Contents

1 Chapter one **3**
Command Line Arguments 3
Variables 4

Chapter 1

Chapter one

Go code is organised 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 packages 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, and then the declarations of the program that are stored in that file.

the `fmt` package contains functions for printing formatted output and scanning input. `Println` is one of the basic output functions in `fmt`; it prints one or more values, seperated by spaces, with a newline character at the end so that the values appear as a single line of output.

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. ofcourse, `main` will normally call upon functions in other packages to do much of the work, such as function `fmt.Println`.

Command Line Arguments

The Variable `os.Args` is a *slice* of strings. Slices are a fundamental notion in Go. A slice is a dynamically sized sequence `s` of array of elements where individual elements can be accessed by `s[i]` and a contiguous subsequence as `s[m:n]`

The number of elements is given by `len(s)`.

The first element of `os.Args`, `os.Args[0]`, is the name of the command itself; The other elements are arguments that were presented to the program when it started execution

```
for _, arg := range os.Args[1: ] {
    s += sep + arg
    sep = " "
}
fmt.Println(s)
```

here range produces two values, `index` and `value` of the element at that index so arg handles `value` and `_` handles the index

Variables

The version of strings above uses short variable declaration There are several other ways to declare a variable in go

```
s := ""
var s = ""
var s string
var s string = ""
```