

Minitest Cheat Sheet – The Basics

Assert-Style Testing

Test cases are written as standard Ruby classes inheriting from `Minitest::Test`. Public instance methods matching the pattern `test_*` are treated as tests.

```
class TpsReportTest < Minitest::Test
  def test_report_must_have_cover_sheet
    report = TpsReport.new(cover_sheet: nil)
    assert report.invalid?
  end
end
```

setup – Executes before each test

```
def setup
  @report = TpsReport.new(cover_sheet: true)
end
```

teardown – Executes after each test

```
def teardown
  @report = nil
end
```

skip – Ends the test immediately, result is neither pass or fail

```
def test_memo_received
  skip "Hasn't been written yet"
  # possible other assertions...
end
```

flunk – Fail the test immediately

```
def test_tps_report_has_cover_sheet
  flunk "No cover sheet ever"
end
```

Mocking and Stubbing

Mock#expect – Defines a method expectation on a mock object

```
mock_report = Minitest::Mock.new
mock_report.expect(:date, Date.today)
```

Mock#verify – Check that all method expectations were fulfilled

```
mock_report.verify
```

Object#stub – Replace a method within the scope of a block

```
report.stub(:date, Date.today) do
  assert_equal Date.today, report.date # passes
end
```

Spec-Style Testing

Specs are written using a specialized DSL similar to the one used by RSpec. Test cases and tests are defined using block helper methods, and assertions use a syntax meant to more closely mirror natural language.

```
describe TpsReport do
  it "must have a cover sheet" do
    report = TpsReport.new(cover_sheet: nil)
    expect(report).must_be :invalid?
  end
end
```

before – Executes before each test in a given describe scope

```
before do
  @report = TpsReport.new(cover_sheet: true)
end
```

after – Executes before each test in a given describe scope

```
after do
  @report = nil
end
```

describe – Defines a scope for a collection of related tests

```
describe TpsReport do
  describe "Validations" do
    # ...
  end
end
```

it – Defines a test method body

```
it "includes all TpsReports for the week" do
  @inbox.must_include @peters_report
end
```

let – Creates a named lazy initializer using the requested block

```
let(:report) { TpsReport.first }
```

subject – Creates a lazy initializer named subject

```
subject { TpsReport.first }
```

expect – Wrap a value before making expectations on it

Aliased as: `_`, **value**

```
expect(@tps_report.date).must_equal Date.today
```

Minitest Cheat Sheet – Assertions

Assertion / Refutation	Description	Examples
assert/refute	Returns a “truthy” / “falsy” value	<code>assert @report.has_cover_sheet?, “No cover sheet”</code>
assert_empty/refute_empty	Responds to <code>#empty?</code> and returns <code>true</code> / <code>false</code>	<code>assert_empty @inbox</code>
assert_equal/refute_equal	Expected and actual values are equal (<code>==</code>)	<code>assert_equal :paper_jam, @printer.current_status</code>
assert_in_delta refute_in_delta	Values' absolute difference falls in specified range	<code>assert_in_delta Math::PI, (22.0 / 7.0), 0.01</code>
assert_in_epsilon refute_in_epsilon	Values' relative difference falls in specified range	<code>assert_in_epsilon 22.55, @item.price / 2.0, 0.01</code>
assert_includes refute_includes	Collection includes the requested object	<code>assert_includes @calendar.days_of_week, “Monday”</code>
assert_instance_of refute_instance_of	Requested object is an instance of the given class	<code>assert_instance_of String, “Initech”</code>
assert_kind_of refute_kind_of	Requested object inherits from the class/module	<code>assert_kind_of Numeric, @report.page_count</code>
assert_match/refute_match	RegExp argument matches the actual String	<code>assert_match /synergies/, @report.text</code>
assert_nil/refute_nil	Tested object is <code>nil</code>	<code>assert_nil @report.cover_sheet</code>
assert_operator refute_operator	Binary expression prepared using the given arguments evaluates to <code>true</code>	<code>assert_operator @report.page_count, :>=, 20</code>
assert_output assert_silent	Block produces the expected output on <code>\$stdout</code> or <code>\$stderr</code> or remains silent on both	<pre># exact match on \$stdout assert_output("OK") { system("echo OK") } # partial match with Regexp on \$stderr assert_output("", /borked/i) { system("status") } assert_silent { system "find . -name *~ -delete" }</pre>
assert_predicate refute_predicate	Message composed using the parameters returns <code>true</code> when sent to the tested object	<code>assert_predicate @report, :submitted?</code>
assert_raises	Block raises an error like the one specified	<code>assert_raises(ReportFormatError) { @report.submit }</code>
assert_respond_to refute_respond_to	Tested object <code>#responds_to?</code> requested message	<code>assert_respond_to “Bill”, :length</code>
assert_same/refute_same	Both parameters refer to the same object	<code>assert_same @new_report, @report_from_db</code>
assert_send	Message sent to the receiver with the requested arguments returns <code>true</code>	<code>assert_send [@calendar, :no_meetings?, :saturday]</code>
assert_throws	Block throws the expected symbol	<code>assert_throws(:found) { @haystack.find(“needle”) }</code>

Minitest Cheat Sheet – Expectations

Expectations	Description	Examples
must_be/wont_be	Statement prepared using the requested arguments evaluates to a “truthy” / “falsy” value	<code>expect(@report).must_be :complete?, “Incomplete”</code> <code>expect(@report.date).must_be :>, Date.today</code>
must_be_empty wont_be_empty	Responds to <code>#empty?</code> and returns <code>true</code> / <code>false</code>	<code>expect(@inbox).must_be_empty</code>
must_equal wont_equal	Expected and actual values are equal (<code>==</code>)	<code>expect(@printer.current_status).must_equal :paper_jam</code>
must_be_within_delta wont_be_within_delta	Values' absolute difference falls in specified range Alias: <code>must_be_close_to/wont_be_close_to</code>	<code>expect(22.0/7.0).must_be_within_delta Math::PI, 0.01</code>
must_be_within_epsilon wont_be_within_epsilon	Values' relative difference falls in specified range	<code>expect(@price/2.0).must_be_within_epsilon 22.55, 0.01</code>
must_include wont_include	Collection includes the requested object	<code>expect(@calendar.days_of_week).must_include “Monday”</code>
must_be_instance_of wont_be_instance_of	Requested object is an instance of the given class	<code>expect(“Initech”).must_be_instance_of String</code>
must_be_kind_of wont_be_kind_of	Requested object inherits from the class/module	<code>expect(@report.page_count).must_be_kind_of Numeric</code>
must_match/wont_match	RegExp argument matches the actual String	<code>expect(@report.text).must_match /synergies/</code>
must_be_nil/wont_be_nil	Tested object is <code>nil</code>	<code>expect(@report.cover_sheet).must_be_nil</code>
must_output	Output captured from <code>\$stdout</code> / <code>\$stderr</code> matches the expected output	<i># exact match on \$stdout</i> <code>callable = -> { system(“echo OK”) }</code> <code>expect(callable).must_output(“OK”)</code> <i># partial match with Regexp on \$stderr</i> <code>callable = -> { system(“service-status”) }</code> <code>expect(callable).must_output(“”, /borked/i)</code>
must_raise	Callable raises an error like the one specified?	<code>callable = proc { “TPS Report”.submitted? }</code> <code>expect(callable).must_raise(NoMethodError)</code>
must_respond_to wont_respond_to	Tested object <code>#responds_to?</code> requested message	<code>expect(“Bill”).must_respond_to :length</code>
must_be_same_as wont_be_same_as	Both parameters refer to the same object	<code>expect(@report_from_db).must_be_same_as @new_report</code>
must_be_silent	Block produces no output on <code>\$stdio</code> or <code>\$stderr</code>	<code>callable = proc { system “find . -name *~ -delete” }</code> <code>expect(callable).must_be_silent</code>
must_throw	Block throws the expected symbol	<code>callable = -> { @haystack.search(“needle”) }</code> <code>expect(callable).must_throw(:found)</code>