Ruby master - Feature #15580

Proposal: method addition to class String called .indices (String#indices)

01/31/2019 11:13 PM - shevegen (Robert A. Heiler)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Hello,

I am not sure whether this proposal has a realistic chance to be added to Ruby; but I think it is ok to suggest it nonetheless and let matz and the core team decide whether this may be a useful addition to ruby (at the least a bit), or whether it may not be a useful addition or not necessary. Also, I am trying to learn from sawa on the issue tracker here, making useful suggestions. :)

I propose to add the following **new method** to **class String** directly:

```
String#indices
```

This would behave similar to String#index in that it will return the position of a substring, but rather than return a single number or nil, it should **return an Array** of all positions found between the main (target) String; and a substring match. If no match is found, nil should be returned, similar to String#index. (It may be possible to extend String#index to provide this functionality, but I do not want to get into the problem of backwards compatibility; and #indices seems to make more sense to me when reading it than #index, since the intent is a different one - hence why I suggest this new method addition.)

Right now **.index** on class String will return a result like this:

```
'abcabcabc'.index 'a' # => 0
'abcabcabc'.index 'd' # => nil
```

So either the number of the first member found ('a', at 0), or nil if no result is found (in the example of 'd').

In general, the proposal here is to keep #indices behaviour the very same as #index, just with the sole difference being that an Array is returned when at the least one index is found; and all positions that are found are stored in that array.

What is the use case for this proposal or why would I suggest it?

Actually, the use case I have had was a very simple one: to find a DNA/RNA "subsequence" of just a single nucleotide in a longer DNA/RNA string. As you may know, most organisms use double stranded DNA (dsDNA) consisting of four different bases (A,T,C,G); and RNA that is usually single stranded (ssRNA), with the four different bases being (A,T,C,U).

For example, given the RNA sequence of a String like 'AUGCUUCAGAAAGAGAAAGAGAAAGGUCUUACGUAG' or a similar String, I wanted to know at which positions 'U' (Uracil) would be in that substring. So ideally an Array of where the positions were. So that was my use case for String#indices.

We can of course already get the above as-is via existing ruby features.

One solution is to use .find_all - which I am actually using (and adding +1, because nucleotide positions by default start not at 0 but at 1). So I do not really need this addition to class String to begin with, since I can use find_all or other useful features that ruby has as-is just

fine.

However had, I also thought that it may be useful for others if a String#indices method may exist directly, which is why I propose it here. Perhaps it may simplify some existing code bases out there to a limited extent if ruby users could use the same method/functionality.

There may be other use cases for String#indices, but I will only refer to the use case that I have found here. If others wish to add their use case please feel free to do so at your own leisure if you feel like it.

Please also do feel free to close this issue here at any moment in time if it is considered to be not necessary. It is not really a high priority suggestion at all - just mostly a convenience feature (possibly).

Thanks!

PS: I should also add that of course in bioinformatics you often deal with very large datasets, gigabytes/terabytes of genome sequencing data / Next generation sequencing dataset, but if you need more speed anyway then you may use C or another language to do the "primary" work; and ruby could do very fine with smaller datsets just as well; "big data" is not necessarily everywhere.

I only wanted to mention this in the event that it may be pointed out that String#indices may not be very fast for very long target strings/substrings - there are still many use cases for smaller substrings, for example. Perl was used very early in the bioinformatics field to good success, for instance.

As for documentation, I think the documentation for String#index could be used for String#indices too, just with the change that an Array of the positions found may be returned.

History

#1 - 02/06/2019 09:35 AM - duerst (Martin Dürst)

Just a quick question: Should the results include overlaps or not? I.e. is it
'abababa'.indices('aba') # => [0, 2, 4]
or is it just
'abababa'.indices('aba') # => [0, 4]?

#2 - 07/29/2020 12:12 AM - TylerRick (Tyler Rick)

See also #6596, which proposes adding a similar method to Array (Array#indexes).

I don't know which term ("indices" or "indexes") is preferred. (They are both widely used plural forms of "index".) But whichever way it is decided, we should be consistent in the spelling between Array and String.

Speaking of big data sets, if you have a very large sequence, you might be interested in using a lazy enumerator, like the one I proposed here. I don't know if there is a way (or benefit) to use it directly on a string, but if your input is a near-endless stream and you have some way to convert your input into a Enumerator::Lazy, then it would let you do something like this:

```
enum = 'abcabc'.chars.lazy.indexes('c')
#=> #<Enumerator::Lazy: ...>

enum.next
#=> 2
```

#3 - 07/29/2020 12:14 AM - TylerRick (Tyler Rick)

duerst (Martin Dürst) wrote in #note-1:

> Just a quick question: Should the results include overlaps or not? I.e. is it
> 'abababa'.indices('aba') # => [0, 2, 4]
> or is it just
> 'abababa'.indices('aba') # => [0, 4]?

Good question! For what it's worth, it looks like the String#index_all version of this method in Facets fines it in a non-overlapping way by default but has an option to do an overlapping search ("The reuse flag allows the trailing portion of a match to be reused for subsquent matches."):

```
> require 'facets/string/index_all'
=> true

> 'abababa'.index_all('aba')
=> [0, 4]

'abababa'.index_all('aba', reuse = true)
=> [0, 2, 4]
```

If this were to be included into core Ruby, would we want that to be an option, since one way may be more appropriate than the other given the specific use case? I would recommend making it a keyword arg, though. How about allow_overlap: true?

It seems like for the bioinformatics use case, it would be more useful to allow matching in an overlapping way. This is analogous to the difference between Enumerable#each_cons and Enumerable#each_slice. I think you want to scan each consecutive sequence of n characters for a match, rather than just each slice.

By the way, to implement this (in Ruby), you could either iteratively call index (with a different start-search index each time) like Facets did:

```
class String
  def index_all(s, reuse=false)
    s = Regexp.new(Regexp.escape(s)) unless Regexp===s
    ia = []; i = 0
    while (i = index(s,i))
      ia << i
      i += (reuse ? 1 : $~[0].size)
    end
    ia
  end
end

'abababa'.index_all('aba', reuse = true)
=> [0, 2, 4]
```

or using each_cons like this:

```
class String
  def index_all(sought)
    indexes = []
    chars.each_cons(sought.length).with_index do |cons, i|
      cons_str = cons.join
      indexes << i if cons_str == sought
    end
    indexes
  end
end

> 'abababa'.index_all('aba')
=> [0, 2, 4]
```

#4 - 07/29/2020 01:54 AM - sawa (Tsuyoshi Sawada)

> Also, I am trying to learn from sawa on the issue tracker here, making useful suggestions. :)

If I have provided here anything that is worth your learning from, the first thing among them is probably to try to write things concisely.