Optional argument treated as kwarg

When you define a method with an optional argument and keyword arguments (whether explicitly or with options splat) the defaulted argument can not take a hash argument, instead it is interpreted as keyword arguments:

class Foo
  def self.options(value = nil, **options)
    puts value.inspect
    puts options.inspect
  end

  def self.kwarg(value = nil, kw: nil)
    puts value.inspect
    puts kw.inspect
  end

  def self.splat(*args, kw: nil)
    puts args.inspect
    puts kw.inspect
  end
end

Foo.options({})
# nil
# {}
Foo.kwarg({})
# nil
# nil
Foo.splat({})
# []
# nil

Foo.options({ key: :value })
# nil
# {:key=>:value}
Foo.kwarg({ key: :value })
# ArgumentError: unknown keyword: key
Foo.splat({ key: :value })
# ArgumentError: unknown keyword: key

I would expect the output to be:

Foo.options({})
# {}
# {}
Foo.kwarg({})
# {}
# nil
Foo.splat({})
# [{}, nil]

Foo.options({ key: :value })
Related issues:

- Related to Ruby master - Feature #14183: "Real" keyword argument
  - Closed
- Has duplicate Ruby master - Bug #13336: Default Parameters don't work
  - Closed

### History

#### #1 - 12/14/2017 07:24 AM - hsbt (Hiroshi SHIBATA)
- Related to Feature #14183: "Real" keyword argument added

#### #2 - 09/05/2018 03:22 PM - marcandre (Marc-Andre Lafortune)
I believe this is as designed.

As I stated previously ([https://bugs.ruby-lang.org/issues/11967#note-3](https://bugs.ruby-lang.org/issues/11967#note-3)), my understanding is that:

- after all mandatory unnamed arguments are filled
- if the last remaining argument is hash-like
- and all its keys are symbols
- and the method called uses keyword arguments => then that parameter is used for keyword arguments.

I believe we should close this issue.

#### #3 - 09/05/2018 03:26 PM - marcandre (Marc-Andre Lafortune)
- Has duplicate Bug #13336: Default Parameters don't work added

#### #4 - 08/30/2019 09:46 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Open to Closed

With the changes in #14183:

```ruby
Foo.options({})
# (rb):21: warning: The last argument for 'options' (defined at (rb):6) is used as the keyword parameter
# nil
# {}

Foo.kwarg({})
# (rb):22: warning: The last argument for 'kwarg' (defined at (rb):11) is used as the keyword parameter
# nil
# nil

Foo.splat({})
# (rb):23: warning: The last argument for 'splat' (defined at (rb):16) is used as the keyword parameter
# []
# nil

Foo.options({ key: :value })
# (rb):24: warning: The last argument for 'options' (defined at (rb):6) is used as the keyword parameter
# nil
# {:key=>:value}

Foo.kwarg({ key: :value })
# (rb):25: warning: The last argument for 'kwarg' (defined at (rb):11) is used as the keyword parameter
# ArgumentError (unknown keyword: :key)

Foo.splat({ key: :value })
# (rb):26: warning: The last argument for 'splat' (defined at (rb):16) is used as the keyword parameter
# ArgumentError (unknown keyword: :key)
```

In Ruby 3, the behavior will be:

```ruby
Foo.options({})
# {}
```
I'm going to close this now as the deprecation warnings for the cases where behavior will change in Ruby 3 have been added.