Hello ruby core,

This is about module_function(). As the documentation rightfully states, it adds a private method when included ("mixed in") into a class.

The following code shows that this works:

```ruby
module Foo
  module_function
  def hi
    puts 'hi from method hi() in module Foo'
  end
end

class Bar
  include Foo
  def test
    hi
  end
end

Bar.new.test # This works
Bar.new.hi  # This here leads to an error message "private method `hi' called"
```

I am sure that the private method by default makes sense, but I wondered in those case where I would like the above to be a public variant.

Is there some functionality like module_function but that works as a public method instead? So that the last example, Bar.new.hi, would also work?

Something like

```ruby
public_module_function
```

If I would have a public variant, I assume that I could get rid of "def self.foo" statements in some of my modules.

History

#1 - 09/03/2016 06:07 AM - nobu (Nobuyoshi Nakada)
- Description updated

```ruby
class Bar
  include Foo
  public :hi
end
```

#2 - 09/04/2016 04:27 PM - jwmittag (Jörg W Mittag)
If module_function returned its argument(s), we could write...
module Foo
  public module_function def hi
    puts 'hi from method hi() in module Foo'
  end
end

But I'm not sure that's worth it. The whole point of "functions" is that they don't have receiver! Why would you want to call them with one?

#3 - 11/25/2016 06:58 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Rejected

I understand your proposal, but I don't see what it is for.
Do you have any no-artificial use-case for this feature?
If you come up with real-world use-case, please re-open the issue.

Matz.