I just learned that these two pieces of code are not the same:

```ruby
def test
  if result = calculate_result
    return result
  end

  ...
end

def test
  return result if result = calculate_result

  ...
end
```

The second one will give this unhelpful error message:

```
NameError: undefined local variable or method `result' for main:Object
Did you mean?  result
```

I think this is confusing, especially since tools like RuboCop sometimes encourage users to change conditionals to inline ones ("guard statements"). I'd like to propose that Ruby be changed to treat these as the same.

History

#1 - 05/23/2018 08:50 PM - shevegen (Robert A. Heiler)
Interesting. I personally very rarely do assignment of variables followed by if-conditionals in my ruby code.

Perhaps there may be some parsing limitation? Otherwise it does indeed looks like the same code to me (from the semantics), just spread out onto more than one line in the first case).

As for tools such as rubocop - while this can be ignored for the suggestion here (since your suggestion is just as valid without any tools making recommendation), ruby hackers should never become autobots working for the tools; it should be the other way around, including making decisions. But that is just my personal opinion - rubocop can be customized and adjusted as far as I know anyway, so it's not a big deal. :)

#2 - 05/24/2018 05:47 AM - nobu (Nobuyoshi Nakada)
Seems the recent did_you_mean omits result in that case. It may be able to improve the message, I guess.

#3 - 05/24/2018 01:34 PM - jacobevelyn (Jacob Evelyn)
I agree the did_you_mean message is not good, but I would rather change the behavior so that line of code works and doesn't produce an error at all.

#4 - 05/24/2018 02:12 PM - jeremyevans0 (Jeremy Evans)
jacobevelyn (Jacob Evelyn) wrote:

  I agree the did_you_mean message is not good, but I would rather change the behavior so that line of code works and doesn't produce an error at all.

02/22/2020
Doing that would change the behavior of code that currently works:

```ruby
def calculate_result
  :calculate_result
end

def result
  :result
end

def test
  return result if result = calculate_result
  :test
end
```

Currently, test returns :result. With the change you are proposing, test would return :calculate_result.

---

**#5 - 05/24/2018 04:46 PM - jacob Evelyn (Jacob Evelyn)**

Ah you're right Jeremy, I hadn't thought about that case. I still think it's unexpected though that in your example the behavior would change if we split the conditional into an if...end, but I don't know if others agree, and I don't know how a proposal like this that could possibly break existing code should move forward.

---

**#6 - 05/25/2018 05:46 AM - nobu (Nobuyoshi Nakada)**

```ruby
def test
  result = calculate_result and return result
  # ...
end
```