Ruby master - Bug #14824

Endless Range Support in irb

06/04/2018 07:26 PM - jeremyevans0 (Jeremy Evans)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.0preview2 (2018-05-31 trunk 63539) [x86_64-openbsd]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

irb currently doesn't have great support for endless ranges, forcing you to use explicit parentheses around the endless range. Without explicit parentheses, it treats the endless range as a line continuation.

```
irb(main):001:0> 1..
irb(main):002:0* ;
=> 1..
irb(main):003:0> (1..)
=> 1..
irb(main):004:0>
```

Ranges with ends do not require parentheses in irb, and endless ranges should have the same behavior.

Related issues:

Related to Ruby master - Feature #14808: Last token of endless range should h...	**Rejected**

History

#1 - 06/04/2018 11:04 PM - shevegen (Robert A. Heiler)

Agreed.

#2 - 06/05/2018 06:26 AM - aycabta (aycabta .)

- Related to Feature #14808: Last token of endless range should have EXPR_END added

#3 - 06/05/2018 06:28 AM - aycabta (aycabta .)

- Status changed from Open to Rejected

It's a correct behavior. This ticket is the same arguments for #14808.

#4 - 06/05/2018 07:32 AM - mame (Yusuke Endoh)

First of all, I have no strong opinion about this issue. The current behavior of irb looks weird to me, but it may be okay because irb users can work around the issue easily.

In principle, I expect irb to cut the shortest lines that parses. For example, consider the following input:

```
foo
.bar
```

Irb first evaluates only the first line foo, and the second line .bar causes SyntaxError. This is the behavior that I expect. In the same logic, I expect 1.. to be cut because it parses.

```
irb(main):001:0> 1..
```

BTW, a line obj.+ causes line continuation. This is a behavior that I don't expect:

```
irb(main):001:0> obj = Object.new
=> #<Object:0x0000560275fa4b00>
irb(main):002:0> def obj.+(); 42; end
=> :+
irb(main):003:0> obj.+
irb(main):004:0*
irb(main):005:0* ;
=> 42
```

If the method name is different than +, say foo, irb does not wait for the next line. Looks very inconsistent.

```
irb(main):006:0> def obj.foo(); 42;end
=> :foo
irb(main):007:0> obj.foo
=> 42
```

So, I cannot see any consistent policy about line continuation.

But again, it's okay to me in the case of irb. Few users will encounter such a corner case. Even if someone does, s/he can avoid such a behavior by explicit semicolon or parens or something else.

#5 - 06/05/2018 10:45 AM - aycabta (aycabta .)

- Status changed from Rejected to Open

mame (Yusuke Endoh) wrote:

> In principle, I expect irb to cut the shortest lines that parses.

Ah, well, you got that right. I understand an importance of this issue. So I re-open this.

> BTW, a line obj.+ causes line continuation. This is a behavior that I don't expect:
>
> ```
> irb(main):001:0> obj = Object.new
> => #<Object:0x0000560275fa4b00>
> irb(main):002:0> def obj.+(); 42; end
> => :+
> irb(main):003:0> obj.+
> irb(main):004:0*
> irb(main):005:0* ;
> => 42
> ```

It will be fixed by https://bugs.ruby-lang.org/issues/14683, but this issue will not. Therefore, I guess that https://bugs.ruby-lang.org/issues/14808 is needed for this issue too.