**Testing**

Testing is built right into the go tools and the standard library. Testing needs to be a vital part of the development process because it can save you a tremendous amount of time throughout the life cycle of the project. Benchmarking is also a very powerful tool tied to the testing functionality. Aspect of your code can be setup to be benchmarked for performance reviews. Profiling provides a view of the interations between functions and which functions are most heavily used.

**Notes**

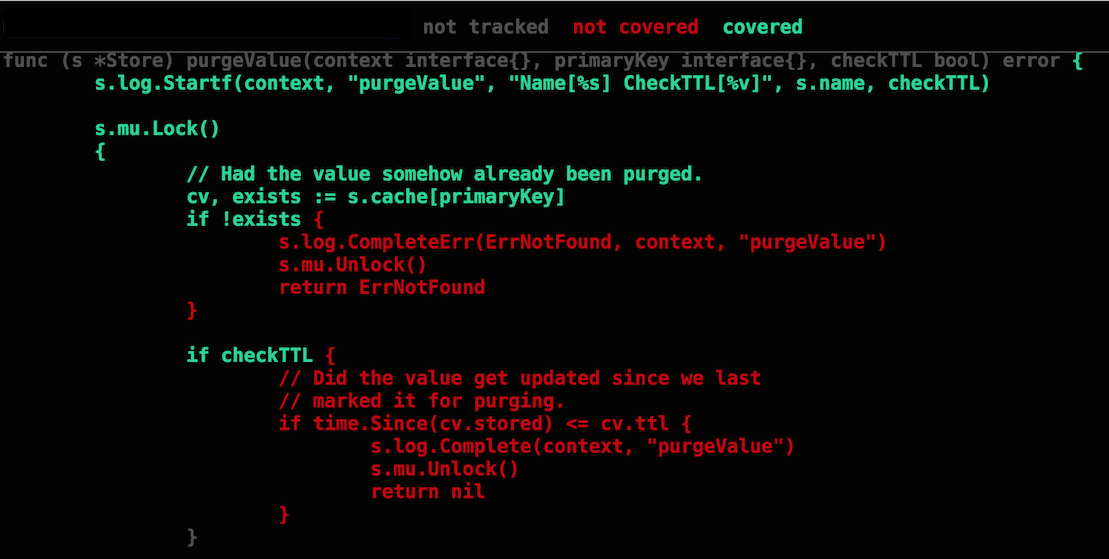
* The Go toolset has support for testing and benchmarking.
* The tools are very flexible and give you many options.
* Write tests in tandem with development.
* Example code serve as both a test and documentation.
* Benchmark throughout the dev, qa and release cycles.
* If performance problems are observed, profile your code to see what functions to focus on.
* The tools can interfere with each other. For example, precise memory profiling skews CPU profiles, goroutine blocking profiling affects scheduler trace, etc. Rerun tests for each needed profiling mode.

## Coverage

Making sure your tests cover as much of your code as possible is critical. Go's testing tool allows you to create a profile for the code that is executed during all the tests and see a visual of what is and is not covered.

go test -coverprofile cover.out

go tool cover -html=cover.out

[](https://github.com/ardanlabs/gotraining/blob/master/topics/go/testing/tests/testing_coverage.png)