Go Basics

# Resources

|  |  |
| --- | --- |
| <https://go.dev/doc/effective_go> | <https://go.dev/doc/> |
| <https://gobyexample.com/> | <https://gowebexamples.com/> |
| <https://go.dev/ref/spec> | <https://go.dev/ref/mod> |
| <https://go.dev/blog/> | <https://go.dev/blog/using-go-modules> |
| <https://pkg.go.dev/std> | <https://go.dev/ref/mem> |

# Basics

Let’s jump straight to some code:

package main

import "fmt"

func main() {

fmt.Println("Hello, Chinese letter")

}

* Go is a compiled language. The Go toolchain converts a source program and the things it depends on into instructions in the **native machine language of a computer**.
* These tools are accessed through a single command called go that has a number of subcommands.
* The simplest of these subcommands is **run, which compiles the source code from one or more source**

**files whose names end in .go, links it with libraries, then runs the resulting executable file.**

$ go run **helloworld.go**

Hello, Chinese letter

* Go natively handles Unicode, so it can process text in all the world’s languages.

If the program is more than a one-shot experiment, it’s likely that you would want to compile

it once and save the compiled result for later use. That is done with go build:

$ go build helloworld.go

**This creates an executable binary file called helloworld that can be run any time without further**

**processing:**

**$ ./helloworld**

**Hello, Chinese letter**

* Go code is organized into packages, which are similar to libraries or modules in other languages.
* A package consists of **one or more .go** source files **in a single directory** that **define what the package does.**
* Each source file:
  + begins with a package declaration, here package main, that states **which package the file belongs to**
  + followed by a list of other packages that it imports. We must tell the compiler what packages are needed by this source file.

You must import exactly the packages you need. A prog ram will not compile if there are missing imports or if there are unnecessary ones.

* + and then the declarations of the program that are stored in that file.
* The Go **standard library** has over 100 packages **for common tasks** like input and output, sorting, and text manipulation.
* For instance, the **fmt** package contains functions for **printing formatted output** **and scanning input**.

**Package main is special**. **It defines a standalone executable program**, **not a library.**

**Within package main the function main is also special**—**it’s where execution of the program begins**. **Whatever main does is what the program does.**

After the import section, a program consists of the **declarations of functions**, **variables**, **constants**, and **types** (introduce d by the key words **func, var, const, and type**); for the most part, the order of declarations does not matter.

Go does not require semicolons at the ends of statements or declarations, except where two or

more appear on the same line.

In effect, newlines following certain tokens are converted into semicolons, so where newlines are placed matters to proper parsing of Go code.

For instance, the opening brace { of the function must be on the same line as the end of the func declaration, not on a line by itself, and in the expression x + y, a newline is permitted after but not before the + operator.

Go takes a strong stance on code formatting. **The gofmt tool rewrites code into the standard format**, and the go tool’s fmt subcommand **applies gofmt to all the files in the specified package**, **or the ones in the current directory by default**.

you should get into the habit of running your code through gofmt. Declaring a standard format by fiat eliminates a lot of pointless debate about trivia and, more importantly, **enables a variety of automated source code transformations that would be infeasible if arbitrary formatting were allowed.**

Many text editors can be configured to run gofmt each time you save a file.

A related tool, goimports, additionally manages the insertion and removal of import declarations as needed. **It is not part of the standard distribution** but you can obtain it with this command:

$ go get golang.org/x/tools/cmd/goimports

## Command-Line Arguments