An Exploration of Go

Admin

- Syllabus
  - Just to cover all the bases
- Who is here?
  - Reverse roll call

This Class

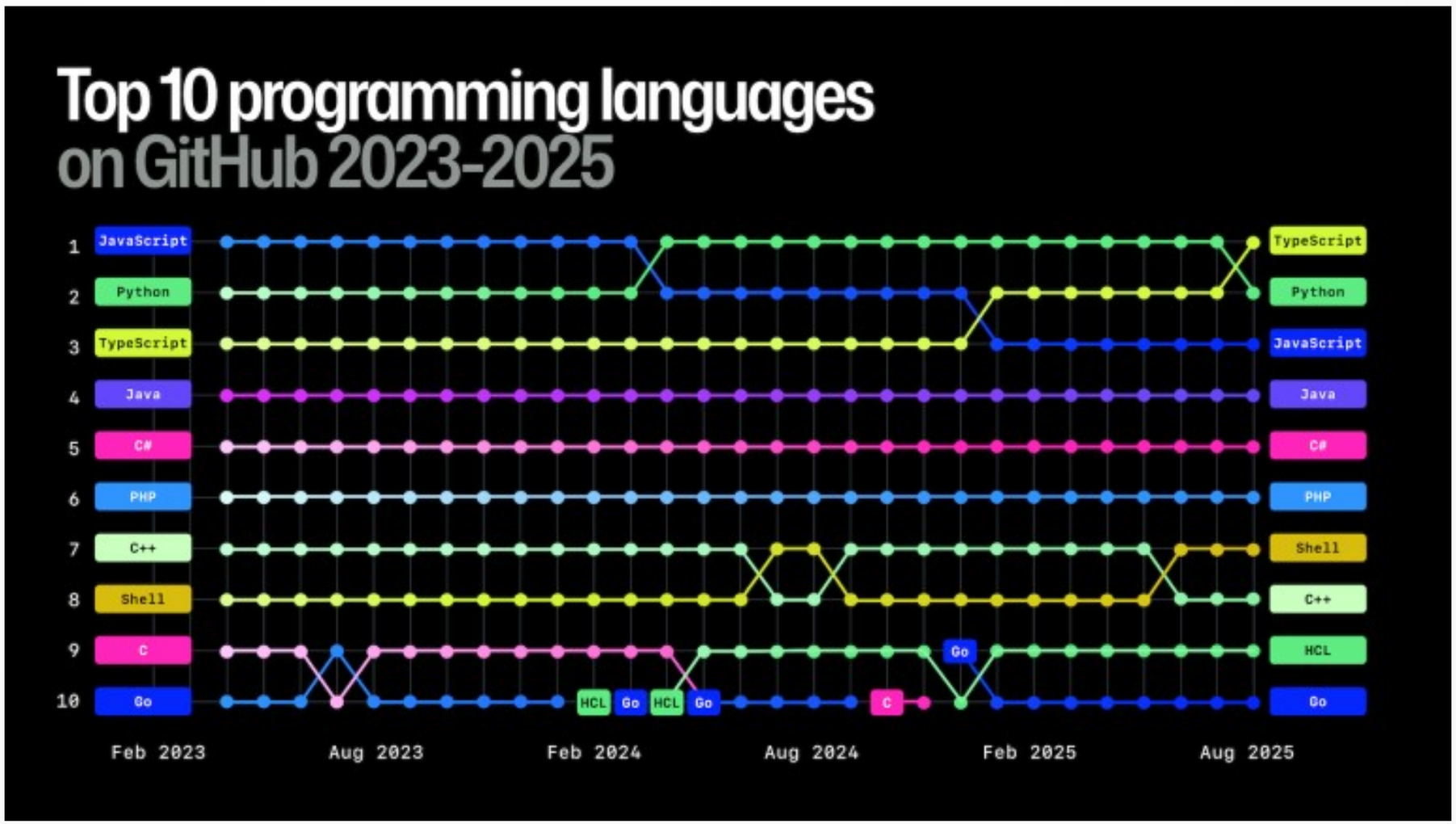
- What is this class all about?
  - A fairly deep dive into the go programming language
    - Go beat out C++ and Dart (C++ by only a little) in your voting
    - Fairly deep, won’t be looking into the internals of the compiler
  - Go
    - a new(ish) programming language.

Why Go?

- IEEE Programming language usage (probably best)
  - https://spectrum.ieee.org/top-programming-languages-2025
- And in recent survey of job demand, at position 6
  - https://www.itransition.com/developers/in-demand-programming-languages

Why Go?

- In All of Github (including non-open source) go is only a top 10 language.
-



Very Punny

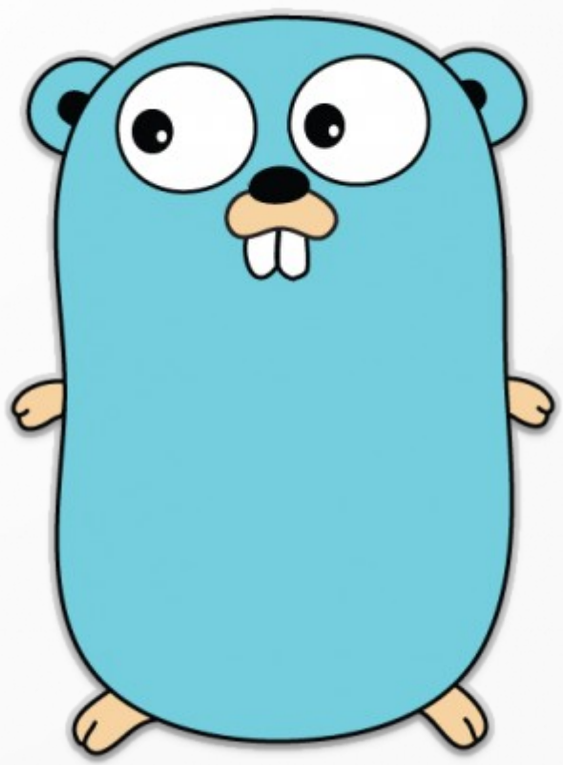
- You call a python programmer? .....
- What do you call a Go programmer?

Very Punny

- You call a python programmer? a pythonista
- What do you call a Go programmer?

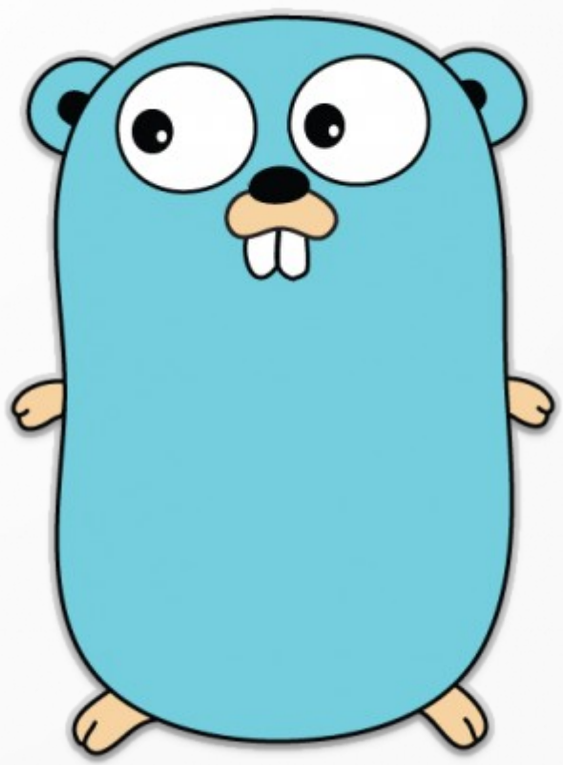
Very Punny

- What do you call a Go programmer?
  - A gopher
- The pun is better when you see the go tools



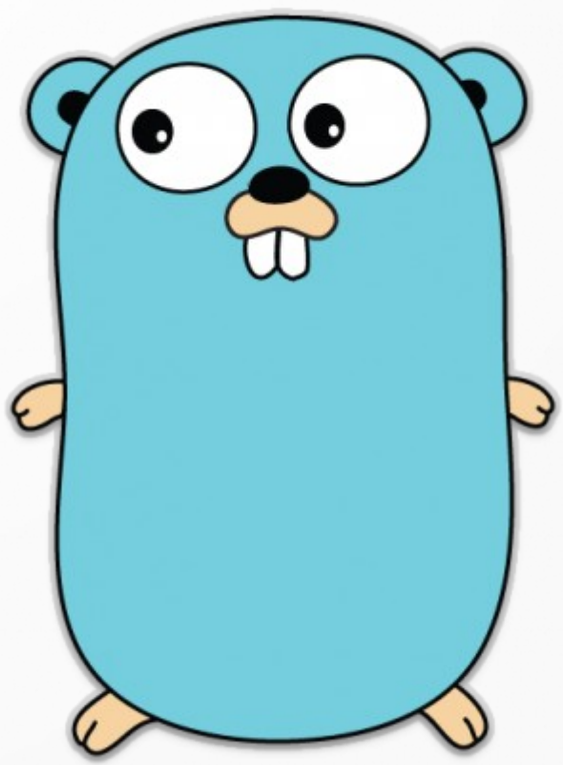
Very Punny

- What do you call a Go programmer?
  - A gopher
- The pun is better when you see the go tools
- How do you install a package at the python command line by default?



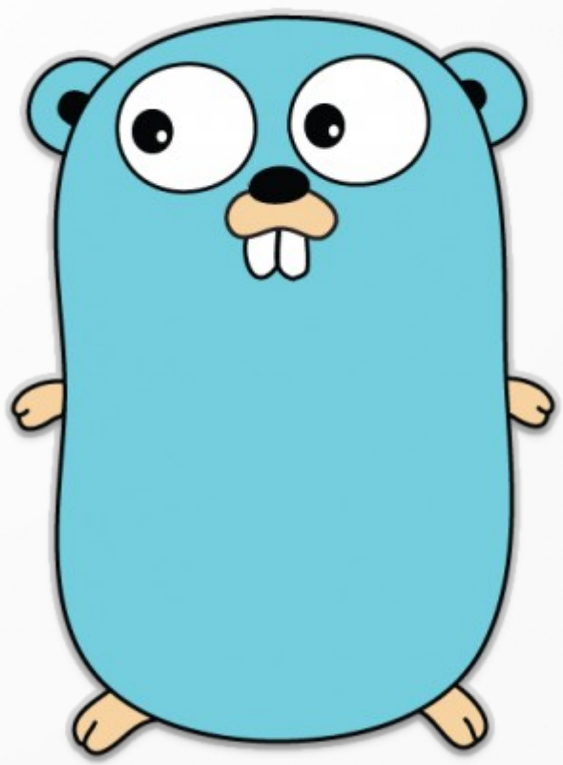
Very Punny

- What do you call a Go programmer?
  - A gopher
- The pun is better when you see the go tools
- How do you install a package at the python command line by default?
  - pip install <package>



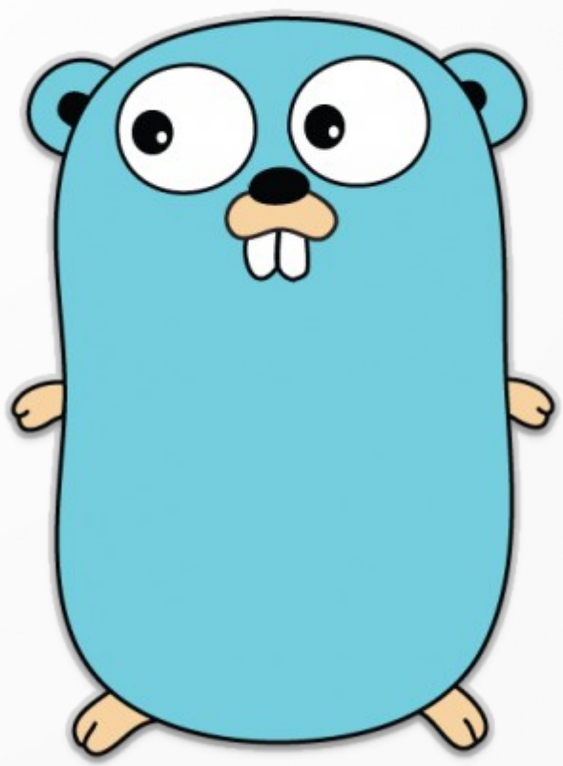
Very Punny

- What do you call a Go programmer?
  - A gopher
- The pun is better when you see the go tools
- How do you install a package at the go command line?



Very Punny

- What do you call a Go programmer?
  - A gopher
- The pun is better when you see the go tools
- How do you install a package at the go command line?
- go get <repo>
- gofer this, gofer that
  - Though modules are replacing go get.



Hello world

- Following the tradition set down by Brian Kernighan for BCPL, lets look at the “hello world” program for these languages
  - See the minimal program that produces output

Go (Golang) Hello World

- The canonical hello world from https://tour.golang.org/welcome/1

```
package main

import "fmt"

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

- Note the deliberate highlighting of multilanguage support

boring

- Go sets out to be ‘boring’
  - Code in industry/real world is often read far more often than written.
    - Especially in the LLM world
  - Go wants to present easy to read code to your colleagues.
- Go 1.x compatibility promise
  - Go adds features, but doesn’t break any existing code.
- Deliberately slow to add features:
  - https://drewdevault.com/2019/03/25/Rust-is-not-a-good-C-replacement.html

Compiled

- Go is a compiled language
  - So what does that mean?
    - Lucky Volunteer™?

Compiled

- Go is a compiled language
  - So what does that mean?
    - The code is converted to machine code for the machine you will run on, you end up with an app/exe that runs natively
    - Unlike python (interpreted) or java (compiled to ‘jvm’ then interpreted)
    - That native binary exists after ‘linking’
      - So what is linking?
      - Lucky Volunteer™?

Compiled

- Go is a compiled language
  - So what does that mean?
    - The code is converted to machine code for the machine you will run on, you end up with an app/exe that runs natively
    - That native binary exists after 'linking'
      - So what is linking?
      - Lucky Volunteer™?
    - Compile creates a bunch of binary 'object files' usually one per source file linking takes all of those objects and any supporting library code and puts it all together for a final executable.

Linking

- Two kinds of linking
  - Static Linking
  - Dynamic Linking
  - Anyone know the difference between them?
    - We don’t cover this in required classes before OS any more I suspect.

Linking

- Two kinds of linking
  - Static Linking
    - The external library code needed for the program to run is incorporated into the actual final program.
      - Program is bigger, but contains everything
  - Dynamic Linking
    - The external libraries are shipped at dlls (windows) or so (shared object on linux) and shared between all of the programs that use that library.
    - The program, when it needs to call a function from that library, just goes out and calls it from the dll/so
  - Which one seems better in 2026? let’s discuss

Lets try go in golang

- Command line possible,
  - Post go 1.11 module support (which was years ago)
- I suggest IDE usage.
  - We'll start with go module usage (discuss)
  - Might go with go dep later
- Lets create a go project
  - Command line : `go mod init <project name>`
  - In IDE <new project> will do the above for you.

Modern Tool Chain

- The go toolchain handles
  - Building (compiling and linking)
  - Dependency management
  - Package management
  - Toolchain management
- Most done through go.mod file for a project.
  - Since go 1.21, the go mod file can specify a go version and the go toolchain will update and download a new version of go if needed.

Assignment

- End of intro
  - Now go install the compiler
    - at least go 1.26 (go 1.27 comes out in Aug – but the I've refreshed everything in the summer before it came out.)
  - Install IDE tooling – get goland from jetbrains
    - As students you can get the professional version for free.
    - VS code fine if you support it – I need to specialize.
  - Install Git if you haven't already.
  - Send me your github username/id (even if you did it for a previous class)
  - See resources page.
  - Read preface and chapter 1 in Learning Go