Ruby master - Misc #16464

Which core objects should support deconstruct/deconstruct_keys?

12/28/2019 01:04 PM - zverok (Victor Shepelev)

Status: Open
Priority: Normal
Assignee:

Description
Now, when pattern matching is out, I believe it is worth discussing which core and standard library objects should be matchable "out of the box".

My proposals, as of now, are:

1. **Object#deconstruct, returning [self].**

   Justification:
   
   ```ruby
   # this works:
   1 in Integer
   # this works:
   1 in ..0
   # this does NOT:
   1 in Integer(..0)
   # NoMatchingPatternError (1)
   ```

   I believe the latter example looks pretty logical (and can be used in some flexible methods like "if it is a positive integer, it is index in the array, if it is negative integer, it is backward index, and if it is float, it should be calculated as a mean of nearby elements")

2. **Time#deconstruct_keys**

   Justification is obvious:
   
   ```ruby
   case created_at
   when year: 2019, month: 11..12 => m, day:
     p "Created at #{day}.#{m} this year"
   else
     # ...
   end
   ```

   (Probably the same for Date and DateTime)

3. **Set#deconstruct**

   Seems "logical" as set is a sequence, but I can't think of a good realistic example :)

History

#1 - 12/28/2019 01:15 PM - shevegen (Robert A. Heiler)

I think when possible every object should support deconstructing IF it makes sense. If it walks like a duck, talks like a duck, then it can be dequacked (deconstructed like a duck).

I guess one question may be how useful something it is, whether there is an actual use case or not - people to use that. Perhaps it may be better to see and wait for (several) people who really had a use case to do so and evaluate again in a few months.

```
1 in Integer(..0)
```

"I believe the latter example looks pretty logical "

To me this looks very, very, very strange.

Is that still ruby at all? :P
I guess it follows from a logical continuation, e.g. "if x in y" works, and "beginless ranges" work, then the above should work too. But the syntax is so strange - I wonder if I am the only one feeling about that so if that is the case I'll happily quiet down. But to my eyes it looks very strange.

#2 - 12/30/2019 01:31 AM - Dan0042 (Daniel DeLorme)

1 in Integer(..0)

"I believe the latter example looks pretty logical"

To me this looks very, very, very strange.

Same here. It looks like trying to convert a beginless range to an integer. I understand how Object#deconstruct would work with the array pattern to allow this, but it feels a bit hacky. An object with deconstruct should have some kind of enumerable/tuple-like quality such that it makes sense to represent it as an array.

Given that "Alternative pattern" is represented as pat | pat | ..., using & like this would look more intuitive to me:

1 in Integer & ..0

Time#deconstruct_keys is pretty obvious indeed.

Set#deconstruct might make sense but I think that a Set is usually considered un-ordered? But an array pattern matches in order with the values provided by deconstruct, so Set[1,2,3] in Set[3,2,1] would be false. Quite a gotcha, no?

#3 - 12/30/2019 07:54 AM - decuplet (Nikita Shilnikov)

These probably should be filed as separate feature requests. I'm already using PM in production and the idea about deconstructing Time instances also came into my mind. Object#deconstruct is a lot more subtle, I wouldn't say it's that obvious, I'd rather have it for some built-in classes such as Time or Integer.