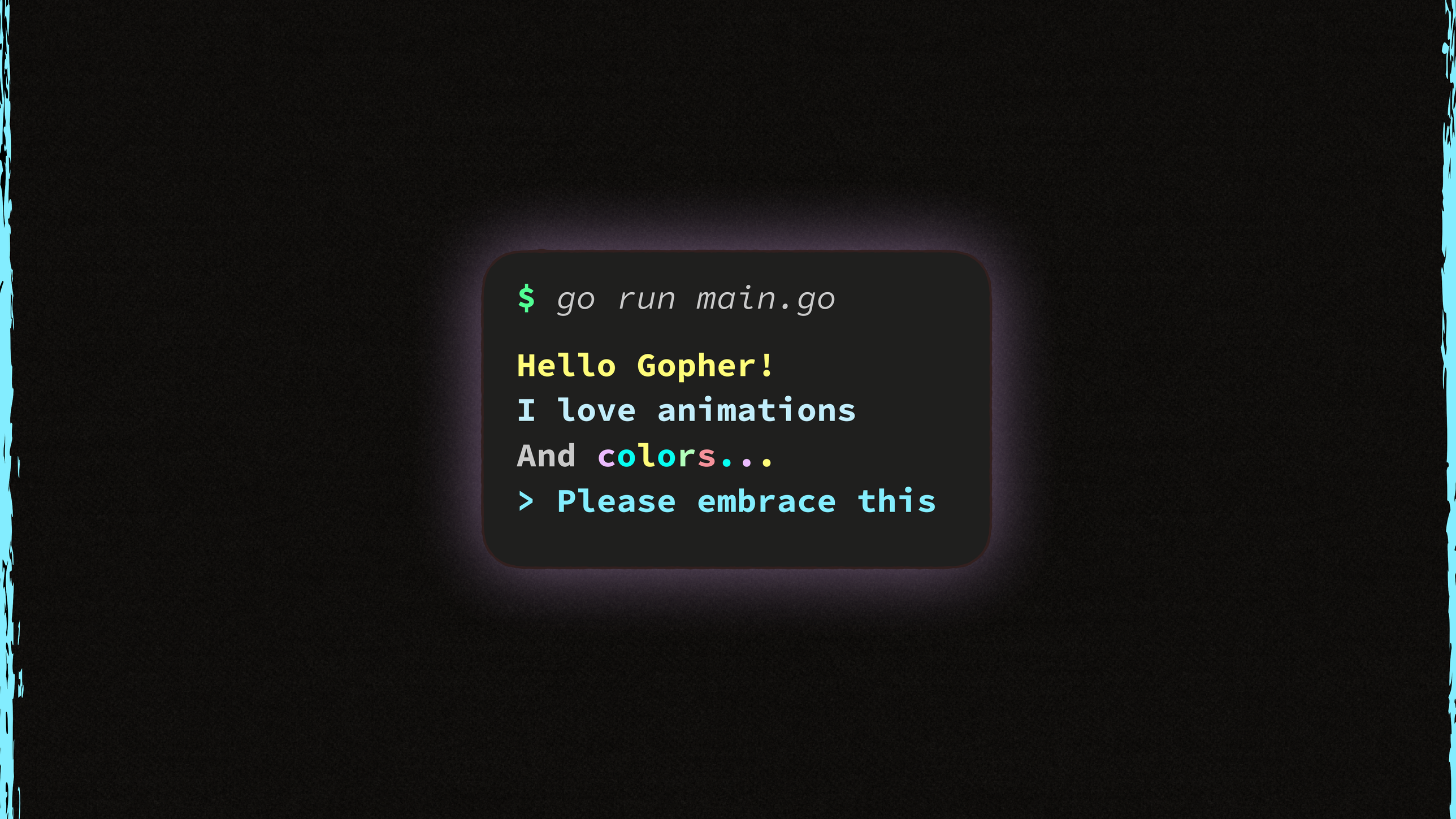

$ go run main.go
Hello Gopher!
I love animations
And colors...
> Please embrace this

CODE

Let's print something to the console!

```go
package main
import "fmt"

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

CODE

Let's code your first Go program

main.go

```
package main

func main() {
    function's code
    will be in here
}
```

package clause
this should be the first code

***name** of the package*
which this file belongs to

function
reusable and executable block of code

func main
a special func that tells Go where to start executing

HOW TO CODE WITH ME

I'll always be providing the full source-code for the examples

sometimes,
i won't show these
common statements
in here

```
package main
import "fmt"

func main() {
}
```

full source-code
will always be
available on
github
&
playground

github
https://github.com/inancgumus/learngo

playground
https://play.golang.org/

COMPILE-TIME

vs

RUNTIME

SOURCE CODE

```
package main
import "fmt"

func main() {
   fmt.Println("hello!")
}
```

$ go build

Compiling

COMPILED CODE

```
MOVQ GS:0x8a0, CX
CMPQ 0x10(CX), SP
JBE 0x108e7c7
SUBQ $0x48, SP
MOVQ BP, 0x40(SP)
...
```

Source-Code is like a seed
(compiled or not)

RUNTIME

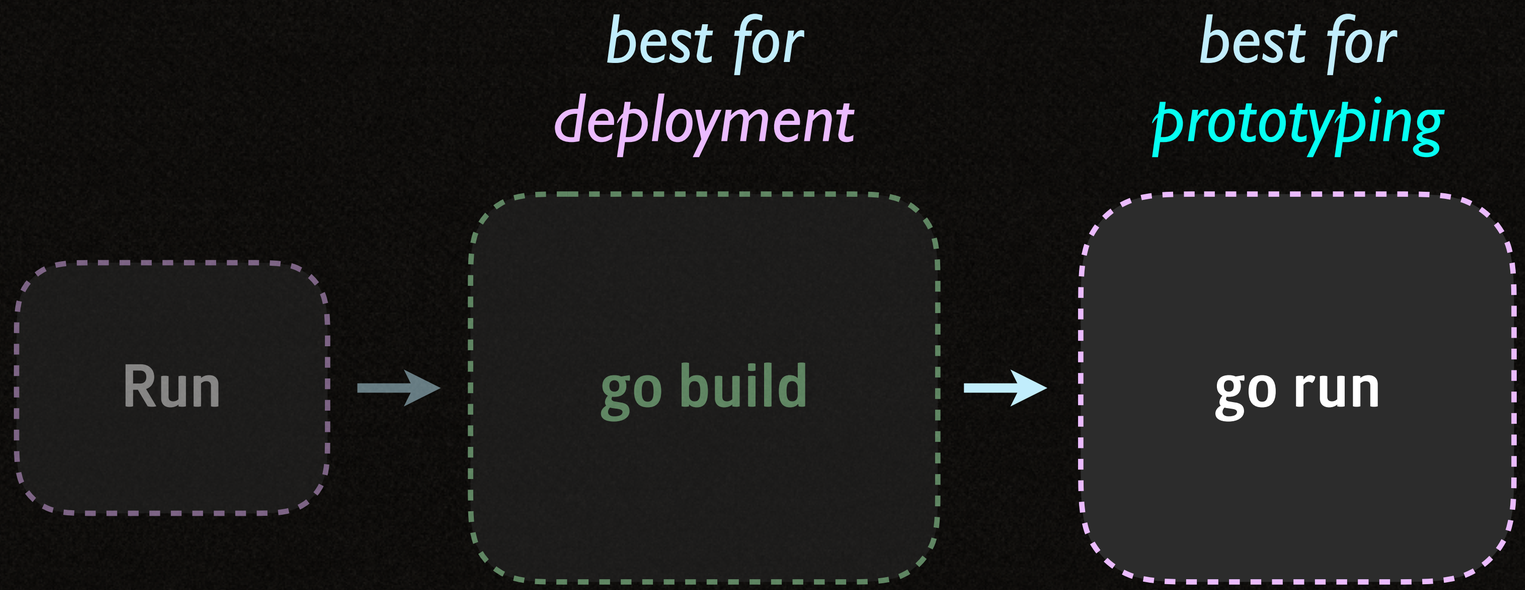
memory

cpu

```
$ ./first
hello!
```

GO RUN

You're going to learn how to compile your Go program using `go run` tool.




```
go run main.go
```

STATEMENTS

instructs Go to execute something

one statement at a time
(*unless you use semicolons* **;**)

STATEMENTS

Statements tell Go to execute something
and they can change the execution flow of code

```
package main
import "fmt"

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

STATEMENTS

Almost all the code are statements here
They can stand on their own on a single line of code

```
package main
import "fmt"

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

STATEMENTS

Statements control the execution flow

```go
package main
import "fmt"

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

STATEMENTS

Statements control the execution flow. Look at how if statement changes the execution.

```go
package main
import "fmt"

func main() {
    fmt.Println("Hello!")

    if 5 > 1 {
        fmt.Println("bigger")
    }
}
```

STATEMENTS

Now, there are two statements: both are fmt.Println function calls.

```
package main
import "fmt"

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

STATEMENTS

Go adds a semicolon between statements, behind the scenes.

```
package main
import "fmt"

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

There are still two statements here

EXPRESSIONS

computes one or multiple values

expressions should be used with or within a *statement*

some statements like func calls can also act like expressions

EXPRESSIONS

Expressions calculate (evaluate) and return one or more values (multiple).

```go
package main
import "fmt"

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

EXPRESSIONS

There's only one expression here: "Hello!" string literal.

```
package main
import "fmt"

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

EXPRESSIONS

Operators allow you to combine expressions.

```
package main
import "fmt"

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

"Hello!!"

EXPRESSIONS

Functions can be used as expressions inside statements.

```
package main
import (
    "fmt"
    "runtime"
)

func main() {
    fmt.Println(runtime.NumCPU())
}
```

EXPRESSIONS

You can use operators with function call expressions.

```
package main
import (
    "fmt"
    "runtime"
)

func main() {
    fmt.Println(runtime.NumCPU() + 1)
}
```

EXPRESSIONS

You can use operators with function call expressions.

```
package main
import (
   "fmt"
   "runtime"
)

func main() {
   fmt.Println(runtime.NumCPU() + 1)
}
```

COMMENTS

documentation

```
// single line comments

/*
  multi line comments
  more than single line
*/
```

GO DOC

creates documentation automatically

EXPORTING

allows a **package** to make its **functionalities**
available to **other packages**

to export a name just make its first letter an uppercase letter