## In this lecture, we will discuss…

- ✧ Functions / Methods
  - Definitions
  - How do you call them?
  - What and how do they return?
  - Default args
- ✧ How to make methods more expressive
- ✧ What is “splat”

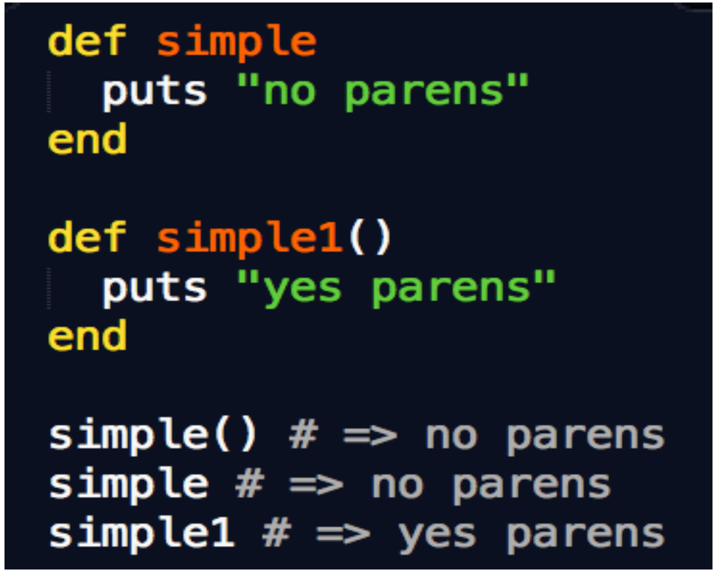
# ~~Functions~~ and Methods

- ✧ Technically, a **function** is defined **outside** of a class and a **method** is defined **inside** a class
- ✧ In Ruby, **every** function/method has at least one class it belongs to
  - Not always written inside a class

**Conclusion: Every function is really a method in Ruby**

# Methods

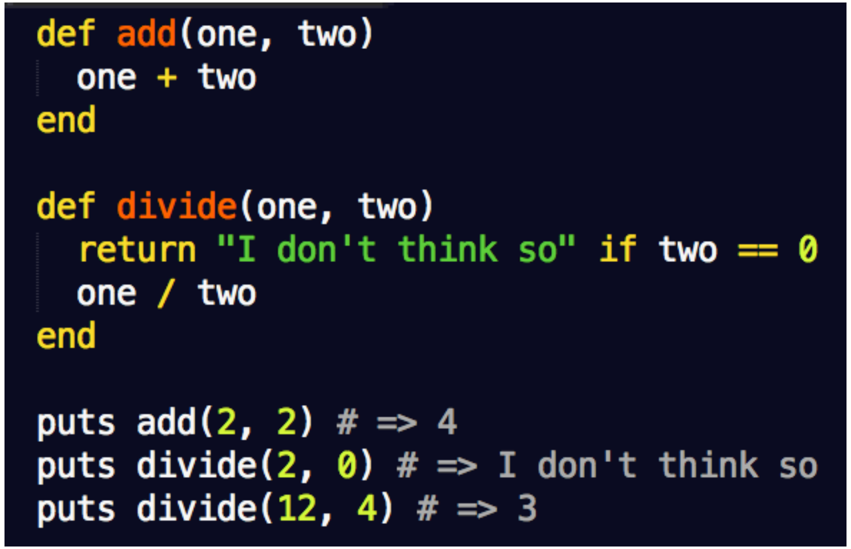
- ✧ Parentheses are **optional** both when **defining and calling** a method
  - Used for **clarity**



```
def simple
  puts "no parens"
end

def simple1()
  puts "yes parens"
end

simple() # => no parens
simple # => no parens
simple1 # => yes parens
```

## Return

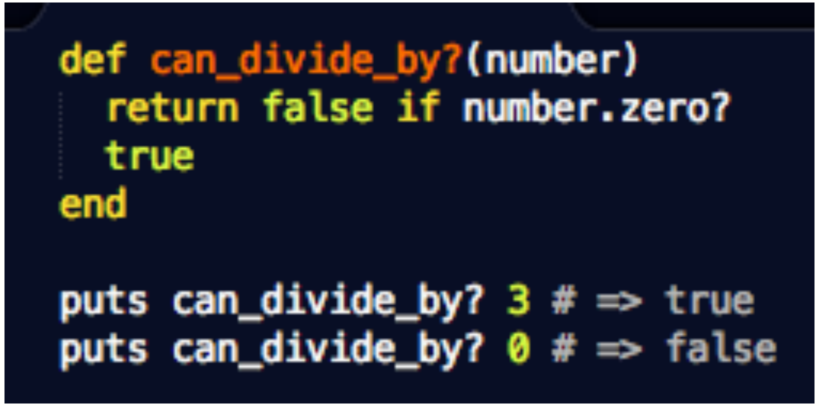
- **No need** to declare type of parameters
- Can return **whatever you want**
- `return` keyword is optional (last executed line returned)



```ruby
def add(one, two)
  one + two
end

def divide(one, two)
  return "I don't think so" if two == 0
  one / two
end

puts add(2, 2) # => 4
puts divide(2, 0) # => I don't think so
puts divide(12, 4) # => 3
```

# Expressive Method Names

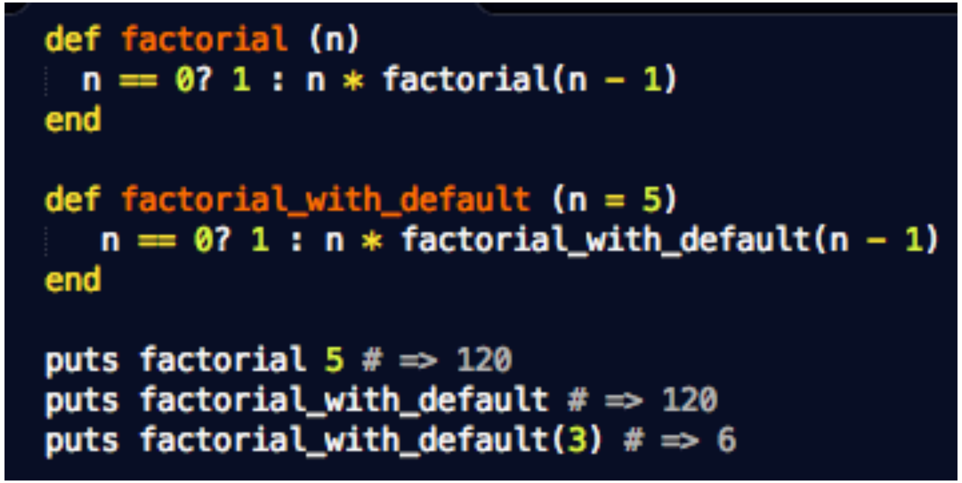
- Method names can end with:
  - **'?'** - Predicate methods
  - **'!'** - Dangerous side-effects (*example later by strings*)



```
def can_divide_by?(number)
  return false if number.zero?
  true
end

puts can_divide_by? 3 # => true
puts can_divide_by? 0 # => false
```

# Default Arguments

- Methods can have **default arguments**
  - If a value is passed in – use that value
  - Otherwise – use the default value provided

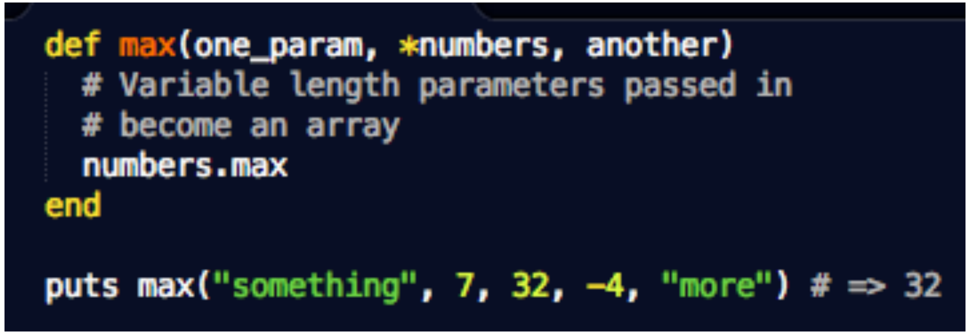


```ruby
def factorial (n)
  n == 0? 1 : n * factorial(n - 1)
end

def factorial_with_default (n = 5)
   n == 0? 1 : n * factorial_with_default(n - 1)
end

puts factorial 5 # => 120
puts factorial_with_default # => 120
puts factorial_with_default(3) # => 6
```

Ternary operator:
condition ? true : false

# Splat

- * **prefixes parameter inside** method definition
  - Can even apply to **middle parameter**, not just the last



```
def max(one_param, *numbers, another)
  # Variable length parameters passed in
  # become an array
  numbers.max
end

puts max("something", 7, 32, -4, "more") # => 32
```

# Summary

- ✧ There is **no need** to declare parameter type passed in or returned (dynamic)
- ✧ **`return`** is **optional** – the last executable line is “returned”
- ✧ You can construct methods with **variable number** of arguments or default arguments

**What’s next?**

- ✧ Blocks