

### MiniTest Generic Assertions

`assert( test, msg=nil )`

inverse: `refute`

`assert_equal( obj, val, msg=nil )`

inverse: `refute_equal`

`assert_instance_of( klass, obj, msg=nil )`

inverse: `refute_instance_of`

`assert_kind_of( klass, obj, msg=nil )`

inverse: `refute_kind_of`

`assert_match( regex, obj, msg=nil )`

inverse: `refute_match`

`assert_nil( obj, msg=nil )`

inverse: `refute_nil`

`assert_operator( obj, operator, val, msg=nil )`

inverse: `refute_operator`

`assert_respond_to( obj, method, msg=nil )`

inverse: `refute_respond_to`

`assert_same( obj, obj2, msg=nil )`

inverse: `refute_same`

### MiniTest Block Assertions

`assert_block( msg=nil ) do ... end`

`assert_output( stdout=nil, stderr=nil ) do ... end`

`assert_raises( exception ) do ... end`

`assert_silent do ... end`

`assert_throws( sym, msg=nil )`

### MiniTest Enumerable Assertions

`assert_empty( obj, msg=nil )`

`assert_includes( collection, obj, msg=nil )`

### MiniTest Float Assertions

`assert_in_delta( val, obj, delta=0.001, msg=nil )`

`assert_in_epsilon( val, obj, e=0.001, msg=nil )`

### MiniTest::Spec Enumerable Expectations

`obj.(must|wont)_be_empty()`

`obj.(must|wont)_include( obj2 )`

### MiniTest::Spec Float Expectations

`obj.(must|wont)_be_close_to( val, [delta] )`

equivalent to `assert_in_delta`

`obj.(must|wont)_be_within_delta( val, [delta] )`

Fails unless `obj` and `val` are within `delta` of each other

`obj.(must|wont)_be_within_epsilon( val, [e] )`

Fails unless `obj` and `val` have a relative error less than `e`

### MiniTest::Spec Generic Expectations

`obj.(must|wont)_be( operator, expected )`

`10.must_be :<, 11`

`obj.(must|wont)_be_instance_of( klass )`

`obj.(must|wont)_be_kind_of( klass )`

`obj.(must|wont)_be_nil()`

`obj.(must|wont)_be_same_as( obj2 )`

`obj.(must|wont)_equal( obj2 )`

`obj.(must|wont)_match( regex )`

`obj.(must|wont)_respond_to( message )`

### MiniTest::Spec Block Expectations

`->{ }.must_be_silent()`

Fails if block outputs anything to `stderr` or `stdout`.

`->{ }.must_output( stdout, [stderr] )`

The block should have certain output on `stdout` or `stderr`. Set `stdout` to `nil` just to check `stderr`.

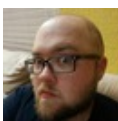
`->{ }.must_raise( exception )`

Fails unless block raises `exception`

`->{ }.must_throw( sym )`

Fails unless block throws `sym`

Works on **procs**, **blocks**, and **lambdas**



By **Ryan Johnson** (CITguy)  
[cheatography.com/citguy/](http://cheatography.com/citguy/)

Published 11th September, 2013.  
Last updated 11th May, 2016.  
Page 1 of 1.

Sponsored by **Readability-Score.com**  
Measure your website readability!  
<https://readability-score.com>