Learn Go: Introduction

Go Comments

Comments are useful for documentation in a Go file and are ignored by the compiler. There are two types of comments:

- a single-lined comment is preceded by a double forward slash, `//`, and ends at the end of the line.
- a multi-lined comment begins with `/*` followed by one or more lines of comments and ends with `*/`

```
// one line comment
/*
  this comment
  is on multiple lines
  and ends here
*/
```

Go Documentation

In Go, comments can be used as built-in documentation. To check the role of a function, in the command line, use the command go doc followed by a package or the function of a package. For example:

```
$ go doc fmt
```

To find more information about a package’s function:

```
$ go doc fmt.println
```

Import Multiple Packages

To import multiple packages in a Go file, use the `import` keyword followed by the package name enclosed in double-quotes and repeat this statement for every imported package on its own line, or write a single `import` keyword to import multiple packages, one per line, in enclosed parentheses, (...).

```
import "fmt"
import "math"
import "time"
```

or

```
import (
  "fmt"
  "math"
  "time"
)
```

Go Compiler

As a compiled language, Go does not run until its source file is processed through a separate software called a compiler to produce a final executable program. The Go compiler can be accessed on the command line via a generic command such as:
`go <command> [arguments]`

Packages in Go

A Go package is a directory made up of a collection of Go source files that are compiled together. This collection of reusable code typically contains functions related to a specific topic or concept. To use code from a particular package, we simply import it into our Go source file.
For example, to import the `fmt` package which contains functions for formatting input and output strings, we type the keyword `import` followed by the package name.

```
import "fmt"
```

Running Files in Go

The Go compiler can execute Go code from the source file without producing an executable file. Instead of `build`, use `run`. To do this, type the following in the command line:

```
$ go run exampleFile.go
```

Compile Go

The Go compiler takes a Go source file with a `.go` extension, processes it and produces an executable file without any extension. To compile a Go source file, `test.go`, type at the command line:

```
$ go build test.go
```

This will produce an executable file, `test`. To run `test`, type in the command line:

```
$ ./test
```

codecademy

Go Import Package

To import a single package in a Go file, use the keyword `import` followed by the package name in double-quotes.

```go
import "time"
```