Basic Installation of Go and Writing Your First Program

Before diving into complex projects, you need to set up your local environment so you can run, build, and compile Go code. Once the environment is ready, we will write a simple "Hello World" application to test it out.

Part 1: Installing the Go Runtime

To build and execute Go programs, you must install the Go Runtime.

1. Open your browser and navigate to `golang.org/dl`.
2. Find the download link for your operating system (macOS, Windows, or Linux) and grab the installer.
3. Run the installer and click through the standard installation prompts.
4. **Verify the installation:** Open your computer's terminal and type the single word `go`, then hit enter. You should see a long help message appear on the screen. This `go` command is the primary tool you will use to interact with the Go language.

Part 2: Configuring the Editor (VSCode)

While you can use any editor (like Atom or Sublime Text), **Visual Studio Code (VSCode)** is highly recommended because it offers some of the best built-in integration with Go.

1. Download VSCode from `code.visualstudio.com` and install it.
2. Open VSCode, navigate to the top menu bar, click on **View**, and select **Extensions**.
3. Search for "Go" and install the extension named **"Rich Go language support for Visual Studio"**.
4. **Important Step:** To ensure the extension can successfully install its underlying command-line tools, you must completely quit and restart the VSCode editor.
5. Open a new file and change the language mode in the bottom right corner to **Go**. A yellow prompt will appear saying **"Analysis Tools Missing"**—click **Install** to allow a terminal

window to grab the final tools needed to analyze your code.

Part 3: Writing Your First Program

Now that the environment is ready, let's write a tiny application.

1. Create a new folder on your computer called `Hello World` and open this folder in your code editor.
2. Inside this directory, create a new file named `main.go`.
3. Add the following code exactly as it appears. Ensure you use double quotes (not single quotes) around your strings:

```
package main

import "fmt"

func main() {
    fmt.Println("Hi there")
}
```

Part 4: How to Run the Code

To run your project, open your terminal and navigate inside your `Hello World` directory. You can use the Go Command-Line Interface (CLI) to execute the code in two different ways:

- `go run main.go`: This command takes your file, compiles it, and immediately executes the result. When you run this, you will instantly see `Hi there` printed on the screen.
- `go build main.go`: This command will *only* compile your program; it does not execute it automatically. Running this will spit out a runnable executable file named `main` (or `main.exe` on Windows) directly into your folder. You can then execute that file manually.

Part 5: Breaking Down the Code

Even though this is a simple file, it reveals the fundamental structure of all Go programs.

- `package main`: A package is a collection of common source code files. The name `main` is sacred in Go; it specifically tells the compiler that you are making an *executable* package that will spit out a runnable file, rather than a reusable dependency library. Any time you make an executable package, it must contain a function called `main`.
- `import "fmt"`: By default, your package is isolated. The `import` statement gives your package access to functionality written in another package. `fmt` (short for "format") is a part of Go's Standard Library, and it is primarily used to print information out to the terminal.
- `func main()`: `func` is short for function. We declare a function by providing the keyword `func`, the function's name, a set of parentheses for arguments, and curly braces containing the body of our logic.

Revision #2

Created 2026-03-13 16:37:13 UTC by Carsten

Updated 2026-03-13 17:02:02 UTC by Carsten