# Searching for Help and Documenting the Code

Generating documentation for code is a useful habit. This lesson discusses a tool that lets us document programs and packages.

WE'LL COVER THE FOLLOWING



## Basics of `godoc` command

The command `godoc` extracts and generates documentation for Go programs and packages. It extracts comments that appear before top-level declarations in the source code with no intervening newlines. Together with the declaration, they serve as explanatory text for the item. It can also function as a web server delivering documentation; this is how the golang-website works.

### General format

- To get the *package comment* documentation for package, the following format is used:

```
go doc package
```

For example:

```
go doc fmt
```

This comment will appear first on the output produced by `godoc`.

- To get the documentation for subpackage in package:

```
go doc package/subpackage
```

For example:

```
go doc container/list
```

- To get the documentation for function in package:

```
go doc package function
```

For example:

```
go doc fmt Printf
```

provides explanations for the use of `fmt.Printf()`.

## Other uses of `godoc` command #

The command `godoc` itself has some more options. **Godoc** can also be started as a local webserver: `

```
godoc -http=:6060
```

This starts the server on the command-line, then opens a browser window with the URL: http://localhost:6060, and you have a local documentation server for your installed version without the need for an internet connection. Godoc can also be used to generate documentation for your own Go-programs (see Chapter 7).

For more info, visit this page.

---

Apart from formatting and documenting tools, multiple tools make Go a successful application programming language. The next lesson mentions some of the important tools provided by Go.