

Getting Started with Go – greetme CLI

1. Objective

This toolkit shows you how to:

- Install Go.
- Set up a small Go project using modules.
- Build and run a simple command-line app called `greetme` that:
 - Asks for your name.
 - Prints a greeting.
 - Optionally shouts the greeting with a `--shout` flag.
 - Shows a timestamp.

No prior Go experience is required.

2. What is Go?

Go (Golang) is a compiled, statically typed language created at Google. It's used widely for:

- Backend services and APIs
- Command-line tools
- Cloud and DevOps tooling
- Networking and distributed systems

Real-world examples include Docker and many Kubernetes components, which are written in Go.

3. Requirements

- **OS:** Windows, macOS, or Linux
- **Software:**
 - Go 1.20+
 - Git (recommended)
 - Text editor (VS Code, GoLand, etc.)

Check Go:

```
go version
```

You should see something like: `go version go1.2x.x <os>/<arch>`

If you see command not found, Go isn't installed or not in PATH.

4. Setup

4.1 Install Go (summary)

- Go to: <https://go.dev/dl>
- Download the installer for your OS.
- Install with default settings.
- Open a new terminal and run:

```
go version
```

If it prints a version, you're good.

4.2 Get the project

- Option A – Clone from GitHub

```
git clone https://github.com/<your-username>/greetme-go.git
cd greetme-go
```

- Option B – Create from scratch

```
mkdir greetme-go
cd greetme-go
go mod init github.com/<your-username>/greetme-go
```

Then create `main.go` and paste the code in section 5.

5. Minimal Working Example

5.1 What greetme does

- Asks for your name.
- If you press Enter with no name, it uses "stranger".
- Prints a greeting.
- If you pass `--shout`, the greeting is uppercase.
- Prints the current time.

5.2 Code (`main.go`)`package main`

```
import (  
    "bufio"
```

```
"flag"
"fmt"
"os"
"strings"
"time"
)

// greet takes a raw name string and a shout flag.
// It cleans the name, falls back to "stranger" if empty,
// and returns a greeting. If shout is true, the greeting is uppercased.
func greet(name string, shout bool) string {
    name = strings.TrimSpace(name)
    if name == "" {
        name = "stranger"
    }

    msg := "Hello, " + name + "! Welcome to Go ☺"

    if shout {
        msg = strings.ToUpper(msg)
    }

    return msg
}

func main() {
    // Define --shout flag. Usage example:
    // go run main.go -- --shout
    shout := flag.Bool("shout", false, "Print greeting in uppercase")
    flag.Parse()

    fmt.Print("Enter your name: ")

    reader := bufio.NewReader(os.Stdin)
    name, err := reader.ReadString('\n')
    if err != nil {
        fmt.Println("Error reading input:", err)
        return
    }

    fmt.Println(greet(name, *shout))
    fmt.Println("Generated at:", time.Now().Format(time.RFC1123))
}
```

5.3 Run it

From inside the project folder:

```
go run main.go
```

Example:

```
Enter your name: Chris
Hello, Chris! Welcome to Go 🐹
Generated at: Thu, 27 Nov 2025 21:30:00 EAT
```

With shout:

```
go run main.go -- --shout
```

5.4 Build a binary

```
go build -o greetme .
```

Run:

```
./greetme      # Linux/macOS
greetme.exe    # Windows
```

With shout:

```
./greetme --shout
```

6. AI Prompt Journal

Prompt 1

- “I’m a complete beginner to the Go programming language. I’ve worked a bit with Python and JavaScript, but I’ve never written Go before. I want to understand the absolute basics: Explain the structure of a minimal Go program and show a Hello World example.”

Summary:

Explained package main, func main(), and fmt.Println. Showed how to run with go run.

Helpfulness:

5/5

Result:

Used this to understand the basic Go program shape.

Prompt 2

"Now that I can print 'Hello, Go!' to the terminal, Show me how to read a line of user input from the terminal in Go and handle errors."

Summary:

Introduced `bufio.NewReader(os.Stdin)` and `ReadString("\n")`, plus error checking.

Helpfulness:

5/5

Result:

Used this code pattern to read the name safely.

Prompt 3

"I now want to turn my simple Go greeting program into a slightly more realistic CLI by adding a command-line flag. How do I add a boolean flag like `--shout` in a Go CLI?"

Summary:

Showed `flag.Bool`, `flag.Parse()`, and using `*shout`.

Helpfulness:

4/5

Result:

Implemented the `--shout` option

Prompt 4

"I'm getting this Go error:

`go: cannot find main module; see 'go help modules'`

Context:

- I created a folder for my project.
- I wrote a `main.go` file inside it.
- When I run `go run main.go`, I see the error above.
- I know modules exist in Go but I don't really understand them yet.
- Please explain, like I'm new to Go

Summary:

Explained Go modules and `go mod init`.

Helpfulness:

4/5

Result:

Created `go.mod` and ran the app from the correct folder.

7. Common Issues & Fixes

Issue 1: `go: command not found`

- Cause:

Go not installed or PATH not set.

- Fix:

Install Go from <https://go.dev/dl> and ensure the Go bin directory is on PATH. Restart the terminal and run `go version`.

Issue 2: `go: cannot find main module; see 'go help modules'`

- Cause:

No `go.mod` in the folder or wrong directory.

- Fix:

```
cd greetme-go
go mod init github.com/<your-username>/greetme-go # if go.mod is missing
go run main.go
```

Issue 3: Extra newline in name

- Cause:

`ReadString("\n")` includes newline.

- Fix:

Use `strings.TrimSpace(name)` before using the input.

Issue 4: Binary name confusion

- Cause:

`go build` uses `directory/module name`.

Fix:

```
go build -o greetme .
```

8. References

- Go download: <https://go.dev/dl>
- Install guide: <https://go.dev/doc/install>
- Getting started: <https://go.dev/doc/tutorial/getting-started>
- Tour of Go: <https://go.dev/tour>
- Standard library docs (fmt, bufio, os, strings, flag, time): <https://pkg.go.dev/std>

9. 5-Day Learning Flow (Mapped)

Day 1:

- Choose Go and decide to build a greetme CLI.

Day 2:

- Install Go, verify go version, create module, run "Hello, Go!".

Day 3:

- Add user input and the greet function.

Day 4:

Add `--shout`, timestamp, fix newline issues, test.

Day 5:

- Write this toolkit, record prompts, list issues and references.