

---

# Ruby Monstas



---

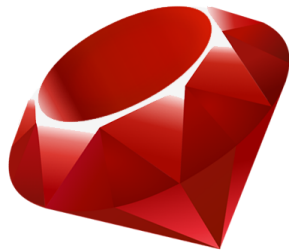
## Session 5

---

# Agenda

---

- Interactive Recap
- Methods
- Exercises



---

# Methods

---

Introduction

---

# Methods

---

You've used them before!

```
"Hello Rubymonstas".length
```

```
"Test string".include?("Something")
```

```
5.odd?
```

```
puts "Tadaaaa!"
```

---

# Method or Function

---

A named block of code which takes input and returns output.

Methods make your code reusable.

Related to the mathematical idea of a function.

---

# Structure

---

- Method name
    - The name a method is known as in the rest of your code
  - Input values (aka arguments, parameters)
    - The information the method needs given to do its job
  - Body
    - The code that does something useful with the input
  - Return value
    - The value that is return to the code that called the method
-

# Structure

---

```
def my_first_method(first_argument, second_argument)
  # This is the method body,
  # Here comes some useful code

  return_value
end
```

---

# Method name

---

- Mandatory
- Naming conventions: Just like variable names!
- But you can use ? and ! at the end of your method name

calculate\_age

enough\_free\_space?

save\_user\_data!

---



# Input values

---

- Optional, Comma-separated in parentheses
- Naming: Just like normal variables
- Only valid within the method, not before, not after

```
def calculate_age(year_of_birth)
  # code here
end
```

```
def save_user_data!(first_name, last_name, home_town)
  # code here
end
```

---

# Method body

---

- Everything between the signature and the `end`
- Any Ruby code
- You can also call methods from within methods

```
def calculate_age(year_of_birth)
  age = 2015 - year_of_birth
end
```

# Return value

---

- A method can return one value
- Anything that can be in a variable, a method can return
- The last thing you do in a method is the return value
- Sometimes we don't need to care about the return value

```
def calculate_age(year_of_birth)
  age = 2015 - year_of_birth
  age
end
```

# Calling methods

---

How do I use (“call”) my method?

```
calculated_value = my_first_method(42, “hello”)
```

```
my_first_method calculated_value, 53.45
```

---

---

# Time to practice



---

Let's get to it!

---