations, such as "5..", which I don't use myself, but one slight concern is this one:

\[
\text{ary}[-1...-2] \rightarrow [] \\
\text{ary}[-10...-9] \rightarrow \text{nil}
\]

Is this certain to not break a lot of code? I have not checked myself and I rarely use #slice anyway, but I do use a lot of [] in general. It's one of my favourite method calls in general, in ruby. :)

Admittedly I actually don't remember off-hand having ever used two negative
indices here ... for some reason, I seem to use 0 or positive numbers a lot more.

No idea how/if other ruby users use or rely on that behaviour though but I think it would be important to get some specific overview about any potential effect (or side-effect) of proposed changes, even if the reasoning given is ok.