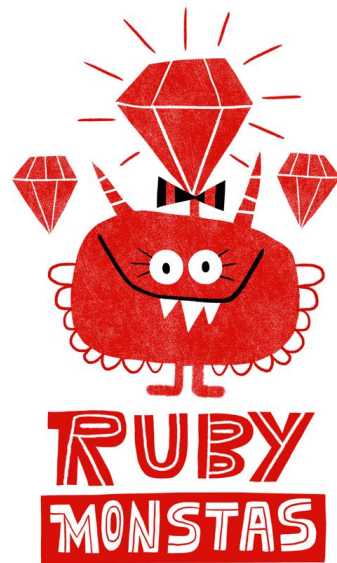


---

# Ruby Monstas



---

Session 11

---

# Agenda

---

- Recap / Questions
- Classes / Objects and OOP
- Exercises



---

# Recap

---

---

# Questions

---

Blocks

Symbols

Ranges

---

---

# Classes & Objects

---

Object-oriented programming

---

# Cookie cutter analogy

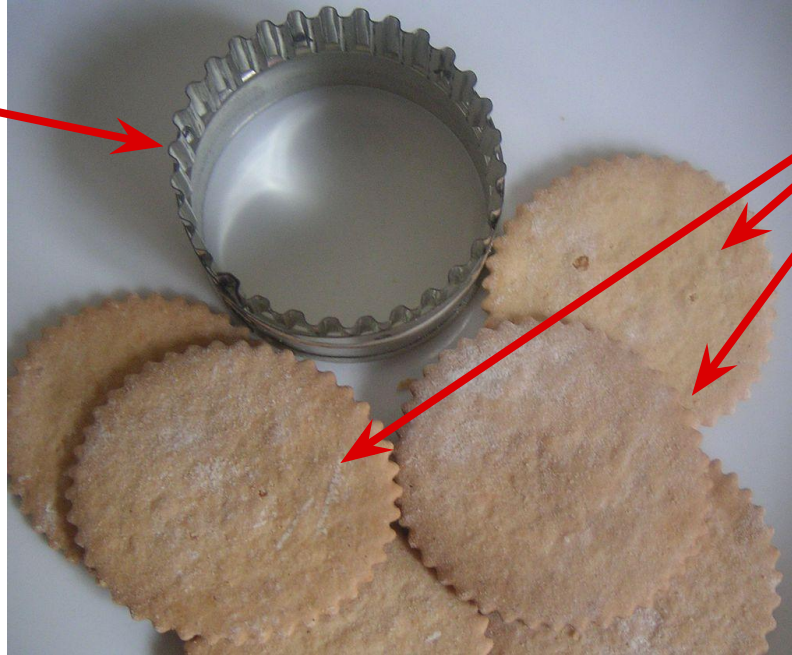
---



# Cookie cutter analogy

---

Cookie cutter

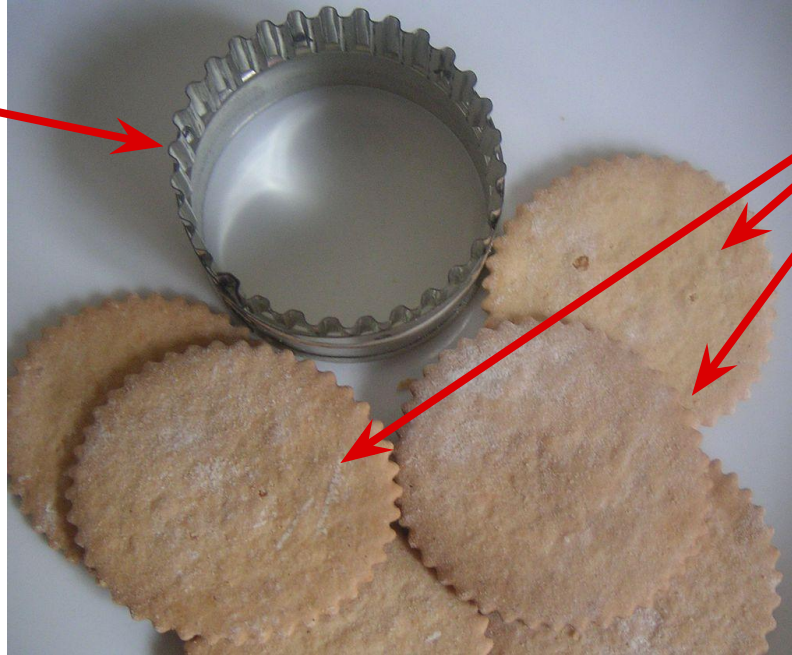


Cookies

# Cookie cutter analogy

---

Cookie cutter  
→ Class



Cookies  
→ Objects



# Cookie cutter analogy

---

The cookie cutter (class) is used to make the dough (memory) into cookies (objects).

---

# Cat example

---

The domain of our application: cats!

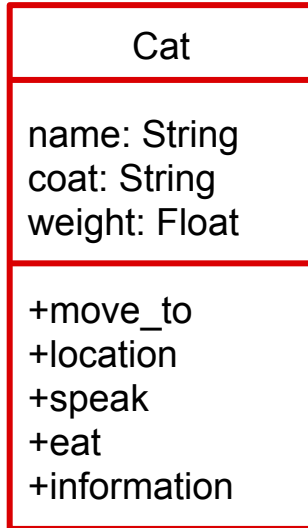
Therefore it makes sense to have a “Cat” class (cookie cutter).

We can later use our “Cat” class to produce concrete (actual) cats.

---

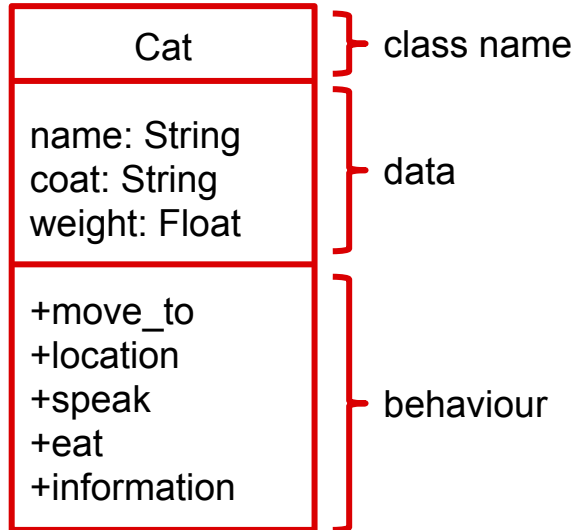
# Cat example: Class diagram

---



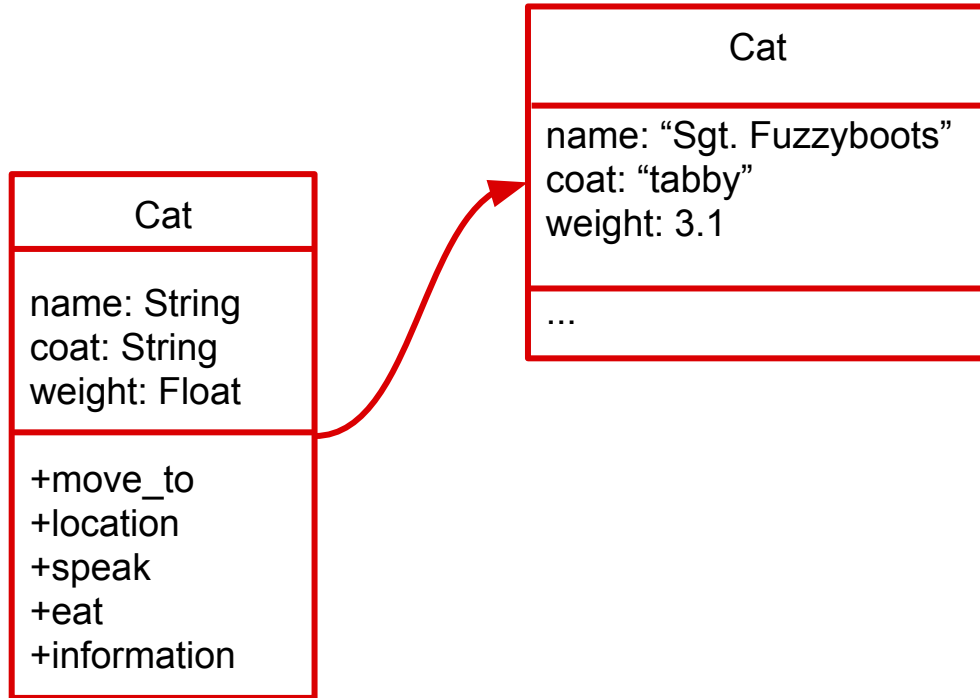
# Cat example: Class diagram

---



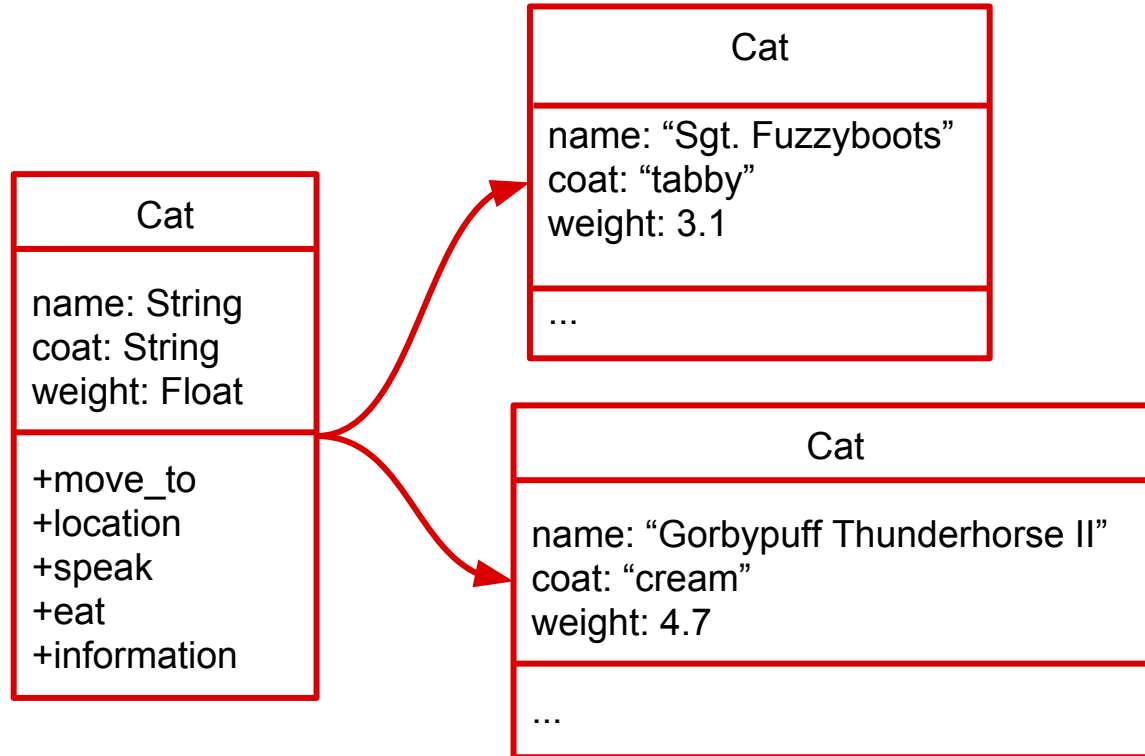
# Cat example: Instantiation

---



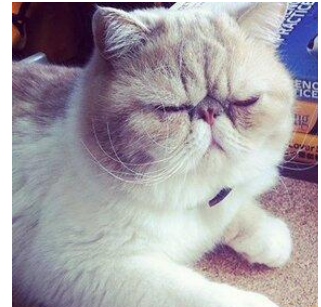
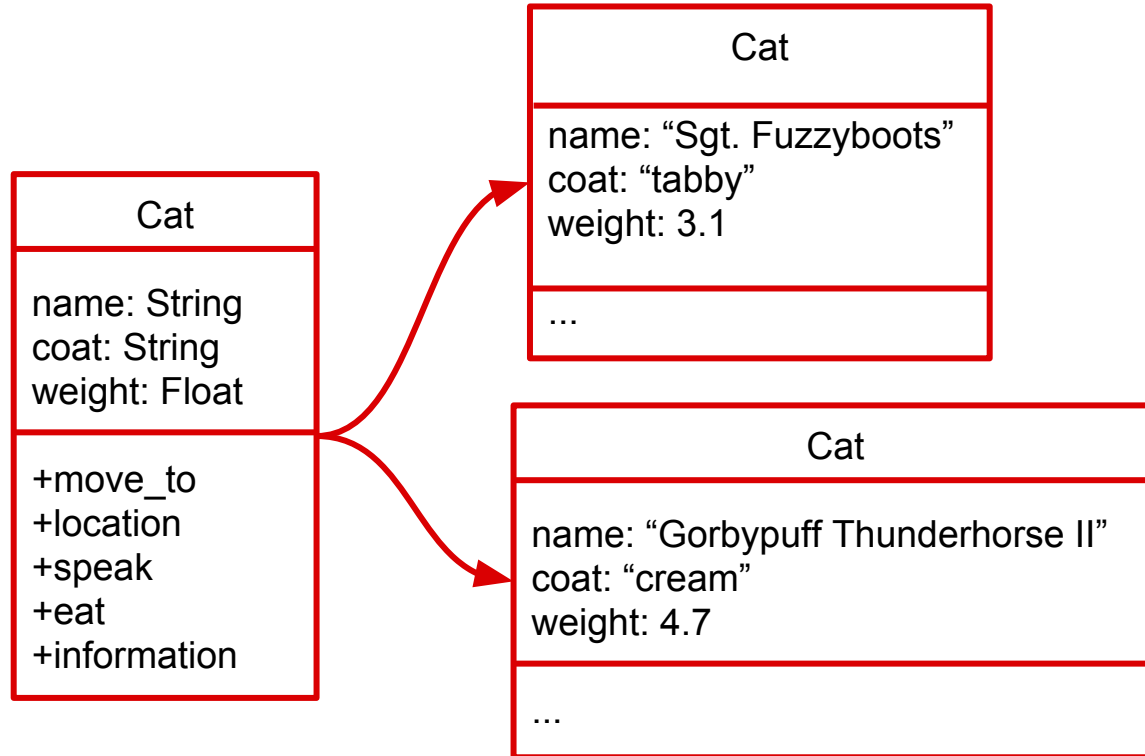
# Cat example: Instantiation

---

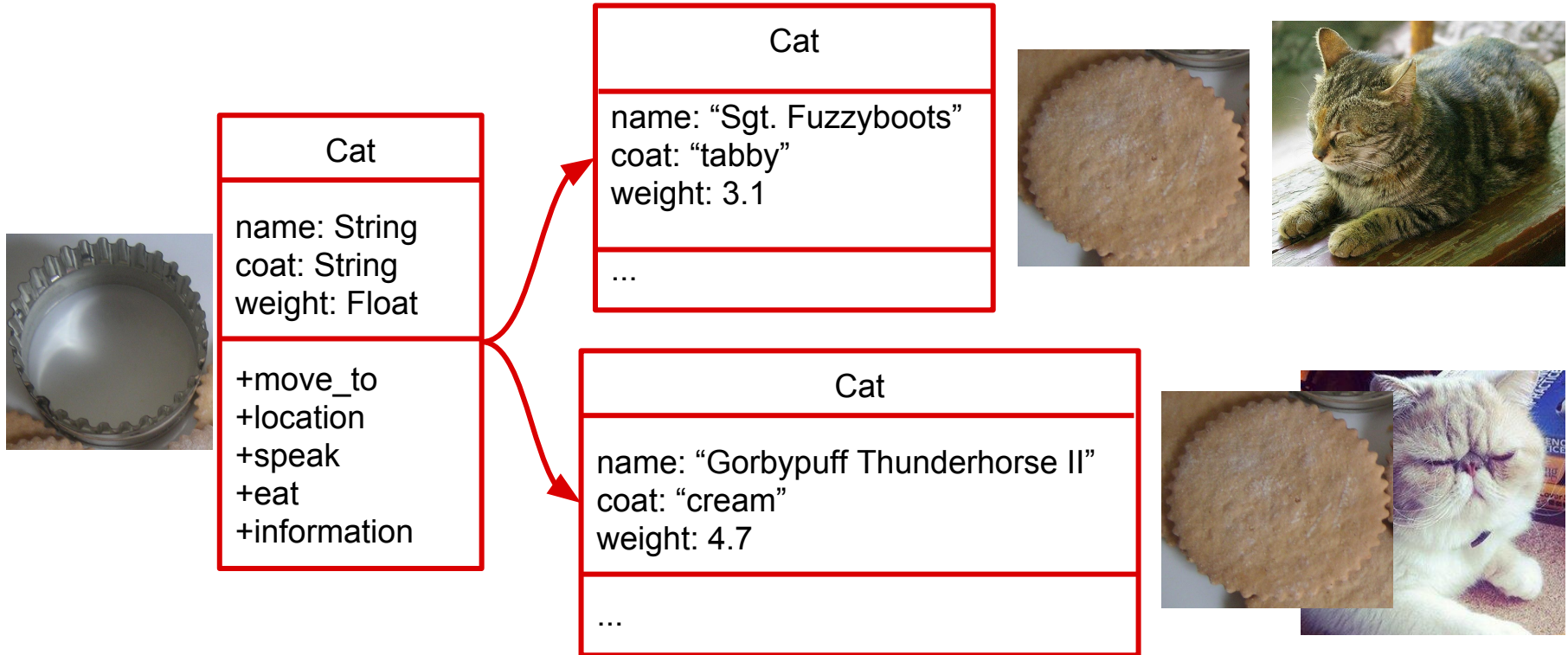


# Cat example: Instantiation

---



# Cat example: Instantiation





# Classes in Ruby

---

Cat
name: String coat: String weight: Float
+move_to +location +speak +eat +information

# Classes in Ruby: initialize method

---

Cat
name: String coat: String weight: Float
+move_to +location +speak +eat +information

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

# Classes in Ruby: initialize method

---

Cat
<u>name</u> : String coat: String weight: Float
+move_to +location +speak +eat +information

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

# Classes in Ruby: initialize method

---

Cat
<u>name</u> : String coat: String weight: Float
+move_to +location +speak +eat +information

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

**initialize** is a special method, that allows us to initialize the object we're creating with some data.

The same concept is called “constructor” in other programming languages.

# Classes in Ruby: Using initialize

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

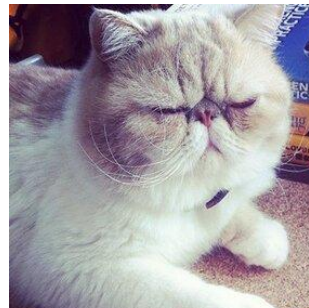
```
sgt_fuzzyboots = Cat.new(
  'Sgt. Fuzzyboots', 'tabby', 3.1
)
```

```
gorbypuff = Cat.new(
  'Gorbypuff Thunderhorse II',
  'cream',
  4.7
)
```

Cat
name: "Sgt. Fuzzyboots" coat: "tabby" weight: 3.1
...



Cat
name: "Gorbypuff Thunderhorse II" coat: "cream" weight: 4.7
...



# Classes in Ruby: Using initialize

---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
```



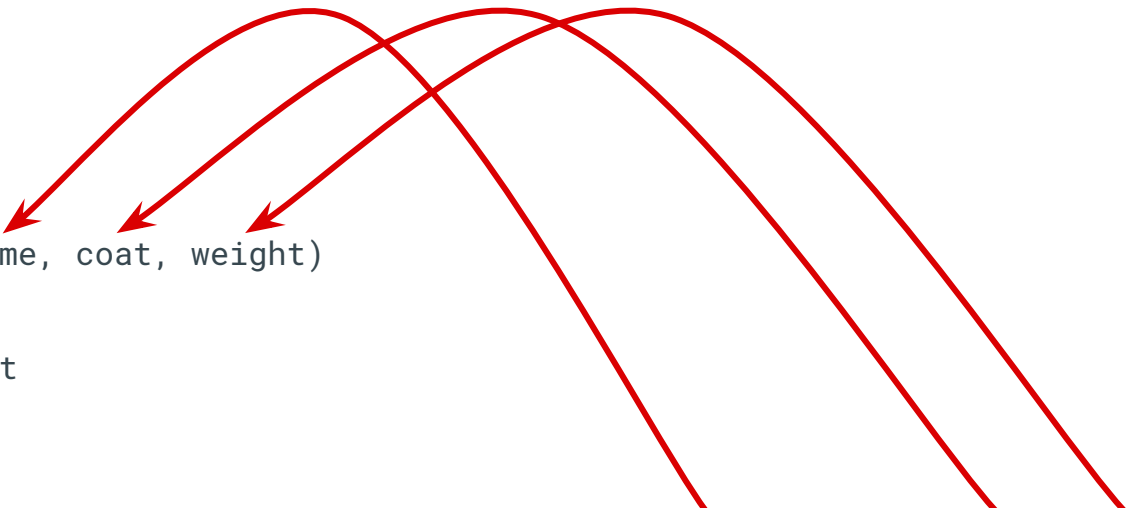
1. `Cat.new` is called, that means `initialize` gets called in the `Cat` class.
-

# Classes in Ruby: Using initialize

---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end

sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
```

A diagram consisting of three red curved arrows. The first arrow starts at the first argument 'Sgt. Fuzzyboots' in the Cat.new call and points to the 'name' parameter in the initialize method definition. The second arrow starts at the second argument 'tabby' and points to the 'coat' parameter. The third arrow starts at the third argument '3.1' and points to the 'weight' parameter.

2. Local variables `name`, `coat` and `weight` get assigned to the values passed.
-

# Classes in Ruby: Using initialize

---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
```

3. Instance variables `@name`, `@coat` and `@weight` get assigned to the values of the local variables `name`, `coat` and `weight`.
-



# Classes in Ruby: Using initialize

---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
```

4. `initialize` method is over. Ruby builds an instance (object) of type `Cat` that contains the data and returns it from the `new` method call.
-

# Classes in Ruby: Using initialize

---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end
```

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
```

5. The returned object of type `Cat` gets assigned to a local variable called `sgt_fuzzyboots` outside the class.
-

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
```

```
puts a
```

---

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
```

```
puts a
```

12

---

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12      def my_method
              a = 12
puts a      puts a
            end

my_method
```

12

---

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12      def my_method
              a = 12
puts a      puts a
              end
```

```
my_method
```

12

12

---

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
```

```
puts a
```

```
12
```

```
def my_method
```

```
  a = 12
```

```
  puts a
```

```
end
```

```
my_method
```

```
12
```

```
def my_method
```

```
  a = 12
```

```
end
```

```
my_method
```

```
puts a
```

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
```

```
puts a
```

```
12
```

```
def my_method
```

```
  a = 12
```

```
  puts a
```

```
end
```

```
my_method
```

```
12
```

```
def my_method
```

```
  a = 12
```

```
end
```

```
my_method
```

```
puts a
```

```
NameError  
(undefined local  
variable or  
method `a' for  
main:Object)
```



# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
puts a
```

12

```
def my_method
  a = 12
  puts a
end
```

my\_method

12

```
def my_method
  a = 12
end
```

my\_method  
puts a

NameError  
(undefined local  
variable or  
method `a' for  
main:Object)

```
def my_method
  a = 12
end
```

a = 23  
my\_method  
puts a

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
puts a
```

12

```
def my_method
  a = 12
  puts a
end
```

```
my_method
```

12

```
def my_method
  a = 12
end
```

```
my_method
puts a
```

NameError  
(undefined local  
variable or  
method `a' for  
main:Object)

```
def my_method
  a = 12
end
```

```
a = 23
my_method
puts a
```

23

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
puts a
```

12

```
def my_method
  a = 12
  puts a
end
```

my\_method

12

```
def my_method
  a = 12
end
```

my\_method  
puts a

NameError  
(undefined local  
variable or  
method `a' for  
main:Object)

```
def my_method
  a = 12
end
```

a = 23  
my\_method  
puts a

23

```
def my_method
  a = 12
  puts a
end
```

a = 23  
my\_method

# Local vs. instance variables

---

**Local variables** are useable from when they're first defined until the end of the scope they're defined in.

```
a = 12
puts a
```

12

```
def my_method
  a = 12
  puts a
end
```

my\_method

12

```
def my_method
  a = 12
end
```

my\_method  
puts a

NameError  
(undefined local  
variable or  
method `a' for  
main:Object)

```
def my_method
  a = 12
end
```

a = 23  
my\_method  
puts a

23

```
def my_method
  a = 12
  puts a
end
```

a = 23  
my\_method

12

# Local vs. instance variables

---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end

sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)

# do something else
```


# Local vs. instance variables


---

```
class Cat
  def initialize(name, coat, weight)
    @name = name
    @coat = coat
    @weight = weight
  end
end

sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)

# do something else
```

 name, coat, weight are in scope here, but not outside initialize!

 name, coat, weight are gone here!

---

# Local vs. instance variables

---

**Instance variables** are useable without definition (default value: `nil`). They are in scope inside an object. Each object has its own instance variables.

```
class MyClass
  def my_method
    @my_variable
  end
end
```

```
my_object = MyClass.new
puts my_object.my_method
```

# Local vs. instance variables

---

**Instance variables** are useable without definition (default value: `nil`). They are in scope inside an object. Each object has its own instance variables.

```
class MyClass
  def my_method
    @my_variable
  end
end
```

```
my_object = MyClass.new
puts my_object.my_method
```

(nothing (nil))

---



# Local vs. instance variables

---

**Instance variables** are useable without definition (default value: `nil`). They are in scope inside an object. Each object has its own instance variables.

```
class MyClass
  def my_method
    @my_variable
  end
end

my_object = MyClass.new
puts my_object.my_method
```

```
class MyClass
  def initialize
    @my_variable = 12
  end

  def my_method
    @my_variable
  end
end

my_object = MyClass.new
my_object.my_method
```

(nothing (nil))

---

# Local vs. instance variables

---

**Instance variables** are useable without definition (default value: `nil`). They are in scope inside an object. Each object has its own instance variables.

```
class MyClass
  def my_method
    @my_variable
  end
end

my_object = MyClass.new
puts my_object.my_method
```

```
class MyClass
  def initialize
    @my_variable = 12
  end

  def my_method
    @my_variable
  end
end

my_object = MyClass.new
my_object.my_method
```

(nothing (nil))

12

---

# Local vs. instance variables

---

**Instance variables** are useable without definition (default value: `nil`). They are in scope inside an object. Each object has its own instance variables.

```
class MyClass
  def my_method
    @my_variable
  end
end

my_object = MyClass.new
puts my_object.my_method
```

(nothing (nil))

```
class MyClass
  def initialize
    @my_variable = 12
  end

  def my_method
    @my_variable
  end
end

my_object = MyClass.new
my_object.my_method
```

12

```
class MyClass
  def initialize(my_number)
    @my_variable = my_number
  end

  def my_method
    @my_variable
  end
end

a = MyClass.new(12)
b = MyClass.new(23)

puts b.my_method
puts a.my_method
```

# Local vs. instance variables

---

**Instance variables** are useable without definition (default value: `nil`). They are in scope inside an object. Each object has its own instance variables.

```
class MyClass
  def my_method
    @my_variable
  end
end

my_object = MyClass.new
puts my_object.my_method
```

(nothing (nil))

```
class MyClass
  def initialize
    @my_variable = 12
  end

  def my_method
    @my_variable
  end
end

my_object = MyClass.new
my_object.my_method
```

12

```
class MyClass
  def initialize(my_number)
    @my_variable = my_number
  end

  def my_method
    @my_variable
  end
end

a = MyClass.new(12)
b = MyClass.new(23)

puts b.my_method
puts a.my_method
```

23

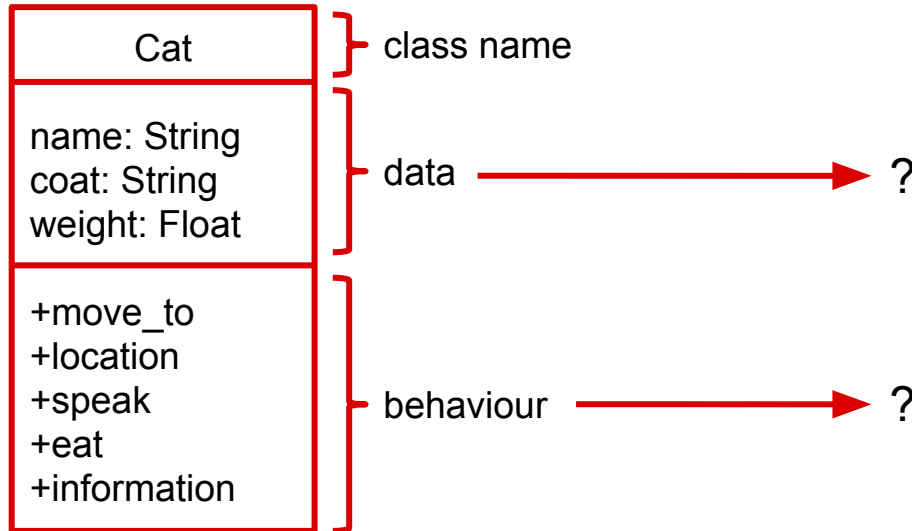
12

# Classes in Ruby: Behaviour

---

## Class diagram

## Ruby

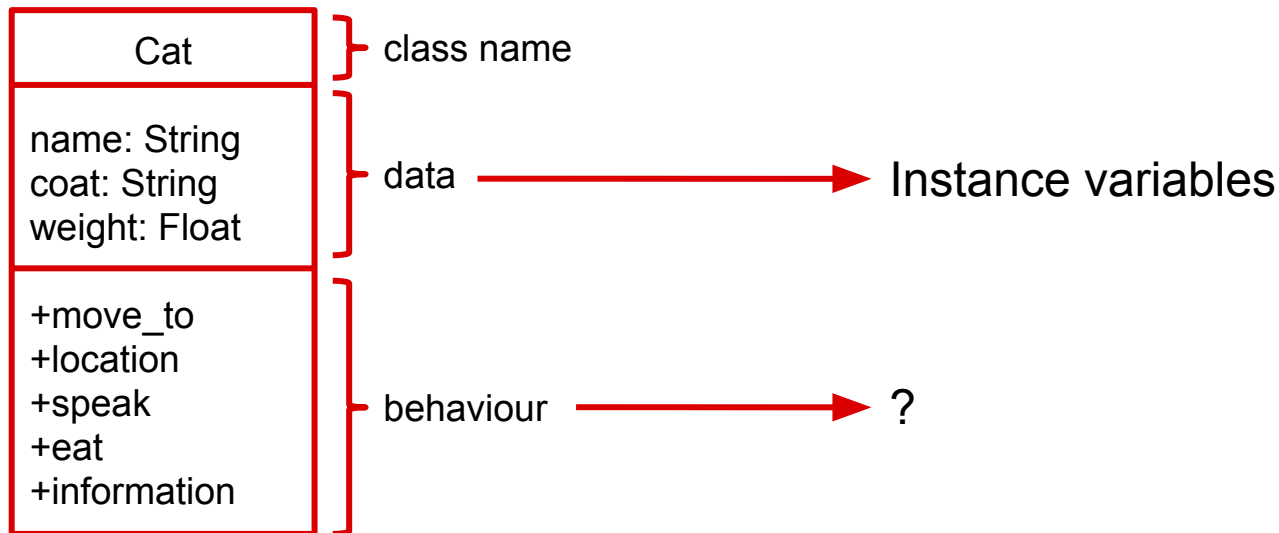


# Classes in Ruby: Behaviour

---

## Class diagram

## Ruby

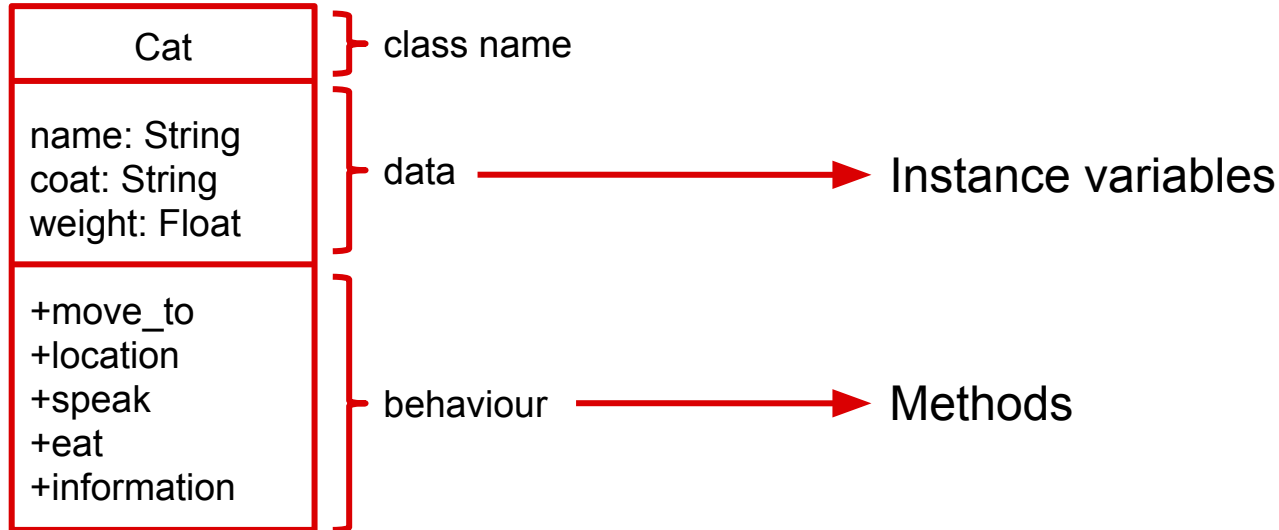


# Classes in Ruby: Behaviour

---

## Class diagram

## Ruby



# Classes in Ruby: Behaviour

---

What we want (our “contract” for the `Cat` class):

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

Meow!

the bed

Sgt. Fuzzyboots has eaten cat food, a treat and is currently here: the bed

---



# Classes in Ruby: Behaviour

---

What we want (our “contract” for the `Cat` class):

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```



Meow!

the bed

Sgt. Fuzzyboots has eaten cat food, a treat and is currently here: the bed

---

# Classes in Ruby: Behaviour

---

What we want (our “contract” for the `Cat` class):

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

Meow!

the bed

Sgt. Fuzzyboots has eaten cat food, a treat and is currently here: the bed

# Classes in Ruby: Behaviour

---

What we want (our “contract” for the `Cat` class):

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

Meow!

the bed

Sgt. Fuzzyboots has eaten cat food, a treat and is currently here: the bed

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def speak

  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def speak
    puts 'Meow!'
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def move_to(location)
    # ...
  end
end
```



# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def move_to(location)
    @location = location
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def move_to(location)
    @location = location
  end

  def location
    # ...
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def move_to(location)
    @location = location
  end

  def location
    @location
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def eat(food)
    # ...
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...

  def eat(food)
    @foods << food
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
  def initialize(name, coat, weight)
    # ...
  end

  def eat(food)
    @foods << food
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
  def initialize(name, coat, weight)
    # ...
    @foods = []
  end

  def eat(food)
    @foods << food
  end
end
```

---



# Classes in Ruby: Behaviour

---


```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

---

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```



Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
end
```

→ Sgt. Fuzzyboots has eaten cat food, a treat and is currently here: the bed

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
end
```


→ Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
  def information
    "#{@name} has eaten"
  end
end
```

 Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
  def information
    "#{@name} has eaten"
  end
end
```

→ Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

→ Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed

```
class Cat
  # ...
  def information
    "#{@name} has eaten #{@foods.join(', ')}"
  end
end
```

# Classes in Ruby: Behaviour

---

```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

```
class Cat
  # ...
  def information
    "#{@name} has eaten #{@foods.join(', ')}"
  end
end
```

→ Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed



# Classes in Ruby: Behaviour

---

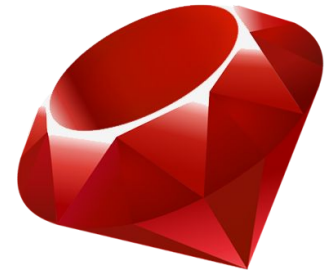
```
sgt_fuzzyboots = Cat.new('Sgt. Fuzzyboots', 'tabby', 3.1)
sgt_fuzzyboots.speak
sgt_fuzzyboots.move_to('the bed')
puts sgt_fuzzyboots.location
sgt_fuzzyboots.eat('cat food')
sgt_fuzzyboots.eat('a treat')
puts sgt_fuzzyboots.information
```

→ Sgt. Fuzzyboots has eaten  
cat food, a treat and is  
currently here: the bed

```
class Cat
  # ...
  def information
    "#{@name} has eaten #{@foods.join(', ')} and is currently here: #{@location}"
  end
end
```

---

# Time to practice



---

Let's get to it!

---