Ruby trunk - Feature #5120

String#split needs to be logical

07/31/2011 01:12 AM - alexeymuranov (Alexey Muranov)

Status:	Rejected
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	Next Major

Description

I would call this a bug, but i am new to Ruby, so i report this as a feature request.

Here are examples showing a surprising and inconsistent behavior of String#split method:

```
"aa".split('a')  # => []
"aab".split('a')  # => ["", "", "b"]

"aaa".split('aa')  # => ["", "a"]
"aaaa".split('aa')  # => []
"aaaaa".split('aa')  # => ["", "", "a"]

"".split('')  # => []
"a".split('')  # => ["a"]
```

What is the definition of *split*?
In my opinion, there should be given a simple one that would make it clear what to expect.
For example:

str1.split(str2) returns a maximal array of non-empty substrings of str1 which can be concatenated with copies of str2 to form str1.

Additional precisions can be made to this definition to make clear what to expect as the result of "baaab".split("aa").

Thanks for attention.

Related issues:

Related to Ruby trunk - Feature #3575: String#split is inconsistent with empt...	**Closed**	**07/15/2010**

History

#1 - 07/31/2011 01:53 AM - mame (Yusuke Endoh)

Hello,

2011/7/31 Alexey Muranov muranov@math.univ-toulouse.fr:

> Â str1.split(str2) returns a maximal array of non-empty substrings of str1 which can be concatenated with copies of str2 to form str1.

So, what does "aab".split('a') return?
I think that only ["aab"] meets the condition.
But it is also surprising to me.

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 07/31/2011 01:53 AM - mame (Yusuke Endoh)

2011/7/31 Yusuke ENDOH mame@tsg.ne.jp:

> Hello,
>
> 2011/7/31 Alexey Muranov muranov@math.univ-toulouse.fr:
>
> > Â str1.split(str2) returns a maximal array of non-empty substrings of str1 which can be concatenated with copies of str2 to form str1.
>
> So, what does "aab".split('a') return?
> I think that only ["aab"] meets the condition.

> But it is also surprising to me.

Oops. It should be ["a", "b"]. But it is very difficult (to me) behavior.

--
Yusuke Endoh mame@tsg.ne.jp

#3 - 07/31/2011 04:23 AM - Anonymous

On Jul 30, 2011, at 12:12 PM, Alexey Muranov wrote:

> Here are examples showing a surprising and inconsistent behavior of String#split method:

With only one argument, split discards trailing empty fields.

```
"aa".split('a')  # => []
```

<empty><a><empty><a><empty> => three empty fields all trailing and discarded

```
"aab".split('a')  # => ["", "", "b"]
```

<empty><a><empty><a><b> => three fields with no empty trailing fields

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

<empty><separator><a> => two fields with no empty trailing field

```
"aaaa".split('aa')  # => []
```

<empty><aa><empty><aa><empty> => all empty fields, all discarded

```
"aaaaa".split('aa')  # => ["", "", "a"]
```

<empty><aa><empty><aa><a> => three fields

```
"".split('')  # => []
"a".split('')  # => ["a"]
```

zero-legth match results in all characters being returned in array:

"ab".split('') #=> ['a', 'b']

> What is the definition of *split*?

The String#split documentation clearly states that empty trailing fields are discarded and the special case of a zero-length match.

Gary Wright

#4 - 07/31/2011 06:13 AM - alexeymuranov (Alexey Muranov)

Thank you for the explanation.
Yes, Yusuke, my definition was wrong, i thought about "a maximal array of non-empty strings whose members don't contain any matching substrings".

Gary, I didn't read the documentation carefully, so i didn't know about the discarding of the trailing empty fields, this is why the result looked illogical to me (why not the beginning empty fields?)
I think str1.split(str2, -1) produces what i would have expected.
I will think more about it, thanks.

Alexey.

#5 - 07/31/2011 06:17 AM - alexeymuranov (Alexey Muranov)

It is still not very clear why

```
"a".split('',-1)  # => ["a", ""]
```

and not ["", "a", ""] or ["a"],
(and why "".split(",-1) # => [])

Alexey.

#6 - 07/31/2011 02:23 PM - duerst (Martin Dürst)

On 2011/07/31 1:30, Yusuke ENDOH wrote:

> So, what does "aab".split('a') return?
> I think that only ["aab"] meets the condition.
> But it is also surprising to me.

It's much easier to think about if you replace 'a' with ',' (and if
necessary, think about a format such as CSV).
So what does ",,b".split(',') return?
=> ['', '', 'b']

Regards,   Martin.

#7 - 07/31/2011 03:24 PM - alexeymuranov (Alexey Muranov)

I understand why

```
",,1".split(',')  # => ["","","1"]
```

and why

```
".5".split('.')  # => ["","5"]
```

But then ",1,".split(',') should return ["", "1", ""].
It is not clear why one needs to do it like that:

```
",1,".split(',',-1)  # => ["", "1", ""]
```

The decision to discard trailing empty elements seems random (maybe targeted at processing particular programming languages or application input where optional parameters are placed at the end?).

However, splitting on the empty string does not make sense (cannot be made consistent with these examples).
In my opinion, splitting on the empty string should be forbidden.
To obtain the array of letters (in the given encoding) it would be more logical to introduce a #letters method or use #split without parameters.
Current implementation of split('') seems inconsistent with the rest: why

```
"ab".split('')  # => ["a", "b"] and not  ["", "a", "b"] or  ["", "a", "", "b"] ?
"ab".split('',-1)  # => ["a", "b", ""] and not  ["", "a", "b", ""] ?
```

Does splitting on the empty string work this way because this is how the general implementation works if fed with the empty string, or is it implemented as a separate case?
In the last case it is not a good solution.

Alexey.

#8 - 08/01/2011 01:37 PM - kirillrdy (Kirill Radzikhovskyy)

Hi,
I also find this behavior confusing
mainly because:

ruby-1.9.2-p290 :001 > 'a,b,,'.split ','
=> ["a", "b"]
ruby-1.9.2-p290 :002 > ',,a,b'.split ','
=> ["", "", "a", "b"]

#9 - 08/01/2011 02:23 PM - Anonymous

On Aug 1, 2011, at 12:37 AM, Kirill Radzikhovskyy wrote:

> I also find this behavior confusing
> mainly because:
>
> ruby-1.9.2-p290 :001 > 'a,b,,'.split ','

#10 - 09/11/2011 09:49 PM - alexeymuranov (Alexey Muranov)

I would like to summarize my feature request:

1. trailing empty fields should not be discarded
   (it would make sense, however, to have a similar method which splits and discards initial and trailing empty fields, and returns as the first element of the array the number of initial empty fields discarded, and possibly as the last element of the array the number of trailing empty fields discarded)

2. a separate method for getting the array of letters (#letters?) should be implemented, split on the empty string should raise an error
   (or otherwise it should always return the empty string as the first and the last elements: "a".split("") # => ["", "a", ""] by analogy with "a".split("a") # => ["", ""] in the proposed implementation, but this is not very practical).

Update: how about String#to_a?

This is my opinion, please comment.

Alexey.

#11 - 09/11/2011 10:23 PM - aprescott (Adam Prescott)

On Sun, Sep 11, 2011 at 1:49 PM, Alexey Muranov
muranov@math.univ-toulouse.fr wrote:

> 1. a separate method for getting the array of letters (#letters?) should be implemented, split on the empty string should raise an error (or otherwise it should always return the empty string as the first and the last elements: "a".split("") # => ["", "a", ""] by analogy with "a".split("a") # => ["", ""] in the proposed implementation, but this is not very practical).

str.split("") already gets you the array of "letters" (as does
str.chars.to_a), but since you feel that str.split("") should raise an
error or have another return value, do you think str.split("") should
break existing code which uses split("") to get characters?

What's the reasoning behind str.split("") raising an error? I can't
see a good reason for it. Equally, I can see no good reason for
treating "a".split("") the same in return value as "a".split("a"). In
the latter, there is more to be considered because the receiver itself
contains "a". Why should "a".split("") return ["", "a", ""]?

#12 - 09/11/2011 11:57 PM - alexeymuranov (Alexey Muranov)

Adam Prescott wrote:

> str.split("") already gets you the array of "letters" (as does
> str.chars.to_a), but since you feel that str.split("") should raise an
> error or have another return value, do you think str.split("") should
> break existing code which uses split("") to get characters?

Thanks for pointing out str.chars.to_a, but i think that it would be more convenient to have a single method that would do this. I understand that this would break existing code, i was discussing the issue from the point of view of "improving" the language, according to what would look like an improvement to me. As a person new to Ruby, i expressed my "astonishment" at the current behavior of #split, and tried to contribute to POLA.

> What's the reasoning behind str.split("") raising an error? I can't
> see a good reason for it. Equally, I can see no good reason for
> treating "a".split("") the same in return value as "a".split("a"). In
> the latter, there is more to be considered because the receiver itself
> contains "a". Why should "a".split("") return ["", "a", ""]?

I think that #split should treat all strings equally, whether empty or not. Maybe i've missed something (then please point me to the explanation), but i do not see how the treatment of empty and non-empty strings can be particular cases of a general rule. What is the general rule, which gives such different results for empty and non-empty strings?

I think that "a".split("") should return ["", "a", ""], because this would be more logical, than when "a".split("",-1) returns ["a", ""], as it does now.
I think that in most other cases #split(str) should behave as #split(str,-1) behaves now, because the decision to discard trailing empty elements seems arbitrary.
By analogy with ",".split(",",-1) currently returning ["",""], i think that:
",".split(",") should return ["",""],
"".split("") should return ["",""] (if not forbidden altogether),
",".split("") should return ["", ",", ""] (if not forbidden).

But, as i said, this is only a suggestion to preserve consistency: use the same rule and same code to split on empty and non-empty strings. (What is the rule now?) I think that if #split treats the empty string "" separately from all other string arguments, then #split("") should be made into a separate

method, or be made clearly distinguished by the number or type of arguments, for example: #split() or #split(:by_chars), or #split(:how => :by_chars).

The easiest way to be consistent, in my opinion, is to forbid splitting on the empty string and to use a different method to get the array of letters. How about String#to_a to return the array of letters (in addition to .chars.to_a)? This would be consistent with the String#[] method in Ruby 1.9.

Last edited 2011-11-21

#13 - 03/02/2012 12:36 AM - alexeymuranov (Alexey Muranov)

I would like add a use case which may be not very useful, but in my opinion illustrates the issue well.

I wanted to normalize some mistyped email addresses in a database and i did like this (because i forgot about this issue):

```
email.split('@').map(&:strip).join('@')
```

This works for complete email address, but when i ran into

```
' sam @ '.split('@').map(&:strip).join('@') # => 'sam@'
```

but

```
'jim@'.split('@').map(&:strip).join('@') # => 'jim'
```

i decided to change to

```
email.split('@', -1).map(&:strip).join('@')
```

which does not make it more readable (sarcasm :-)).

#14 - 03/02/2012 04:07 AM - drbrain (Eric Hodel)

=begin
Tell split you want to keep the @ if you want to keep the @:

[' sam @ ', 'jim@'].map { |e| e.split(/(@)/).map(&:strip).join }" # => ["sam@", "jim@"]
=end

#15 - 03/02/2012 04:49 AM - alexeymuranov (Alexey Muranov)

Well, i didn't really want to keep '@', splitting on it and then joining with it would be fine :).

Thanks anyway.

#16 - 03/25/2012 05:26 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to Next Major

#17 - 02/20/2018 08:33 AM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

At least some of your points are rational. Those behaviors are inherited from Perl.
I don't think we can change the behavior. We are not going to break existing code for the sake of consistency.

Matz.