I'd like to add a new method, Enumerable#slice_between.

It is similar to Enumerable#slice_before but it can use not only the element after the slice position but also the element before the slice position.

```ruby
enum.slice_between(pattern_before, pattern_after=nil) -> an_enumerator
enum.slice_between {{elt_before, elt_after| bool} -> an_enumerator
```

I found several people try to use Enumerable#slice_before for compacting sequence of integers using hyphens:

1,2,4,9,10,11,12,15,16,19,20,21 to 1,2,4,9-12,15,16,19-21.

- [ruby-talk:370132](http://d.hatena.ne.jp/keyesberry/20120107/p1) Dave Thomas and James Edward Gray II
- [ruby-talk:359255](http://stackoverflow.com/questions/6258971/how-do-i-return-a-group-of-sequential-numbers-that-might-exist-in-an-array) split logs where interval is 30s or more.
- [ruby-talk:415057](http://stackoverflow.com/questions/6258971/how-do-i-return-a-group-of-sequential-numbers-that-might-exist-in-an-array) collects same elements. (Enumerable#chunk can be used, though.)

Also, I found several usages for Enumerable#slice_between.

- [ruby-talk:415057](http://stackoverflow.com/questions/6258971/how-do-i-return-a-group-of-sequential-numbers-that-might-exist-in-an-array)

Any idea?
Revision 47652 - 09/20/2014 06:52 AM - akr (Akira Tanaka)

- enum.c (enum_slice_when): New method: Enumerable#slice_when.
  (slicewhen_1): New function.
  (slicewhen_2): New function.

- enumerator.c (InitVM_Enumerator): New method:
  Enumerator::Lazy#slice_when.

Revision 47652 - 09/20/2014 06:52 AM - akr (Akira Tanaka)

- enum.c (enum_slice_when): New method: Enumerable#slice_when.
  (slicewhen_1): New function.
  (slicewhen_2): New function.

- enumerator.c (InitVM_Enumerator): New method:
  Enumerator::Lazy#slice_when.

Revision 47652 - 09/20/2014 06:52 AM - akr (Akira Tanaka)

- enum.c (enum_slice_when): New method: Enumerable#slice_when.
  (slicewhen_1): New function.
  (slicewhen_2): New function.

- enumerator.c (InitVM_Enumerator): New method:
  Enumerator::Lazy#slice_when.

Revision 47652 - 09/20/2014 06:52 AM - akr (Akira Tanaka)

- enum.c (enum_slice_when): New method: Enumerable#slice_when.
  (slicewhen_1): New function.
  (slicewhen_2): New function.

- enumerator.c (InitVM_Enumerator): New method:
  Enumerator::Lazy#slice_when.
Enumerator::Lazy#slice_when.

[ruby-core:62499] [Feature #9826]

Revision 47652 - 09/20/2014 06:52 AM - akr (Akira Tanaka)

- enum.c (enum_slice_when): New method: Enumerable#slice_when.
  (slicewhen_j): New function.
  (slicewhen_ii): New function.

- enumerator.c (InitVM_Enumerator): New method:
  Enumerator::Lazy#slice_when.

[ruby-core:62499] [Feature #9826]

History

#1 - 05/12/2014 09:47 AM - akr (Akira Tanaka)
- File slice_between2.patch added

I updated the patch to simplify argument handling.

#2 - 05/17/2014 06:26 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Feedback

I understand the use-case, and I'd like to add the feature.
But slice_between does not describe what it does.
Try another name.

Matz.

#3 - 05/18/2014 04:09 AM - akr (Akira Tanaka)
- File slice_between3.patch added

I updated the patch to be applied cleanly for HEAD after slice_after merge.

#4 - 05/18/2014 04:11 AM - akr (Akira Tanaka)

Yukihiro Matsumoto wrote:

But slice_between does not describe what it does.
Try another name.

I searched "between" with wordnet.

% wordnet between -over

Overview of adv between

The adv between has 2 senses (first 2 from tagged texts)

1. (1) between, betwixt -- (in the interval; "dancing all the dances with little rest between")
2. (1) between, 'tween -- (in between; "two houses with a tree between")

I agree that enum.slice_between(X,Y) doesn't behave as "in the interval".
There is no interval between X and Y.

However I feel the 2nd meaning is appropriate.
So, slice_between itself may be possible.
slice_in_between may be more clear.

Some idea:

- slice_between
- slice_in_between
I'd like to ask others (especially English native speakers).

Any opinion?

**#5 - 05/18/2014 05:35 AM - sawa (Tsuyoshi Sawada)**

There can be another method that works the opposite to the proposed method with respect to the truthfulness of the block, and I think the two methods should come in a pair. And I have a feeling that cons should somehow be used in the name of these methods. For example, the method implemented by Akira Tanaka may be called slice_cons, and the opposite one may be called chunk_cons, and the following could be equivalent:

```
[1,2,4,9,10,11,12,15,16,19,20,21].slice_cons{|i, j| i+1 != j}
[1,2,4,9,10,11,12,15,16,19,20,21].chunk_cons{|i, j| i+1 == j}
```

But by analogy from the method each_cons, the cons in the names may imply that the arity of the block need not be two. I have no clear idea how this feature can be extended to take a block of other arity, and if the implication is a problem, then the word pair might work better:

```
[1,2,4,9,10,11,12,15,16,19,20,21].slice_pair{|i, j| i+1 != j}
[1,2,4,9,10,11,12,15,16,19,20,21].chunk_pair{|i, j| i+1 == j}
```

**#6 - 09/04/2014 12:08 PM - mrkn (Kenta Murata)**

```
[1,2,4,9,10,11,12,15,16,19,20,21].slice_when{|i, j| i+1 != j}
```

**#7 - 09/04/2014 03:07 PM - matz (Yukihiro Matsumoto)**

I prefer #slice_when. Besides that, could you explain the behavior when no block is given? #slice_when might not suitable for that calling pattern. But maybe we don't need that.

Matz.

**#8 - 09/05/2014 05:35 AM - akr (Akira Tanaka)**

I'm grad to see an acceptable name.

Non-block form can be used to split paragraphs (sequence of non-empty lines with trailing empty lines), for example.

```ruby
$ ruby -e 'lines = [:foo
, "bar
", "\n", "baz
", "\n", "\n", "qux
", "quux\n"]
lines.slice_when(/\A\s*\z/, /\S/).each{|para| p para}

["foo
", "bar
", "\n"]
["baz
", "\n", "\n"]
["qux
", "quux\n"]
```

If the name, slice_when, is not appropriate to this form, I can drop the form.

The block form can same task as follows. (It is bit longer, of course.)

```ruby
$ ruby -e 'lines = [:foo
, "bar
", "\n", "baz
", "\n", "\n", "qux
", "quux\n"]
lines.slice_when{|l1, l2| /\A\s*\z/ =~ l1 && /\S/ =~ l2}.each{|para| p para}

["foo
", "bar
", "\n"]
["baz
", "\n", "\n"]
["qux
", "quux\n"]
```

**#9 - 09/18/2014 04:30 AM - akr (Akira Tanaka)**

- File slice_when.patch added

I updated the patch at today's lunch break: slice_when.patch

It defines Enumerable#slice_when and it doesn't support non-block form.

**#10 - 09/20/2014 06:30 AM - matz (Yukihiro Matsumoto)**

- Status changed from Feedback to Open

- Assignee set to akr (Akira Tanaka)

Agreed with slice_when. Gi ahead.
enum.c (enum_slice_when): New method: Enumerable#slice_when.
  (slicewhen_i): New function.
  (slicewhen_ii): New function.

enumerator.c (InitVM_Enumerator): New method:
  Enumerator::Lazy#slice_when.

[ruby-core:62499] [Feature #9826]

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice_between.patch</td>
<td>10 KB</td>
<td>05/10/2014</td>
<td>akr (Akira Tanaka)</td>
</tr>
<tr>
<td>slice_between2.patch</td>
<td>9.7 KB</td>
<td>05/12/2014</td>
<td>akr (Akira Tanaka)</td>
</tr>
<tr>
<td>slice_between3.patch</td>
<td>9.71 KB</td>
<td>05/18/2014</td>
<td>akr (Akira Tanaka)</td>
</tr>
<tr>
<td>slice_when.patch</td>
<td>7.49 KB</td>
<td>09/18/2014</td>
<td>akr (Akira Tanaka)</td>
</tr>
</tbody>
</table>