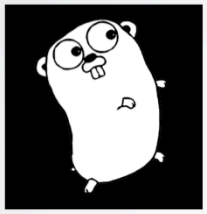


Go

Do you even userland thread?

About Me (Well, the parts you care about)

- Sarcastic
- Says "YOLO"/#swag a lot
- Likes computers
  - And Go
- Rips through slides

If you like…

- Java
- C
- C++
- Python
- Ruby

…Don’t take any part of this talk personally. ☺

Python/Ruby/Node devs:

What do you like about it?

- Dynamic typing
- Batteries-included
- ALL the REPLs
- (Node) Callback-oriented
- Package managers (pip, gem, npm)
- Expressivity

Python/Ruby/Node devs:

What do you **not** like about it?

- Dynamically typed
- (Node) Callback-oriented
- Slow
- CPython/MRI's "threads" suck
- "High performance" libs aren't actually written in Python/Ruby/Node

C/C++/Java devs:

What do you like about it?

- Fast
- Actual threading
- Static typing
- Control over the details

C/C++/Java devs

What do you hate about it?

- Batteries not included ever anywhere period (especially in C)
- Control over the details
- WRITING OUT TYPES
  - `ArrayList<Map<Integer, String>> al = new ArrayList<Map<Integer, String>>();` is for MASOCHISTS
- (C/C++) Makefiles and all of their glory
- (C/C++ [kinda]) Manual memory freeing
- (Java) Classes EVERYWHERE
- (Java) startup times

Oh, also, for that C++ bit…

How C++ says “You forgot a comma”

```
lexer_test.cpp:86:2: error: could not convert ‘{<expression error>, lexer::instruction((*(const std::vector<std::basic_string<char> >*)(& std::vector<std::basic_string<char> >{std::initializer_list<std::basic_stri
ng<char> >{((const std::basic_string<char>*)(& const std::basic_string<char> [1]{std::basic_string<char>(((const char*)"LabelOne"), (*(const std::allocator<char>*)(& std::allocator<char>())))})), 1u}, (*(const all
ocator_type*)(& std::allocator<std::basic_string<char> >()))))), (*(const string*)(& std::basic_string<char>(((const char*)"baz"), (*(const std::allocator<char>*)(& std::allocator<char>()))))), (*(const std::vecto
r<lexer::token>*)(& std::vector<lexer::token>())), 7), lexer::instruction((*(const std::vector<std::basic_string<char> >*)(& std::vector<std::basic_string<char> >())), (*(const string*)(& std::basic_string<char>((
(const char*)"pinkle"), (*(const std::allocator<char>*)(& std::allocator<char>())))), (*(const std::vector<lexer::token>*)(& std::vector<lexer::token>{std::initializer_list<lexer::token>{((const lexer::token*)(&
const lexer::token [1]{lexer::token(std::basic_string<char>(((const char*)"pie"), (*(const std::allocator<char>*)(& std::allocator<char>()))))))), 1u}, (*(const allocator_type*)(& std::allocator<lexer::token>()))))
)), 8), lexer::instruction((*(const std::vector<std::basic_string<char> >*)(& std::vector<std::basic_string<char> >())), (*(const string*)(& std::basic_string<char>(((const char*)"yolo"), (*(const std::allocator<c
har>*)(& std::allocator<char>())))), (*(const std::vector<lexer::token>*)(& std::vector<lexer::token>{std::initializer_list<lexer::token>{((const lexer::token*)(& const lexer::token [1]{lexer::token(std::basic_st
ring<char>(((const char*)"swag"), (*(const std::allocator<char>*)(& std::allocator<char>()))))))), 1u}, (*(const allocator_type*)(& std::allocator<lexer::token>()))))), 12)}’ from ‘<brace-enclosed initializer list
>’ to ‘const std::vector<lexer::instruction>’
```

THIS SINGLE ERROR MESSAGE.
IS LONGER.
THAN OVER HALF OF MY PYTHON PROGRAMS.

WHY

…And don't forget those *blazing fast* compiles

```
dyrevps:assembler:% time g++ -std=c++11 -O2 -c assembler.cpp
g++ -std=c++11 -O2 -c assembler.cpp  1.18s user 0.15s system 99% cpu 1.327 total
dyrevps:assembler:% wc -l assembler.cpp
386 assembler.cpp
dyrevps:assembler:%
```

1.3 seconds. For a file. < 400 lines. Not even using Boost. Or any crazy templates. For stage 1 of compilation. SERIOUSLY?!
(Visual Studio clocked in at around 2.5s)
…For reference, a 182 line Go program takes 0.26s to compile, link, and produce an executable.

So now that we know what we like and hate

What is Go?

- A verb
- A game
- A programming language
- …And probably 800 other things

Why is this annoying?

go basics

Web Images Maps Shopping Videos More Search tools

About 389,000,000 results (0.28 seconds)

The Interactive Way To **Go**
playgo.to/iwtg/en/
30+ items - Java-based interactive tutorial in several languages, in 33 brief ...
50 Kyu Problems Capture Atari Stones
49 Kyu Problems A bit more difficult
The Interactive Way To Go - Komi - 40 Kyu Problems - 48 Kyu Problems

GO For Beginners: **GO Basics** - Drennen Consulting dot Com
www.drennenconsulting.com/Games/**GO**/**go_basics**.htm
The gaining of territory is far more important than the capturing of stones! Territory is vacant intersections entirely within the control of safe stones (having 2 eyes ...

Rules of **Go** - introductory at Sensei's Library
senseis.xmp.net/?RulesOfGoIntroductory
May 23, 2013 - Rules of **Go** - introductory. Difficulty: Introductory Keywords: Opening, Rules. This page introduces the **basics** of **Go** rules. These rules allow you ...
1. Players - 2. Board - 3. Point - 4. Stones

See results about

Go basics
Book by Pete
Includes inte
one of the old

How do we get around it?

golang basics

Web Images Maps Shopping Videos More Search tools

About 19,300 results (0.30 seconds)

A Tour of Go
tour.**golang**.org/
The tour is divided into three sections: **basic** concepts, methods and interfaces, and concurrency. Throughout the tour you will find a series of exercises for you to ...
Go's Declaration Syntax - Slices: usage and internals - Image - Cmplx

Writing Web Applications - The Go Programming Language
golang.org/doc/articles/wiki/
Assumed knowledge: Programming experience; Understanding of **basic** web technologies (HTTP, HTML); Some UNIX/DOS command-line knowledge ...

The Go Programming Language
golang.org/
Documentation, source, and other resources for Google's Go language.
6,185 people +1'd this

What is Go?

- Smart.
- Intelligently statically typed
- Batteries included
- Fast startup
- Fast compiles (yes, compiles)
- Actual threads
  - But smartful.
- Package manager for free if you call now.

The best of both worlds!

…Mostly.

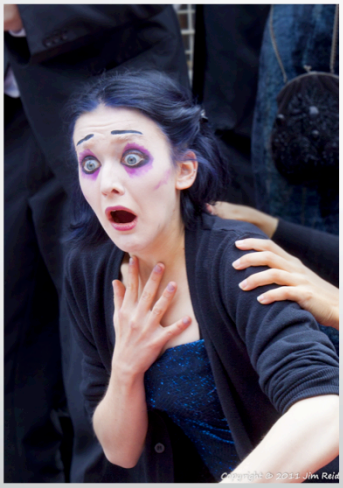
BTDUBS

- http://golang.org
  - Amazing site 10/10 would visit again
  - Has tutorial, etc. http://tour.golang.org
    - Also these are both powered by go

What is Golang?

- A modern systems language

SYSTEMS LANGUAGE?!



ERMAGERD.

Calm down.

Yes, systems language. Moving on.

What is Golang?

- Pioneered by (#namedropping)
  - Rob Pike (co-creator of UTF-8, original UNIX dev at Bell, original Plan 9 dev…)
  - Robert Griesemer (worked on HotSpot, Chrome's JS engine, Cray optimizing compiler)
  - Ken Thompson (co-creator of UTF-8, inventor of B PL, designer/implementer of UNIX, creator of Plan 9…)

(No one cares though)

What is Golang?

- Works on Linux, Mac, FreeBSD, OpenBSD, Plan 9, Windows
  - ARM + x86 support
- So, write once, run anywhere
  - "The biggest lie in history"
    - Well, maybe excluding "I have read and accept the terms and conditions…"

Where Go?

- Embedded environments
- Webservers (dynamic/static)
- Scripting/Automation
- Interfacing with C/C++
- Anything that needs to scale up
- **Instead of** C/C++
  - In some cases
  - Inb4 kernel panic

Where Go?

- Google
  - dl.google.com, golang.org, Vitess system, and more they're not talking about
- Canonical
- Bitly
  - Replacing some of their backend C and Python with Go
- Airbrake
  - Migrated form Ruby
- BBC Worldwide
  - Web crawlers, backend, web services
- Soundcloud
- Torbit
- Nokia
- Carbon Games
- AND LOTS MORE [http://golang.cat-v.org/organizations-using-go]

1.0 came out in 2012. > 40 companies openly say they use it less than a year after getting out of beta. Sexy. As. Fuck.

Why is Go cool?

- Garbage Collection
  - FREEDOM
    - ...From free().

Why is Go cool?

- Both imperative and OO
  - class HelloWorld need not apply

Why is Go cool?

- Statically typed **with plenty of** type inference

  myVar := GetNameMap()

  Vs

  Map<String, AbstractFactoryFacadeAdapterPersonActorImpl> myMap = GetNameMap()

  ...While still proactively checking for

  getLengthOfString(5);

Why is Go cool?

- All* arrays are auto-resized like ArrayList
- They also support syntax like Python:

```
b := a[:len(a)/2]
c := a[len(a)/2:]
```

- Maps (dictionaries, whatever) come built in
- Multiple return values from one function
- Semicolon inference (like Python)
- Lots of other conveniences

Why is Go cool?

- Sensible type names!

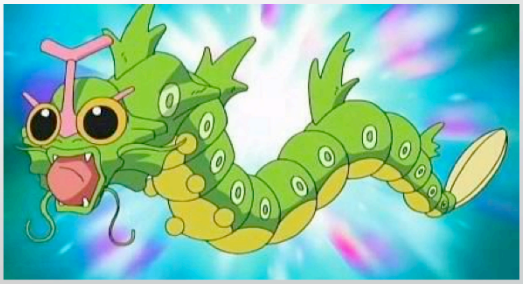
Who's that Type? (Oppan C Style)

Who's that Typemon?

- int** i;
  - A pointer to a pointer to an int

Who's that typemon?

- int** var[]
  - "An array of pointers to pointers to ints"
  - Wait, what?
  - An Array
  - Of Pointers
  - To Pointers
  - To ints



Wait, you DON'T read backwards just for certain types and forwards for most others?

You filthy casual.

Sensible type names

- Same types, but in Go.
  - **int
    - "Pointer to a pointer to an int"
  - []**int
    - "Array of pointers to pointers to ints"

Well,
Shit...

Why is Go cool?

- Designed to be developer-friendly
  - Fast to EVERYTHING
  - Batteries included
  - Holy crap the tools
    - Go test
    - Go doc
    - Go get
    - If I don't give a demo here yell at me

Fast to EVERYTHING means:

Fast to compile (go is meant to build large programs in seconds on a modern desktop)

Fast to run (Approaching speed of C++, but is a bit weak when doing tons of stuff with arrays due to bounds checking. And the GC is a bit weak.)

Fast to code (Python-style syntax and KISS coding make code "how will I do this task?" as opposed to "Should I use pattern matching with switch, or do if n.x == 0 else if n.y == 0 else …

All go tools can be run like:

`go run myscript.go`
`go doc io`
`go get github.com/dyreshark/yolo`
`go vet # Built-in go linter.`

Why is Go cool?

- Oh also, there are pointers, but no ->
  - Yeah, compiler decides when to use -> for you
  - Just use .
  - Ezpz
  - See? Go smahrt.

Now that we're around slide 35, I'm assuming you're interested

If not, stay anyway. Like… Reddit can wait, maaaaaaaan(/mississsssssss).

(Ok the mississsssssss sounded less creeper in my head than it does out loud)

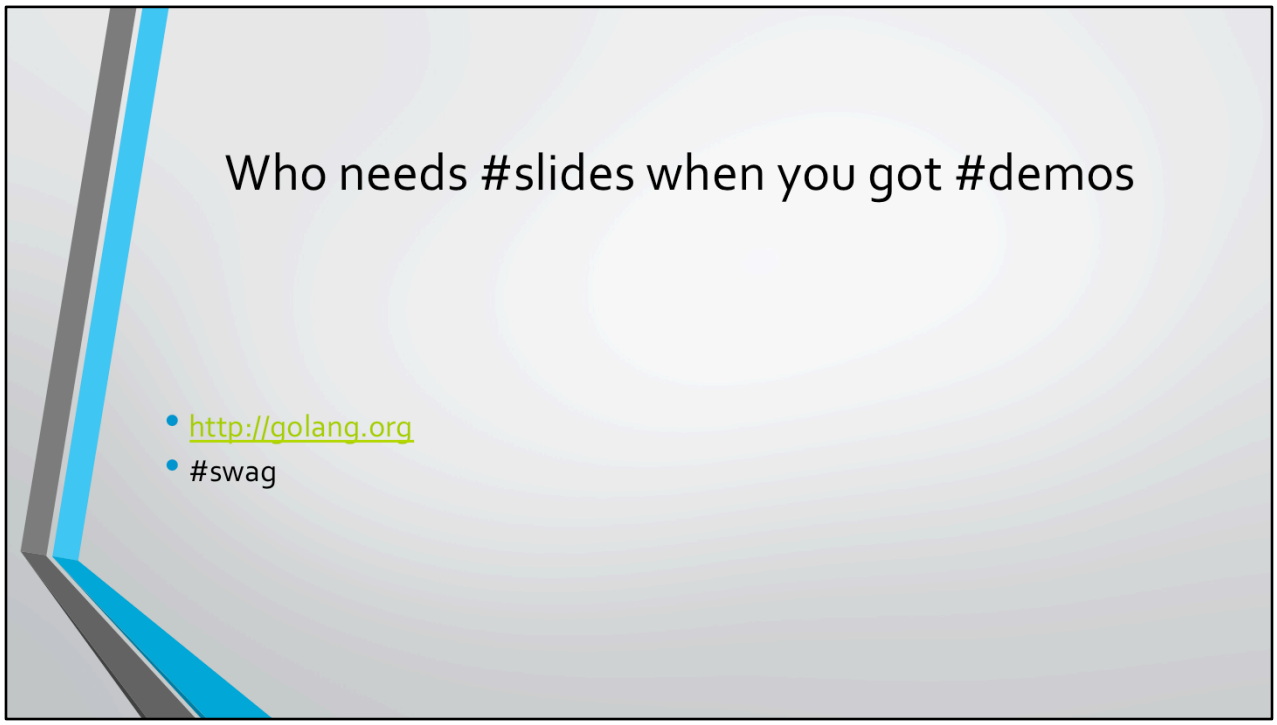
Speaking of slide 35

- We've gone a TON of slides without a full line of actual Go.

Ok this is Go

```
(defn main [[h & rest :as l]]
 (when-not (empty? l)
  (print h)
  (recur rest)))
```

Jk.



I just ran the Hello World! On the front page

Realtalk - Channels

- Lots of analogies for them.
  - Think shared concurrent queue.
  - If you haven't concurrencied before, think queue that coordinates lots of programs
  - One way socket, pipe, ConcurrentBlockingQueueWithOptionalBufferFactoryAdapterFacade (#Java) all work too

Realtalk - Channels

- How they look:
  - As much as code is beautiful in Powerpoint, I'll show you on/in EMACS, VIM, and golang.org.
  - Real-time demos. Awwwww yeeeee.

Code I ran on golang.org:

```
// You can edit this code!
// Click here and start typing.
package main

import "fmt"

func main() {
	ch := make(chan int, 1)
	ch <- 5
	fmt.Println("The number was:", <-ch)
}
```

Real code pls.

- Fibonacci generator!
  - Concurrent
  - Infinite Fibonaccis
  - Wow so fibonacci

<Used my code, fibs.go>
Run it with:

```
go run fibs.go
```

Let's make a few changes…

Or not.

Copypasta

- One friggin line.
- TeeReader is magic.
- Enter-faces

This was on copy.go and copy2.go
I ran copy.go, said “So copy takes two files, amirite?”
Then ran copy2.go, and the NSA recorded all of my things! How nice!
Showed copy2.go source.

Side note: if you hover over io.TeeReader in vim and type `:Godoc` with the Go plugins, it’ll take you to the documentation for io.TeeReader()

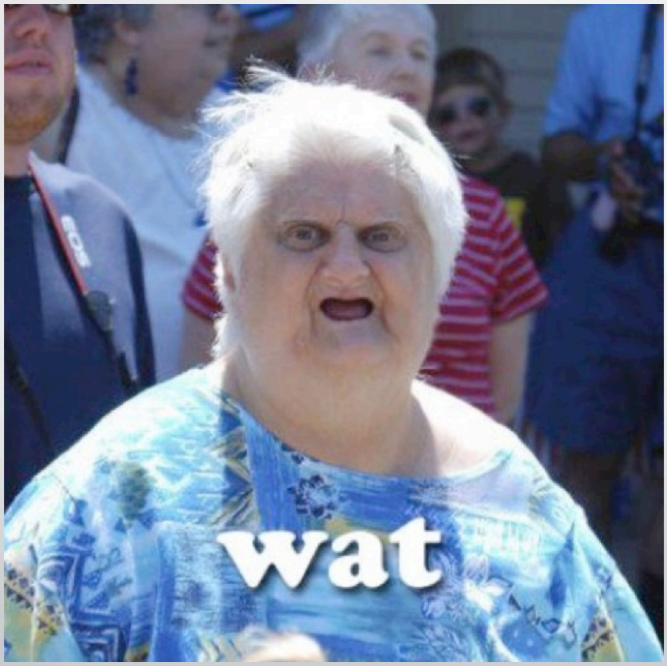
Getting all Socratic on yo brains

- Duck typing: If it walks like a duck and quacks like a duck, it's a duck.

```
def doubleSizeOfList(l):
     return len(l) * 2

>>> doubleSizeOfList({1: 2, 3: 4})
4
```

What is type?
Baby don't hurt me
Don't hurt me
No more


wat

Interfaces! What the struct?!

- Interfaces: Things that say "Some type must provide these methods"
- Structs: Things that carry data that you can tack methods on to
- Go figures out on its own if a type implements an interface
  - None of this class AbstractFactoryPattern implements AdapterFacadeInterface, Enumerator, Enumerable, YOLOSwagInterface

Tl;dr: If I have

```
interface Fooable {
        Foo() int
}
```

And

```
func fn(Fooable f) {
        /* … */
}
```

…fn can be run with ANY STRUCT EVER that has a method with the signature Foo() int. Even if they exist in the standard libs. Wow. Such Go.

Why is this cool?

- Specify arguments in terms of what functionality you need
  - More composability
- Fprintf to a buffer. Wat.

Fprintf takes an io.Writer instead of a file descriptor. So you can pass it a *byte.Buffer because it implements io.Writer.

Break – Questions?

Go is smart/Moar interfaces

- Yolo demo

<Ran yolowriter.go> .

Okok, so interfaces are really cool. What else?

- GOROUTINES
  - AHASKLDJLASJDSLAJ
  - And channels too.

Yes, that's it. Threads on threads. Why is that cool? Runtime will tell you when you have a deadlock (as well as the state of everything). Runtime will block goroutines waiting on mutexes/semaphores (swapping another goroutine on to the OS thread). Runtime will swap goroutines off the current core that are blocking for IO automagically. You can literally have 1,000 concurrent goroutines running on the same thread, no problem.

A FILM BY CHRISTOPHER NOLAN
INCEPTION
FROM THE DIRECTOR OF THE DARK KNIGHT

Why are Goroutines cool?

- Something something scalable
  - Impress ALL the upper management

Why are Goroutines cool?

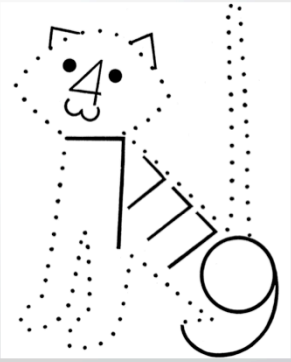
- "There are three types of dishonesty: lies, damn lies, and benchmarks"

So let's look at a benchmark!

<Ran benchmark.go>

Why are Goroutines cool?

- Numbers!
  - So many numbers
  - Normally takes ~30us for a normal context switch.
  - Normally takes > 4K RAM for a new thread.



Why are Goroutines cool?

- Tl;dr: They're FAST.
  - Who doesn't love fast?
- And they're lightweight
  - SO lightweight

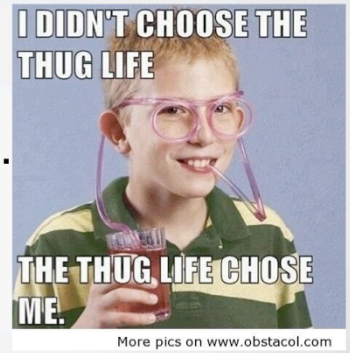
Why are Goroutines cool?

- They let you structure programs to be parallel-ready
  - But without tons of extra overhead everywhere
  - Actor model is sometimes a really natural fit (i.e. MMOs, etc.)
- See: Quicksort
- See: Web crawler at http://tour.golang.org/#71

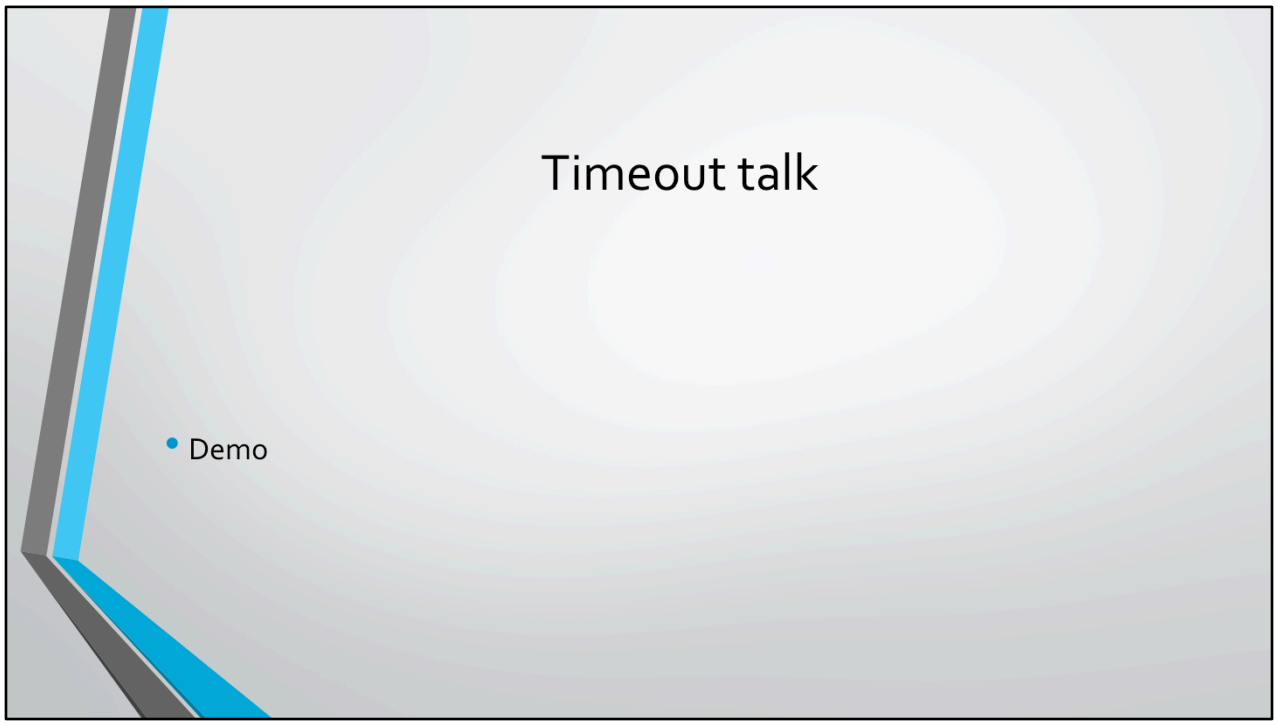
<Ran quicksort.go>

Smart locking mentioned above. Ran bitonic.go. Bitonic is implemented in terms of locks to show that you can have 30,000,000,000,000 things (not literally) waiting on one lock on one thread. So long as something else can run, that thread still does work.

MOAR channels.



- Optionally buffered ways for goroutines to communicate.
- Provide synchronization points
- Can select on multiple at once



<Ran timeout-talk.go>

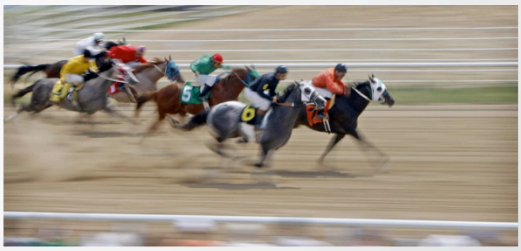
Take a breath.

Questions?

Ready?

Set **Go**! (tee hee)

Mother of concurrency!



- Go also has a race-detector built in.
  - Just compile code with –race, and it will yell at you for race conditions
    - But only if they actually happen
    - Has found tens of race conditions in the standard lib
    - Takes lots of CPU and memory though, so don't use in production

See:

`go run –race foobar`

import “github.com/dyreshark/yolo”

- Yes. You can import things (almost) directly from github.
- It’s just an extra step.
  - `go get github.com/dyreshark/yolo`
  - `go run whatever-needs-github.com/dyreshark/yolo`

See slides.

Go uses a directory structure to determine imports. go get will install packages under the necessary parts of this structure so all you need to do is:
`go get github.com/dyreshark/yolo`
…And in your code

```
import “github.com/dyreshark/yolo”

func IsMySwagOn() bool {
	return yolo.GetSwagStatus() == yolo.ON;
}
```

Code style

- "WEBCAT DOCKED ME 80 POINTS ON AN 85 POINT ASSIGNMENT FOR HAVING A SPACE AFTER MY IF WTF?!"
- Go has a standardized format that everyone uses.
- Don't like it? Too bad.
  - Still want to use it? `go fmt <code>`
  - Demo.

Literally. Just run `go fmt filename` and it will format it according to Go style guides.

But wait, there's more!

- …But this talk would get **massive** if I mentioned it all

So I'll just leave you with this:

- Heaps of fun demo

<Ran heapsoffun.go>

Note that I'm returning pointers to integers allocated on the stack. Go recognizes this and implicitly allocates the integers on the heap for me, so my code doesn't break.
Earlier [in yolowriter.go], I also did

```
func newYoloWriter() *yoloWriter {
	return &yoloWriter{ … } // The same thing applies here.
}
```

Stretch goals!

- Time permitting:
  - How to make a webserver using one weird old trick
    - C PROGRAMMERS HATE THIS MAN

<see server.go>

How to be successful with Go

- Learn from people who have been using the language for > 2 months (i.e. not me)
- Learn to let Go
  - (handle things)
- Think Different™
  - i.e. in terms of goroutines, so your program will scale like crazy
- Join the IRC channel at freenode/#go-nuts

Summary

- Go is cool
- You should use Go
- Focused primarily on scalability and ease-of-development
  - Without being ungodly slow
- Pseudo-duck typing + static typing = <3

Actual Summary

- Go's interfaces are amazing
- Go's dev tools are amazing
  - import straight from github + testing + formatting built in? YES PLEASE
- Go feels Pythonic where it needs to, but gets out of the way when it needs to
- Unified community + unified projects == win
  - None of this "OH NUMPY? LOL IT'S IN C BRO GL READING IT"
- Lightweight threads give you new ways to structure programs
  - Parallel == less scary now

THANKS FRIENDS

You should learn go:
http://tour.golang.org

I MUST GO
My people need me