INO TECHNE

Creative Intelligence for Business

www.inotechne.com

Richard Piacentini
Chief Technical Officer & Founder

Pratompol Jiravanitchakorn
Ruby Developer

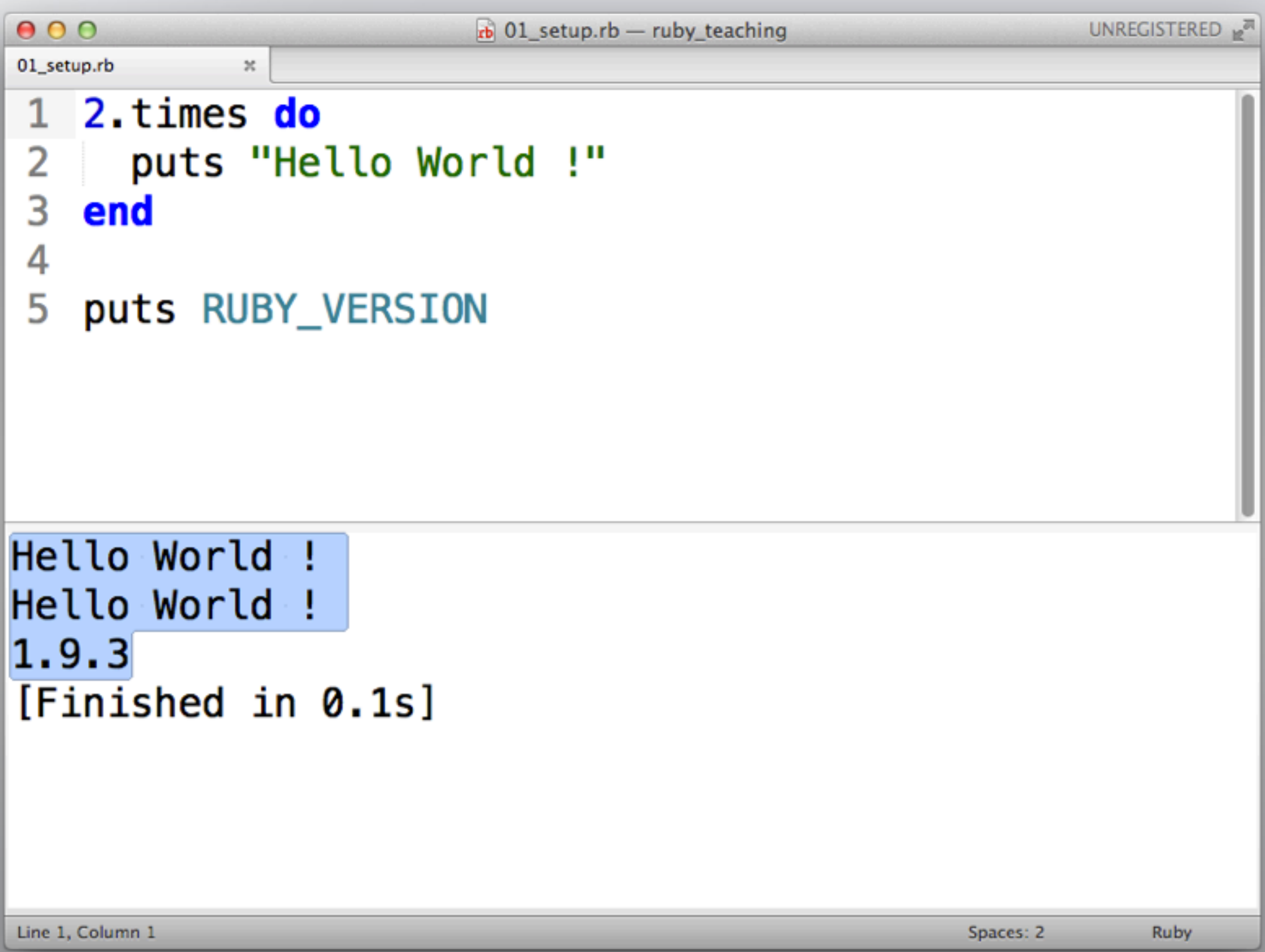
Monsiree Thassanakathitum
Ruby Developer

Virudson Thitilertdecha
Ruby Developer

Mobile Ruby on Rails Design Programming

SETUP

- Create a new directory on your Desktop: **ruby_teaching**
- Launch **Sublime Text Editor**
- Create a new file **ruby_teaching\code.rb**
- Save the file (CTRL + S)



```ruby
2.times do
  puts "Hello World !"
end

puts RUBY_VERSION
```

```
Hello World !
Hello World !
1.9.3
[Finished in 0.1s]
```

INO TECHNE

Creative Intelligence for Business

INTRODUCTION TO RUBY

Kasetsart University, Bangkok
July 19th, 2012

Mobile Ruby on Rails Design Programming

TABLE OF CONTENTS

INTRODUCTION

```ruby
puts "Sawasdee Krub !"
puts "Sawasdee Krub !"
```

```
Sawasdee Krub !
```

WHAT IS RUBY ?

WHAT IS RUBY ?

- Dynamic **Object Oriented** Programming Language

WHAT IS RUBY ?

- Dynamic **Object Oriented** Programming Language
- Inspired by **Perl** and **Smalltalk**

WHAT IS RUBY ?

- Dynamic **Object Oriented** Programming Language
- Inspired by **Perl** and **Smalltalk**
- Influenced by **Eiffel** and **Lisp**

WHAT IS RUBY ?

WHAT IS RUBY ?

- Created in Japan by **Yukihiro “Matz” Matsumoto**

WHAT IS RUBY ?

- Created in Japan by **Yukihiro “Matz” Matsumoto**
- First release on **December 21, 1995**

WHAT IS RUBY ?

- Created in Japan by **Yukihiro “Matz” Matsumoto**
- First release on **December 21, 1995**
- Standard implementation is called **“MRI”**, and is written in **C**

WHAT IS RUBY ?

- Created in Japan by **Yukihiro "Matz" Matsumoto**
- First release on **December 21, 1995**
- Standard implementation is called **"MRI"**, and is written in **C**
- **JRuby** is another popular implementation, written in **Java**

INO TECHNE

Creative Intelligence for Business

ADVANTAGES OF RUBY

www.inotechne.com

ADVANTAGES OF RUBY

ADVANTAGES OF RUBY

☑ Object Oriented

ADVANTAGES OF RUBY

☑ Object Oriented

☑ Managed memory

ADVANTAGES OF RUBY

- ✓ Object Oriented

- ✓ Managed memory
- ✓ General purpose

ADVANTAGES OF RUBY

- ✓ Object Oriented
- ✓ Managed memory
- ✓ General purpose
- ✓ Builtin REPL

ADVANTAGES OF RUBY

- ✓ Object Oriented

- ✓ Managed memory
- ✓ General purpose
- ✓ Simple and elegant
- ✓ Builtin REPL

ADVANTAGES OF RUBY

- ✓ Object Oriented
- ✓ Managed memory
- ✓ General purpose
- ✓ Interpreted
- ✓ Simple and elegant
- ✓ Builtin REPL

ADVANTAGES OF RUBY

- ✓ Object Oriented
- ✓ Dynamic
- ✓ Interpreted
- ✓ Simple and elegant
- ✓ Managed memory
- ✓ General purpose
- ✓ Builtin REPL

PHILOSOPHY OF RUBY ?

PHILOSOPHY OF RUBY ?

INO TECHNE

Creative Intelligence for Business

PHILOSOPHY OF RUBY

PHILOSOPHY OF RUBY

- Designed for **programmer's productivity**

PHILOSOPHY OF RUBY

- Designed for **programmer's productivity**
- Follow the **Principle of least surprise** (POLS)

PHILOSOPHY OF RUBY

- Designed for **programmer’s productivity**
- Follow the **Principle of least surprise** (POLS)
- Focus on the **programmer**, not the computer !

INO TECHNE

Creative Intelligence for Business

FEATURES OF RUBY

FEATURES OF RUBY

- Truly Object Oriented...**Everything is an object**

FEATURES OF RUBY

- Truly Object Oriented...**Everything is an object**
- **Compact** and **flexible** syntax

FEATURES OF RUBY

- Truly Object Oriented...**Everything is an object**
- **Compact** and **flexible** syntax
- **Memory** is **managed** by interpreter through GC

FEATURES OF RUBY

- Truly Object Oriented...**Everything is an object**
- **Compact** and **flexible** syntax
- **Memory** is **managed** by interpreter through GC
- **Dynamic Typing** (type checking performed at run-time)

FEATURES OF RUBY

- Truly Object Oriented...**Everything is an object**
- **Compact** and **flexible** syntax
- **Memory** is **managed** by interpreter through GC
- **Dynamic Typing** (type checking performed at run-time)
- **Duck Typing** (what you can do is more important than who you are !)

INO TECHNE

Creative Intelligence for Business

EVERYTHING IS OBJECT

EVERYTHING IS OBJECT

- `1.class`

EVERYTHING IS OBJECT

- `1.class`
- `(1.5).class`

EVERYTHING IS OBJECT

- `1.class`
- `(1.5).class`
- `(2*5.0).class`

EVERYTHING IS OBJECT

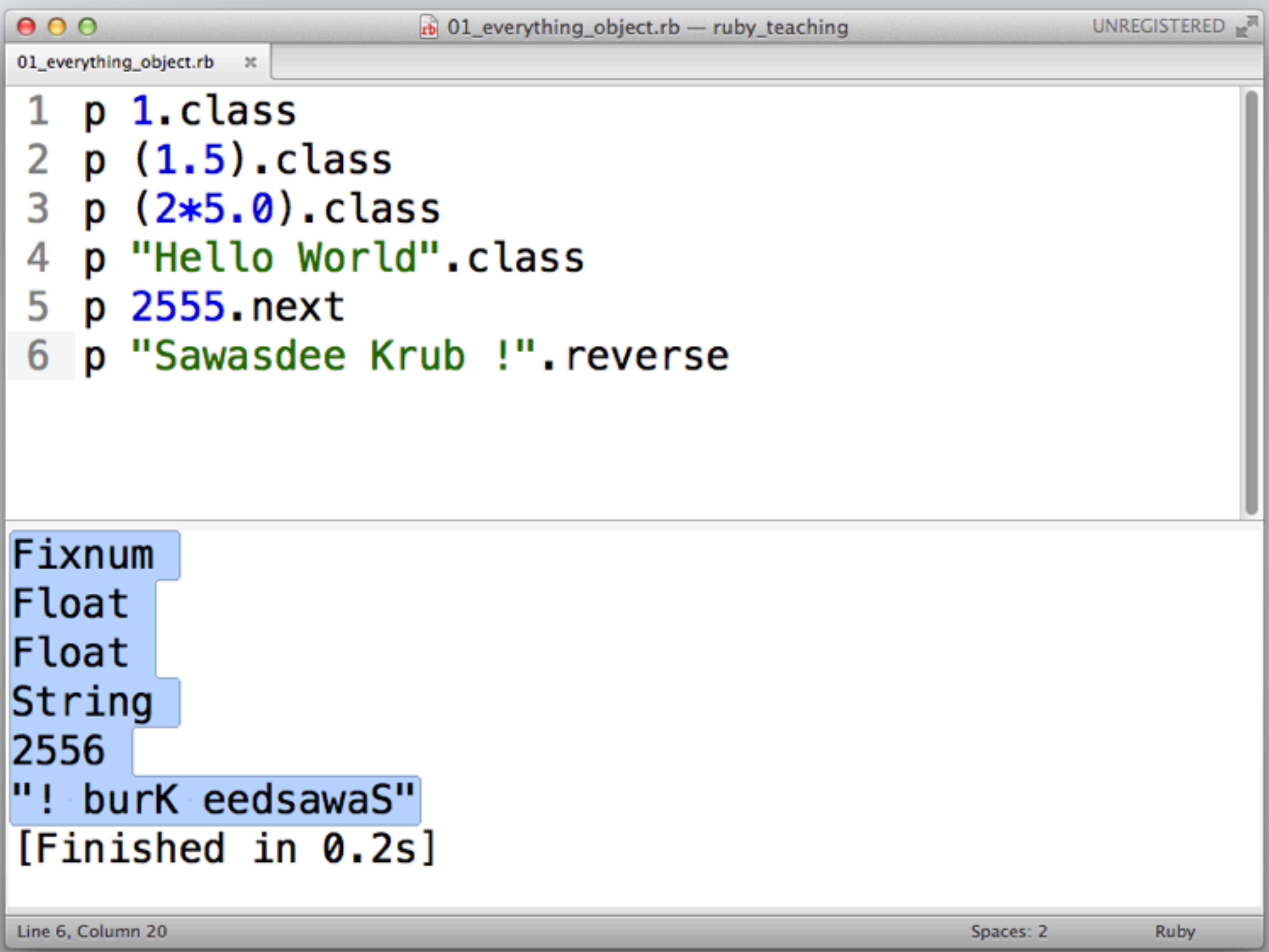
- `1.class`
- `(1.5).class`
- `(2*5.0).class`
- `"Hello World".class`

EVERYTHING IS OBJECT

- `1.class`
- `(1.5).class`
- `(2*5.0).class`
- `"Hello World".class`
- `2555.next`

EVERYTHING IS OBJECT

- `1.class`
- `(1.5).class`
- `(2*5.0).class`
- `"Hello World".class`
- `2555.next`
- `"Sawasdee Krub !".reverse`



```ruby
p 1.class
p (1.5).class
p (2*5.0).class
p "Hello World".class
p 2555.next
p "Sawasdee Krub !".reverse
```

```
Fixnum
Float
Float
String
2556
"! burK eedsawaS"
[Finished in 0.2s]
```

Creative Intelligence for Business

CODING STYLE

CODING STYLE

- Use **SPACES** to **indent** your code

CODING STYLE

- Use **SPACES** to **indent** your code
- Never use **TABS**

CODING STYLE

- Use **SPACES** to **indent** your code
- Never use **TABS**
- The rule is to use 2 spaces per **indent**

CODING STYLE

- Use **SPACES** to **indent** your code
- Never use **TABS**
- The rule is to use 2 spaces per **indent**
- but **why** ??

CODING STYLE

- Use **SPACES** to **indent** your code
- Never use **TABS**
- The rule is to use 2 spaces per **indent**
- but **why** ??
- Because it's the **smallest** indentation that you can **easily see** !

VARIABLES

LOCAL VARIABLES

LOCAL VARIABLES

- Start with a **lower case letter** or an **underscore** _

LOCAL VARIABLES

- Start with a **lower case letter** or an **underscore** _
- Example: *results*, *first_name*, *_ary*

LOCAL VARIABLES

- Start with a **lower case letter** or an **underscore** _
- Example: *results*, *first_name*, *_ary*
- Ruby variables **naming convention** called: **snake_case**

LOCAL VARIABLES

- Start with a **lower case letter** or an **underscore** _
- Example: *results*, *first_name*, *_ary*
- Ruby variables **naming convention** called: **snake_case**
- Multiple words should be separated by an underscore

LOCAL VARIABLES

- Start with a **lower case letter** or an **underscore** _
- Example: *results*, *first_name*, *_ary*
- Ruby variables **naming convention** called: **snake_case**
- Multiple words should be separated by an underscore
- **Never** use a **capital letter** to start a local variable name !

INSTANCE VARIABLES

INSTANCE VARIABLES

- Used to store information per **object instance**

INSTANCE VARIABLES

- Used to store information per **object instance**
- Must start with an **@** sign

INSTANCE VARIABLES

- Used to store information per **object instance**
- Must start with an **@** sign
- Example: *@first_name*, *@results*

INSTANCE VARIABLES

- Used to store information per **object instance**
- Must start with an **@** sign
- Example: *@first_name*, *@results*
- Can start with an uppercase letter: *@First_name* ...

INSTANCE VARIABLES

- Used to store information per **object instance**
- Must start with an **@** sign
- Example: *@first_name*, *@results*
- Can start with an uppercase letter: *@First_name* ...
- ... But you should **avoid** it

CLASS VARIABLES

CLASS VARIABLES

- Used to store information per **class hierarchy**

CLASS VARIABLES

- Used to store information per **class hierarchy**
- Must start with a double **@@** sign

CLASS VARIABLES

- Used to store information per **class hierarchy**
- Must start with a double **@@** sign
- Example: *@@counter, @@total_results*

CLASS VARIABLES

- Used to store information per **class hierarchy**
- Must start with a double **@@** sign
- Example: *@@counter, @@total_results*
- Can start with an uppercase letter: *@@Count* ...

CLASS VARIABLES

- Used to store information per **class hierarchy**
- Must start with a double **@@** sign
- Example: *@@counter, @@total_results*
- Can start with an uppercase letter: *@@Count* ...
- ... But, here also, you should **avoid** it

GLOBAL VARIABLES

GLOBAL VARIABLES

- Easily identifiable by their leading **$** sign

GLOBAL VARIABLES

- Easily identifiable by their leading **$** sign
- Example: *$LOAD_PATH, $:, $/*

GLOBAL VARIABLES

- Easily identifiable by their leading **$** sign
- Example: *$LOAD_PATH, $:, $/*
- **But avoid creating new global variables !!**

GLOBAL VARIABLES

- Easily identifiable by their leading **$** sign
- Example: *$LOAD_PATH, $:, $/*
- **But avoid creating new global variables !!**
- It's (very) **BAD practice**

INO TECHNE

Creative Intelligence for Business

REMEMBER THIS

REMEMBER THIS

- Ruby variables type is determined by the **first character(s)** of their name

REMEMBER THIS

- Ruby variables type is determined by the **first character(s)** of their name

`$GLOBAL_VARIABLE`	`$LOAD_PATH, $bad_practice`
`@@class_variable`	`@@total_count, @@TOTAL_COUNT`
`@instance_variable`	`@results, @first_name`
`CONSTANT`	`VERSION, PI`
`local_variable`	`x, _my_string`

METHODS

METHOD DEFINITION

METHOD DEFINITION

- Follow the **same naming rules** as **local variables**

METHOD DEFINITION

- Follow the **same naming rules** as **local variables**
- Start with keyword *def* followed by **lower case letter** or an **underscore _**

METHOD DEFINITION

- Follow the **same naming rules** as **local variables**
- Start with keyword *def* followed by **lower case letter** or an **underscore** _
- End by keyword *end*

METHOD DEFINITION

- Follow the **same naming rules** as **local variables**
- Start with keyword *def* followed by **lower case letter** or an **underscore** _
- End by keyword *end*
- Method **name** can end with **?, ! or =**

METHOD DEFINITION

- Follow the **same naming rules** as **local variables**
- Start with keyword *def* followed by **lower case letter** or an **underscore** _
- End by keyword *end*
- Method **name** can end with **?, ! or =**
- Example: *create*, *_get_user*, *is_active?*, *replace!*, *result=(value)*

INO TECHNE

Creative Intelligence for Business

NAMING CONVENTION

NAMING CONVENTION

- Methods that returns *true* or *false* should end by *?*

NAMING CONVENTION

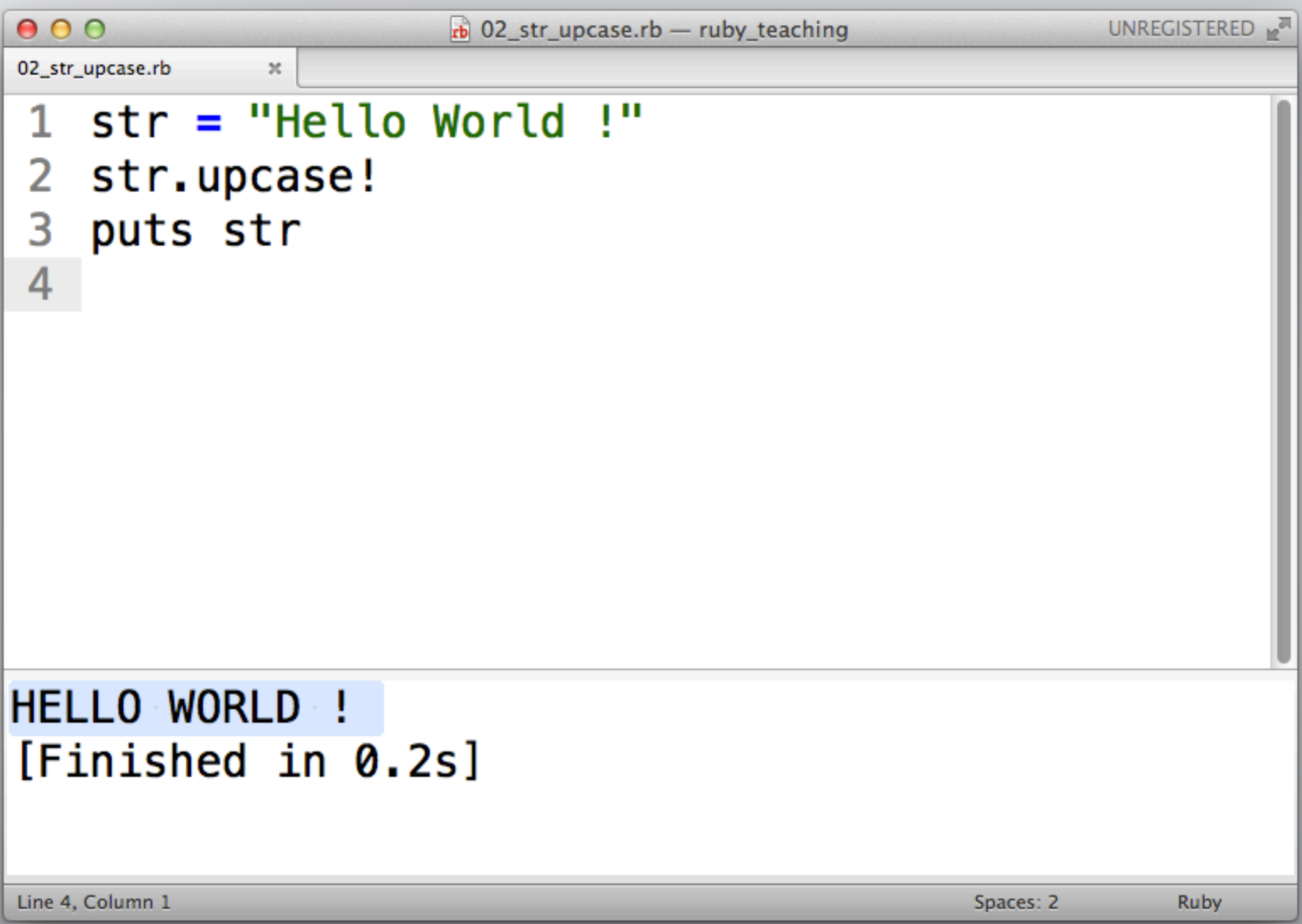
- Methods that returns *true* or *false* should end by *?*
- Methods that modify object in place should end by !

NAMING CONVENTION

- Methods that returns *true* or *false* should end by *?*
- Methods that modify object in place should end by !

```
str = "Hello World !"
str.upcase!
puts str
```

Creative Intelligence for Business



```ruby
str = "Hello World !"
str.upcase!
puts str
```

```
HELLO WORLD !
[Finished in 0.2s]
```

METHODS EXAMPLE

METHODS EXAMPLE

```ruby
def say(message)
  puts "Ruby says: #{message}"
end
```

METHODS EXAMPLE

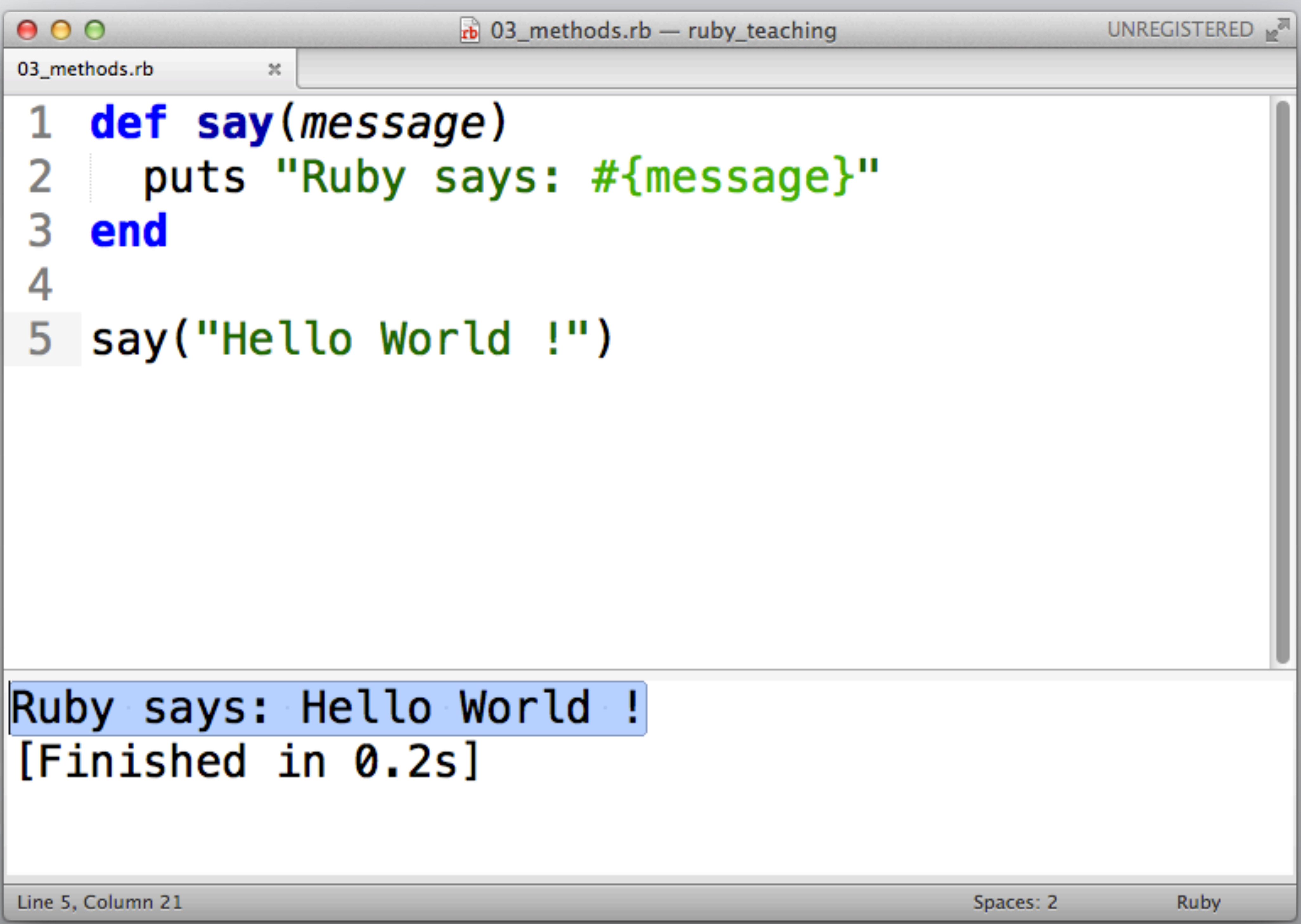
```
def say(message)
  puts "Ruby says: #{message}"
end


def exclude?(str, value)
   !str.include?(value)
end
```

METHODS EXAMPLE

```
def say(message)
  puts "Ruby says: #{message}"
end


def exclude?(str, value)  # no explicit return needed, last evaluated
   !str.include?(value)   expression is returned by default.
end
```

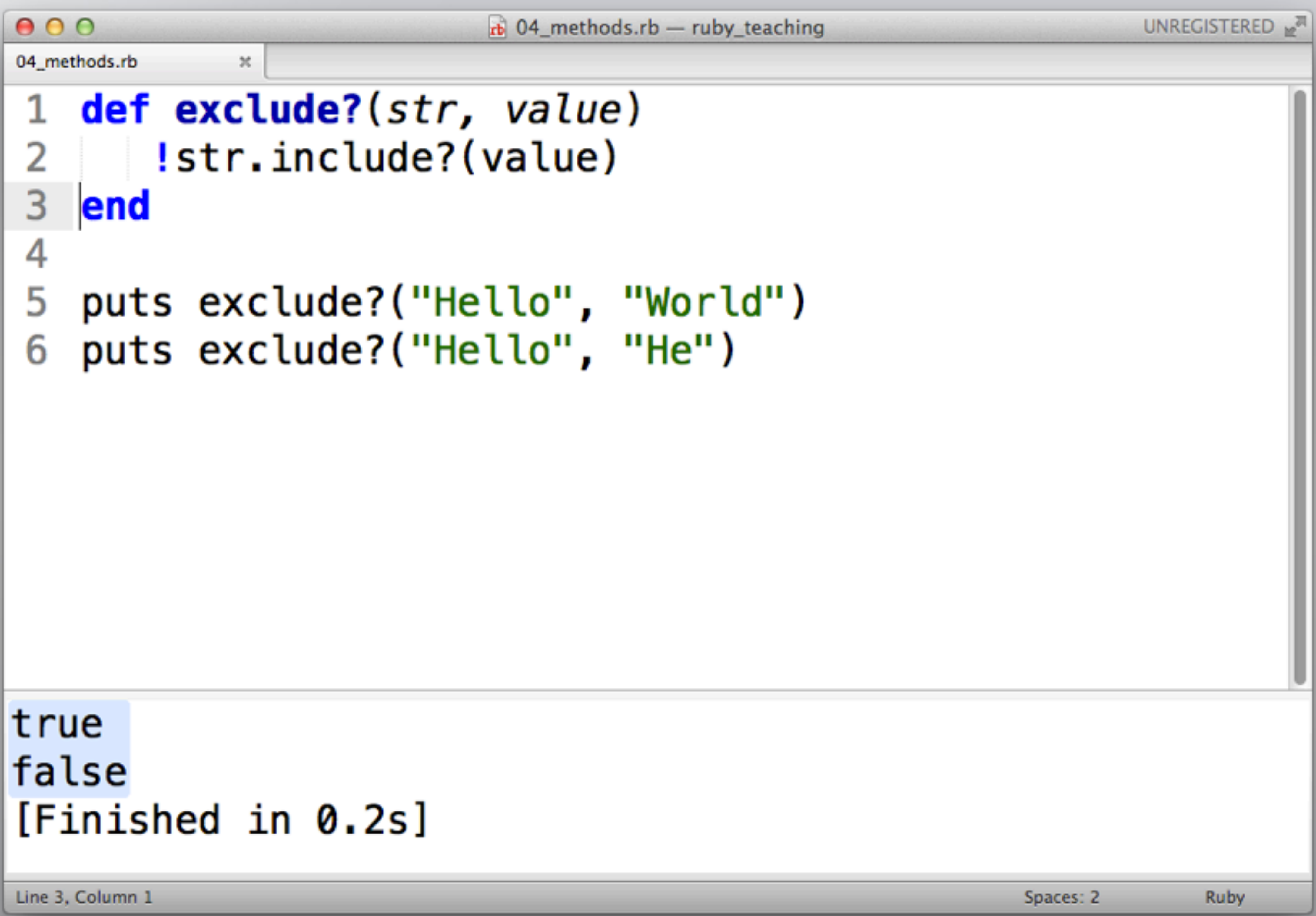


```ruby
def say(message)
  puts "Ruby says: #{message}"
end

say("Hello World !")
```

```
Ruby says: Hello World !
[Finished in 0.2s]
```

Creative Intelligence for Business



04_methods.rb — ruby_teaching

```ruby
def exclude?(str, value)
    !str.include?(value)
end

puts exclude?("Hello", "World")
puts exclude?("Hello", "He")
```

```
true
false
[Finished in 0.2s]
```

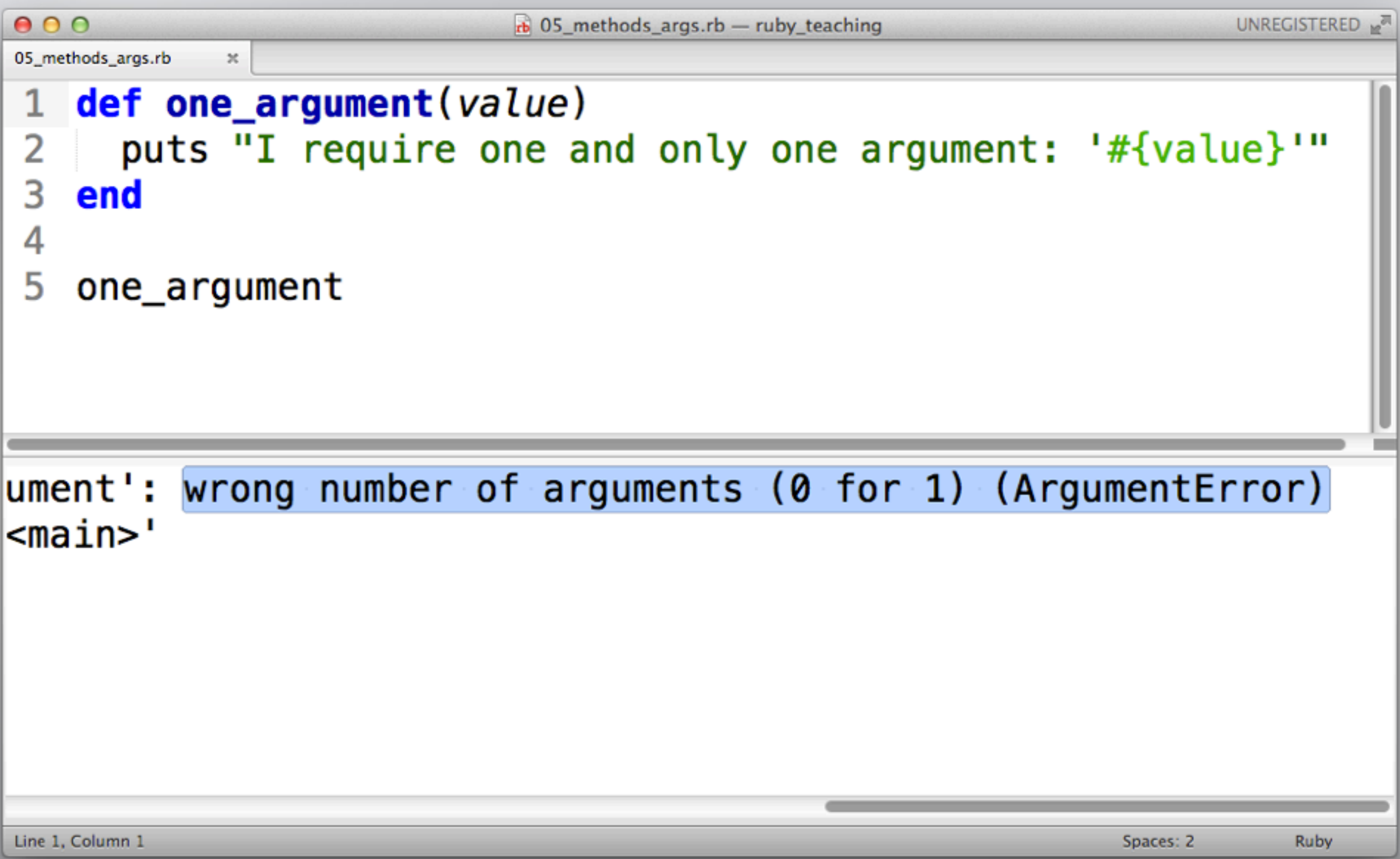
METHOD ARGUMENTS #1

- You have to supply the **correct number** of **arguments**

METHOD ARGUMENTS #1

- You have to supply the **correct number** of **arguments**

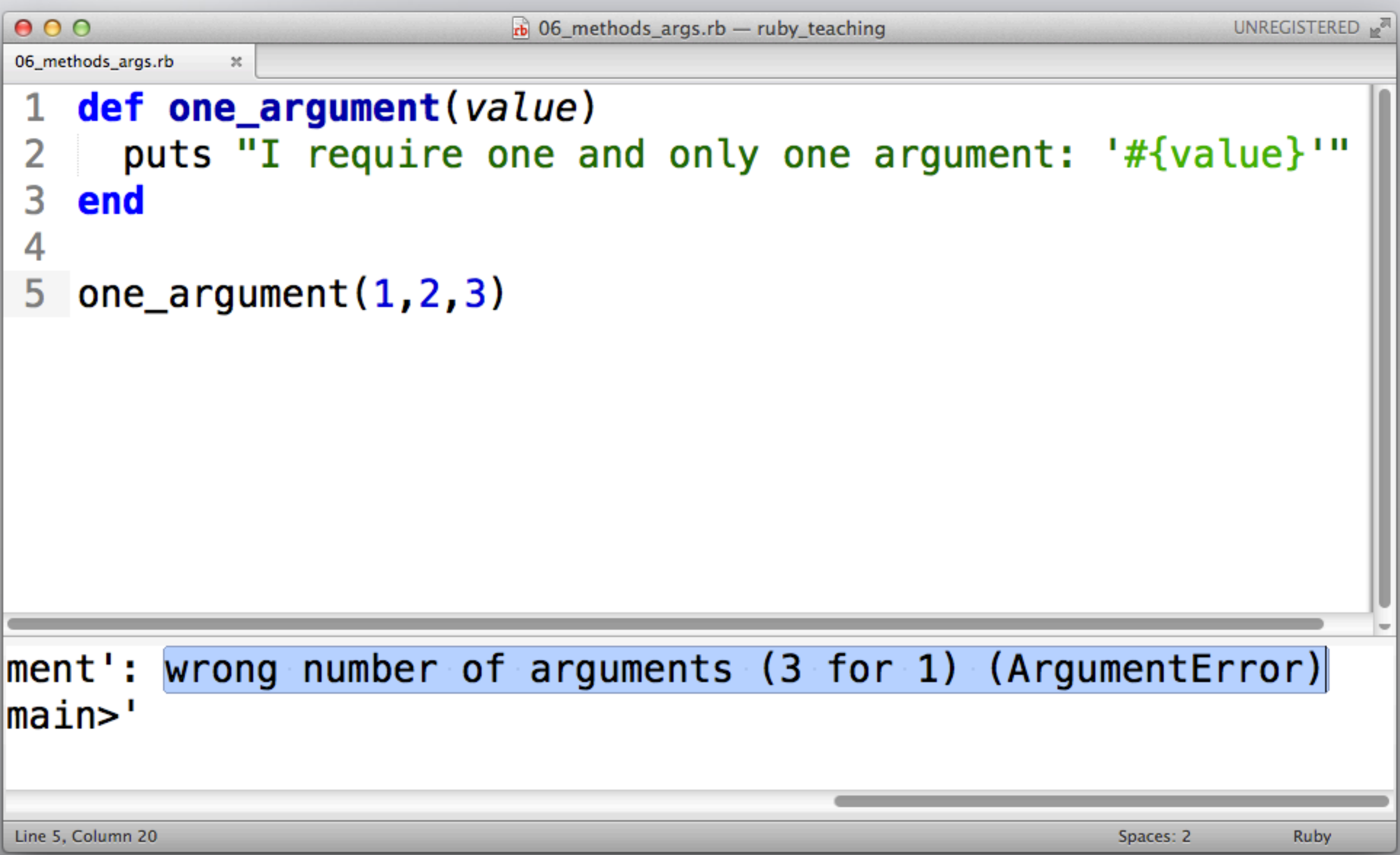
```
def one_argument(value)
 puts "I require one and only one argument: '#{value}'"
end
```



```ruby
def one_argument(value)
  puts "I require one and only one argument: '#{value}'"
end

one_argument
```

```
ument': wrong number of arguments (0 for 1) (ArgumentError)
<main>'
```



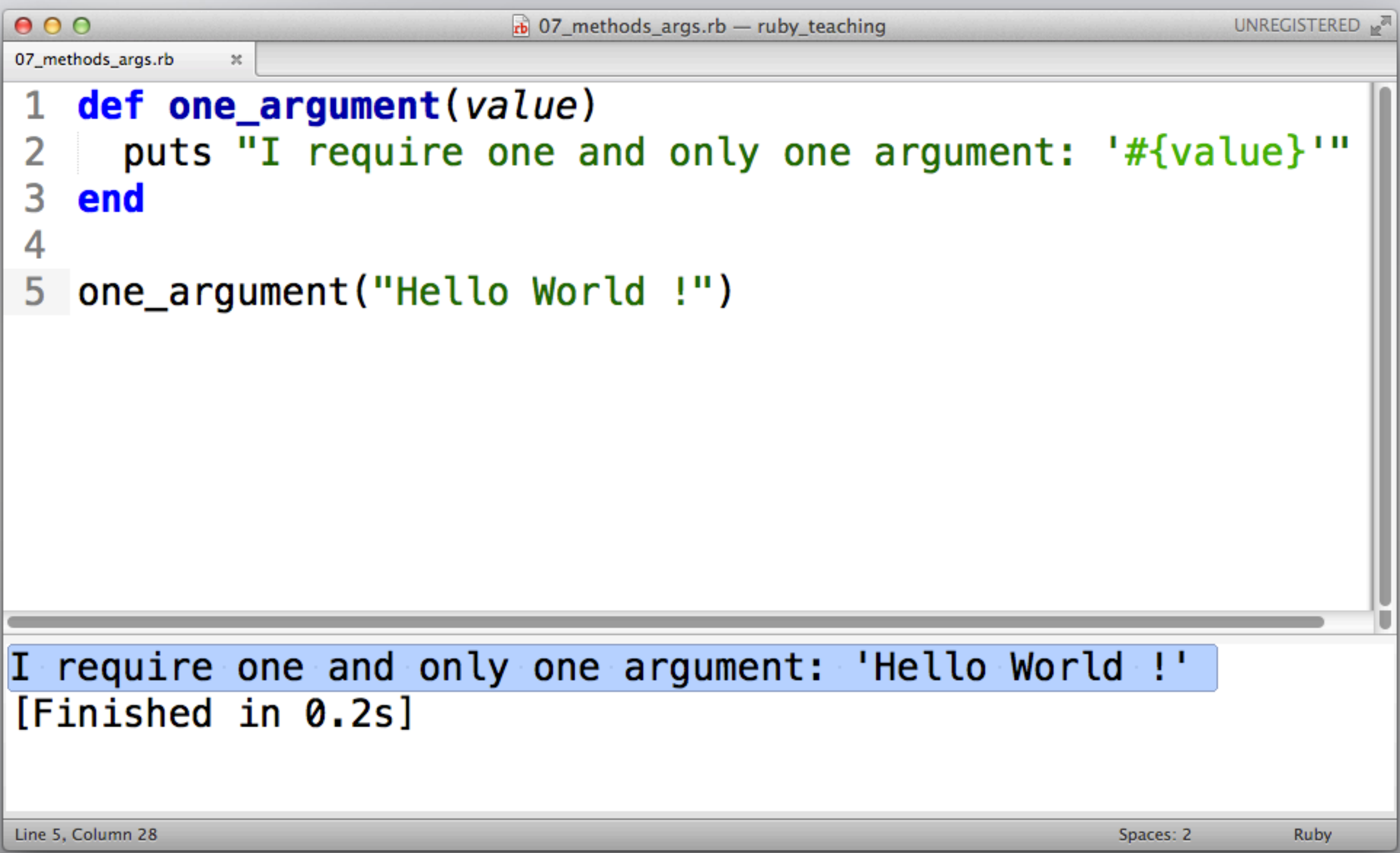
06_methods_args.rb — ruby_teaching

```ruby
def one_argument(value)
  puts "I require one and only one argument: '#{value}'"
end

one_argument(1,2,3)
```

```
ment': wrong number of arguments (3 for 1) (ArgumentError)
main>'
```

Line 5, Column 20 Spaces: 2 Ruby



```ruby
def one_argument(value)
  puts "I require one and only one argument: '#{value}'"
end

one_argument("Hello World !")
```

```
I require one and only one argument: 'Hello World !'
[Finished in 0.2s]
```

MULTIPLE VALUES

MULTIPLE VALUES

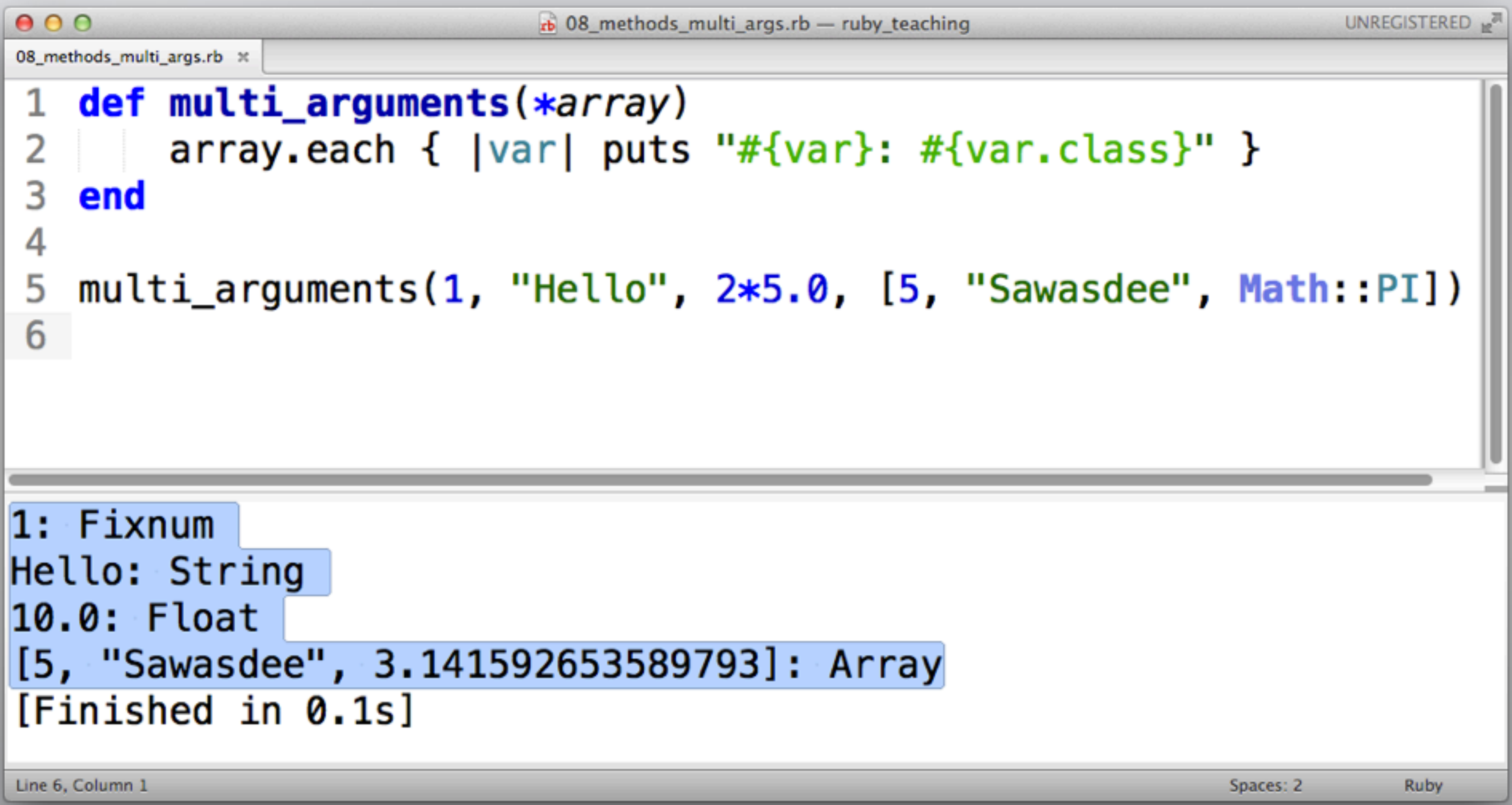
- You can define **methods** that allows **any number** of **arguments**

MULTIPLE VALUES

- You can define **methods** that allows **any number** of **arguments**

```
def multi_arguments(*array)
    array.each { |var| puts "#{var}: #{var.class}" }
end
```

Creative Intelligence for Business



```ruby
def multi_arguments(*array)
    array.each { |var| puts "#{var}: #{var.class}" }
end

multi_arguments(1, "Hello", 2*5.0, [5, "Sawasdee", Math::PI])
```

```
1: Fixnum
Hello: String
10.0: Float
[5, "Sawasdee", 3.141592653589793]: Array
[Finished in 0.1s]
```

DEFAULT VALUES

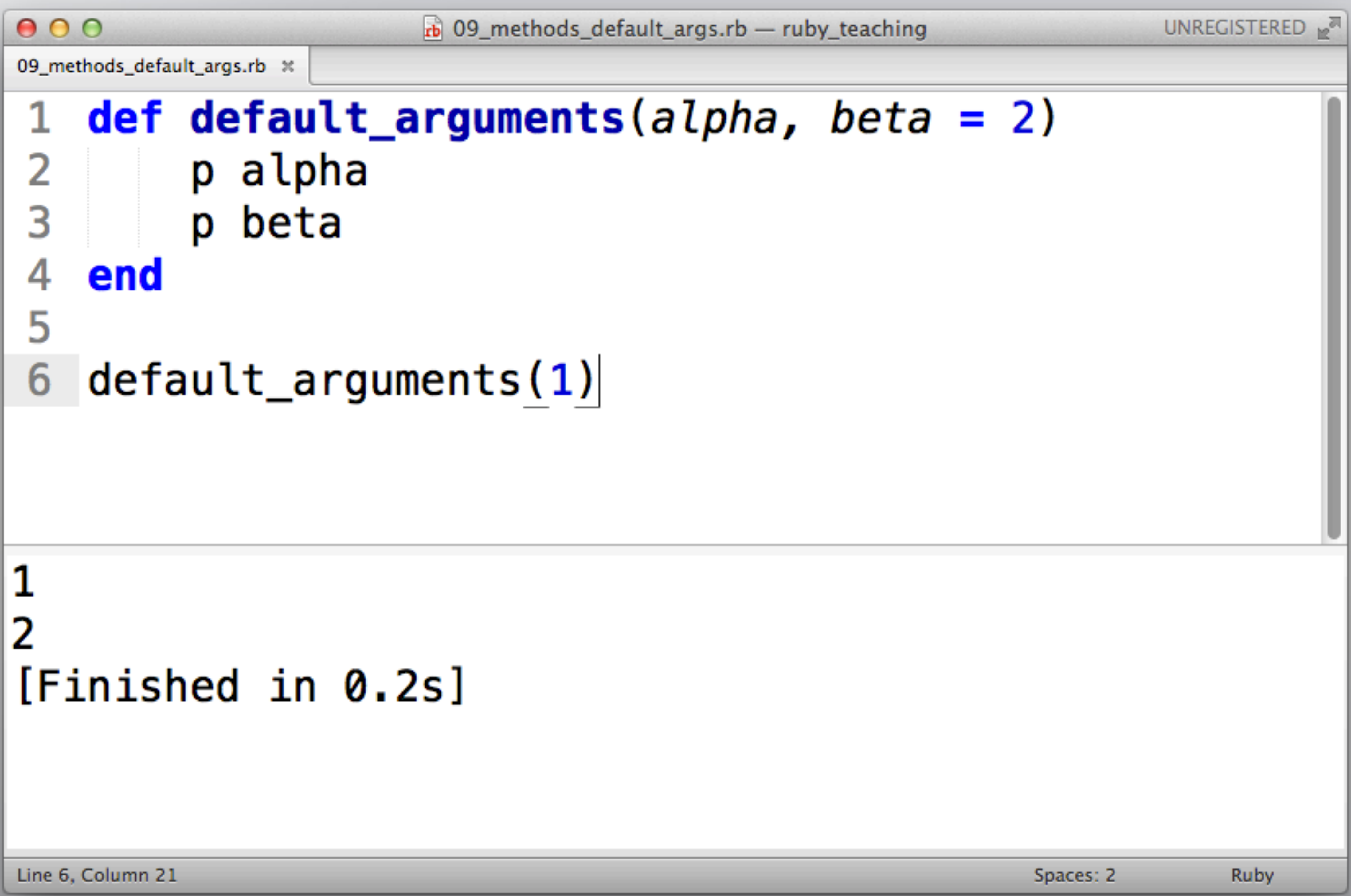
DEFAULT VALUES

- You can define **methods** with **default value** for **arguments**

DEFAULT VALUES

- You can define **methods** with **default value** for **arguments**

```
def default_arguments(alpha, beta = 2)
    p alpha
    p beta
end
```

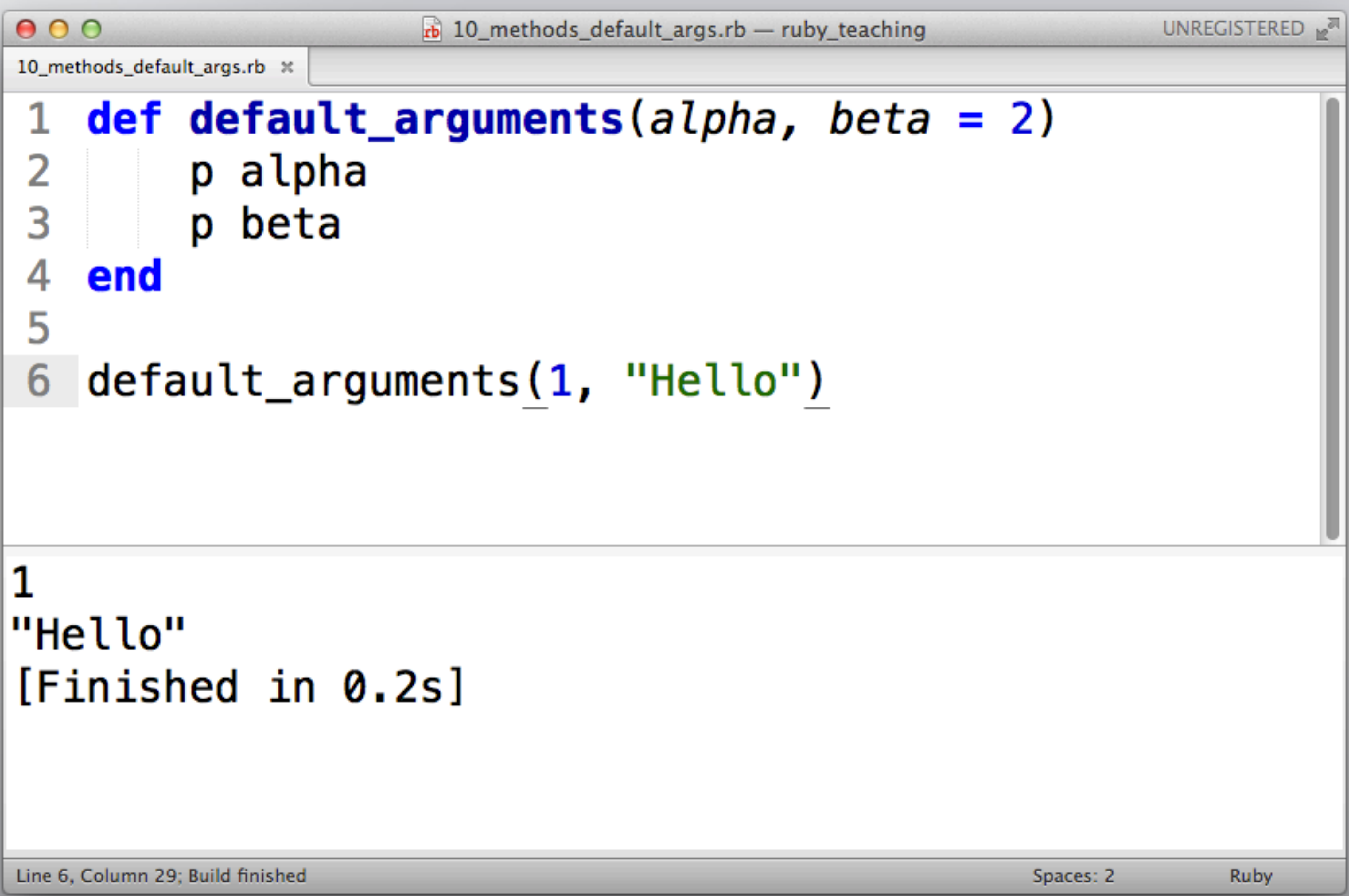

09_methods_default_args.rb — ruby_teaching
UNREGISTERED
09_methods_default_args.rb
Line 6, Column 21
Spaces: 2
Ruby

```ruby
def default_arguments(alpha, beta = 2)
    p alpha
    p beta
end

default_arguments(1)
```

```
1
2
[Finished in 0.2s]
```

Creative Intelligence for Business


10_methods_default_args.rb — ruby_teaching
UNREGISTERED
10_methods_default_args.rb

```ruby
def default_arguments(alpha, beta = 2)
    p alpha
    p beta
end

default_arguments(1, "Hello")
```

```
1
"Hello"
[Finished in 0.2s]
```

Line 6, Column 29; Build finished
Spaces: 2
Ruby

INO TECHNE

Creative Intelligence for Business

REMEMBER THIS

REMEMBER THIS

- Method definition **start** by the **keyword** *def*

REMEMBER THIS

- Method definition **start** by the **keyword** *def*
- Method definition **end** by the **keyword** *end*

REMEMBER THIS

- Method definition **start** by the **keyword** *def*
- Method definition **end** by the **keyword** *end*
- Method **name** start by **lower case letter** or an **underscore** _

REMEMBER THIS

- Method definition **start** by the **keyword** *def*
- Method definition **end** by the **keyword** *end*
- Method **name** start by **lower case letter** or an **underscore** _
- Method **name** can end with **?, ! or =**

INO TECHNE

Creative Intelligence for Business

CLASSES

Mobile Ruby on Rails Design Programming

Creative Intelligence for Business

CLASS DEFINITION

CLASS DEFINITION

- A **class** is a construct used to **create instances** of itself...

CLASS DEFINITION

- A **class** is a construct used to **create instances** of itself...
- ... referred to as *class instances*, *instance objects* or simply **objects**

CLASS DEFINITION

- A **class** is a construct used to **create instances** of itself...
- ... referred to as *class instances*, *instance objects* or simply **objects**
- Each **object** can have his **own state** and **behavior**

CLASS DEFINITION

- A **class** is a construct used to **create instances** of itself...
- ... referred to as *class instances*, *instance objects* or simply **objects**
- Each **object** can have his **own state** and **behavior**
- **Instance variables** enable an object to **maintain** his **state**

CLASS DEFINITION

- A **class** is a construct used to **create instances** of itself...
- ... referred to as *class instances*, *instance objects* or simply **objects**
- Each **object** can have his **own state** and **behavior**
- **Instance variables** enable an object to **maintain** his **state**
- **Methods** enable the **behavior** of **objects**

Creative Intelligence for Business

RUBY CLASS

RUBY CLASS

- Start with keyword *class* followed by an **uppercase letter**

RUBY CLASS

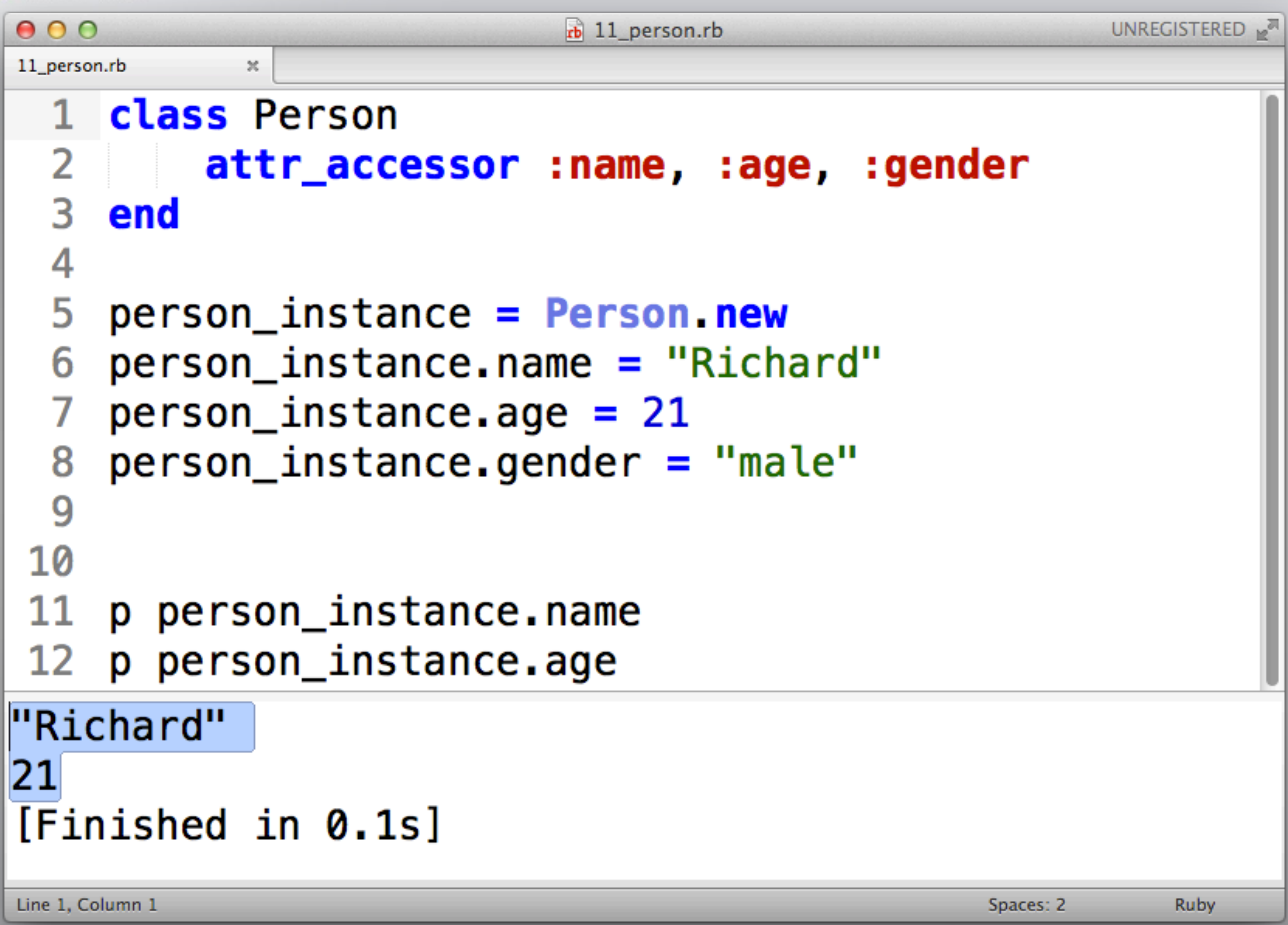
- Start with keyword *class* followed by an **uppercase letter**
- End by keyword *end*

CLASS EXAMPLE

```
class Person
    attr_accessor :name, :age, :gender
end

person_instance = Person.new
person_instance.name = "Richard"
person_instance.age = 21
person_instance.gender = "male"
```

Creative Intelligence for Business



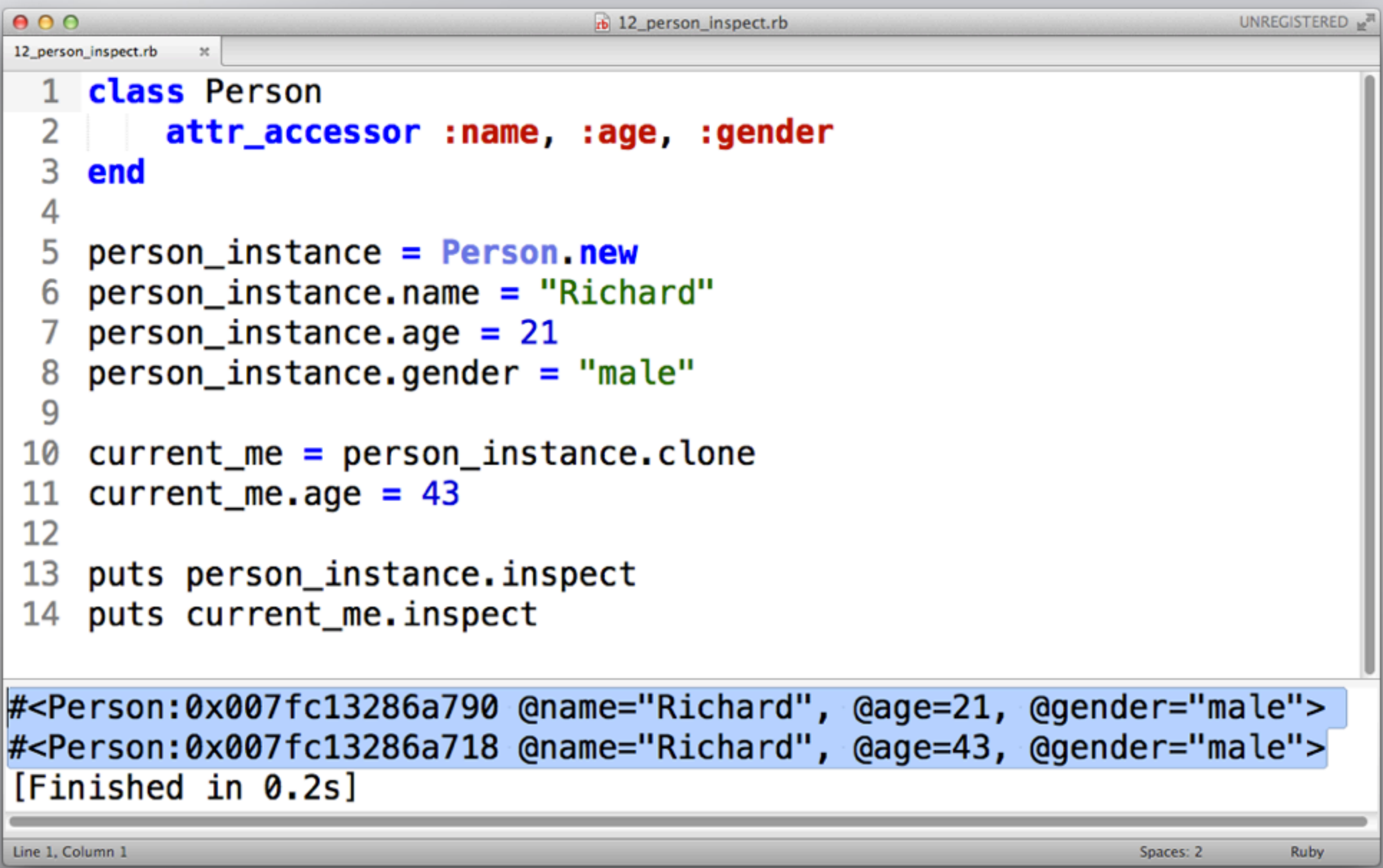
```ruby
class Person
    attr_accessor :name, :age, :gender
end

person_instance = Person.new
person_instance.name = "Richard"
person_instance.age = 21
person_instance.gender = "male"


p person_instance.name
p person_instance.age
```

```
"Richard"
21
[Finished in 0.1s]
```

Creative Intelligence for Business



12_person_inspect.rb

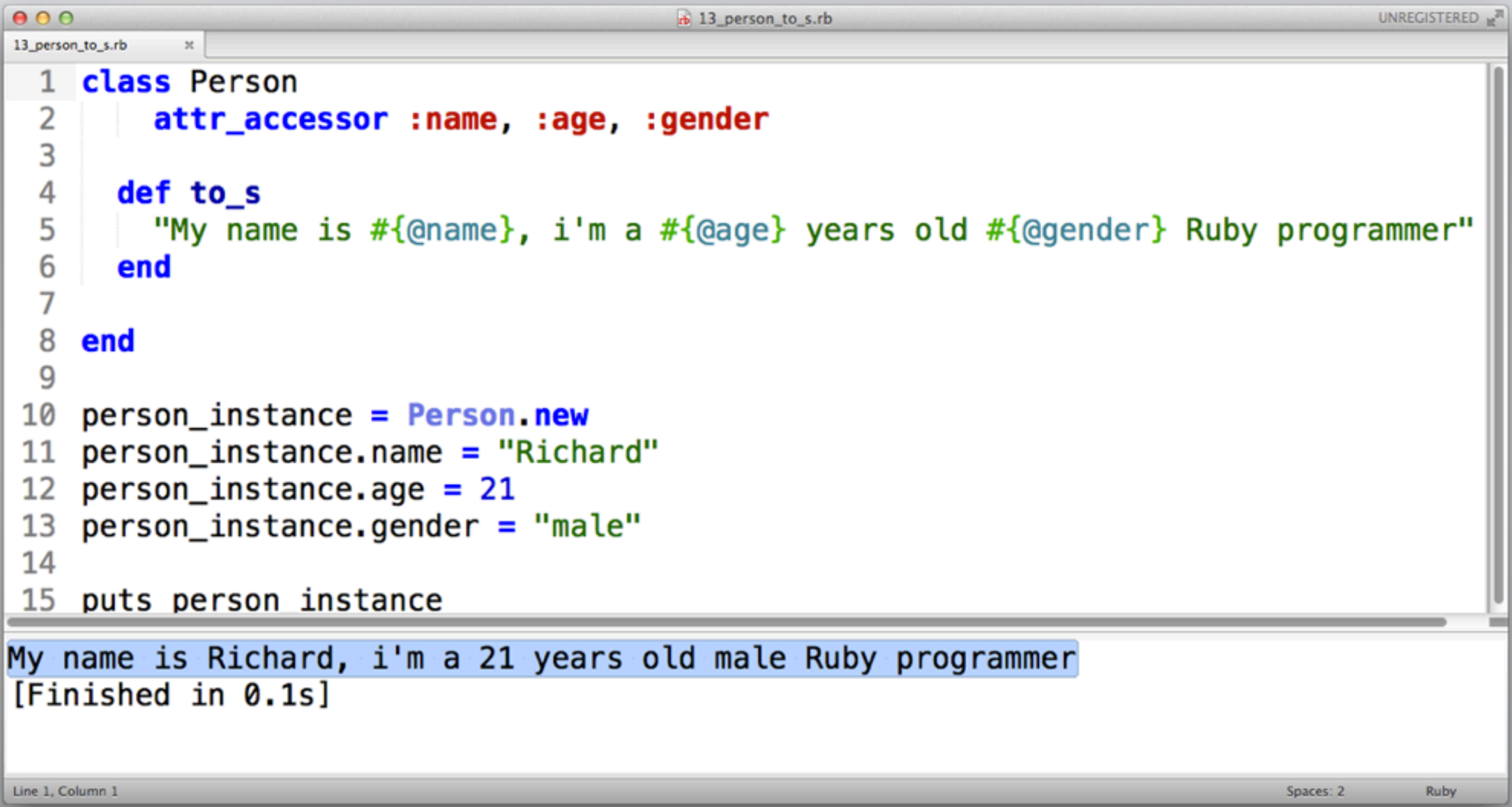
```ruby
class Person
    attr_accessor :name, :age, :gender
end

person_instance = Person.new
person_instance.name = "Richard"
person_instance.age = 21
person_instance.gender = "male"

current_me = person_instance.clone
current_me.age = 43

puts person_instance.inspect
puts current_me.inspect
```

```
#<Person:0x007fc13286a790 @name="Richard", @age=21, @gender="male">
#<Person:0x007fc13286a718 @name="Richard", @age=43, @gender="male">
[Finished in 0.2s]
```



```ruby
class Person
    attr_accessor :name, :age, :gender

  def to_s
    "My name is #{@name}, i'm a #{@age} years old #{@gender} Ruby programmer"
  end

end

person_instance = Person.new
person_instance.name = "Richard"
person_instance.age = 21
person_instance.gender = "male"

puts person_instance
```

```
My name is Richard, i'm a 21 years old male Ruby programmer
[Finished in 0.1s]
```

MORE ABOUT CLASSES

CLASSES IN DETAIL

CLASSES IN DETAIL

- Ruby classes are **executable code**

CLASSES IN DETAIL

- Ruby classes are **executable code**
- The *new* method is a **constructor**

CLASSES IN DETAIL

- Ruby classes are **executable code**
- The *new* method is a **constructor**
- **Classes** are **named** with **constants**

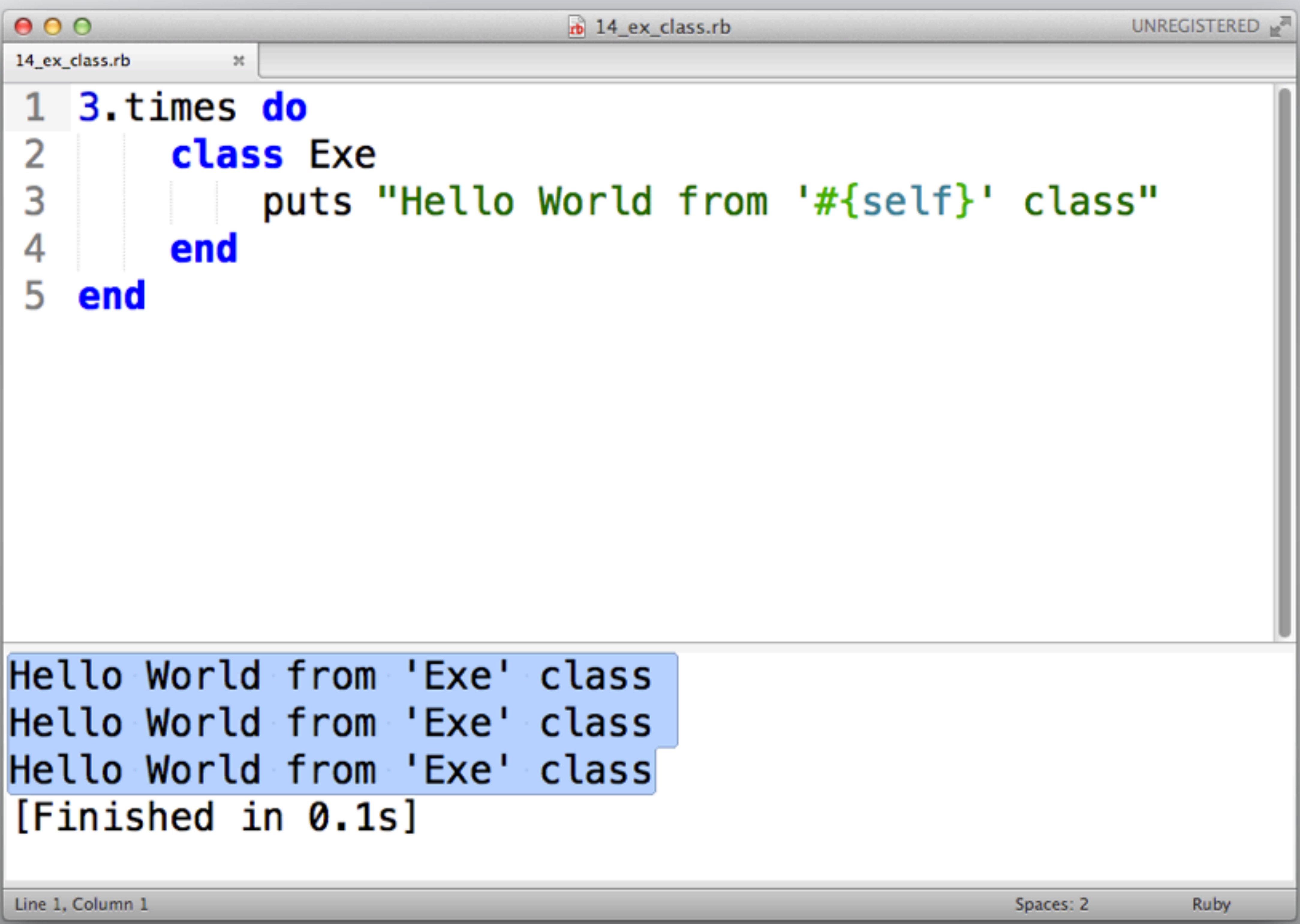
CLASSES IN DETAIL

- Ruby classes are **executable code**
- The *new* method is a **constructor**
- **Classes** are **named** with **constants**
- In Ruby **classes** are **objects** (ouch !)

CLASS IS EXECUTABLE CODE

```ruby
3.times do
    class Exe
        puts "Hello World from '#{self}' class"
    end
end
```

Creative Intelligence for Business



```ruby
3.times do
    class Exe
        puts "Hello World from '#{self}' class"
    end
end
```

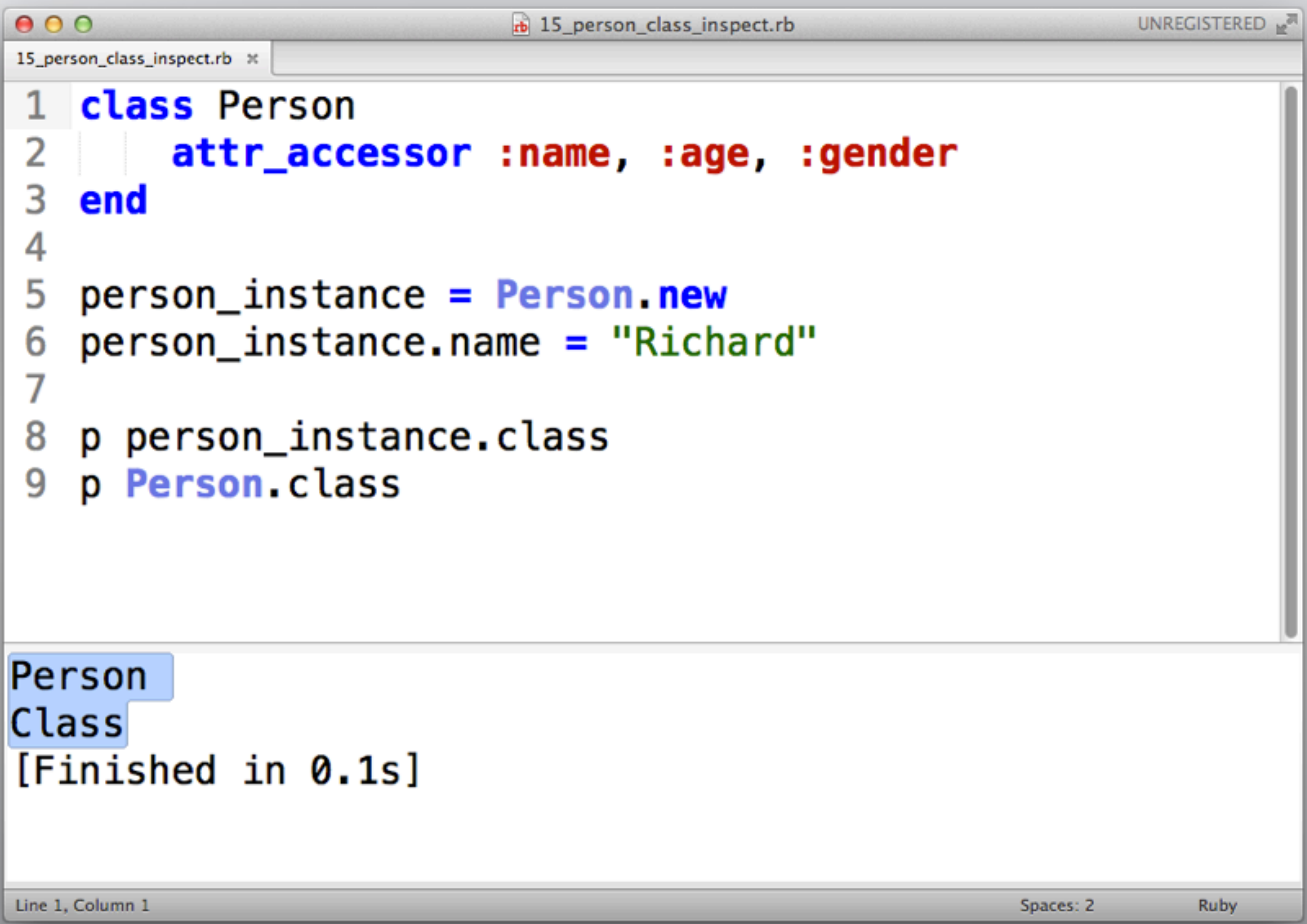
```
Hello World from 'Exe' class
Hello World from 'Exe' class
Hello World from 'Exe' class
[Finished in 0.1s]
```

A CLASS IS AN OBJECT

```
class Person
    attr_accessor :name, :age, :gender
end

person_instance = Person.new
person_instance.name = "Richard"
```

Creative Intelligence for Business



```ruby
class Person
    attr_accessor :name, :age, :gender
end

person_instance = Person.new
person_instance.name = "Richard"

p person_instance.class
p Person.class
```

```
Person
Class
[Finished in 0.1s]
```

REOPENING CLASSES

REOPENING CLASSES

- Ruby **classes** implementation is **not closed**

REOPENING CLASSES

- Ruby **classes** implementation is **not closed**
- Ruby **classes** can be **reopened** and **modified**

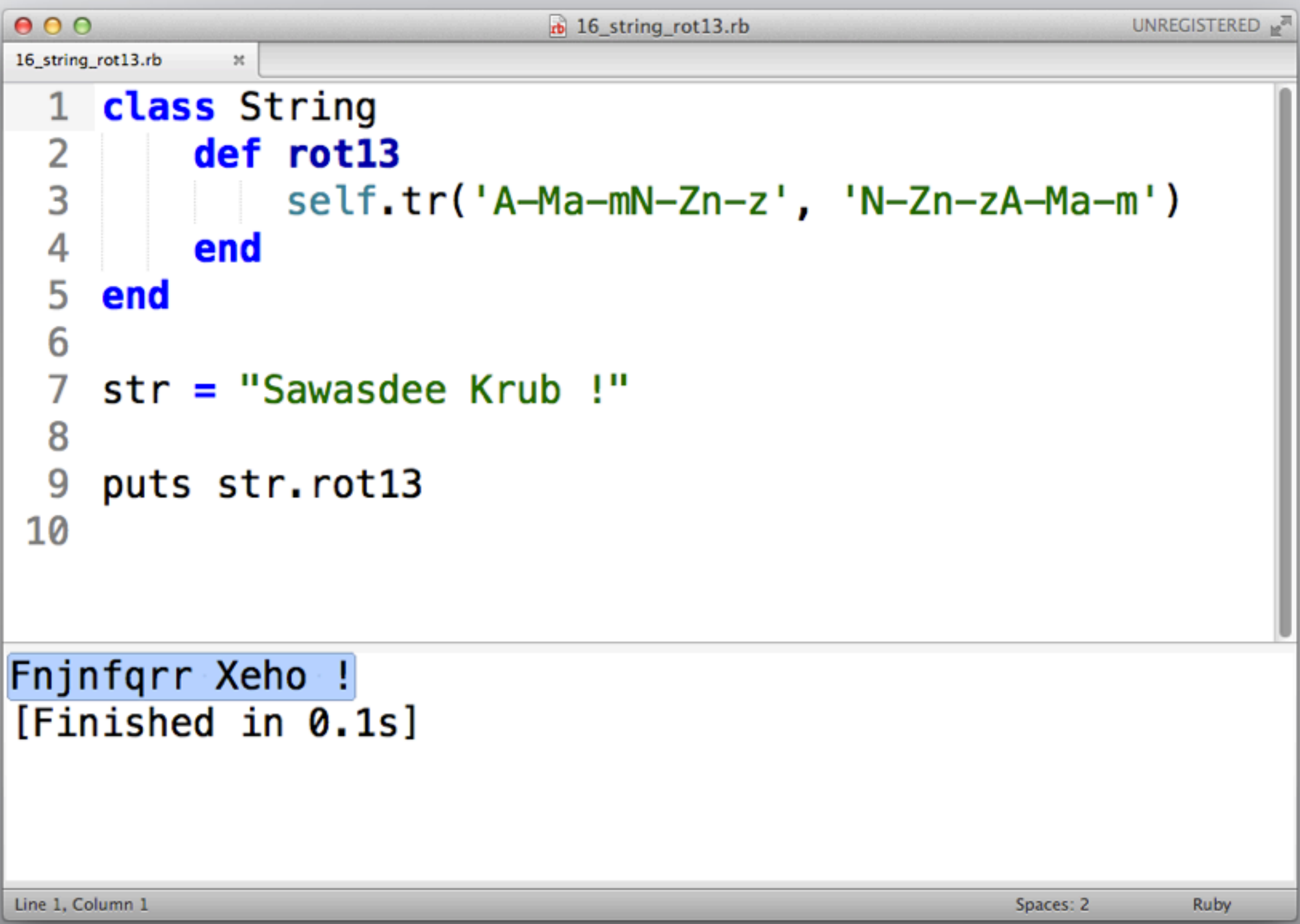
REOPENING CLASSES

- Ruby **classes** implementation is **not closed**
- Ruby **classes** can be **reopened** and **modified**
- **Any class** can be reopened !

REOPENING CLASSES

- Ruby **classes** implementation is **not closed**
- Ruby **classes** can be **reopened** and **modified**
- **Any class** can be reopened !
- This **feature** make Ruby **incredibly flexible**

Creative Intelligence for Business



```ruby
class String
    def rot13
        self.tr('A-Ma-mN-Zn-z', 'N-Zn-zA-Ma-m')
    end
end

str = "Sawasdee Krub !"

puts str.rot13

```

```
Fnjnfqrr Xeho !
[Finished in 0.1s]
```

EXERCISE

EXERCISE

- Add the **exclude?** method to all **String objects**

EXERCISE

- Add the **exclude?** method to all **String objects**
- Original implementation below

EXERCISE

- Add the **exclude?** method to all **String objects**
- Original implementation below

```
def exclude?(str, value)
    !str.include?(value)
end
```

INO TECHNE

Creative Intelligence for Business

COMMAND LINE SWITCHES

Switch	Description	Example of usage
`-c`	Check the syntax of a program file without executing the program	`ruby -c c2f.rb`
`-w`	Give warning messages during program execution	`ruby -w c2f.rb`
`-e`	Execute the code provided in quotation marks on the command line	`ruby -e 'puts "Code demo!"'`
`-v`	Show Ruby version information, and execute the program in verbose mode	`ruby -v`
`-l`	Line mode: print a newline after every line of output	`ruby -le 'print "+ newline!"'`
`-r`*`name`*	Require the named feature	`ruby -rprofile`
`--version`	Show Ruby version information	`ruby --version`

INO TECHNE

Creative Intelligence for Business

Mobile

Ruby on Rails

Design

Programming