Default Arguments and Option Hashes

Default Arguments

```
# Let's make num an optional parameter.
# By default, num will have the value of 1
def repeat(message, num=1)
    message * num
end

p repeat("hi") # => "hi"
p repeat("hi", 3) # => "hihihi"
```

A fairly common design pattern is to set an arg to `nil` by default and have logic based on that scenario:

```
def greet(person_1, person_2=nil)
    if person_2.nil?
        p "Hey " + person_1
    else
        p "Hey " + person_1 + " and " + person_2
    end
end

greet("Chao") # => "Hey Chao"
greet("Chao", "Arittro") # => "Hey Chao and Arittro"
```

To avoid confusion, it's best practice to have optional parameters listed after the required ones. If we stick to this convention, we can always expect arguments to be taken in the same order we pass them in. So avoid writing code like this:

```
def greet(person_1="default", person_2)
    p person_1 + " and " + person_2
end

greet("Chao") # => "default and Chao"
```

The method above is not intuitive because although `"Chao"` is first argument passed in, `person_2` will be assigned `"Chao"`. Avoid this by only assigning default values at the end of the parameter list.

Option Hashes

If you have a method that accepts a hash as an argument, you can omit the braces when passing in the hash:

```
def method(hash)
    p hash  # {"location"=>"SF", "color"=>"red", "size"=>100}
end

method({"location"=>"SF", "color"=>"red", "size"=>100})

# this also works:
method("location"=>"SF", "color"=>"red", "size"=>100)
```

This can really clean things up when you have other arguments before the hash:

```
def modify_string(str, options)
    str.upcase! if options["upper"]
    p str * options["repeats"]
end

# less readable
modify_string("bye", {"upper"=>true, "repeats"=>3}) # => "BYEBYEBYE"

# more readable
modify_string("bye", "upper"=>true, "repeats"=>3)   # => "BYEBYEBYE"
```

Combining this with the default arguments we covered in the previous section can make our code even more flexible:

```
def modify_string(str, options={"upper"=>false, "repeats"=>1})
    str.upcase! if options["upper"]
    p str * options["repeats"]
end

modify_string("bye")   # => "bye"
modify_string("bye", "upper"=>true, "repeats"=>3)   # => "BYEBYEBYE"
```