Ruby - Feature #9116

String#rsplit missing

11/16/2013 05:35 PM - Anonymous

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

There's nothing corresponding to Python's rsplit(). A quick glance at rb_str_split_m() tells me that it should be pretty trivial to implement. Is there any specific reason it hasn't already been done?

History

#1 - 11/16/2013 07:23 PM - phluid61 (Matthew Kerwin)

On Nov 16, 2013 6:35 PM, "artagnon (Ramkumar Ramachandra)" <
artagnon@gmail.com> wrote:

> There's nothing corresponding to Python's rsplit(). A quick glance at rb_str_split_m() tells me that it should be pretty trivial to implement. Is there any specific reason it hasn't already been done?

What is rsplit? How does it differ from split (with or without a second paramater)? What is the use-case/demand for it?

Sent from my phone, so excuse the typos.

#2 - 11/17/2013 06:05 PM - alexeymuranov (Alexey Muranov)

Out of curiosity, i have looked it up: http://docs.python.org/3/library/stdtypes.html#str.rsplit

#3 - 11/19/2013 10:00 AM - shevegen (Robert A. Heiler)

I am still not sure how it differs from #split().

#4 - 11/19/2013 10:01 AM - shevegen (Robert A. Heiler)

Oh, now I see:

"Except for splitting from the right, rsplit() behaves like split() which is described in detail below."

So it basically splits from the right, not left, unlike #split() which splits from the left, I think.

I suppose in Ruby you can do .reverse.split, but perhaps rsplit may be more convenient. (I myself don't think I have had a need to use something similar to rsplit so far).

#5 - 11/19/2013 11:05 AM - phluid61 (Matthew Kerwin)

I, too, looked up and read the documentation, a couple of times.

I understand that the difference only applies when a limit parameter is given, and so examples of the new API would be:

```
'a.b.c'.rsplit('.')    #=> ["a", "b", "c"], same as #split
'a.b.c'.rpslit('.', 2) #=> ["a.b", "c"]
```

I would want to clarify some of the other edge cases (from String#split) before continuing:

- If pattern is a String, then its contents are used as the delimiter when splitting str. If pattern is a single space, str is split on whitespace, with leading whitespace and runs of contiguous whitespace characters ignored.

Would this have some right-handed equivalent in #rsplit? E.g. "...with trailing whitespace and runs..."? Or would it remain the same as #split? Or some third option?

E.g.:

```
' x  y '.rsplit(' ')    #=> ["x", "y"], same as split?
' x  y '.split(' ',-1) #=> ["", "x", "y"] or ["x", "y", ""] or ..?
```

- If the limit parameter is omitted, trailing null fields are suppressed. If limit is a positive number, at most that number of fields will be returned (if limit is 1, the entire string is returned as the only entry in an array). If negative, there is no limit to the number of fields returned, and trailing null fields are not suppressed.

Similarly, would this become: "...leading null fields..." in both instances?

E.g.:

```
'..x..'.rsplit('.')    #=> ["x", "", ""] or ["", "", "x"] or ..?
'..x..'.rsplit('.',-1) #=> ["", "", "x", "", ""], same as #split?
```

Note that this would be another difference from #split, which *doesn't* depend on the limit parameter.

Seems like a lot of work. What is the demand for this feature?

#6 - 11/19/2013 07:25 PM - alexeymuranov (Alexey Muranov)

phluid61 (Matthew Kerwin) wrote:

> I understand that the difference only applies when a limit parameter is given

It is not only when limit parameter is given:

```
"aaa".split("aa")  # => ["", "a"]
"aaa".rsplit("aa") # => ["a", ""]
```

Maybe with a regex there can be a more meaningful example.

#7 - 11/19/2013 07:37 PM - alexeymuranov (Alexey Muranov)

phluid61 (Matthew Kerwin) wrote:

> Would this have some right-handed equivalent in #rsplit? E.g. "...with trailing whitespace and runs..."? Or would it remain the same as #split? Or some third option?

IMO, if it is introduced, i would say it must be completely symmetric with split.

#8 - 11/19/2013 07:44 PM - phluid61 (Matthew Kerwin)

alexeymuranov (Alexey Muranov) wrote:

> It is not only when limit parameter is given:
>
> ```
> "aaa".split("aa")  # => ["", "a"]
> "aaa".rsplit("aa") # => ["a", ""]
> ```

Ah, I see. Thank you.

> Maybe with a regex there can be a more meaningful example.

I'm interested to see how it would be achieved with regex, from an implementation point of view.

#9 - 01/29/2017 06:26 PM - stomar (Marcus Stollsteimer)

I'd like to revive the discussion about String#rsplit.

Here one use case I stumbled upon recently: splitting the digest off the end of a cookie (taken from Rack::Session::Cookie, see https://github.com/rack/rack/blob/master/lib/rack/session/cookie.rb#L139):

```
session_data = "session--data--digest"

digest, session_data = session_data.reverse.split("--", 2)
digest.reverse!        if digest
session_data.reverse!  if session_data

session_data  # => "session--data"
digest        # => "digest"
```

Note that each substring needs to be reversed back (for higher limits this would probably be done using #map), which seems inefficient and unhandy.

With rsplit this would become:

```
session_data = "session--data--digest"

session_data, digest = session_data.rsplit("--", 2)
```

#10 - 03/13/2017 03:10 PM - shyouhei (Shyouhei Urabe)

We looked at this issue in today's developer meeting.

Someone there pointed out that in general, rsplit(..., 2) could be avoided by using regexp, because you can match $ (for instance, you can match /(.+)--([^-]+)$/ for the session cookie). So the cookie case is a bit short for us to add new method.

It might make sense when the limit is bigger than 2. Is there such example?

#11 - 04/27/2017 10:00 AM - naruse (Yui NARUSE)

I find a use case.

```
test_priv=# SELECT * FROM pg_database;
       datname        | datdba | encoding | datcollate  |  datctype   | datistemplate | datallowconn | dat
connlimit | datlastsysoid | datfrozenxid | datminmxid | dattablespace |                datacl
----------------------+--------+----------+-------------+-------------+---------------+--------------+----
----------+---------------+--------------+------------+---------------+-----------------------------------
------
 test_prev             |  16384 |        6 | en_US.UTF-8 | en_US.UTF-8 | f             | t            |
       -1 |         12668 |         1917 |           1 |          1663 | {=Tc/pg,pg=CTc/pg,"\"foo=bar\"=CTc
/pg"}
```

On PostgreSQL, I can retrieve databases information by SQL, and its ACL data is "datacl" column as key-value.
Its key is rolename (user/group name) and the value is the privilege, with separated "=".
The rolename may include "=" as above.