Level 1–1

3, 2, 1… Go!

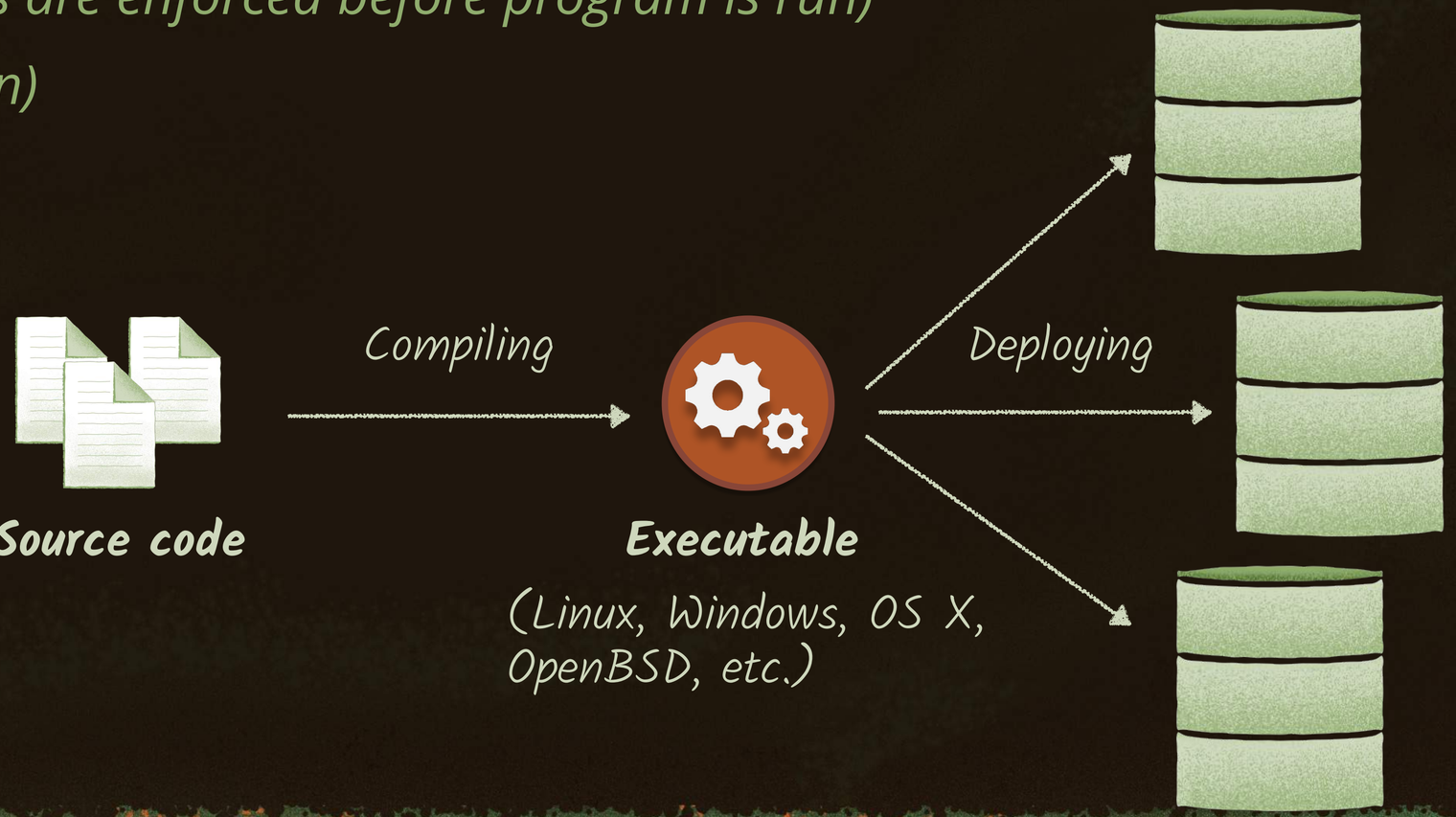
Taking the First Steps

ON TRACK with GOLANG

What Is Go?

Go is an **open-source** programming language created by Google in 2007. It makes it easy to build **simple, efficient** programs. Here are some of its main characteristics:

- Compiled *(compiler generates single executable file)*
- Statically typed *(types are enforced before program is run)*
- Fast *(concurrency built in)*
- Easy to deploy
- Fun to write!



Systems Programming

Go is a great language choice for writing lower-level programs that provide services to other systems. This type of programming is called **systems programming.**

Allows users to perform tasks
(Very common — your friends and family all use these)

Application Programs

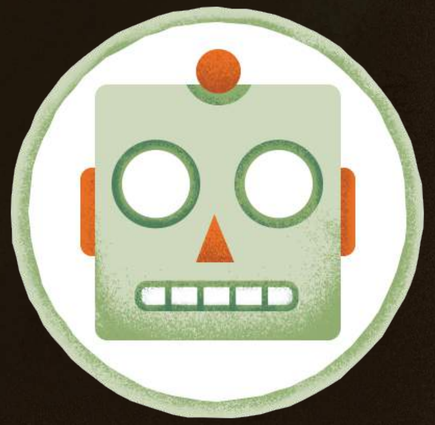
- E-commerce
- To-do lists
- Text editors
- Music players

vs.

Provides services to other systems
(By developers, for developers)

System Programs

- APIs
- Game engines
- Network applications
- CLI apps *(command line)*



What We'll Learn

In this course, we'll look at the most common features used when writing programs in Go.

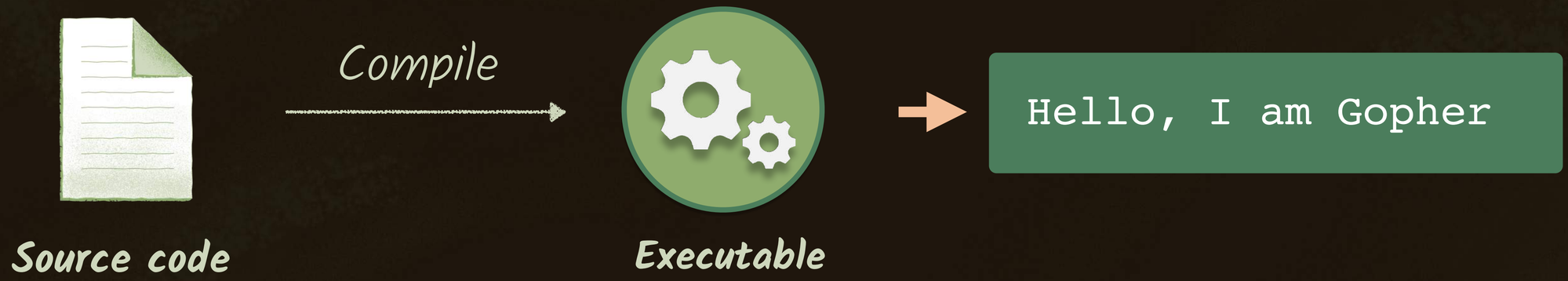
Things we'll learn:

- Building and running programs
- Importing and creating packages
- Basic constructs *(functions, variables, loops, conditionals)*
- Data types
- Concurrency with **goroutines**

ON TRACK with GOLANG

Our First Go Program

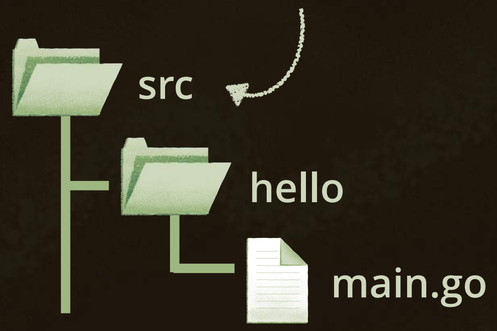
We'll start with a simple *Hello World* program in a single source code file. Once compiled, we'll use the executable file to run the program, which will print a message to the console.



main — the Entry Point

All runnable Go programs must have a main package and one main function.

The Go environment assumes all projects live under a src folder.



It's a convention in Go to use main as the name of single file programs.

src/hello/main.go

```
package main

func main() {

}
```

A package definition is always the very first thing in a Go source file.

The main() function is the entry point for all Go programs. It MUST have this name.

The func keyword declares a new function.

Printing From main()

Package **imports** go after package definition. The Println() function belongs to the fmt package.

src/hello/main.go

```
package main

import "fmt"

func main() {
  fmt.Println("Hello, I am Gopher")
}
```

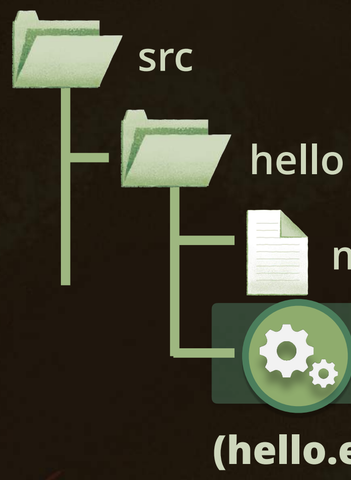
Packages used by the program must be explicitly imported.

No semicolons necessary!

This function from the fmt package prints a message to the standard output (a.k.a., console).

Building and Running With go build

A **compiler** reads source code and produces one executable file. We call this the **build process.**



(hello.exe *on Windows)*

Must be run from the project root

Output of the build command on Unix machines

src/hello/main.go

```
...
```

The build command is shipped with Go...

```
$ go build
```

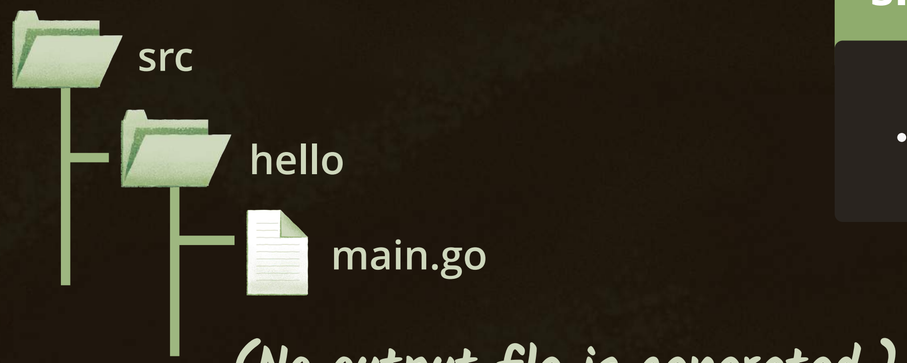
...it produces a single executable file named after the project.

```
$ ./hello
```

```
Hello, I am Gopher
```

Running with go run

Using go run**, we can build and run programs in one command, which makes things a bit easier.**



(No output file is generated.)

src/hello/main.go

```
...
```

The run command is shipped with Go...

```
$ go run main.go
```

```
Hello, I am Gopher
```

...it builds AND runs our code straight from the source file.

The gofmt Command

The gofmt command ships with Go and can be used to **automatically format** source code.

Valid syntax but hard to understand

src/hello/main.go

```
package
main
import"fmt"
func main(){
fmt.Println("Hello, I am Gopher")}
```

```
$ gofmt -w main.go
```

The -w flag is used to write the results to the original source file instead of printing to console.

Valid syntax AND easy to understand!

src/hello/main.go

```
package main

import "fmt"

func main() {
  fmt.Println("Hello, I am Gopher")
}
```

Formatting Source Code With gofmt

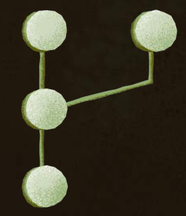
There are many benefits when following gofmt's formatting rules.

- Uncontroversial decisions *(less time spent arguing about code style)*
- Easier to **write, read**, and **maintain** programs

Many plugins are available for text editors, IDEs, and version control systems that make it easy to run gofmt automatically.

gofmt ***could be configured to run on every file save...***

...or before each commit.



http://go.codeschool.com/gofmt