Unit testing

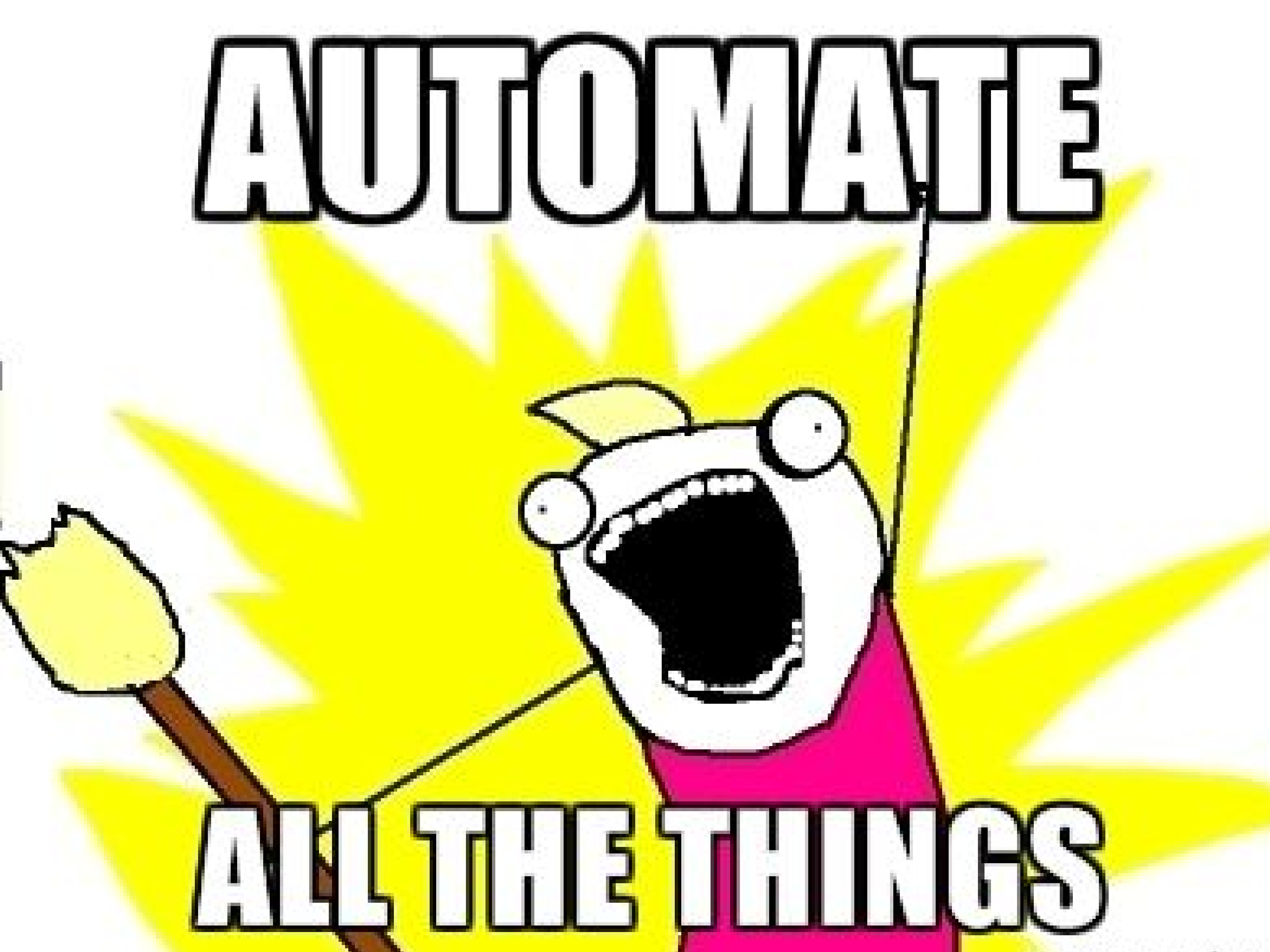
Up until now, how did we check that our programs were working?

Iteration 1

Create a simple String calculator object with a method add(numbers) that takes a string as a parameter and returns an integer.

The method takes a string with 0, 1 or 2 digits, and will return their sum (for an empty string it will return 0) for example “” or “1” or “1,2”

AUTOMATE



ALL THE THINGS

Write code

Write code that checks that the code you wrote before is OK

Repeat

TYPING IS NOT THE BOTTLENECK

Iteration 2

Allow the Add method to handle an undetermined amount of numbers

RSpec

RSpec is testing tool for the Ruby programming language

Test Suite

```ruby
RSpec.describe "String calculator" do
    it "returns 0 for the empty string" do
        expect(StringCalc.new.add("")).to eq(0)
    end
end
```

Suite Description

```
RSpec.describe "String calculator" do
    it "returns 0 for the empty string" do
        expect(StringCalc.new.add("")).to eq(0)
    end
end
```

Unit Test

```ruby
RSpec.describe "String calculator" do
    it "returns 0 for the empty string" do
        expect(StringCalc.new.add("")).to eq(0)
    end
end
```

Unit Test Description

```
RSpec.describe "String calculator" do
    it "returns 0 for the empty string" do
        expect(StringCalc.new.add("")).to eq(0)
    end
end
```

Assertion

```
RSpec.describe "String calculator" do
    it "returns 0 for the empty string" do
        expect(StringCalc.new.add("")).to eq(0)
    end
end
```

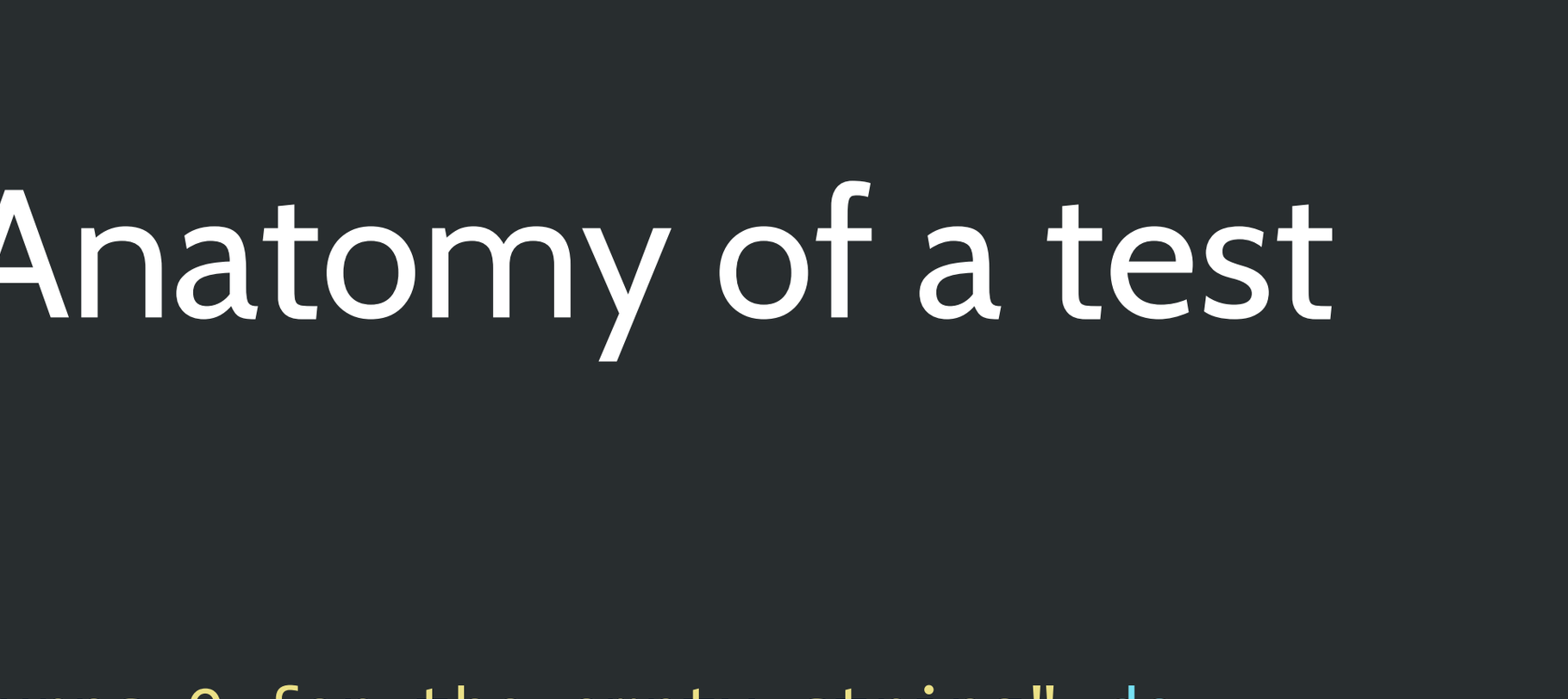
Another unit test

```
RSpec.describe "String calculator" do
    it "returns 0 for the empty string" do
        expect(StringCalc.new.add("")).to eq(0)
    end

    it "returns 3 for only that number" do
        expect(StringCalc.new.add("3")).to eq(3)
    end
end
```

Anatomy of a test

```
it "returns 0 for the empty string" do
    input = ""
    expect(StringCalc.new.add(input)).to eq(0)
end
```



Iteration 3

Allow the add method to handle new lines between numbers (instead of commas).

Unit test

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use.

A good unit test should be

- Automated
- Repeatable
- Easy to implement
- Should remain for future use
- Should run at the push of a button
- Should be quick

Some advice from our experts

https://www.flickr.
com/photos/enerva/9068467267

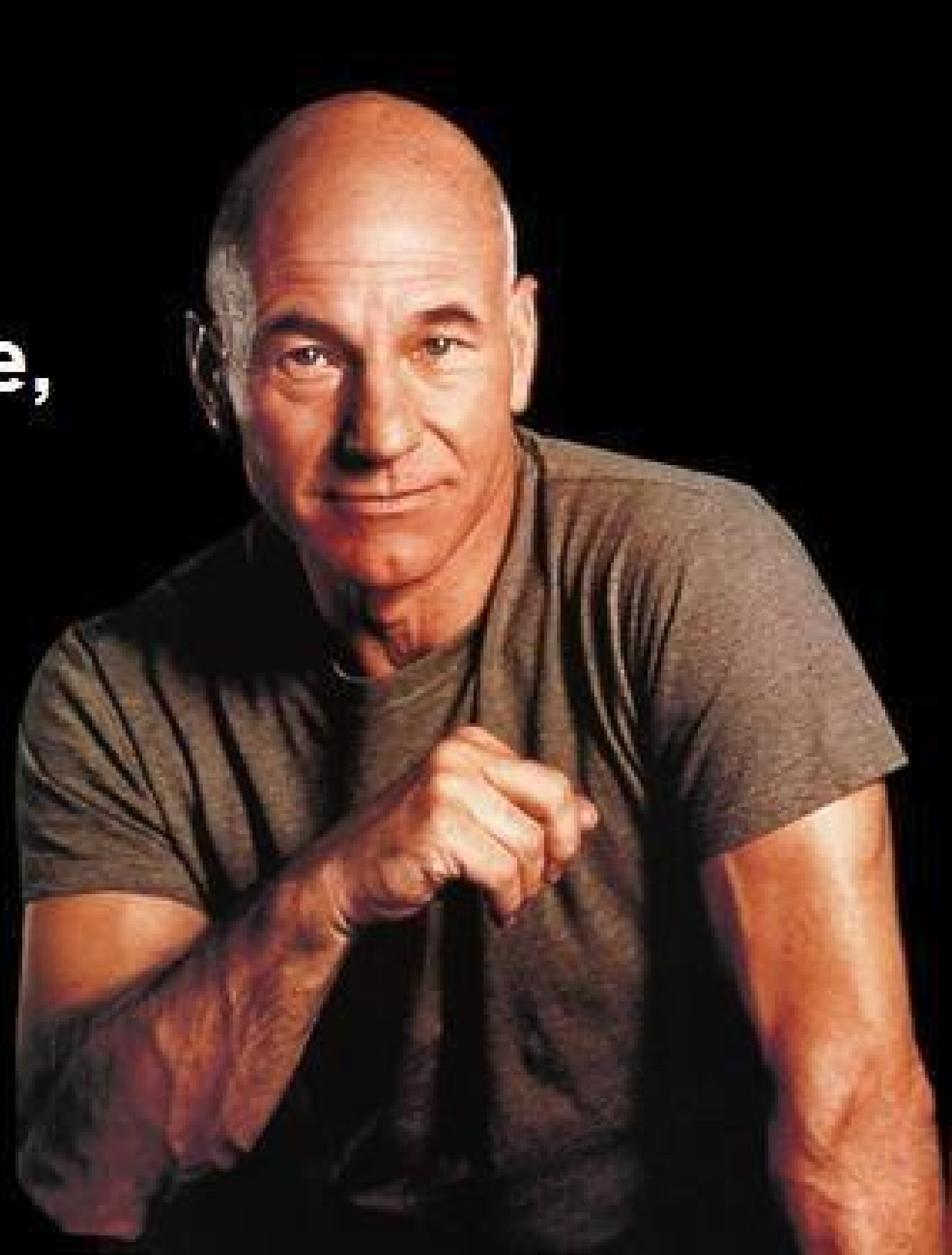
A nice test suite is worth a thousand minions

Matchers

```
expect(StringCalc.new.add("3")).to eq(3)
```

"Use the force, Harry"

- Gandalf



Use the correct matcher

```
expect("Hi guys!".include?("hello")).to be_truthy
```

```
Failure/Error: expect("Hi guys!".include?("hello")).to be_true
  expected false to respond to `true?`
```

Use the correct matcher

```
expect("Hi guys!").to include("hello")
```

```
Failure/Error: expect("Hi guys!").to include("hello")
  expected "Hi guys!" to include "hello"
```

```ruby
it "should return 0 for an empty string" do
  expect(StringCalculator.new.add("")).to eq(0)
end
it "should return 2 when only that number" do
  expect(StringCalculator.new.add("2")).to eq(2)
end
it "should return 3 if adding 1 and 2" do
  expect(StringCalculator.new.add("2,1")).to eq(3)
end
```

```ruby
before :each do
  @calculator = StringCalculator.new
end

it "should return 0 for an empty string" do
  expect(@calculator.add("")).to eq(0)
end
it "should return 2 when only that number" do
  expect(@calculator.add("2")).to eq(2)
end
it "should return 3 if adding 1 and 2" do
  expect(@calculator.add("2,1")).to eq(3)
end
```

Running code between tests

```
before :all do
end

before :each do
end

after :each do
end

after :all do
end
```

```ruby
before :each do
  @calculator = StringCalculator.new
end

it "should return 0 for an empty string" do
  expect(@calculator.add("")).to eq(0)
end
it "should return 2 when only that number" do
  expect(@calculator.add("2")).to eq(2)
end
it "should return 3 if adding 1 and 2" do
  expect(@calculator.add("2,1")).to eq(3)
end
```

```ruby
before :each do
  @calculator = StringCalculator.new
end

it "should return 0 for an empty string" do
  expect(@calculator.add("")).to eq(0)
end
it "should return 2 when only that number" do
  expect(@calculator.add("2")).to eq(2)
end
it "should return 3 if adding 1 and 2" do
  expect(@calculator.add("2,1")).to eq(3)
end
it "nasty test" do
  expect(StringCalculator.new("A parameter").add("2,1")).to eq(3)
end
```

Different creation

Lazy-memoized creation

```
let(:calculator) { StringCalculator.new }

it "should return 0 for an empty string" do
  expect(calculator.add("")).to eq(0)
end
it "should return 2 when only that number" do
  expect(calculator.add("2")).to eq(2)
end
it "should return 3 if adding 1 and 2" do
  expect(calculator.add("2,1")).to eq(3)
end
```

https://www.flickr.com/photos/shutterhacks/4474421855

Tests are your living documentation