Upon more testing apparently this has been fixed in 3.0.2p107 (2021-07-07 revision 0db68f0233).

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>ruby -v: 3.0.0p0 (2020-12-25 revision 95aff21468)</td>
</tr>
</tbody>
</table>

### Description

These results with lazy enumerators are as expected:

```ruby
(2..10).take(0).to_a # => []
(2..10).take(0).map(&:itself).to_a # => []
(2..10).lazy.take(0).to_a # => []
```

However, once another operation is added after take(0), if first element will pass through directly it will leak through:

```ruby
(2..10).lazy.take(0).map(&:itself).to_a # => [2]
(2..10).lazy.take(0).select(&:even?).to_a # => [2]
(2..10).lazy.take(0).select(&:odd?).to_a # => []
(2..10).lazy.take(0).reject(&:even?).to_a # => []
(2..10).lazy.take(0).reject(&:odd?).to_a # => [2]
(2..10).lazy.take(0).take(1).to_a # => [2]
(2..10).lazy.take(0).take(0).take(1).to_a # => [2]
(2..10).lazy.take(0).drop(0).to_a # => [2]
(2..10).lazy.take(0).find_all { |_| true}.to_a # => [2]
(2..10).lazy.take(0).zip((12..20)).to_a # => [(2, 12)]
(2..10).lazy.take(0).uniq.to_a # => [2]
(2..10).lazy.take(0).sort.to_a # => []
(2..2).lazy.take(0).sort.to_a # => []
```

Non lazy versions all return [] as expected.

In 3.1.0 All of them behave as expected as well:

```ruby
(2..10).lazy.take(0).map(&:itself).to_a # => []
(2..10).lazy.take(0).select(&:even?).to_a # => []
(2..10).lazy.take(0).select(&:odd?).to_a # => []
(2..10).lazy.take(0).reject(&:even?).to_a # => []
(2..10).lazy.take(0).reject(&:odd?).to_a # => []
(2..10).lazy.take(0).take(1).to_a # => []
(2..10).lazy.take(0).take(0).take(1).to_a # => []
(2..10).lazy.take(0).drop(0).to_a # => []
(2..10).lazy.take(0).find_all { |_| true}.to_a # => []
(2..10).lazy.take(0).zip((12..20)).to_a # => []
(2..10).lazy.take(0).uniq.to_a # => []
(2..10).lazy.take(0).sort.to_a # => []
(2..2).lazy.take(0).sort.to_a # => []
```

### History

#### #1 - 08/23/2022 03:36 AM - Voileexperiments (Library Voile)
- ruby -v changed from 3.0.0 to 3.0.0p0 (2020-12-25 revision 95aff21468)

#### #2 - 08/23/2022 03:52 AM - Voileexperiments (Library Voile)
- Description updated

Upon more testing apparently this has been fixed in 3.0.2p107 (2021-07-07 revision 0db68f0233).
However I can't find a commit between p0 and p107 related to this change.

#3 - 08/23/2022 05:58 AM - jeremyevans0 (Jeremy Evans)
Are you sure this has been fixed? I tried with Ruby 2.7.3.1 and current master and did not get all [] results on any Ruby version I tested.

There is specific code to deal with an argument of 0, though I don't understand what it does. It comes from commit 29f7309ca934cfaf7b51d1de4d22933ab56dc602.

One way I found to work around this is to use an intermediate cycle(0) enumerator. I submitted a pull request for that (https://github.com/ruby/ruby/pull/6273), though I expect @nobu (Nobuyoshi Nakada) could come up with a more elegant solution.

#4 - 08/23/2022 06:44 AM - Voileexperiments (Library Voile)
You're right. I just tested again and apparently even 3.1.2p20 has this issue.

#5 - 08/23/2022 06:44 AM - Voileexperiments (Library Voile)
- Description updated

#6 - 08/23/2022 06:47 AM - Voileexperiments (Library Voile)
- Description updated

#7 - 08/25/2022 05:35 AM - nobu (Nobuyoshi Nakada)
Although not so elegant, I was thinking to add "precheck" to enumerators. https://github.com/nobu/ruby/tree/lazy_take0