Ruby - Misc #16507

=~ vs include? or match?

01/12/2020 11:27 PM - MSP-Greg (Greg L)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

While working on getting the mswin build working on Actions, I thought adding mswin? and ci? methods to Minitest::Unit::Guard in tool/lib/minitest/unit.rb would be helpful.

Currently some tests are failing that are guarded/skipped based on ENV['APPVEYOR']. For ci?, I'd combine Travis, AppVeyor, & Actions.

There are quite a few instances where =~ is used for a boolean return. Would it be considered appropriate to replace those calls with include? or match?

History

#1 - 02/12/2023 09:57 AM - rubyFeedback (robert heiler)

I can't speak for the ruby core team, but here is my rule-of-thumb in my own ruby code "bases" out there.

I usually try to stick with the simplest choice possible, which in many ways is .include?() for me. Normally I do not use the () there, but I just wanted to make it more explicit here.

Sometimes this is not possible, though, such as when the input is not so simple (e. g. where a regex is more appropriate such as /\d{0,6}.\d{3,6}/ like when the input can vary). Regexes are great in ruby; I think matz liked them from the perl days. :)

I almost never use .match() or .match?() - and I think these are also rare in other people's code bases out there.

I tend to use methods ending via "?" a lot. It is a bit like DSL-design for me:

```
if object.can_do_this? or object.can_do_that?
end
```

I think it reads very nice. That is actually my recommendation too - if you feel the readability is improved then use the methods.

The method .ci?() seems a bit short-ish. I would recommend against such short method names; they rarely seem to improve a lot. I tend to use them mostly when I need to do things where being succinct is important.

.mswin?() makes more sense. Even then I would perhaps consider using two words, if that makes sense for the code you are working with.

In my opinion, when it really helps you maintain the code, and when it leads to some clarity, using helper-methods is a good idea.

Perhaps if you want some more feedback from ruby core devs, it may help if you could give a few more specific examples so they can look at the instances where you thought that removing =~ would be a good idea. Sometimes different ruby developers have a different focus and priority for using certain constructs in their code. For me my own code is very simple and readable, but I am sure for others it can be very messy and unreadable (I try to stick to simplicity at all times and avoid too complicated constructs; for instance I never use "->" and other things, but I use blocks a LOT, so my code will have tons of "if block_given?; yield = yielded" just to keep on working with the variable. Blocks are great).