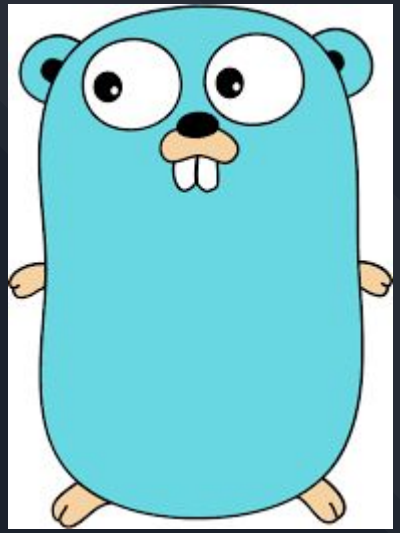
Why GoLang???



-Jap Leen

Introduction

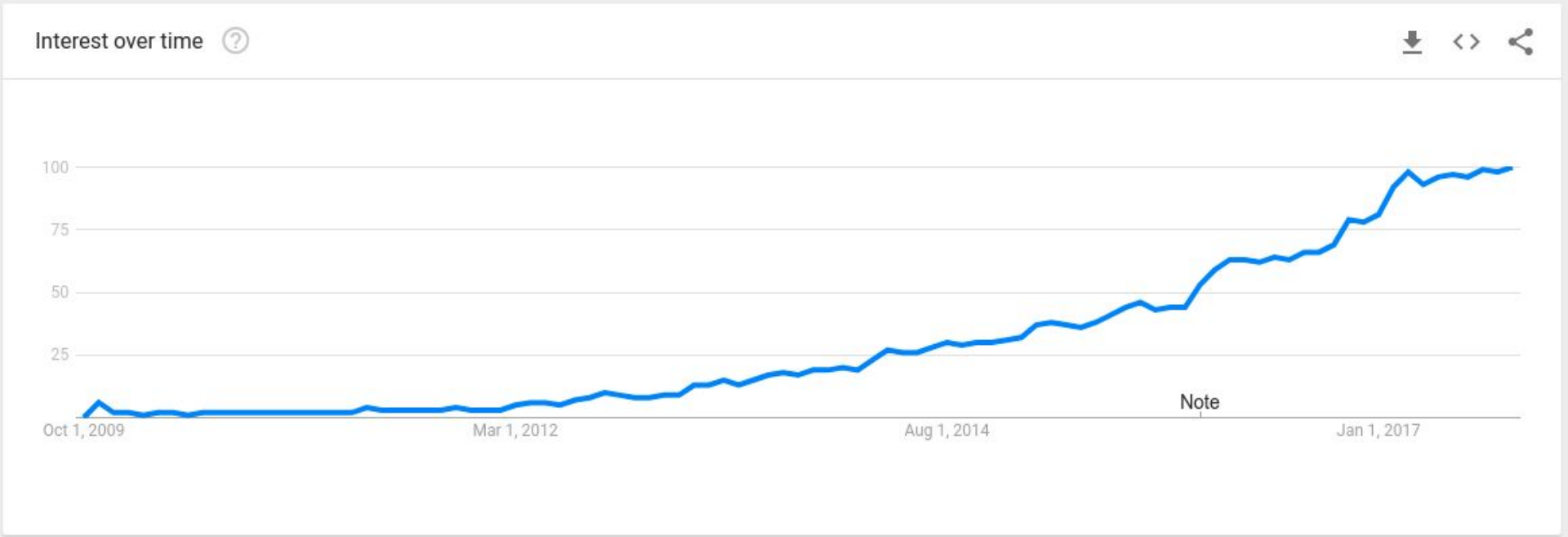
> A programming language created at Google, co-invented by Ken Thompson.

> An open source project, yayy!

> Meant to be simple to learn, straightforward to work with, and easy to read by other developers.

Some interesting facts and figures...

Interest over Time Graph



> Approximately one million users worldwide.

> In the 2017 Octoverse by GitHub, **Go has become the #9 most popular language**, surpassing C.

> The fastest growing language on GitHub in 2017, in the top 10, with **52% growth over the previous year**. In growth, Go swapped places with Javascript, which fell to the second spot with 44%.

> In Stack Overflow's 2017 developer survey , Go was the only language that was both on the **top 5 most loved and top 5 most wanted** languages.

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted

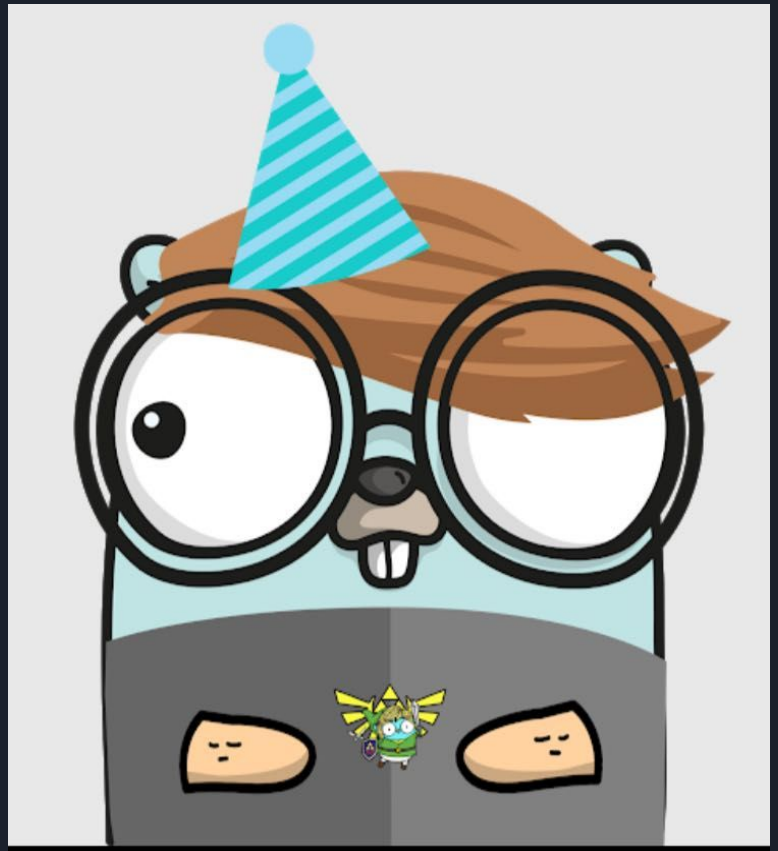
Loved | Dreaded | Wanted

By 2017, Go had emerged as the language of cloud infrastructure. Today, **every single cloud company has critical components of their cloud infrastructure implemented in Go** including Google Cloud, AWS, Microsoft Azure, Digital Ocean, Heroku and many others.

> A major force in the world of open source, powering some of the most popular projects and enabling innovations across many industries.

You can find thousands of additional applications and libraries at https://awesome-go.com/ .

You can find other such amazing latest survey results for 2017 at https://blog.golang.org/survey2017-results

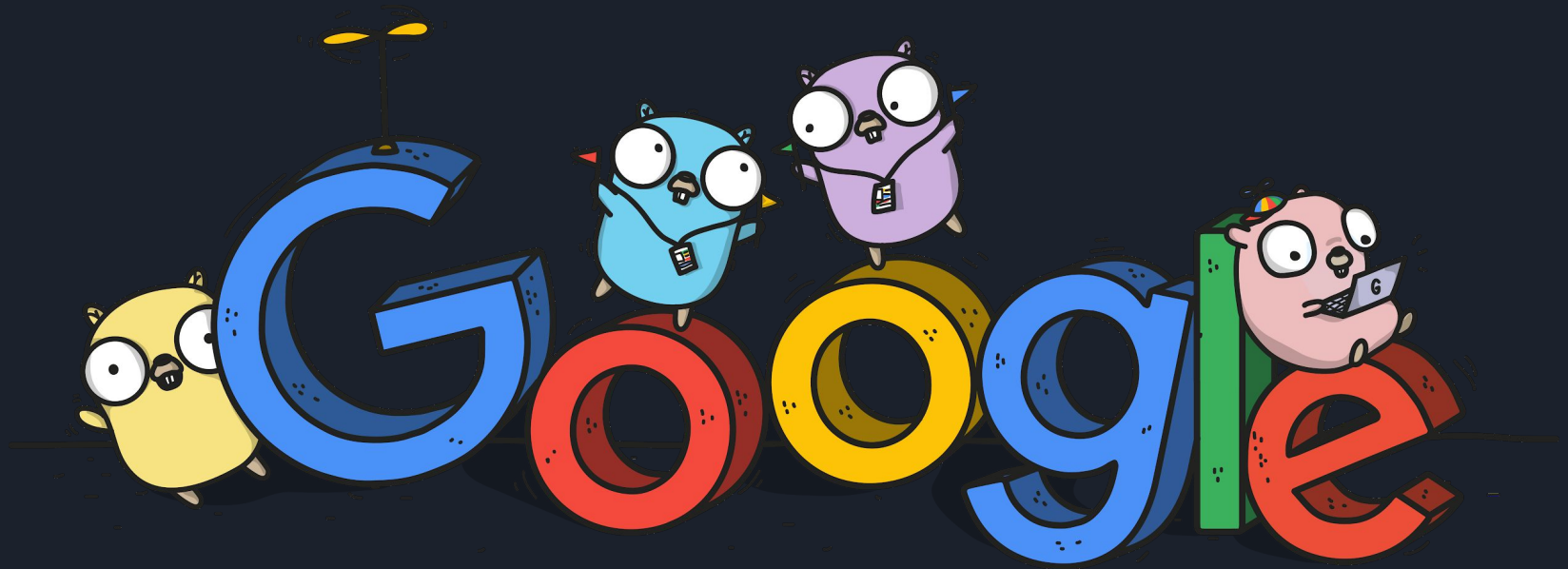


Some salient features and advantages of GoLang...

>> Easy to learn

>> Excellent in performance

>>Concurrency



>> Concurrency

> Large programs are often made up of many smaller sub-programs.

> For example a web server handles requests made from web browsers and serves up HTML web pages in response. Each request is handled like a small program.

> It would be ideal for programs like these to be able to run their smaller components at the same time (in the case of the web server to handle multiple requests).

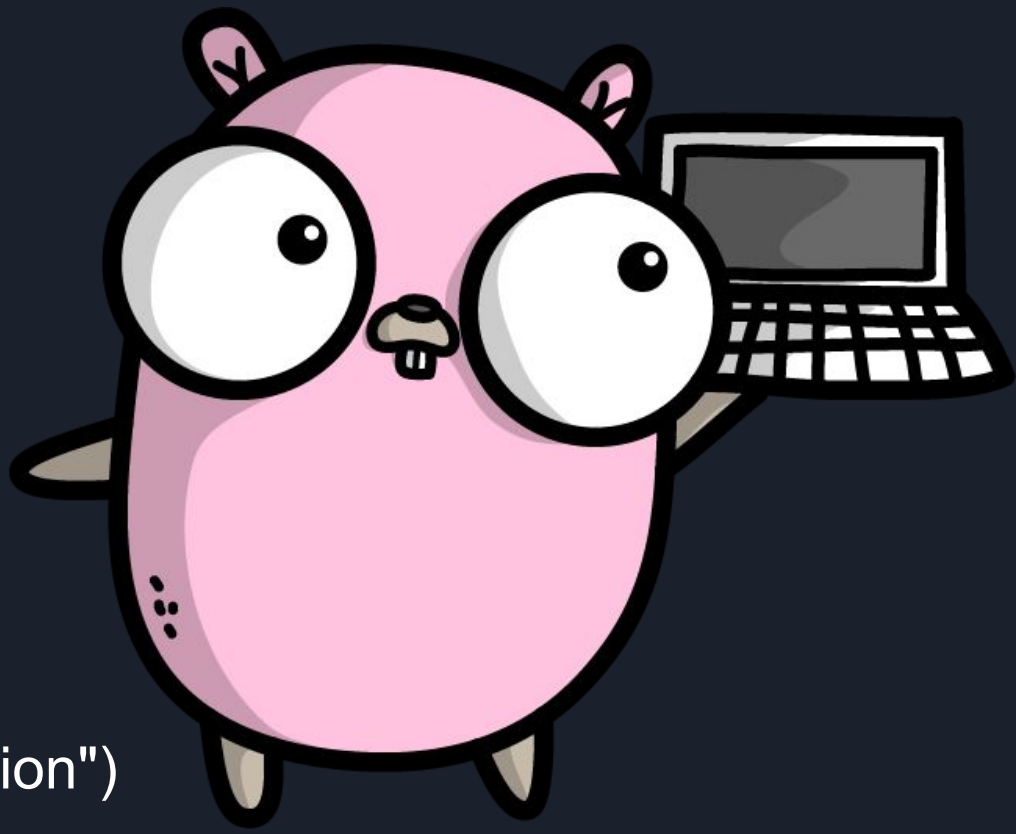
> Making progress on more than one task simultaneously is known as concurrency. Go has rich support for concurrency using goroutines and channels.

A *goroutine* is a function that is capable of running concurrently with other functions. To create a goroutine we use the keyword go followed by a function invocation.

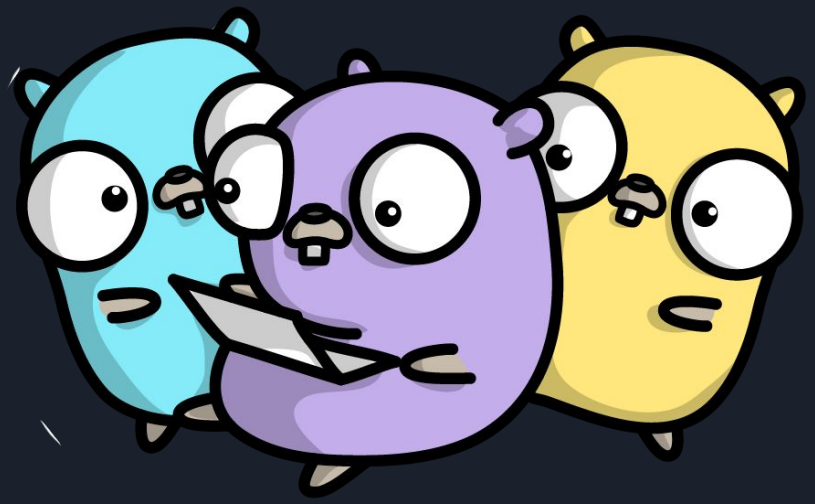
Goroutines are lightweight and we can easily create thousands of them.

Channels provide a way for two goroutines to communicate with one another and synchronize their execution.

```
package main
import ( "fmt")
func hello() {
    fmt.Println("Hello!")
}
func main() {
    go hello()
    fmt.Println("main function")
}
```



```
package main
import (      "fmt"   ;   "time"    )
func hello() {
    fmt.Println("Hello!")
}
func main() {
    go hello()
    time.Sleep(1 * time.Second)
    fmt.Println("main function")
}
```

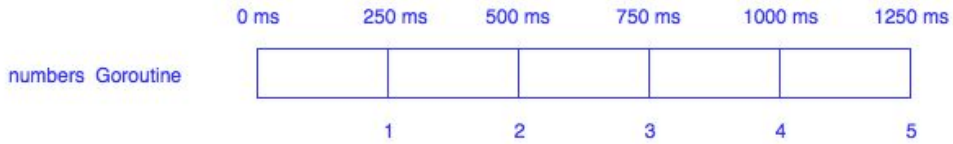


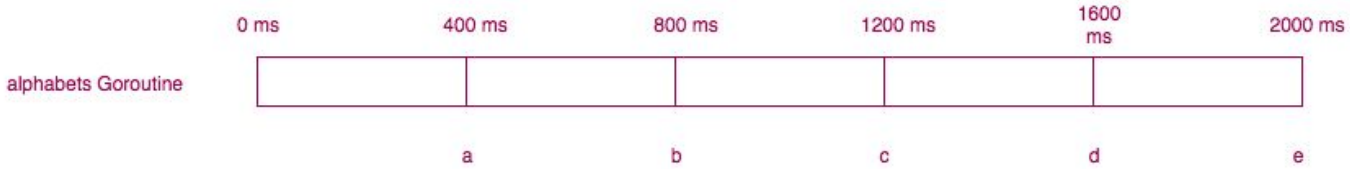
Challenge Time...!!!

Guess the output of
the code snippet. You have
2 minutes.

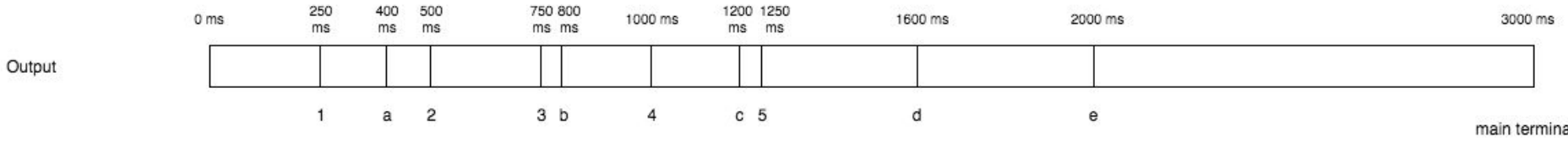
```
func numbers() {
    for i := 1; i <= 5; i++ {
        time.Sleep(250 * time.Millisecond)
        fmt.Printf("%d ", i)     }    }
func alphabets() {
    for i := 'a'; i <= 'e'; i++ {
        time.Sleep(400 * time.Millisecond)
        fmt.Printf("%c ", i)     }    }
func main() {
    go numbers()
    go alphabets()
    time.Sleep(3000 * time.Millisecond)
    fmt.Println("main terminated")      }
```

Answer: 1 a 2 3 b 4 c 5 d e main terminated


0 ms
250 ms
500 ms
750 ms
1000 ms
1250 ms
numbers Goroutine
1
2
3
4
5


0 ms
400 ms
800 ms
1200 ms
1600 ms
2000 ms
alphabets Goroutine
a
b
c
d
e

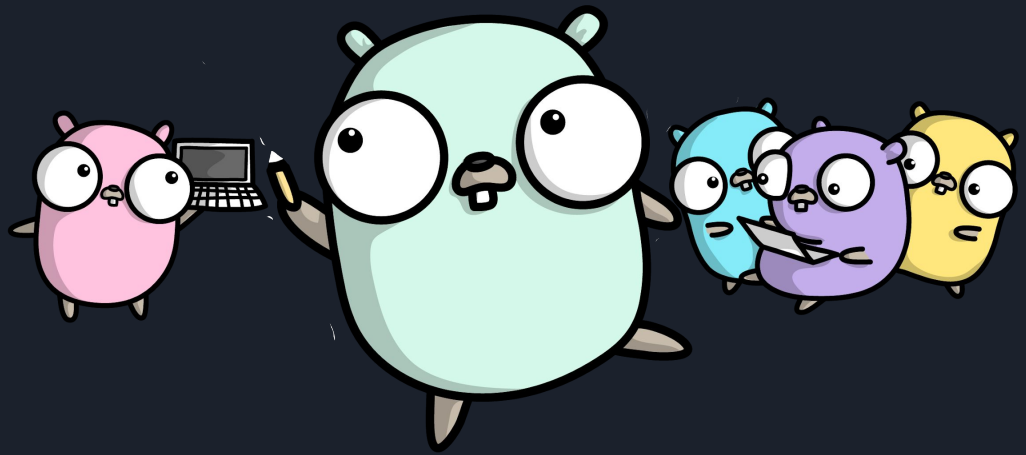

3000 ms
main Goroutine
main terminated


0 ms
250 ms
400 ms
500 ms
750 ms
800 ms
1000 ms
1200 ms
1250 ms
1600 ms
2000 ms
3000 ms
Output
1
a
2
3
b
4
c
5
d
e
main terminated

For more information on concurrency, goroutines and channels, head on to these links:

https://www.golang-book.com/books/intro/10

https://golangbot.com/goroutines/



Thank you!

All questions are welcome!

:)