Loops (while/until) should return last statement value if any, like if/unless

05/02/2010 04:14 AM - Eregon (Benoit Daloze)

Status: Rejected

Priority: Normal

Assignee:

Target version: 2.0.0

Description

```
In conditions:
if true
  2
end #=> 2
```

```
In loops:
i = 0
while i < 3
  i += 1
  "return value"
end #=> nil
```

I suggest the return value of loops to be the return value of
the last "block" evaluated between "while/until" and "end".

It would be nil if the condition is not met from the first evaluation.

Is this reasonable to implement?

It would come very handy in situations like:

```
while condition
  ...
  my_method_return_value
end
```

because you could remove the last line.

The idea came up on ruby-talk: [http://www.ruby-forum.com/topic/208978](http://www.ruby-forum.com/topic/208978)

Any thoughts?

Regards,
B.D.

```
=end
```

History

#1 - 05/02/2010 08:57 AM - matz (Yukihiro Matsumoto)

```
=begin
Hi,

In message "Re: [ruby-core:29920] [Feature #3232] Loops (while/until) should return last statement value if any, like if/unless" on Sun, 2 May 2010 04:14:10 +0900, Benoit Daloze redmine@ruby-lang.org writes:

|In conditions:
|if true
|  2
|end #=> 2
|
|In loops:
|i = 0
|while i < 3
|  i += 1
```
I suggest the return value of loops to be the return value of the last "block" evaluated between "while/until" and "end". The last evaluated expression in this case is "i < 3", so the loop statement would have the return value of false value, even if it would return the last value.

matz.

#2 - 05/02/2010 01:40 PM - murphy (Kornelius Kalnbach)

=begin
On 02.05.10 01:56, Yukihiro Matsumoto wrote:

The last evaluated expression in this case is "i < 3", so the loop statement would have the return value of false value, even if it would return the last value.
So, shouldn't it return false then?

[murphy]
=begin

#3 - 05/02/2010 02:16 PM - tmat (Tomas Matousek)

=begin
It returns the argument of break:

x = while true; break 123; end #=> 123

If you don't break out of the loop it makes sense that the result is nil.

Tomas

-----Original Message-----
From: Kornelius Kalnbach [mailto:murphy@rubychan.de]
Sent: Saturday, May 01, 2010 9:40 PM
To: ruby-core@ruby-lang.org
Subject: [ruby-core:29924] Re: [Feature #3232] Loops (while/until) should return last statement value if any, like if/unless

On 02.05.10 01:56, Yukihiro Matsumoto wrote:

The last evaluated expression in this case is "i < 3", so the loop statement would have the return value of false value, even if it would return the last value.
So, shouldn't it return false then?

[murphy]
=begin

#4 - 05/02/2010 07:20 PM - Eregon (Benoit Daloze)

=begin
On 2 May 2010 01:56, Yukihiro Matsumoto matz@ruby-lang.org wrote:

Hi,

The last evaluated expression in this case is "i < 3", so the loop statement would have the return value of false value, even if it would return the last value.

matz.

My bad, I forgot about condition evaluation,
But that result is really not interesting, as it will always be false (or true for until).
Is it possible, then, to keep the result of the last expression in the loop, before it fails?

It returns the argument of break:

```ruby
x = while true; break 123; end #=> 123
```

If you don't break out of the loop it makes sense that the result is nil.

Sure, so a way to solve my example would be:

```ruby
i = 0
loop do
  i += 1
  break "return value" unless i < 3
end #=> "return value"
```

But there are a few disadvantages to use this way:

- using 'loop' and 'break', while you perfectly now when you want to stop
- therefore you need an 'unless', or the negative condition of what you expected to write after while/until
- you need to specify again explicitly the return value to break, so you don't use any shortcut.

But my example is not clear, let's see this one:

```ruby
def method
  array = []
  while
    array << do_sth
    # With loop & break, here would be: break array unless
    end
  array # I would like to remove this "superfluous" statement
end
```

This would be a nice feature, and would remove the need to explicitly mention the return value, again after the loop.

I think it would be more consistent with if/unless (even if that case is simpler because it is actually the last evaluated expression).

It would be a kind of inject/each_with_object, but with a condition instead of an Enumerable (and no initial value).

B.D.

```ruby
=end
```

---

**#5 - 05/02/2010 10:25 PM - nobu (Nobuyoshi Nakada)**

`=begin`

Hi,

At Sun, 2 May 2010 19:20:41 +0900, Benoit Daloze wrote in [ruby-core:29931]:

> Is is possible, then, to keep the result of the last expression in the loop, before it fails?

Possible.

[http://github.com/nobu/ruby/commit/b83915cdda5645d57f83947bc948abe76fcae59e0](http://github.com/nobu/ruby/commit/b83915cdda5645d57f83947bc948abe76fcae59e0)

```ruby
def method
  array = []
  while
    array << do_sth
    # With loop & break, here would be: break array unless
    end
  array # I would like to remove this "superfluous" statement
end
```

This would be a nice feature, and would remove the need to explicitly mention the return value, again after the loop.
It feels too confusing to me.

--
Nobu Nakada

#6 - 05/03/2010 05:10 AM - Eregon (Benoit Daloze)

On 2 May 2010 15:24:52 UTC+2, Nobuyoshi Nakada nobu@ruby-lang.org wrote:

It feels too confusing to me.

Nobu Nakada

Thanks for the patch, Nobu.

So, how would this cause unexpected behavior?

Let's say 'ret' is the return value of the last statement in the body of the loop.

- If 'break' is not used in the loop (while/until), it will now result in 'ret', or false if never evaluated.

Before, it always returned nil, which is not interesting and should never have been used.
So that case is not relevant I think.

- if 'break' is used, it can return 'ret', if the condition of the loop is met before break's condition.

So:

```ruby
i = 0
ret = while i < 3
  i += 1
  break "break" if i > 3
  "loop return value"
end #=> ret = "loop return value" instead of nil
```

Would someone check for 'ret' to be nil?

Is there any real use case of using 'ret' (which would be nil) in this situation?

I think if you don't break, you would not use any value depending on the return value of the loop.

Please say if you think this can break compatibility, and of course any opinion is appreciated!

Regards,
B.D.

#7 - 05/03/2010 09:15 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:29940] Re: [Feature #3232] Loops (while/until) should return last statement value if any, like if/unless" on Mon, 3 May 2010 05:10:30 +0900, Benoit Daloze eregontp@gmail.com writes:

|So, how would this cause unexpected behavior? |

It is neither consistent (other statements returns the value from the "last" expression), nor useful (by returning nil, you can use the value from break safely), nor compatible.

matz.

#8 - 05/04/2010 03:07 AM - Eregon (Benoit Daloze)
Hi,

On 3 May 2010 02:15, Yukihiro Matsumoto matz@ruby-lang.org wrote:

Hi,

It is neither consistent (other statements returns the value from the "last" expression),

Actually it is not, but here the value of the "last" expression is always false/true, so really not interesting
(and it returns nil, so it's neither consistent now about that)

nor useful (by returning nil, you can use the value from break safely), nor compatible.

That's the problem.
The idea was: I think nobody wrote code using the fact it is nil if the loop did not break.

But it is useful:
it adds a useful return value to the loop,
which it does not have (nil), except if 'break' is used.

matz.

I suppose you've got the final point, and I do agree it can be confusing/inconsistent.

My idea was more of a thought, but the result is interesting though (and could be useful for me (and some others I think)).

Special thanks to Nobu,
and thanks for sharing your thoughts.

Regards,
B.D.

#9 - 05/04/2010 06:06 AM - marcandre (Marc-Andre Lafortune)
- Status changed from Open to Rejected

=begin

=end