Documenting Your Packages

This lesson provides information on documenting a custom package.

The `go doc` tool also shows the comments in your package: the comments must start with // and precede the declarations (package, types, functions ...) with no blank line in between. `go doc` will produce a series of Html-pages, one for each go file. For example:

- In the folder **doc_example** we have the folders **sort** and **sort_main**, containing the go-files **sort** and **sortmain**,respectively, with some comments in the sort file (the files need not be compiled).
- Navigate on the command-line to the folder **doc_example** and issue the command:

```
go doc doc_example/sort
```

This prints out on the command-line:

```
Sorting using a general interface
func Float64sAreSorted(a []float64) bool
func IntsAreSorted(a []int) bool
func IsSorted(data Interface) bool
func Sort(data Interface)
func SortFloat64s(a []float64)
func SortInts(a []int)
func SortStrings(a []string)
func StringsAreSorted(a []string) bool
type Float64Array []float64
type IntArray []int
type Interface interface{ ... }
type StringArray []string
```

```
go doc doc_example/sort_main
```

Which prints out:

```
This package gives an example of how to use a custom package with interfac
es
```

Go doc can also be used as a web server showing your source files in a browser. To see them, run the command `godoc -http=:6060` in the folder **$GOPATH/src**, and navigate in your browser to http://localhost:6060.

Now the documentation of the custom package is ready. In the next lesson, you’ll learn some testing and installation processes.