Unit Testing and You, a guide

Unit tests are a way of ensuring the "correctness" of a program's behavior along side making sure that particular chunks of the program do not "regress" and cease functioning as expected. In a complex application where many functions depend on the bug-free performance of another, it is vital to make sure things behave as expected--and if they don't, you should be alerted immediately!

A good unit testing framework for the .NET platform (on which Boo runs!) is NUnit.

Here's a basic unit test in Boo using NUnit.

```
import NUnit.Framework from "nunit.framework"

[TestFixture]
class SampleFixture:

 [Test]
 def PassTest():
  assert 1 == 1
  assert true == true
  assert "McDonalds sucks!" == "McDonalds sucks!"
 [Test]
 def FailTest():
  assert 0 == 1
```

The TestFixture attribute marks a class as containing tests to be run by NUnit.

The Test attribute marks methods as being methods to be run when the test fixture (SampleFixture) is being tested. This is an important attribute, because there are helper methods we might not want to be considered as tests.

"assert" is a macro in Boo that evaluates one true/false expression. If true, nothing happens. If false, however, an exception will be thrown, and NUnit considers any unhandled exceptions being thrown a test failure; therefore, we do not need to use NUnit's "Assert" class to test expression. The assert macro also accepts a second parameter, a message to be printed to the console (STDOUT) if the assertion fails.

Crack open NUnit-Gui, open the assembly we've just created, and click "run." The SampleFixture test fixture will fail. Why? Because although PassTest was fine, FailTest failed its assertion (0 is NOT equal to 1), and this causes the entire test fixture to fail. This too is important because as part of a text fixture, if any given method fails its test, than anything relying on that method will ultimately fail in the future. Having one or two passing tests is not good enough.