Testing in Ruby
I’m Darcy
Fellow student here
I make APIs for iOS Apps in Ruby
Q: What is testing?
Automated Testing
Verifiable contracts
Given this,
I expect that.
“Test Suites”
Q: Why test?
Testing as a contract
Prevent regressions.
Verify your assumptions
Confirm you’re understanding
Things you forget.
Real World Example:
Merging data.
Testing as a design tool
Test-first development
Red

Green

Refactor
Write a failing test
require 'test_helper'

class PersonTest < ActiveSupport::TestCase
  
  test "making a full name" do
    person = Person.new first_name: "Bob", last_name: "Smith"
    assert_equal "Bob Smith", person.full_name
  end

end
# Running tests:

E

Finished tests in 0.057601s, 17.3608 tests/s, 0.0000 assertions/s.

1) Error:
PersonTest#test_making_a_full_name:
NoMethodError: undefined method `full_name' for #<Person:0x007fcc4e4dea50>
  test/models/person_test.rb:7:in `block in <class:PersonTest>'

1 tests, 0 assertions, 0 failures, 1 errors, 0 skips
==============================================================================
Make it pass
class Person < ActiveRecord::Base

  def full_name
    first_name + " " + last_name
  end

end
# Running tests:

. 

Finished tests in 0.132392s, 7.5533 tests/s, 7.5533 assertions/s.

1 tests, 1 assertions, 0 failures, 0 errors, 0 skips

=================================
Refactor it.
class Person < ActiveRecord::Base
  def full_name
    "#{first_name} #{last_name}".strip
  end
end
# Running tests:

```
.
```

Finished tests in 0.177100s, 5.6465 tests/s, 5.6465 assertions/s.

1 tests, 1 assertions, 0 failures, 0 errors, 0 skips

==============================================================================
We’re all human.
Work out what you want
Write as little as possible to make it happen
Work out what you don’t know
Testing as documentation
Testing in Rails
Rails <3’s testing
Testing is a standard part of Rails
$ rake
Run options: --seed 13019

# Running tests:

...FE

Finished tests in 0.077580s, 64.4496 tests/s, 51.5597 assertions/s.

1) Failure:
PersonTest#test_making_a_name_with_no_given_names [~/Users/sutto/Code/test_project/test/models/person_test.rb:22]:
Expected: nil
Actual: ""

2) Error:
PersonTest#test_making_searching_for_users:
NoMethodError: undefined method `search' for #<Class:0x007fc835b1efe0>
  test/models/person_test.rb:28:in `block in <class:PersonTest>'

5 tests, 4 assertions, 1 failures, 1 errors, 0 skips
==============================================================================
Run options: --seed 5103

# Running tests:

.

Finished tests in 0.086914s, 11.5056 tests/s, 11.5056 assertions/s.

1 tests, 1 assertions, 0 failures, 0 errors, 0 skips
==============================================================================
Generates automatically generate stubs for you.
xUnit
JUnit
Test::Unit
MiniTest
Assertions
assert <a boolean expression>
assert_equal 2, (1 + 1)
Unit / Model Tests
Small, independent bits of functionality.
Minimal set of dependencies
test "making a name with first and last name" do
  person = Person.new first_name: "Bob", last_name: "Smith"
  assert_equal "Bob Smith", person.full_name
end

test "making a name with first name only" do
  person = Person.new first_name: "Bob"
  assert_equal "Bob", person.full_name
end

test "making a name with last name only" do
  person = Person.new last_name: "Smith"
  assert_equal "Smith", person.full_name
end

test "making a name with no given names" do
  person = Person.new
  assert_equal nil, person.full_name
end
Test edge cases
Small, fast and easy to test assumptions.
test "making searching for users" do
  person = Person.create! first_name: "Bob",
             last_name: "Smith"

  # Actually hit the database and get a result...
  Person.search('Bob Smith').should == [person]
end
Functional / Controller Tests
Test how the model, view and controller interact
require 'test_helper'

class PeopleControllerTest < ActionController::TestCase

  test "should list users" do
    person = Person.create!
    get :index
    assert_equal [person], assigns(:people)
  end

end
test "redirecting when not logged in" do
  get :index
  assert_response :redirect
  assert_redirected_to some_other_page
end

test "visible when logged in" do
  get :index
  assert_response :success
end
Guarantees*.
* Only test what you specify, nothing more.
Checks the glue.
Integration tests.
The bigger picture
E.g. user stories
User logs in, purchases an item, order is placed.
Slower to run, but test everything together.
Taking it Further
Full Stack Tests
Coverage Tools
Simplecov
class Person < ActiveRecord::Base
  def full_name
    "#{first_name} #{last_name}".strip
  end
  def something_else
    # Do something horrible...
    World.destroy_all
  end
end
See code that isn’t tested
BDD
minitest-spec
What I’ve learnt about testing
Over testing is bad.
Slow tests sucks.
Testing should be instinctive
Don’t get stuck on numbers (coverage, number of tests)
Good tests inspire confidence
Only test what you implement
If you have to be told to test, it’s probably not going to help
It takes time to learn to write good tests
Don’t be afraid to write tests before code
Most importantly, writing tests costs less long term.