Go by Example: Command-Line Arguments

Command-line arguments are a common way to parameterize execution of programs. For example, `go run hello.go` uses `run` and `hello.go` arguments to the `go` program.

```
package main

import (
    "fmt"
    "os"
)

func main() {
```

`os.Args` provides access to raw command-line arguments. Note that the first value in this slice is the path to the program, and `os.Args[1:]` holds the arguments to the program.

```
    argsWithProg := os.Args
    argsWithoutProg := os.Args[1:]
```

You can get individual args with normal indexing.

```
    arg := os.Args[3]
```

```
    fmt.Println(argsWithProg)
    fmt.Println(argsWithoutProg)
    fmt.Println(arg)
}
```

To experiment with command-line arguments it’s best to build a binary with `go build` first.

```
$ go build command-line-arguments.go
$ ./command-line-arguments a b c d
[./command-line-arguments a b c d]
[a b c d]
c
```

Next we’ll look at more advanced command-line processing with flags.

Next example: Command-Line Flags.

by Mark McGranaghan and Eli Bendersky | source | license