Ruby master - Bug #16458

Ruby 2.7 warning firing in the wrong situations

12/27/2019 05:55 PM - rafaelfranca (Rafael França)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.7.0p0	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

I'm trying to remove all warnings from my library and there is a warning related to keyword arguments that is firing when I believe it should not.

The warning is:

```
/Users/rafaelfranca/src/rails/globalid/lib/global_id/global_id.rb:23: warning: Using the last argu
ment as keyword parameters is deprecated; maybe ** should be added to the call
/Users/rafaelfranca/src/rails/rails/activesupport/lib/active_support/core_ext/object/try.rb:146: w
arning: The called method `try' is defined here
```

This is the method call:
https://github.com/rails/globalid/blob/bdcbc0300c29292709e4f16b7eb69f8cff9be993/lib/global_id/global_id.rb#L23

This is the method definition:
https://github.com/rails/rails/blob/fe097fa629f0975316736f08c3ae00600055ef06/activesupport/lib/active_support/core_ext/object/try.rb#L7-L17

As you can see. The argument that Ruby believes is a keyword argument is a regular hash. I tried to simplify the case writing the following script but I could run that script without warning.

```
require 'active_support'
require 'active_support/core_ext/object/try'

def a(options = {})
  A.public_send(:b, 1, options.merge(a: "a"))
end

class A
  def self.b(number, options = {})
    B.b(number, options)
  end
end

class B
  class << self
    def b(number, options = {})
      C.new.try(:c, options)
    end
  end
end

class C
  def c(options = {})
    puts(options)
  end
end

a
a(b: "b")
```

History

#1 - 12/27/2019 07:46 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Rejected

The method is defined in line *146*, i.e. it is nil's #try method:

```
def try(_method_name = nil, *, **)
  nil
end
```

https://github.com/rails/rails/blob/fe097fa629f0975316736f08c3ae00600055ef06/activesupport/lib/active_support/core_ext/object/try.rb#L146-L148

So the warning isn't actually wrong.