Tutorial: Get started with Go

Table of Contents

In this tutorial, you'll get a brief introduction to Go programming. Along the way, you will:

- Install Go (if you haven't already).
- Write some simple "Hello, world" code.
- Use the `go` command to run your code.
- Use the Go package discovery tool to find packages you can use in your own code.
- Call functions of an external module.

Note: For other tutorials, see Tutorials.

Prerequisites

- **Some programming experience.** The code here is pretty simple, but it helps to know something about functions.
- **A tool to edit your code.** Any text editor you have will work fine. Most text editors have good support for Go. The most popular are VSCode (free), GoLand (paid), and Vim (free).
- **A command terminal.** Go works well using any terminal on Linux and Mac, and on PowerShell or cmd in Windows.

Install Go

Just use the Download and install steps.

Write some code

Get started with Hello, World.

1. Open a command prompt and cd to your home directory.

   On Linux or Mac:

   ```
   cd
   ```

On Windows:

```
cd %HOMEPATH%
```

2. Create a hello directory for your first Go source code.

   For example, use the following commands:

   ```
   mkdir hello
   cd hello
   ```

3. Enable dependency tracking for your code.

   When your code imports packages contained in other modules, you manage those dependencies through your code's own module. That module is defined by a go.mod file that tracks the modules that provide those packages. That go.mod file stays with your code, including in your source code repository.

   To enable dependency tracking for your code by creating a go.mod file, run the `go mod init` command, giving it the name of the module your code will be in. The name is the module's module path.

   In actual development, the module path will typically be the repository location where your source code will be kept. For example, the module path might be `github.com/mymodule`. If you plan to publish your module for others to use, the module path *must* be a location from which Go tools can download your module. For more about naming a module with a module path, see Managing dependencies.

   For the purposes of this tutorial, just use `example/hello`.

   ```
   $ go mod init example/hello
   go: creating new go.mod: module example/hello
   ```

4. In your text editor, create a file hello.go in which to write your code.

5. Paste the following code into your hello.go file and save the file.

   ```
   package main

   import "fmt"

   func main() {
       fmt.Println("Hello, World!")
   }
   ```

   This is your Go code. In this code, you:

   - Declare a `main` package (a package is a way to group functions, and it's made up of all the files in the same directory).

- Import the popular `fmt` package, which contains functions for formatting text, including printing to the console. This package is one of the standard library packages you got when you installed Go.
- Implement a `main` function to print a message to the console. A `main` function executes by default when you run the `main` package.

6. Run your code to see the greeting.

```
$ go run .
Hello, World!
```

The `go run` command is one of many `go` commands you'll use to get things done with Go. Use the following command to get a list of the others:

```
$ go help
```

Call code in an external package

When you need your code to do something that might have been implemented by someone else, you can look for a package that has functions you can use in your code.

1. Make your printed message a little more interesting with a function from an external module.

    1. Visit pkg.go.dev and search for a "quote" package.
    2. Locate and click the `rsc.io/quote` package in search results (if you see `rsc.io/quote/v3`, ignore it for now).
    3. In the **Documentation** section, under **Index**, note the list of functions you can call from your code. You'll use the `Go` function.
    4. At the top of this page, note that package `quote` is included in the `rsc.io/quote` module.

    You can use the pkg.go.dev site to find published modules whose packages have functions you can use in your own code. Packages are published in modules -- like `rsc.io/quote` -- where others can use them. Modules are improved with new versions over time, and you can upgrade your code to use the improved versions.

2. In your Go code, import the `rsc.io/quote` package and add a call to its `Go` function.

    After adding the highlighted lines, your code should include the following:

```
package main

import "fmt"

import "rsc.io/quote"

func main() {
```

```
        fmt.Println(quote.Go())
    }
```

3. Add new module requirements and sums.

   Go will add the `quote` module as a requirement, as well as a go.sum file for use in authenticating the module. For more, see Authenticating modules in the Go Modules Reference.

```
$ go mod tidy
go: finding module for package rsc.io/quote
go: found rsc.io/quote in rsc.io/quote v1.5.2
```

4. Run your code to see the message generated by the function you're calling.

```
$ go run .
Don't communicate by sharing memory, share memory by communicating.
```

   Notice that your code calls the `Go` function, printing a clever message about communication.

   When you ran `go mod tidy`, it located and downloaded the `rsc.io/quote` module that contains the package you imported. By default, it downloaded the latest version -- v1.5.2.

Write more code

With this quick introduction, you got Go installed and learned some of the basics. To write some more code with another tutorial, take a look at Create a Go module.