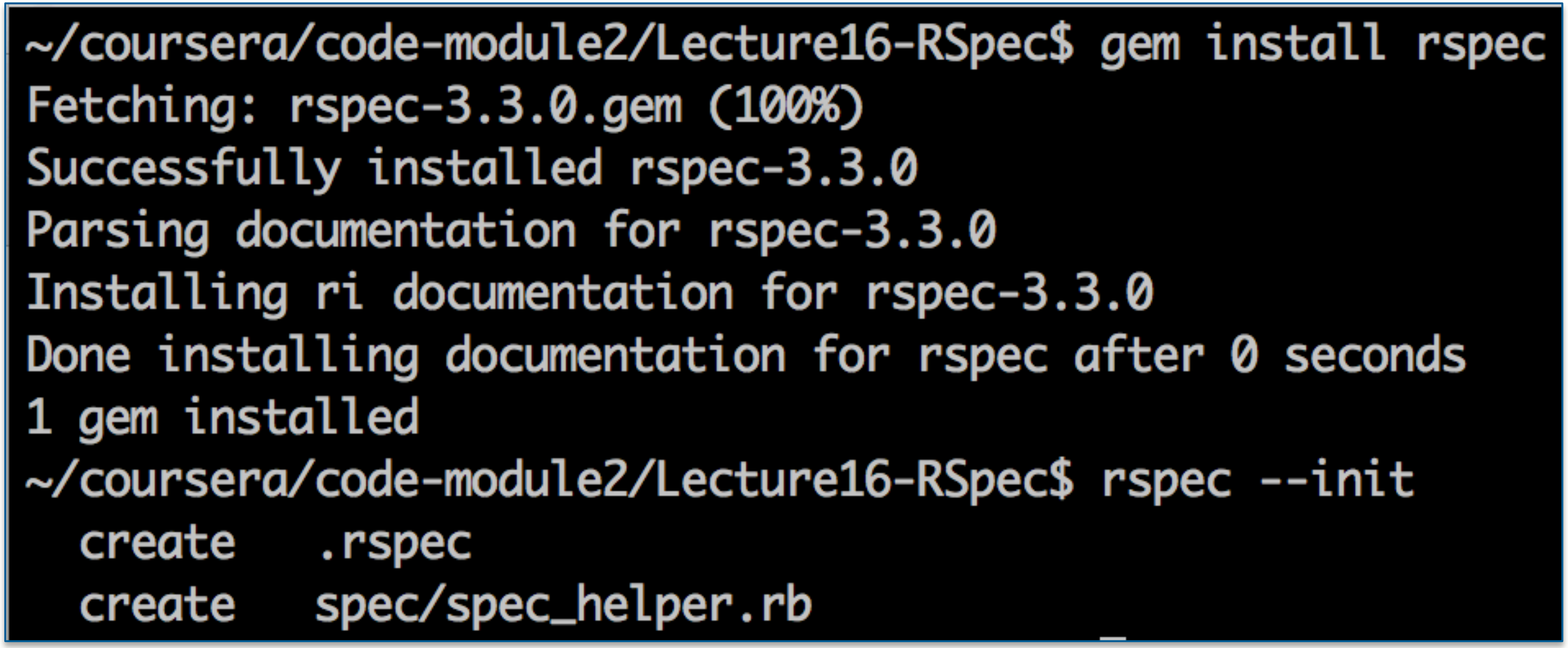
In this lecture, we will discuss…

- ✧ RSpec

Testing With RSpec

- **Test::Unit** “does the job”, but it would be nice if tests would be more **descriptive**, more **English-like**
- The **writing** of the tests is more **intuitive** as well as the **output** from running the tests

Installing RSpec



```
~/coursera/code-module2/Lecture16-RSpec$ gem install rspec
Fetching: rspec-3.3.0.gem (100%)
Successfully installed rspec-3.3.0
Parsing documentation for rspec-3.3.0
Installing ri documentation for rspec-3.3.0
Done installing documentation for rspec after 0 seconds
1 gem installed
~/coursera/code-module2/Lecture16-RSpec$ rspec --init
  create   .rspec
  create   spec/spec_helper.rb
```

Creates a spec directory where “specs” go

describe()

- Set of **related tests** (a.k.a. *example group*)
- Takes either a **String** or **Class** as argument
- All specs **must be inside a describe block**
- No class to subclass
    - Unlike Test::Unit which always subclasses TestCase class

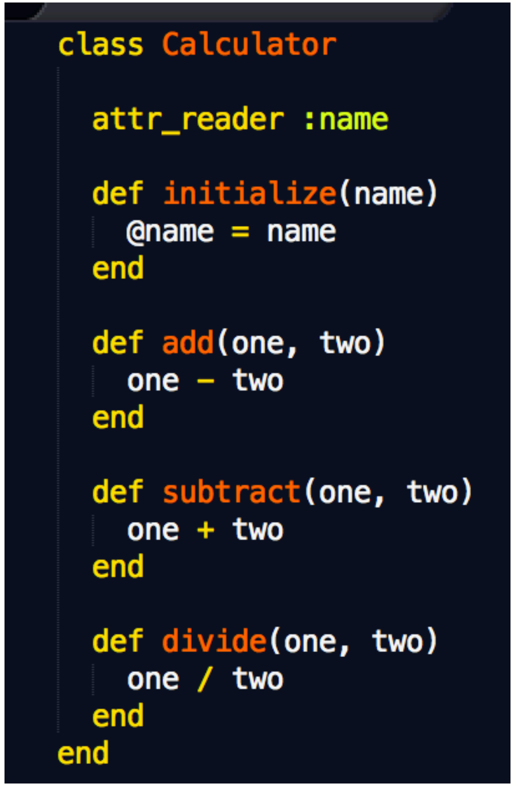
before() and after() methods

- `before()` and `after()` methods are similar to `setup()` and `teardown()` in MiniTest
- Can pass in either `:each` or `:all` (infrequently used) to **specify** whether the block will run **before/after each test** or **once before/after all tests**
- `before :all` **could be useful**, if for example you only want to connect to DB once

it() method

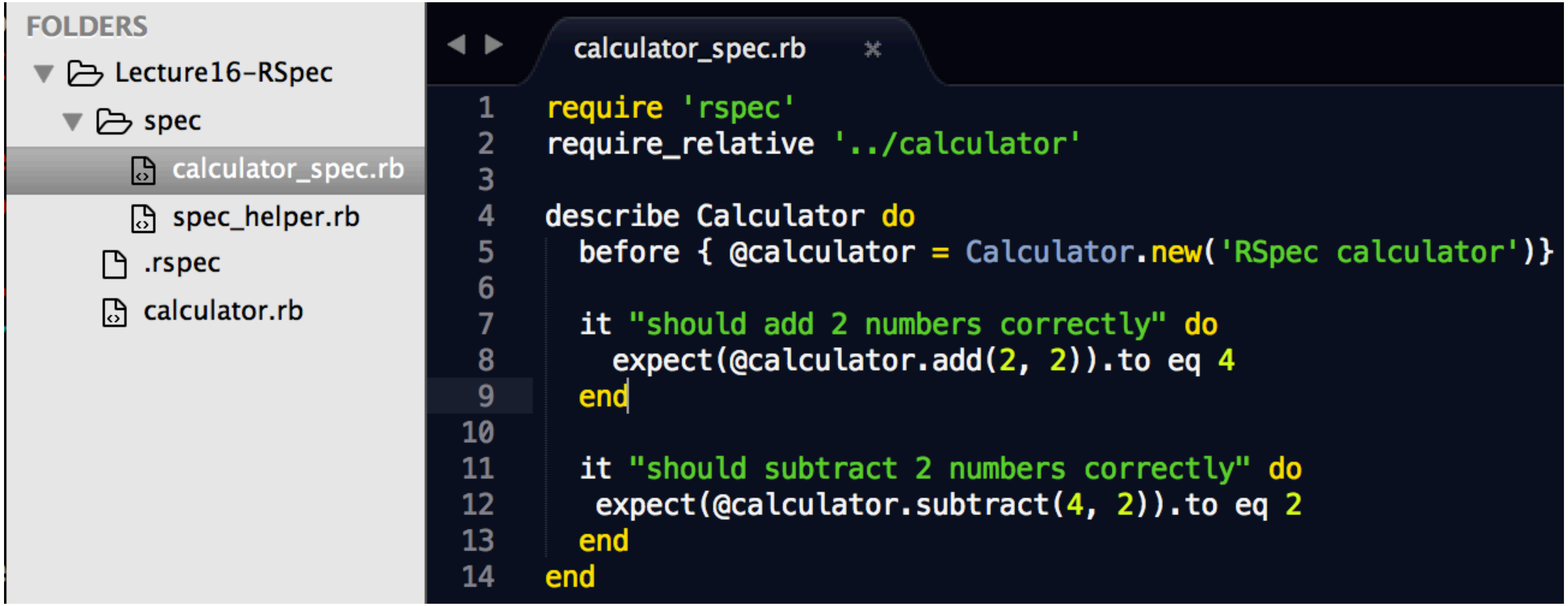
- Used to define the **actual** RSpec specifications/ examples
- Takes an **optional string** that **describes the behavior** being tested

calculator.rb



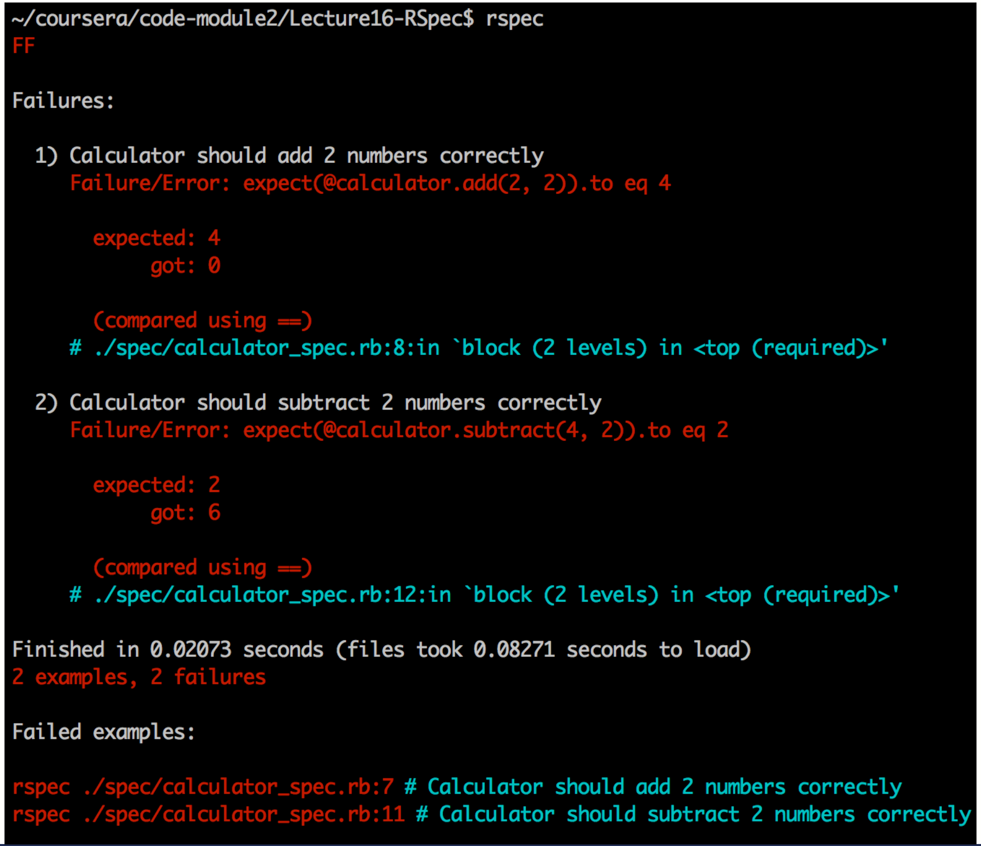
```ruby
class Calculator

  attr_reader :name

  def initialize(name)
    @name = name
  end

  def add(one, two)
    one - two
  end

  def subtract(one, two)
    one + two
  end

  def divide(one, two)
    one / two
  end
end
```

calculator_spec.rb



```ruby
require 'rspec'
require_relative '../calculator'

describe Calculator do
  before { @calculator = Calculator.new('RSpec calculator')}

  it "should add 2 numbers correctly" do
    expect(@calculator.add(2, 2)).to eq 4
  end

  it "should subtract 2 numbers correctly" do
   expect(@calculator.subtract(4, 2)).to eq 2
  end
end
```

Output



```
~/coursera/code-module2/Lecture16-RSpec$ rspec
FF

Failures:

  1) Calculator should add 2 numbers correctly
     Failure/Error: expect(@calculator.add(2, 2)).to eq 4

       expected: 4
            got: 0

       (compared using ==)
     # ./spec/calculator_spec.rb:8:in `block (2 levels) in <top (required)>'

  2) Calculator should subtract 2 numbers correctly
     Failure/Error: expect(@calculator.subtract(4, 2)).to eq 2

       expected: 2
            got: 6

       (compared using ==)
     # ./spec/calculator_spec.rb:12:in `block (2 levels) in <top (required)>'

Finished in 0.02073 seconds (files took 0.08271 seconds to load)
2 examples, 2 failures

Failed examples:

rspec ./spec/calculator_spec.rb:7 # Calculator should add 2 numbers correctly
rspec ./spec/calculator_spec.rb:11 # Calculator should subtract 2 numbers correctly
```

Summary

✧ RSpec makes testing more intuitive

What’s next?

✧ RSpec Matchers