Ruby master - Feature #12262

Anti-loop

04/08/2016 07:21 AM - sawa (Tsuyoshi Sawada)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

The loop method continues by default, and requires the keyword break to escape. This is good when the continuing cases are the norm and the escaping cases are exceptional:

```
loop do
  ...
  if ...
    ...
  elsif ...
    ...
  elsif ...
    ...
    break # breaks on exceptional cases
  elsif ...
    ...
  else
    ...
  end
end
```

But when the continuing cases are exceptional and the escaping cases are the norm, the construction requires a lot of break, and it becomes cumbersome:

```
loop do
  ...
  if ...
    ...
    break # lot of breaks
  elsif ...
    ...
    break # lot of breaks
  elsif ...
    ...
    break # lot of breaks
  elsif ...
    ...
  else
    ...
    break # lot of breaks
  end
end
```

I actually see this use case a lot when user input is asked with validation on a command line script.

I request a loop-like method that works in the opposite way to loop, that is, it escapes (i.e., runs only once) by default, and requires a keyword to continue (perhaps next). The second code above would then be written like:

```
some_loop_like_method do
  ...
  if ...
    ...
  elsif ...
    ...
  elsif ...
    ...
```

```
  elsif ...
    ...
    next # continues on exceptional cases
  else
    ...
  end
end
```

History

#1 - 04/08/2016 09:16 AM - jwille (Jens Wille)

You can make your last example work with loop by just adding a break at the end of the loop body. I don't think that warrants a new method.

#2 - 04/09/2016 05:30 AM - shevegen (Robert A. Heiler)

I don't have any big pro or contra opinion, but there is one thing I am wondering:

Is this still called a loop in the second case? Because the default is to break after the first run. And a loop implies to continue, until one ends it or? :-)

However had, what I might find interesting, is to have other means to force the end of a loop. Like, "break" is used, but what if we could designate another way to end a loop? If we could do that, then perhaps your suggestion might be implied to work in the second case, because you could somehow specify that loop would en when a "return nil" would be implied.

E. g. something like (the syntax does not work, it just is an example):

```
loop(break_on: nil) {
  if ...
    ...
  elsif ...
    ...
  elsif ...
    ...
    next # continues on exceptional cases
  else
    ...
  end
}
```

Where the default would be a loop like:

```
loop(break_on: :break)
```

Which can be omitted. (The symbol :break would then default on the keyword break).

Please consider this just as food-for-thought, I actually do not really suggest it - I am just playing with the thought here. :)

(I myself probably would prefer "the simpler, the better" which is why I do not suggest a change, but as said, I am neutral on this suggestion.)