Ruby - Feature #16355

Raise NoMatchingPatternError when `expr in pat` doesn't match

11/20/2019 12:58 AM - ktsj (Kazuki Tsujimoto)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

Currently, single line pattern matching(expr in pat) returns true or false.

```
[1, 2, 3] in [x, y, z] #=> true (with assigning 1 to x, 2 to y, and 3 to z)
[1, 2, 3] in [1, 2, 4] #=> false
```

I think expr in pat should raise an exception when it doesn't match.
Because if a user doesn't check the return value of expr in pat, matching failure occurs implicitly and it may cause problems in subsequent processes.

```
expr in [0, x] # A user expects it always matches, but if it doesn't match...
...
(snip)
...
x.foo #=> NoMethodError (undefined method `foo' for nil:NilClass)
```

I also propose that expr in pat returns the result of expr if it matches.
It is similar to assignment.

```
x, y, z = 1, 2, 3       #=> [1, 2, 3]
[1, 2, 3] in [x, y, z] #=> [1, 2, 3]
```

Related issues:

Related to Ruby - Feature #15865: `<expr> in <pattern>` expression	**Closed**
Related to Ruby - Feature #16370: Pattern matching with variable assignment (...	**Open**
Related to Ruby - Feature #17371: Reintroduce `expr in pat`	**Closed**

Associated revisions

Revision 8b4ee5d6ba92a385eedc9235ce0a2d5618deecf0 - 11/28/2019 04:47 AM - nobu (Nobuyoshi Nakada)

Raise NoMatchingPatternError when expr in pat doesn't match

- expr in pattern should raise NoMatchingError when unmatched
- expr in pattern should return nil. (this is unspecified, but this feature is experimental, at all)

[Feature #16355]

Revision 8b4ee5d6ba92a385eedc9235ce0a2d5618deecf0 - 11/28/2019 04:47 AM - nobu (Nobuyoshi Nakada)

Raise NoMatchingPatternError when expr in pat doesn't match

- expr in pattern should raise NoMatchingError when unmatched
- expr in pattern should return nil. (this is unspecified, but this feature is experimental, at all)

[Feature #16355]

Revision 8b4ee5d6 - 11/28/2019 04:47 AM - nobu (Nobuyoshi Nakada)

Raise NoMatchingPatternError when expr in pat doesn't match

- expr in pattern should raise NoMatchingError when unmatched
- expr in pattern should return nil. (this is unspecified, but this feature is experimental, at all)

[Feature #16355]

Revision d1ef4fd08e60adcbcb4feeb55f767ff3d80b65a0 - 11/29/2019 03:15 PM - nobu (Nobuyoshi Nakada)

Make single line pattern matching void expression

Instead of returning nil, raise a syntax error if its value is
used. [Feature #16355]

Revision d1ef4fd08e60adcbcb4feeb55f767ff3d80b65a0 - 11/29/2019 03:15 PM - nobu (Nobuyoshi Nakada)

Make single line pattern matching void expression

Instead of returning nil, raise a syntax error if its value is
used. [Feature #16355]

Revision d1ef4fd0 - 11/29/2019 03:15 PM - nobu (Nobuyoshi Nakada)

Make single line pattern matching void expression

Instead of returning nil, raise a syntax error if its value is
used. [Feature #16355]

History

#1 - 11/20/2019 01:00 AM - ktsj (Kazuki Tsujimoto)

- Related to Feature #15865: `<expr> in <pattern>` expression added

#2 - 11/20/2019 01:01 AM - ktsj (Kazuki Tsujimoto)

- Description updated

#3 - 11/20/2019 08:07 PM - palkan (Vladimir Dementyev)

Agree, that it could save users from unexpected behavior.

On the other hand, raising an exception drastically limits the application of online pattern matching: it won't be possible to use it with if ... else ... or in select/filter statements (here is a great example).

If users won't to ensure that a pattern matches, they can write (expr in ptrn) || raise "smth".

Having one-line patterns return true or false brings a lot of possibilities, IMO.

P.S. I was actually scanning the tracker for the mentions of the following situation I've just encountered:

```
assert_block do
  {a:0, b: 1} in {a:, **nil}
  a.nil?
#=> this is not nil, which confused me at first; but now I think that this is a less evil than raising an exce
ption
end
```

#4 - 11/24/2019 12:17 PM - Eregon (Benoit Daloze)

I think this basically breaks [Feature #15865].

We should decide if expr in pattern can be used as a condition (such as in if) or not.

As @palkan said, it's already easy to use || raise NoMatchingPatternError for assignment cases, but it's impossible (or very ugly) to use if expr in pattern if we do the proposed change.

#5 - 11/28/2019 04:25 AM - matz (Yukihiro Matsumoto)

I accept this proposal for two reasons:

- as OP described, returning boolean values from in pattern matching masks match failure. This can cause serious problems.
- By this change, we cannot use in pattern matching in if conditionals. But this style can be easily expressed by case pattern matching.

Matz.

#6 - 11/28/2019 04:51 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset git|8b4ee5d6ba92a385eedc9235ce0a2d5618deecf0.

Raise NoMatchingPatternError when expr in pat doesn't match

- expr in pattern should raise NoMatchingError when unmatched
- expr in pattern should return nil. (this is unspecified, but this feature is experimental, at all)

[Feature #16355]

#7 - 11/29/2019 03:57 PM - ktsj (Kazuki Tsujimoto)

- Related to Feature #16370: Pattern matching with variable assignment (the priority of `in` operator) added

#8 - 12/06/2020 02:55 PM - ktsj (Kazuki Tsujimoto)

- Related to Feature #17371: Reintroduce `expr in pat` added

Files

expr-in-pat-raises-error.patch	2.59 KB	11/20/2019	ktsj (Kazuki Tsujimoto)