Ruby master - Feature #7132

Alternation between named / ordered method arguments and aliases for method arguments.

10/09/2012 09:17 PM - Anonymous

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

=begin
Hi everyone. I am using Ruby for >1 year and I would like to share with you my dreams regarding the named method arguments planned for 2.0. Let us imagine a class Thief with 3 properties (name, hit_points and dexterity), which has constructor #new_thief:

```
new_thief name: "John Fingers", hit_points: 14, dexterity: 15
```

I dream about this constructor accepting alternative syntax:

```
new_thief "John Fingers", hit_points: 14, dexterity: 15
```

and accepting synonyms (aliases) :hp for :hit_points and :dex for dexterity:

```
new_thief "John Fingers", hp: 14, dex: 15
```

To explain my motivation, I am creating a DSL in biology. I am facing the challenge of explaining to my colleagues why is it better than their current Java app. Biologists love synonyms, and I'd like the users calling constructors for biological objects to have freedom to use the synonym they like. Of course, I can already achieve it in 1.9.3:

```
def new_thief *args
  oo = args.extract_options!
  raise ArgumentError if args[0] and oo[:name] and args[0] != oo[:name]
  name = args[0] || oo[:name] || "John Doe"
  raise ArgumentError if oo[:hp] and oo[:hit_points] and oo[:hp] != oo[:hit_points]
  hp = oo[:hp] || oo[:hit_points] || 9
  raise ArgumentError if oo[:dex] and oo[:dexterity] and oo[:dex] != oo[:dexterity]
  dex = oo[:dex] || oo[:dexterity] || 11
  Thief.new( name: name, hit_points: hp, dexterity: dex )
end
```

But I find myself doing this over and over, and even if I write a library to make it easier, it's still too cumbersome.

Proposal no. 1: I propose that alternation between named / ordered arguments and aliases (synonyms) for named method arguments be catered for already by core syntax.

Proposal no. 2: I propose that the syntax for this be as follows:

```
def new_thief( name=**:name="John Doe", hp: **:hit_points=14, dex: **:dexterity=15 )
  Thief.new name: name, hit_points: hp, strength: str, dexterity: dex
end
```

where expressions **:arg_name would refer to the named argument :arg_name.

Please judge the two proposals independently. I really dream about having the alternation between named / ordered arguments and aliases for named arguments in Ruby 2.0. But I feel less sure that the syntax *:arg_name that I am proposing is the best possible syntax for this. Mind me, I don't think it is bad, I am just not sure it is *best.

PS: Please forgive me suggesting such a complicated feature, while being unable to write a single line in C.
=end

History

#1 - 10/19/2012 10:05 AM - wardrop (Tom Wardrop)

=begin

I can't imagine anyone wanting to implement this. Your use case for the named aliases is way too niche (and to be honest, unconvincing; why can't your colleagues just learn the interface) and just adds unnecessary complexity. As for named, unnamed argument switching. That could work, but it raises a couple of problems which would need to be addressed, such as, what happens in this case:

def new_thief(name, hit_points, dexterity)
blah
end

new_thief "John", 6, name: "Bob"

What should Ruby do in this case, where you've assigned a name twice? Should the name argument override the unnamed/ordered argument, or should it raise an exception? It's nothing that can't be resolved, but the selected behaviour needs to be logical and the least surprising.
=end

#2 - 10/19/2012 04:20 PM - nobu (Nobuyoshi Nakada)

- Description updated

=begin
def new_thief(n = "John Doe", name: n, hp: 9, dex: 11, hit_points: hp, dexterity: dex)
=end

#3 - 10/21/2012 08:05 AM - Anonymous

nobu (Nobuyoshi Nakada): Daring syntax. If there is no snag, I'd love it that way.

#4 - 10/22/2012 08:56 PM - Anonymous

wardrop (Tom Wardrop): @Tom Wardrop:

> Your use case for the named aliases is way too niche (and to be honest, unconvincing; why can't your colleagues just learn the interface) and just adds unnecessary complexity.

I would never dare to suggest this proposal to the Python community, with their principle of "one obvious way etc." I do understand the disadvantage of synonyms: you have to learn all the possibilities to understand others' code. But on the other hand, consider human language, which has countless ways of expressing the same, yet we always understand.

So, the first reason is pragmatism. When I use my DSL interactively, I want to save keystrokes, so I type:

```
new_thief "John", hp: 12, dex: 19
```

On the other hand, when I call it from a reusable program, I don't save keystrokes, I want to make it readable, because as we know, code is write once, read many times:

```
new_thief name: "John", hit_points: 12, dexterity: 19
```

Now, some people are used to 'health' instead of 'hit_points'. These people will still understand :hit_points, but when they themselves type the method call, they are likely to say something like:

```
new_thief "John", health: 12, dexterity: 19
```

I want them to still get the expected behavior, even though they speaking habits are different. Or, using a real example:

enzyme 'TMPK', initial_concentration: 7.nanomolar,
activity: {
'EC 2.6.4.9' => [:ternary_reversible_sequential,
TMP + ATP >> TDP + ADP | [kcat: 6.39, hill: 1.0],
site_1: [TMP: 3.2.micromolar, TDP: 3.2.micromolar, TTP: 720.0.micromolar],
site_2: [ADP: 1.nanomolar, ATP: 1.nanomolar]
]
}

should accept alternatives for named arguments, such as :site1, :site_1, and :site2, :site_2, and :kcat :k_cat, :K_cat, :Kcat, and :hill, :Hill, :hill_coeff, :Hill_coeff, :hill_coefficient, :Hill_coefficient etc. I suspect that in real engineering world, synonyms are all over the place, and not just in biology. For example, in mechanical engineering, English speakers are not sure whether to say :young_modulus, :Young_modulus, :youngs_modulus or :Youngs_modulus, while we on the continent don't like to recognize Young, who had elasticity modulus named after him in English only because he was British, so we say alternatively :modulus_of_elasticity, :elastic_modulus or :tensile_modulus. Each speaker has zer own preference, but no matter which alternative is spoken, everyone in the audience understands. So when one creates a materials science DSL, ze has to support all these synonyms. (Btw., I'm glad I'm not writing one, because I don't like *any* of these terms; if I was a ruler of the Universe, I'd order everyone to call it :modulus_of_stiffness :-)

Apart from pragmatism, the second, smaller reason for having named argument synonyms is consistency. Method keywords, that is, noun-like and verb-like DSL elements, have their officialy sanctioned aliasing. But from the user's point of view, names of the named args are adjective-like DSL keywords, and one would expect that they, too, have an officialy sanctioned way of aliasing.

> I can't imagine anyone wanting to implement this.

Like I said, I cannot implement this, because of my zero abilities as a 'real programmer'. But when it comes to eating quiche, then I want :), and I guess that many other users also would.

> As for named, unnamed argument switching. That could work, but it raises a couple of problems which would need to be addressed, such as, what happens in this case:
>
> def new_thief(name, hit_points, dexterity)
> # blah
> end
>
> new_thief "John", 6, name: "Bob"
>
> What should Ruby do in this case, where you've assigned a name twice? Should the name argument override the unnamed/ordered argument, or should it raise an exception? It's nothing that can't be resolved, but the selected behaviour needs to be logical and the least surprising.

For this case, I vote unanimously for raising ArgumentError, unless the object is the same bt :object_id. Such anomalous syntax shouldn't be pampered. If the user wishes for anomalous behaviour, ze can still achive it by handling the argument list manually:

```
def new_thief *ordered_args; named_args = ordered_args.extract_options!
  # handle ordered arguments and named arguments manually
end
```

#5 - 11/24/2012 09:17 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

#6 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)