
ลงทะเบียนอบรมหลักสูตร
Go Basic Workshop Round 1
22-23 April 2024 I 09:30: - 17:30 I The PARQ, 5th Floor - Incubator Area
Anuchito

https://github.com/pallat/intensive_go_basic_workshop

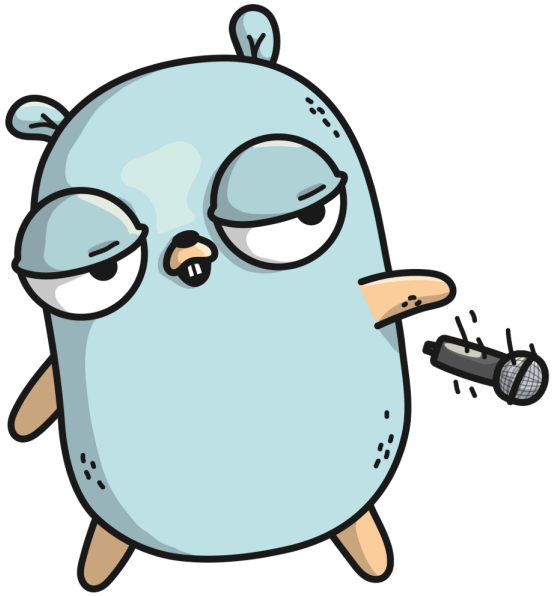
Basic Go & API

Pallat Anchaleechamaikorn

> Technical Coach
> Infinitas by KrungThai
> Arise by Infinitas

yod.pallat@gmail.com
https://github.com/pallat
https://dev.to/pallat

https://go.dev/tour (Thai)
https://github.com/uber-go/guide (Thai)



Installations

https://go.dev/

Hello World

```go
package main

import "fmt"

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

The most popular Go using

- API
- Batch
- CLI

Add Integer

$$sum = a + b$$

example: $1 + 2 = 3$

```
func(a, b int) int
```

```
addInt(1, 2)
```

Exercise: Add Floating Point number

```
addFloat(1.2, 1.3)
```

Exercise: Add String

```
addStringNumber("1","2")
```

> google: golang strings

คำนวณยอดชำระค่าผ่อนรถ

$$Finance\,amount = price - down$$

$$MonthlyInstallment = \frac{(Finance\,amount \times Annual\,interest\,rate \times Year) + Finance\,amount}{Installments}$$

```
ex:  500,000-50,000 = 450,000

(((450,000 x 3%) x 4ปี) + 450,000)  / 48 = 10,500
```

Demo: sum of even number

```go
var numbers = []int{11, 8, 5, 1, 4, 10}

func sumOfEven(numbers []int) int {

}
```

> 22

TDD Kata: String calulator

```go
func AddString(numbers string) int
```

ex:

```go
AddString("1,2")
```

Array & Slice

This is an array

```
var a [1]int
```

This is a slice

```
var a []int
```

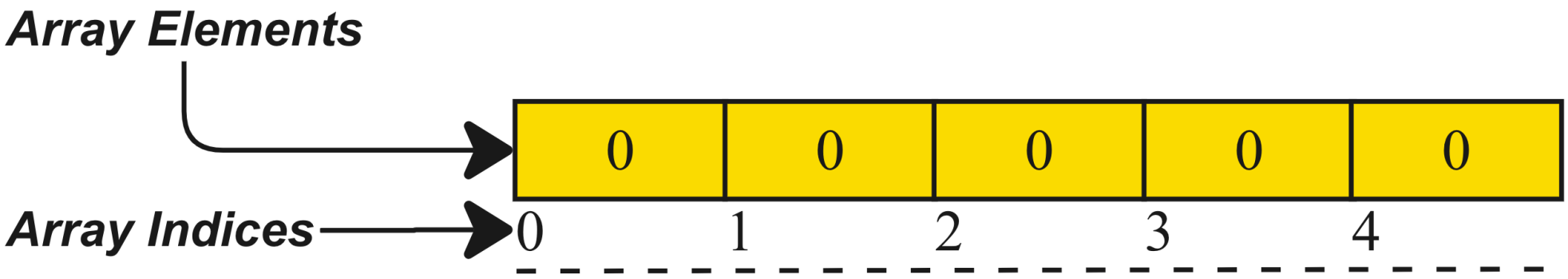
Array vs Slice

Array	Slice
[size]int	[]int
fixed size	variable size
allocated can't change	dynamically resized
value type	reference type

Array

```
var name [n]T
```

```
var array [5]int
```



Array: assign/access

```go
var array [5]int

array[0] = 1
array[1] = 2

elem := array[4]
```

How to loop through an Array

```
fibonacci := [10]int{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}

for i:= 0; i < 10; i ++ {
    fmt.Println(fibonacci[i])
}
```

using len to get the length

```
fibonacci := [10]int{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}

for i:= 0; i < len(fibonacci); i ++ {
    fmt.Println(fibonacci[i])
}
```

range: to iterate over item

```
fibonacci := [10]int{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}

for i, v := range fibonacci {
    fmt.Printf("index: %d, value %d\n", i, v)
}
```

range: _ to ignore not used value

```go
fibonacci := [10]int{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}

for i, _ := range fibonacci {
    fmt.Printf("only index is %d\n", i)
}
```

```go
for i := range fibonacci {
    fmt.Printf("only index is %d\n", i)
}
```

```go
for _, v := range fibonacci {
    fmt.Printf("only value is %d\n", v)
}
```

Exercise: Array

Reverse Elements

```
list := [4]int{1, 3, 4, 2}
r := reverse(list)

fmt.Printf("%T %v\n", r, r)
```

> [4]int [2 4 3 1]

Array auto counting

```
fibonacci := [...]int{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}

fmt.Printf("type: %T\n", fibonacci)
```

> type: [10]int

Slice

An array has a fixed size.

A slice, on the other hand

Slice literals

var name []T

```
var array = [3]int{-1, 0, 1}
var slice = []int{-1, 0, 1}
```

more flexible than array

```
fibonacci := []int{0, 1, 1, 2, 3, 5}

for i, v := range fibonacci {
    fmt.Println(fibonacci[i])
}
```

append

> func append(s []T, vs ...T) []T

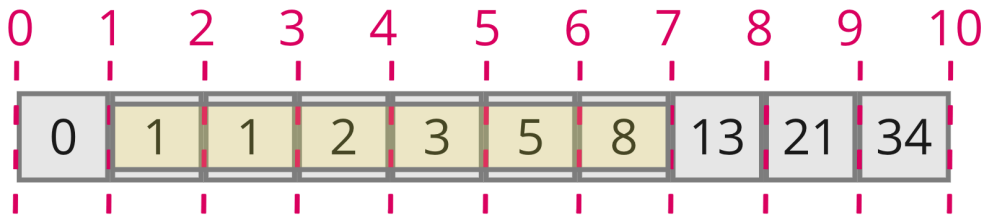
```
fibonacci := []int{0}
fibonacci = append(fibonacci, 1)
```

> fibonacci = []int{0, 1}

slice

> a[low : high]

```
fibonacci := []int{0, 1, 1, 2, 3, 5, 8, 13, 21, 34}
fibonacci = fibonacci[1:7]
```

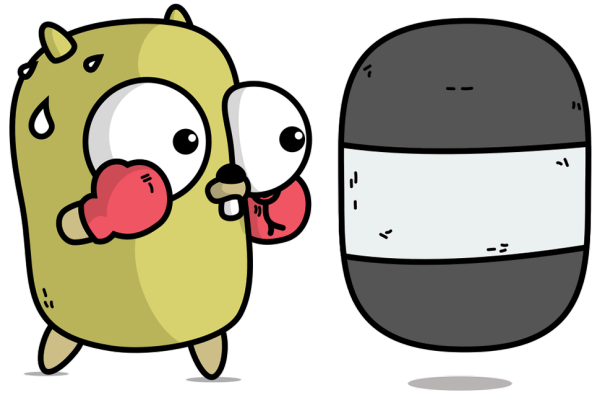


> fibonacci = []int{1, 1, 2, 3, 5, 8}

Test Slice

```
a := [...]int{-1, 0, 1, 2, 3, 4, 5, 6}
s := a[0:5]
```

```
[]int{-1, 0, 1, 2, 3}
```



ignore the low bound if zero

```
a := [...]int{-1, 0, 1, 2, 3, 4, 5, 6}
s := a[:5]
```

```
[]int{-1, 0, 1, 2, 3}
```

ignore the high bound for the last one

```
a := [...]int{-1, 0, 1, 2, 3, 4, 5, 6}
s := a[2:]
```

```
[]int{1, 2, 3, 4, 5, 6}
```

Zero value of slice is `nil`

```go
var s []string // nil
```

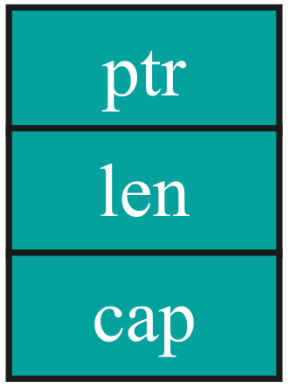
```go
var s []string
s[0] = "The"
```

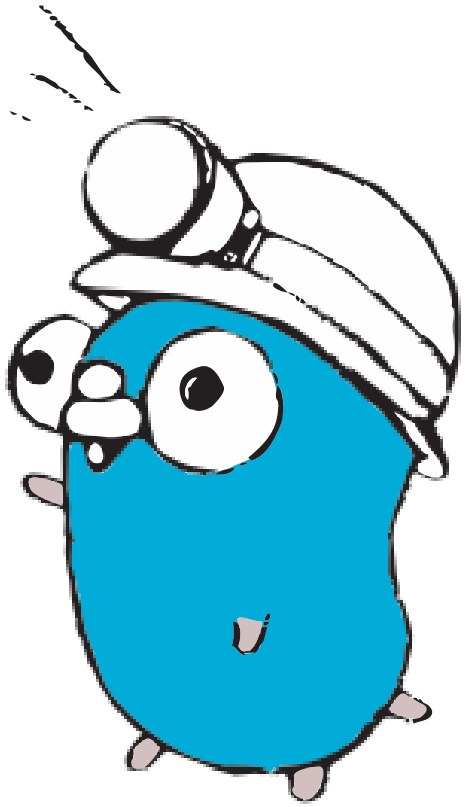
make slice

> make([]T, length, capacity)

```
var s []string
s = make([]string, 1)
s[0] = "The"
```

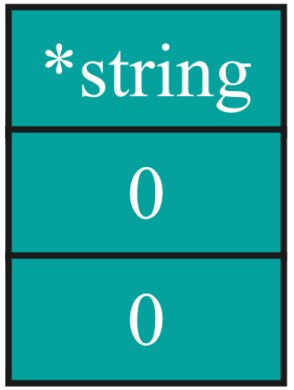
Structure of Slice





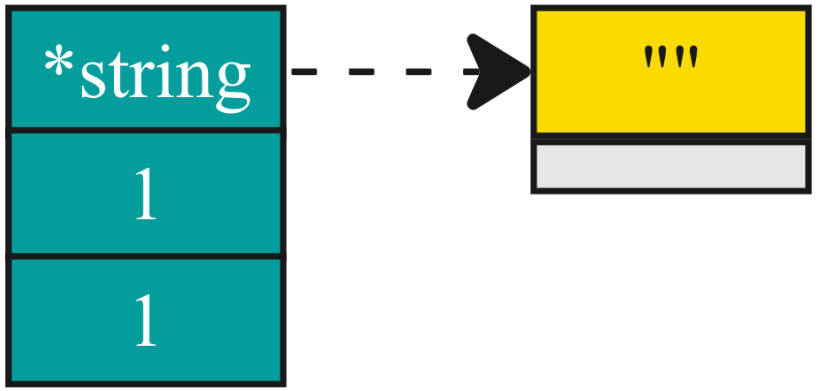
Make 0 length of slice

```
s := make([]string, 0)
```



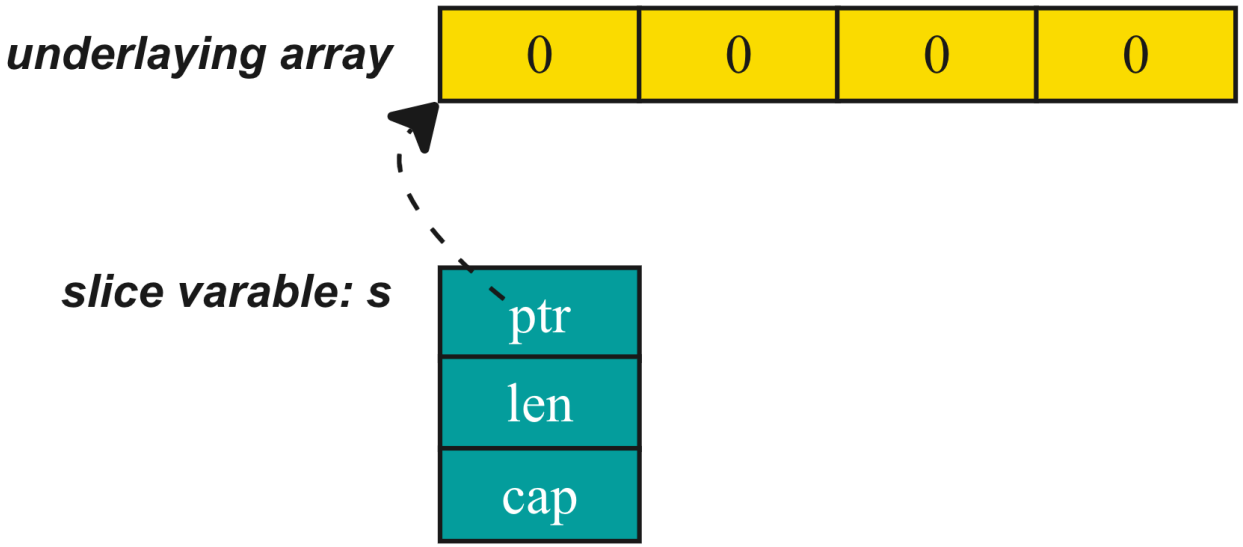
make 1 length

```
s := make([]string, 1)
```



underlaying of slice is an array

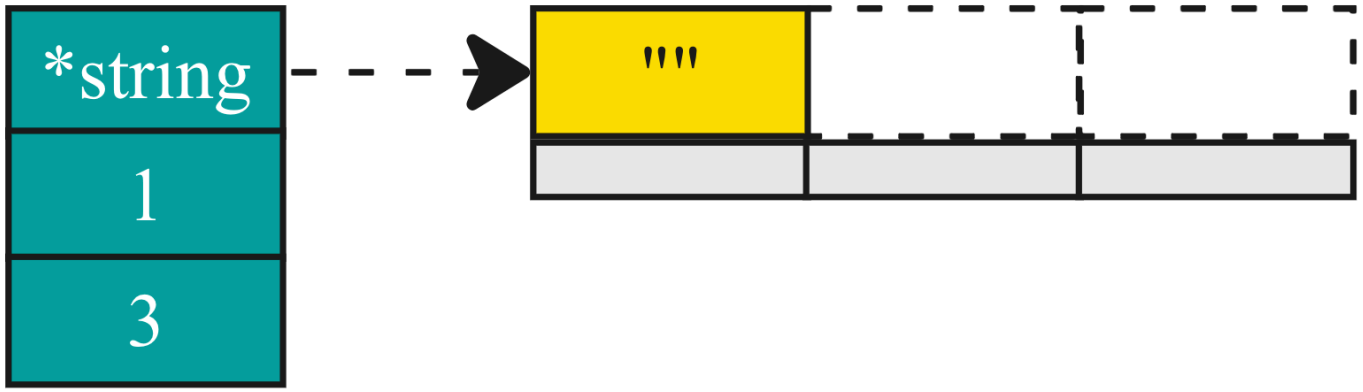
```
s := make([]int, 4)
```



> len = 4
>
> cap = 4

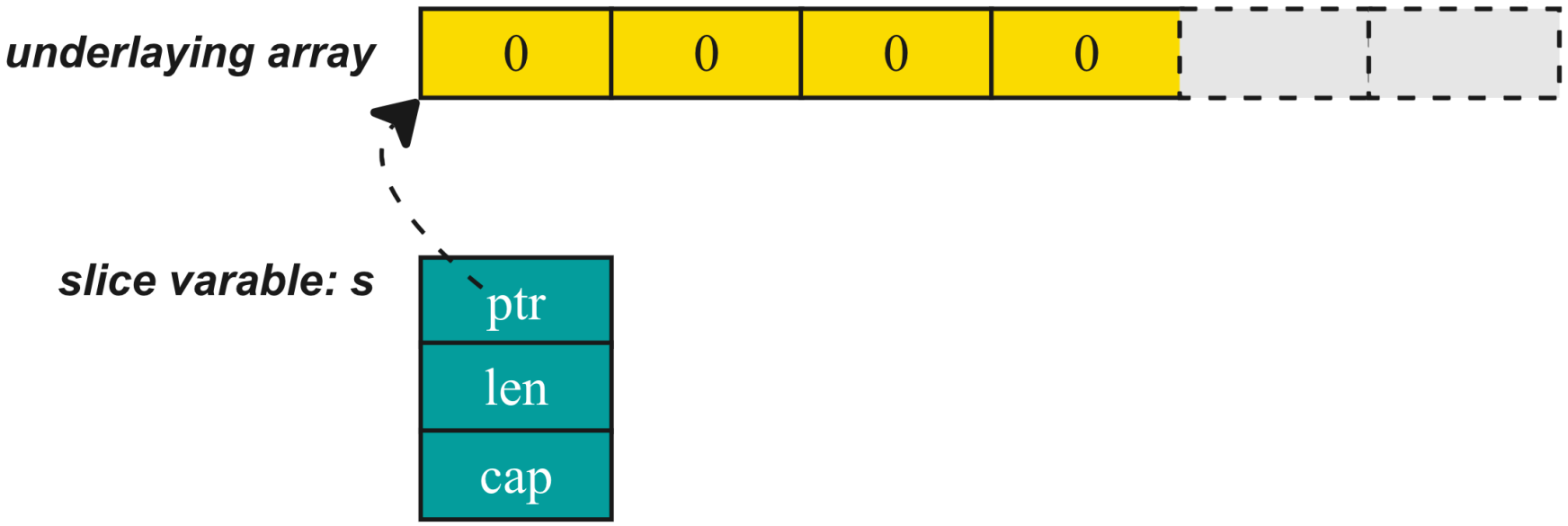
make 1 length and 3 cap

```
s := make([]string, 1, 3)
```



Make Slice with cap

```
s := make([]int, 4, 6)
```



> len = 4
>
> cap = 6

assign a value to slice

```go
var i = []int{-2, -1, 0, 1, 2}
```

```go
var i []int
i = []int{-2, -1, 0, 1, 2}
```

assign an array to slice

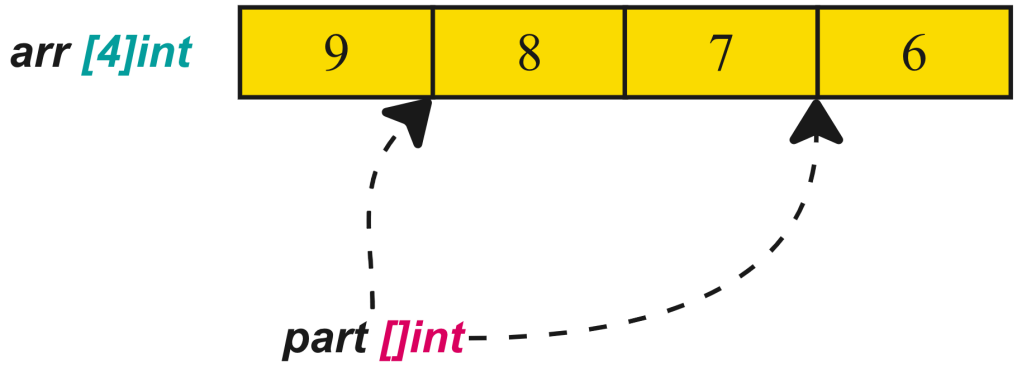
```
var i []int
var arr = [...]int{-4, -3, -2, -1, 0, 1, 2, 3, 4}
i = arr[0:9]
```

Slice: Two-Index Slices

slice point to an array

```
var arr := [4]int{9, 8, 7, 6}
part := arr[1:3]
```

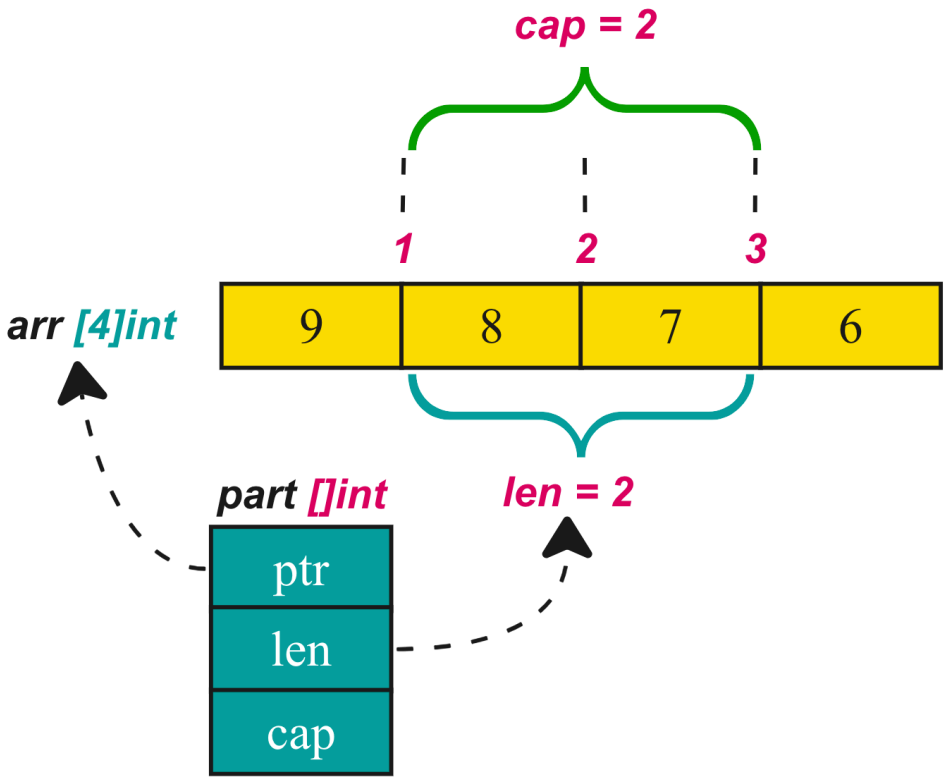
> part = []int{8, 7}



Slice: Three-Index Slices

slice point to an array

```
var arr := [4]int{9, 8, 7, 6}
part := arr[1:3:3]
```



Test about slice

```go
arr := [...]int{111, 101, 110, 123, 321}
s := arr[:]

fmt.Printf("type: %T,  val: %v\n", s ,s)
fmt.Printf("cap: %d, len: %d\n", cap(s), len(s))
```

```
type: []int, val: [111, 101, 110, 123, 321]
```

```
cap: 5, len: 5
```

Test about slice

```go
arr := [...]int{997, 111, 101, 110, 123, 321}
s := arr[1:3:4]

fmt.Printf("type: %T,  val: %v\n", s ,s)
fmt.Printf("cap: %d, len: %d\n", cap(s), len(s))
```

```
type: []int, val: [111, 101]

cap: 3, len: 2
```

Test about slice

```go
arr := []int{997, 111, 101, 110, 123, 321}
s := arr[2:]

fmt.Printf("type: %T,  val: %v\n", s ,s)
fmt.Printf("cap: %d, len: %d\n", cap(s), len(s))
```

```
type: []int, val: [101, 110, 123, 321]
```

```
cap: 4, len: 4
```

Test about slice

```go
arr := []int{997, 111, 101, 110, 123, 321}
s := arr[:2]

fmt.Printf("type: %T,  val: %v\n", s ,s)
fmt.Printf("cap: %d, len: %d\n", cap(s), len(s))
```

```
type: []int, val: [997, 111]
```

```
cap: 6, len: 2
```

Slice just refer to an array

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]

fmt.Printf("value: %v\n", p1)
fmt.Printf("len(p1): %d, cap(p1): %d\n",len(p1), cap(p1))

p1[0] = 3
fmt.Printf("arr value: %v\n", arr)
```

```
value: ?
len(p1): ?, cap(p1): ?
arr value: ?
```

Slice just refer to an array

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]

fmt.Printf("value: %v\n", p1)
fmt.Printf("len(p1): %d, cap(p1): %d\n",len(p1), cap(p1))

p1[0] = 3
fmt.Printf("arr value: %v\n", arr)
```

```
value: [0]
len(p1): 1, cap(p1): 2
arr value: [-1 3 1]
```

Slice just refer to an array(2)

append not returns a new array if not over cap

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]

fmt.Printf("p1 value: %v\n", p1)
fmt.Printf("p1 len(p1): %d, cap(p1): %d\n", len(p1), cap(p1))

p1 = append(p1, 2)
fmt.Printf("arr value: %v\n", arr)
```

```
value: ?
len(p1): ?, cap(p1): ?
arr value: ?
```

Slice just refer to an array(2)

append not returns a new array if not over cap

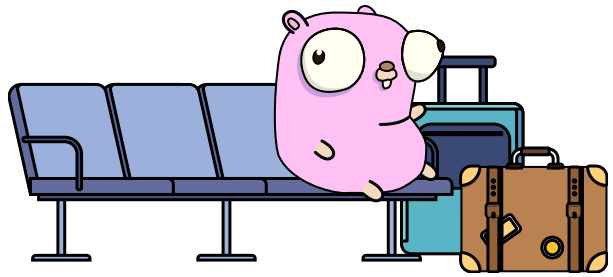
```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]

fmt.Printf("p1 value: %v\n", p1)
fmt.Printf("p1 len(p1): %d, cap(p1): %d\n", len(p1), cap(p1))

p1 = append(p1, 2)
fmt.Printf("arr value: %v\n", arr)
```

```
value: 0
len(p1): 1, cap(p1): 2
arr value: [-1 0 2]
```

Slice just refer to an array(3)

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]
p2 := arr[:]

p1 = append(p1, 8)

fmt.Printf("arr: %v\n", arr)
fmt.Printf("p1: %v\n", p1)
fmt.Printf("p2: %v\n", p2)
```

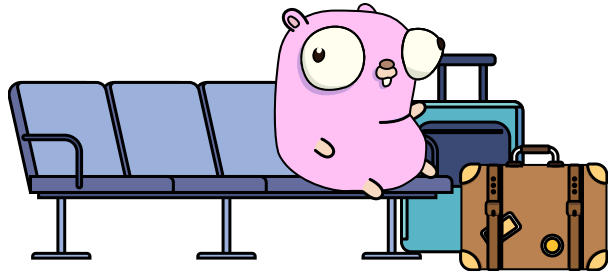
```
arr: ?
p1: ?
p2: ?
```



Slice just refer to an array(3)

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]
p2 := arr[:]

p1 = append(p1, 8)

fmt.Printf("arr: %v\n", arr)
fmt.Printf("p1: %v\n", p1)
fmt.Printf("p2: %v\n", p2)
```

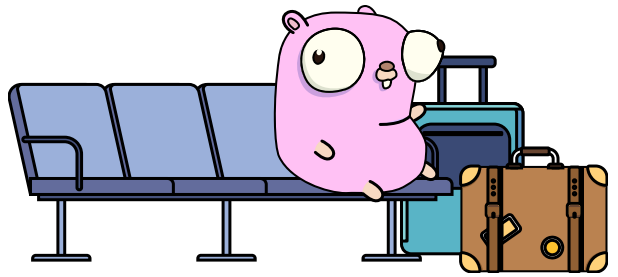
```
arr: [-1 0 8]
p1: [0 8]
p2: [-1 0 8]
```



Slice just refer to an array(4)

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]
p2 := arr[:]

p1 = append(p1, 8, 9)

fmt.Printf("arr: %v\n", arr)
fmt.Printf("p1: %v\n", p1)
fmt.Printf("p2: %v\n", p2)
```

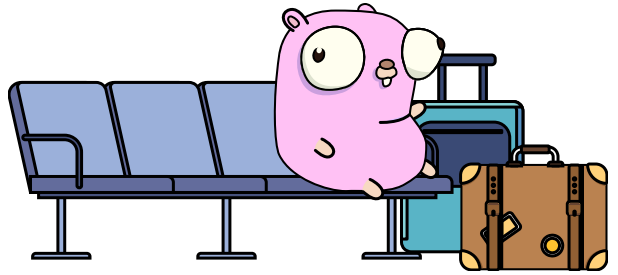
```
arr: ?
p1: ?
p2: ?
```



Slice just refer to an array(4)

```
arr := [3]int{-1, 0, 1}
p1 := arr[1:2]
p2 := arr[:]

p1 = append(p1, 8, 9)

fmt.Printf("arr: %v\n", arr)
fmt.Printf("p1: %v\n", p1)
fmt.Printf("p2: %v\n", p2)
```

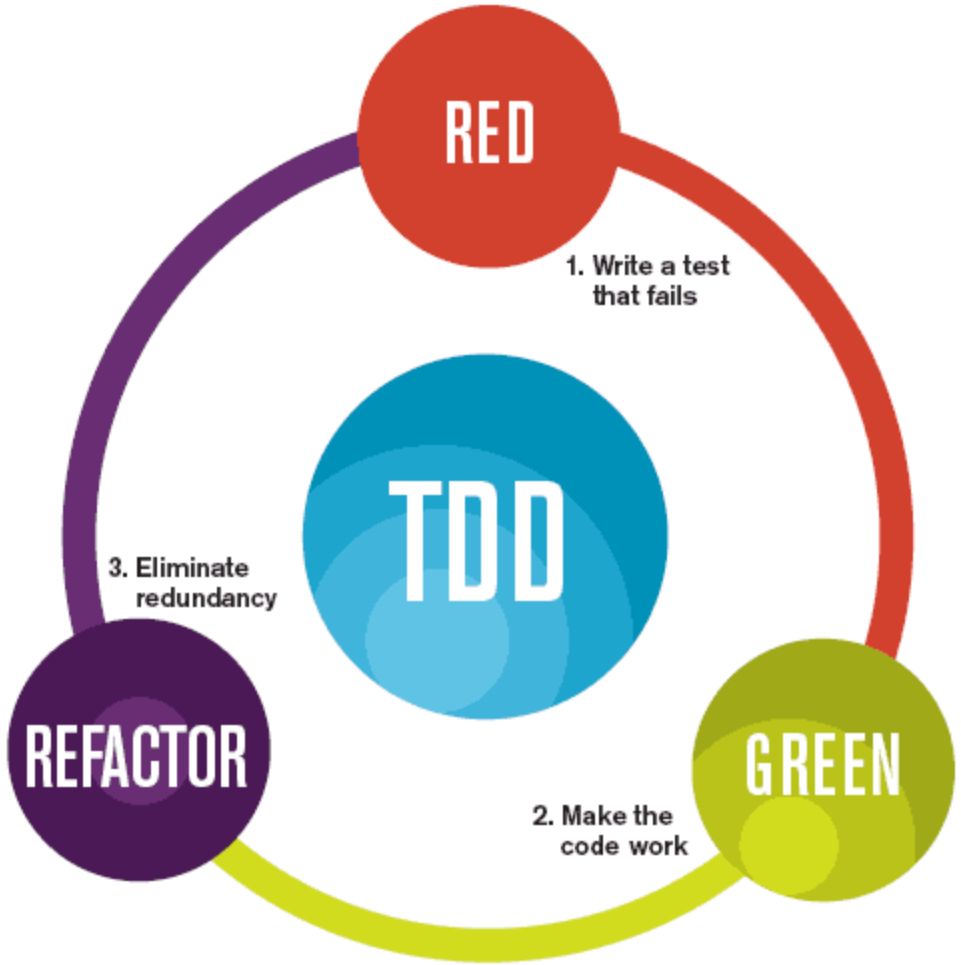
```
arr: [-1 0 1]
p1: [0 8 9]
p2: [-1 0 1]
```



Exercise: filter only prime number

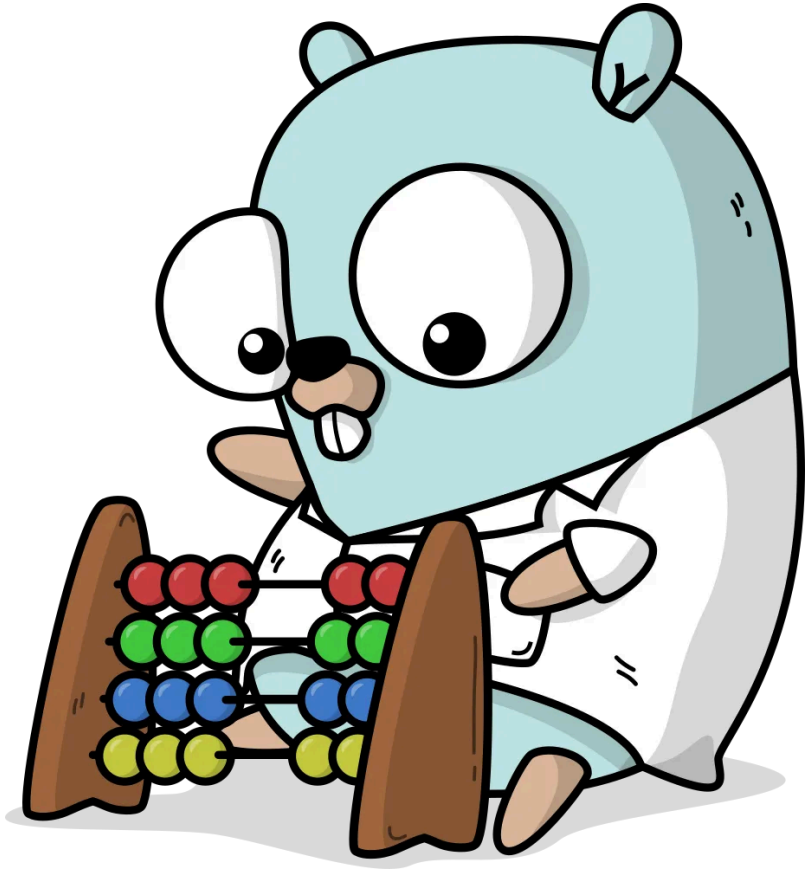
```go
var numbers = []int{4, 5, 6, 7, 11, 21, 37}

func primes(numbers []int) []int {

}
```

TDD



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Unit testing in go

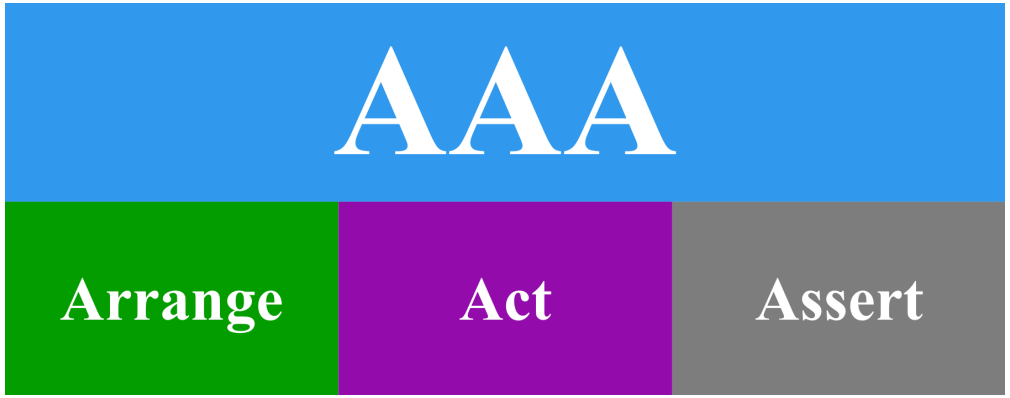


3 Conditions

1. filename has suffix **_test.go** such as **foobar_test.go**
2. function name prefix is **Test**
3. the test function only get 1 parametter type ***testing.T**

```go
import "testing"

func TestACase(t *testing.T) {

}

func Test_a_case(t *testing.T) {

}
```

Unit testing



AAA

```go
// Arrange
given := 1
want := "1"

// Act
get := foobar(given)

// Assert
if want != get {
    // error report
}
```

Rule

1. เขียนเทสก่อนเขียนโค้ด
2. เทสพัง เพราะยังไม่เขียน feature ตามเทสเท่านั้น
3. ห้ามลบเทสที่ตัวเองไม่ได้เขียน (บางทีก็ต้องถามตัวเองก่อน)
4. เขียนเทส ทีละ 1 เคส
5. Refactor เทสได้ ไม่บาป
6. commit ทุก รอบ

Greeting #1

```
fmt.Println(greet("Bob"))
```

"Hello, Bob."

Greeting #2

ถ้าไม่ส่งชื่อมาให้ใช้คำแทนเช่น my friend

```
fmt.Println(greet())
```

"Hello, my friend."

Greeting #3

ถ้าชื่อเป็น Upper case ทั้งหมด ให้ตะโกนกลับ

```
fmt.Println(greet("JERRY"))
```

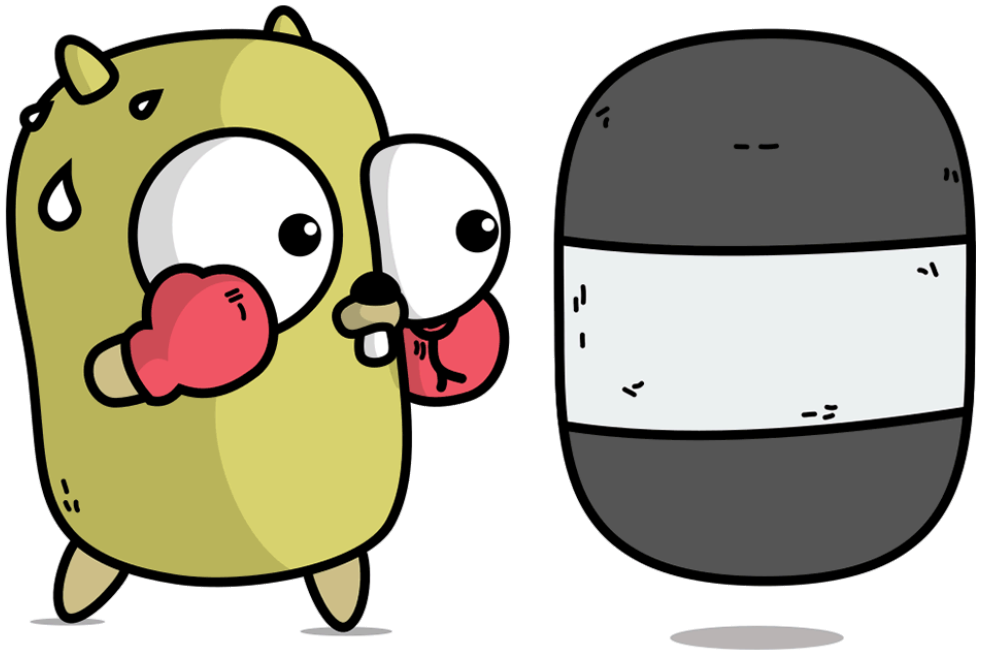
"HELLO, JERRY!"

Greeting #4

สวัสดีหลายๆคนได้

```
fmt.Println(greet("Amy", "Brian", "Charlotte"))
```

"Hello, Amy, Brian and Charlotte."



TDD Kata: String calulator: continue

```go
func AddString(numbers string) int
```

```go
AddString("") // 0
```

```go
AddString("1") // 1
```

```go
AddString("1,2") // 3
```

```go
AddString("1,2\n3") // 6
```

AddStringNumber CLI

```
$ add 1 2
```

Kata: Password input field validation, 8 characters

The password must be at least 8 characters long. If it is not met, then the following error message should be returned: “Password must be at least 8 characters”

```go
func ValidatePassword(password string) error
```

```go
ValidatePassword("1234567") // error: the password must be at least 8 characters
```

Kata: Password input field validation, 2 numbers

The password must contain at least 2 numbers. If it is not met, then the following error message should be returned: “The password must contain at least 2 numbers”

```
ValidatePassword("abcdefgh") // error: the password must contain at least 2 numbers
```

Kata: Password input field validation, at least one capital letter

The password must contain at least one capital letter. If it is not met, then the following error message should be returned: “password must contain at least one capital letter”

```
ValidatePassword("abcdef12")
// error: “the password must contain at least one capital letter”
```

Kata: Password input field validation, at least one special character

The password must contain at least one special character. If it is not met, then the following error message should be returned: “password must contain at least one special character”

```
ValidatePassword("abcdeF12")
// error: “the password must contain at least one special character”
```

Exercise: validate password CLI

```
$ validate 12345678
```

Thai ID Validation

1-2345-67890-12-1

Thai ID Validation 2

1-2345-67890-12-1

นำเลขมาคูณเลขประจำตำแหน่ง

13	12	11	10	9	8	7	6	5	4	3	2	1

X

1	2	3	4	5	6	7	8	9	0	1	2	1

=

13	24	33	40	45	48	49	48	45	0	3	4	_

Thai ID Validation 3

ผลรวมของหลักที่ 2 - 13

13	24	33	40	45	48	49	48	45	0	3	4	_

```
13+24+33+40+45+48+49+48+45+0+3+4 = 352
```

mod *352* ด้วย 11

352 *mod* 11 = **0**

Thai ID Validation 3

นำ 11 มาตั้งแล้วลบผลลัพธ์ที่ได้

```
11 - 0
```

ผลลัพธ์ = 11

ให้ใช้เลขหลักหน่วยมาเป็น check digit คือเลข `1`

1	2	3	4	5	6	7	8	9	0	1	2	1
												

ID Validate function

```
func Validate(id string) error
```

```
input: "1234567890121"
input: "1-2345-67890-12-1"
```

Thai ID Validation conclusion

13	12	11	10	9	8	7	6	5	4	3	2	1
x												
1	2	3	4	5	6	7	8	9	0	1	2	**1**
=												
13	24	33	40	45	48	49	48	45	0	3	4	_

```
13  + 24  + 33  +  40 +  45 +  48 +  49 +  48 +  45 +  0  + 3  + 4  = 352
352 mod 11 = 0
11 - 0 = 11
```

ID Validate test cases

```
Validate("1234567890121") // nil
Validate("123456789012") // error: id digits incorrect
Validate("1234567890122") // error: id incorrect
```

Variable Scoping

Using reference to loop iterator variable

```
func main() {
    var out []*int
    for i := 0; i < 3; i++ {
        out = append(out, &i)
    }
    fmt.Println("Values:", *out[0], *out[1], *out[2])
    fmt.Println("Addresses:", out[0], out[1], out[2])
}
```

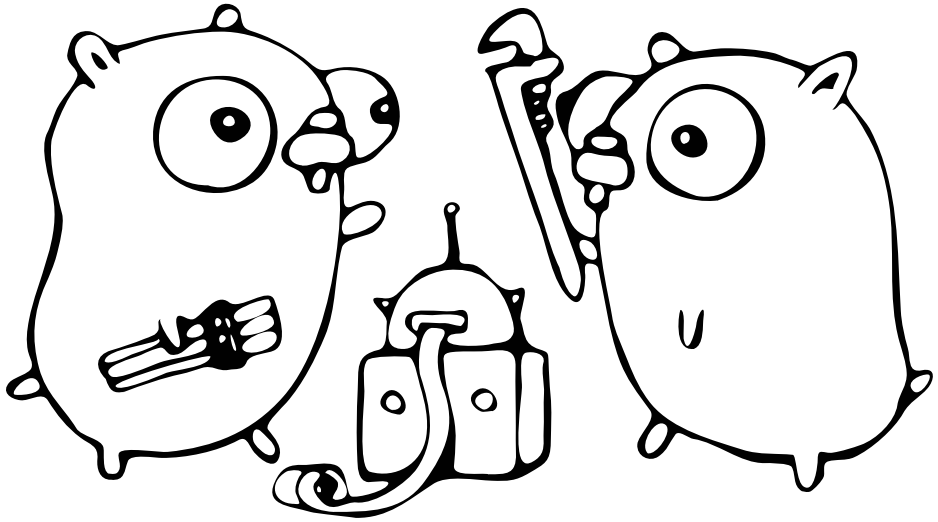
Using slice to loop iterator variable

```
func main() {
    var out [][]int
    for _, i := range [][1]int{{1}, {2}, {3}} {
        out = append(out, i[:])
    }
    fmt.Println("Values:", out)
}
```

Using goroutines on loop iterator variables

```
for _, val := range values {
    go func() {
        fmt.Println(val)
    }()
}
```

API



Web Framework Benchmark

https://www.techempower.com/benchmarks/#section=data-r19&hw=ph&test=plaintext

https://github.com/smallnest/go-web-framework-benchmark

GoFiber

https://docs.gofiber.io/

https://github.com/gofiber/fiber

GoFiber: Example

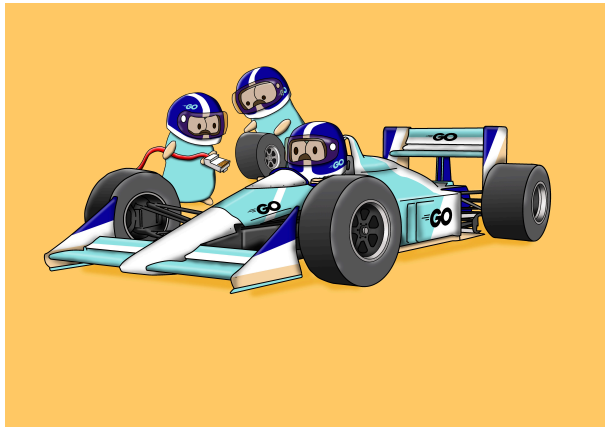
```go
package main

import "github.com/gofiber/fiber/v2"

func main() {
    app := fiber.New()

    app.Get("/", func(c *fiber.Ctx) error {
        return c.SendString("Hello, World 👋!")
    })

    app.Listen(":3000")
}
```

GoFiber.io

★ Some values returned from *fiber.Ctx are not immutable by default

Issue

https://github.com/gofiber/fiber/issues/426



Testing

https://github.com/pallat/fibertest

Go Tutorial

https://go.dev/doc/tutorial/

https://go.dev/doc/tutorial/web-service-gin

Gin-Gonic

https://github.com/gin-gonic/gin

https://gin-gonic.com/

Gin Handler

```go
package main

import (
  "net/http"

  "github.com/gin-gonic/gin"
)

func main() {
  r := gin.Default()
  r.GET("/ping", func(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
      "message": "pong",
    })
  })
  r.Run() // listen and serve on 0.0.0.0:8080 (for windows "localhost:8080")
}
```

anonymous function

```go
func() {
    fmt.Println("my function")
}()
```

```go
func(name string) {
    fmt.Println("my function name:", name)
}("anonymous")
```

First Class Function

```
var fn = func(relativePath string) {
    fmt.Printf("relativePath is %s\n", relativePath)
}
```

Higher Order Function

```
r.GET("/ping", func(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
      "message": "pong",
    })
  })
```

```
func GET(relativePath string, handlers ...func(c *Context))
```

Method

```go
type router struct{}

func (r rounter) GET(relativePath string, handlers ...func(c *Context))
```

Method belongs to a custom type

```
type integer int

func (i integer) Add(a integer) {
    i = i + a
}

var i, j integer = 1, 2
i.Add(j)
fmt.Println(i)
```

```
Is it works?
```

Method is just a function

```go
type integer int

func (i *integer) Add(a integer) {
    i = i + a
}
```

```go
func Add(i *integer, a integer) {

}
```

structure: struct{}

struct{} is a type just like **int** or **string**

```
type myStruct struct {}

type creature struct {
    Name string
}
```

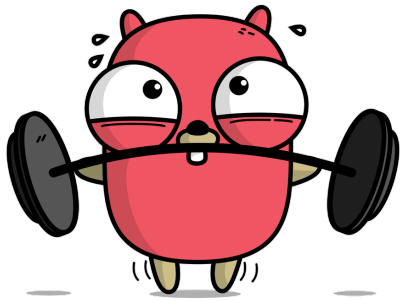
Method: continue

```
type router struct{}

func (r rounter) GET(relativePath string, handlers ...func(c *Context))
```

```
func GET(r rounter,relativePath string, handlers ...func(c *Context))
```

move anonymous handler to function

```go
package main

import (
  "net/http"

  "github.com/gin-gonic/gin"
)

func main() {
  r := gin.Default()
  r.GET("/ping", func(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
      "message": "pong",
    })
  })
  r.Run() // listen and serve on 0.0.0.0:8080 (for windows "localhost:8080")
}
```



Thai ID Validator API

POST /thai/ids/verify

```
{
    "id": "1-2345-67890-12-1"
}
```

200

```
{
    "valid": true
}
```

Thai ID Validator API: invalid

POST /thai/ids/verify

```
{
    "id": "1-2345-67890-12-1"
}
```

???

```
{
    "valid": false
}
```

http status

make a decision together

```
200 : normal workflow
500 : show error message
```

```
200 : normal workflow
200 : normal error workflow
500 : error and break the normal workflow
```

Configurations

https://12factor.net/

direnv

https://direnv.net/

> brew install direnv

> vi ~/.config/direnv/direnv.toml

```
[global]
load_dotenv=true
```

Add .env to the project

```
PORT=8081
```

direnv: activate values

direnv: error ~/workspace/project/.env is blocked. Run `direnv allow` to approve its content

direnv automatic allow

```
[global]
load_dotenv=true
[whitelist]
prefix=["~/workspace"]
```

Dealing with complex config

`goexamples/configurations`

```
import "github.com/caarlos0/env/v10"
```

Dealing with file configuration

https://github.com/spf13/viper

Ask for clarity on what they want

> "If the things are too complex, don't ask for the solution but the reason."

Connect to DB in Go

```
import (
        "database/sql"

        _ "github.com/mattn/go-sqlite3"
)
```

the Blank identifier

(underscore)

import with need the side effects

_ "github.com/mattn/go-sqlite3"

windows

_ modernc.org/sqlite

Open DB

```
db, err := sql.Open("sqlite3", "./foo.db")
if err != nil {
    log.Fatal(err)
}
defer db.Close()
```

defer

A defer statement defers the execution of a function until the surrounding function returns.

The deferred call's arguments are evaluated immediately, but the function call is not executed until the surrounding function returns.

```
res, _ := http.DefaultClient.Do(req)

defer res.Body.Close()
body, _ := ioutil.ReadAll(res.Body)
```

a defer

```
defer fmt.Println("end")

fmt.Println("Hello, Gophers")
```

defer is stack

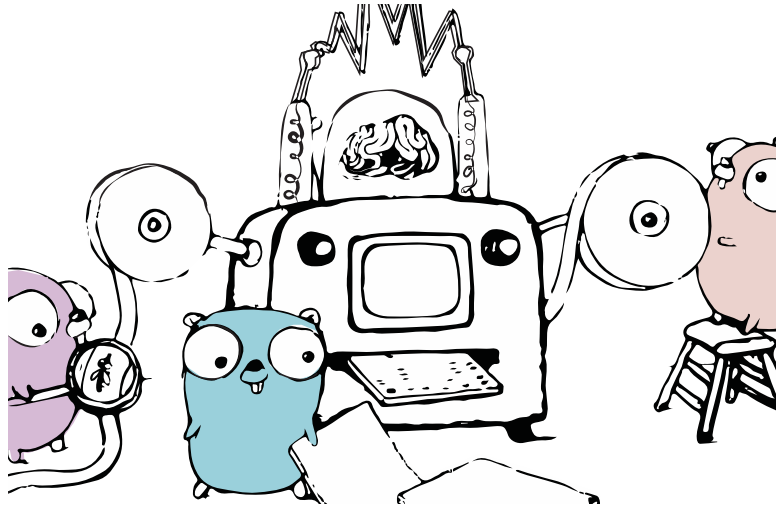
```
func greeting(name string) {
        defer fmt.Println()
        defer fmt.Print(".")
        fmt.Print("Hello")
        fmt.Print(" ")
        fmt.Print(name)
}

greeting("John")
```

Hello John.

Questions? What is the result

```go
func greeting(name string) {
        greet := "Hello"
        defer fmt.Println(greet)
        greet += " " + name + "."
}

func main() {
    greeting("John")
}
```

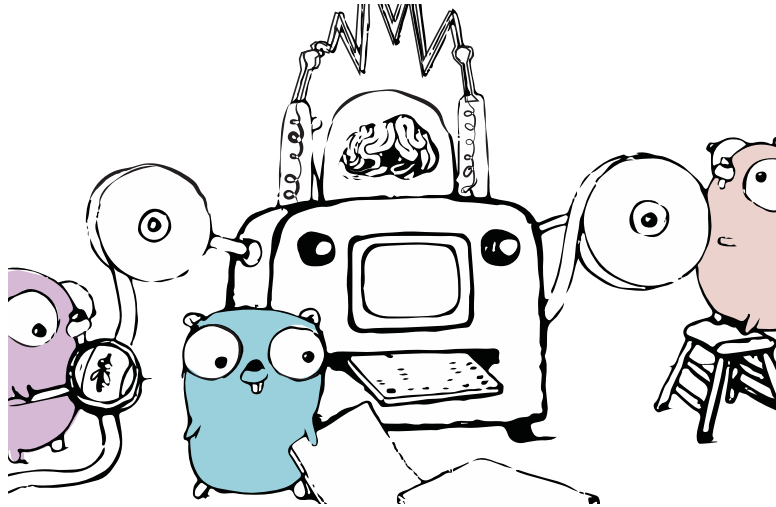
```
Hello
```



Questions? One more

```
func greeting(name string) {
        greet := "Hello"
        defer func() {
                fmt.Println(greet)
        }()
        greet += " " + name + "."
}

func main() {
    greeting("John")
}
```

```
Hello John.
```

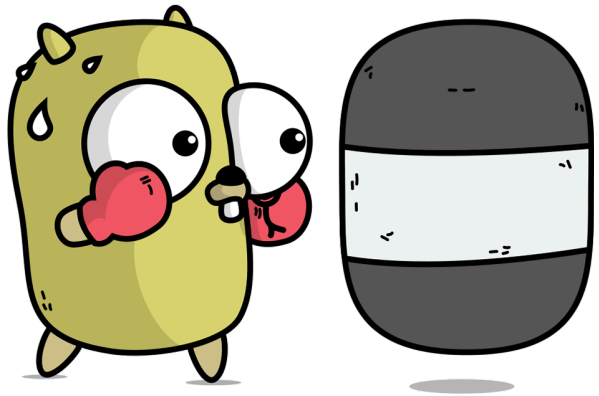


defer for recovering

```go
func catchMe() {
    defer func() {
        if r := recover(); r != nil {
            fmt.Println(r)
        }
    }()

    s := []int{}

    fmt.Println(s[1])
}
```

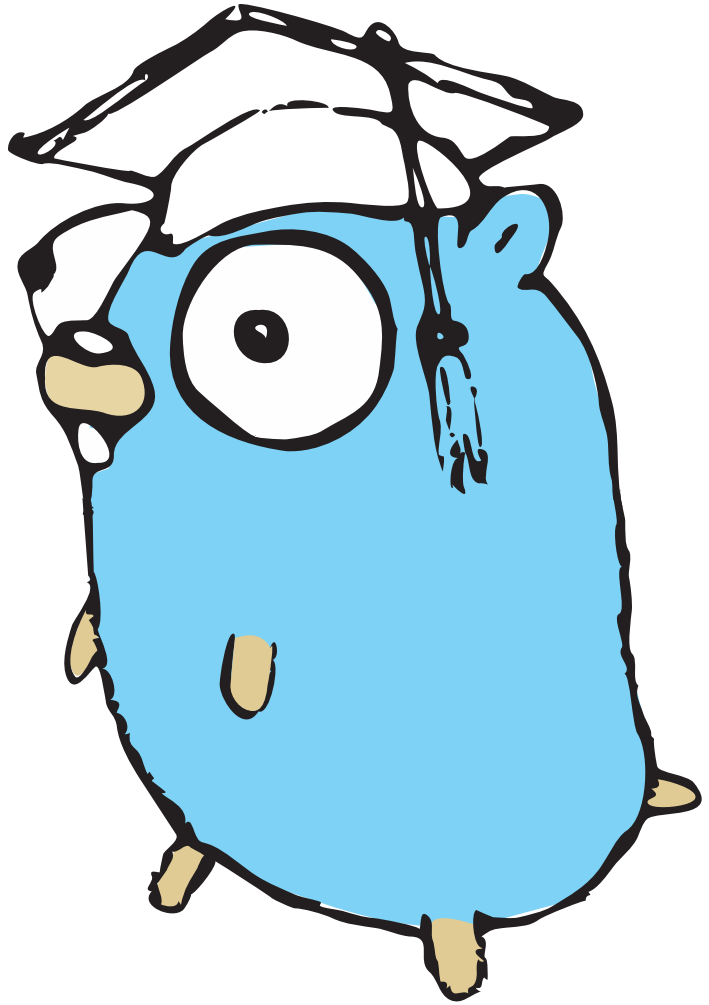
Play with defer

```go
func main() {
        var fn = func() error { return errors.New("fail") }
        b, _ := do(fn, "")
        fmt.Println(b.String())
}

func do(fn func() error, s string) (fmt.Stringer, error) {
        if err := fn(); err != nil {
                return nil, err
        }

        return stringer(s), nil
}

type stringer string

func (s stringer) String() string {
        return string(s)
}
```



ทำ recovery ไม่ให้ app crash

Cross Functional Requirement

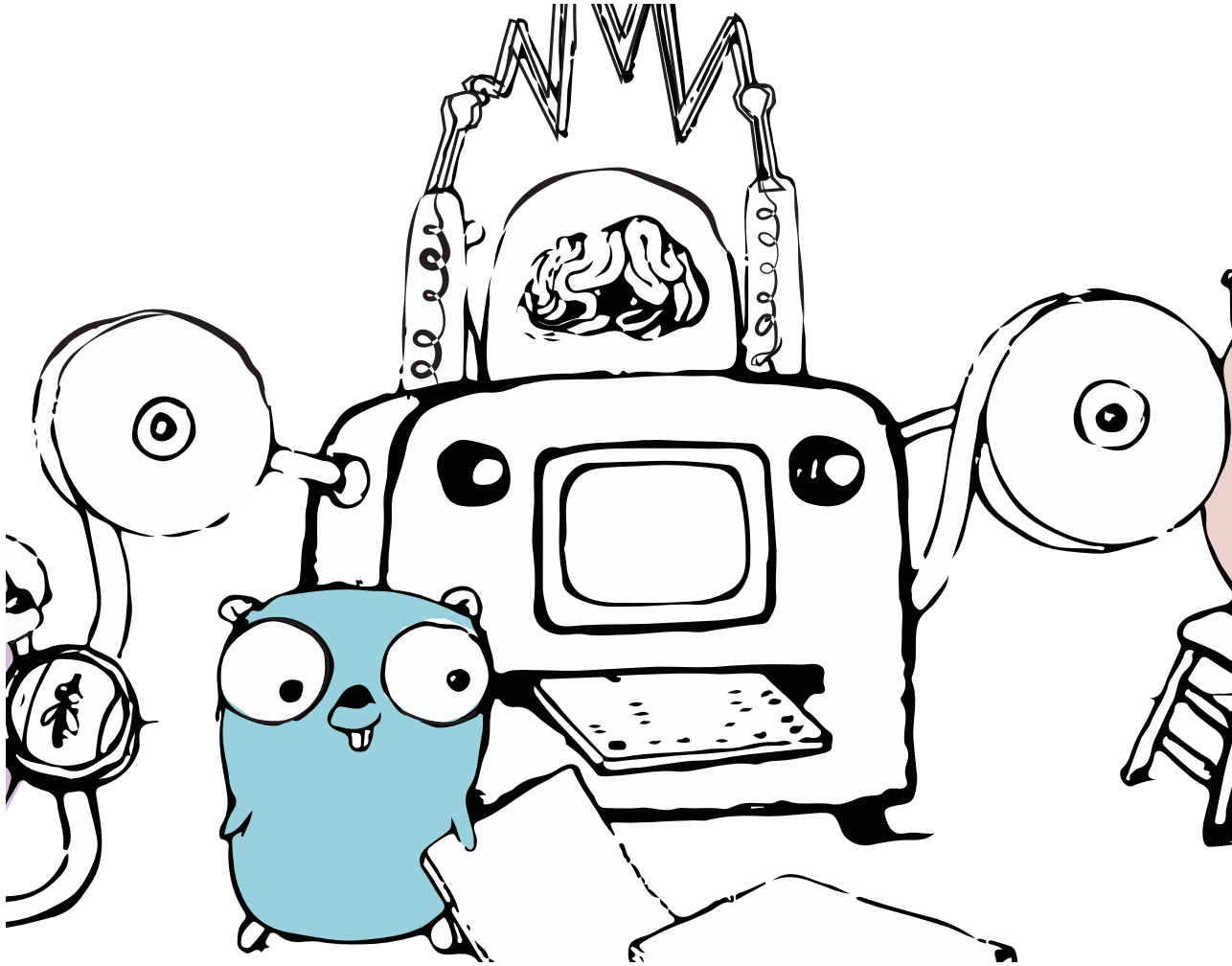


Simple API

```go
package main

import (
        "io"
        "log"
        "net/http"
)

func main() {
        // Hello world, the web server

        helloHandler := func(w http.ResponseWriter, req *http.Request) {
                io.WriteString(w, "Hello, world!\n")
        }

        http.HandleFunc("/hello", helloHandler)
        log.Fatal(http.ListenAndServe(":8080", nil))
}
```

Checklist

- [ ] env configuration
- [ ] graceful shutdown
- [ ] golangci-lint
- [ ] Dockerfile
- [ ] git-hook
- [ ] logging
- [ ] gitlab-ci
- [ ] testable



Configurations

> .env

```
PORT=8910
```

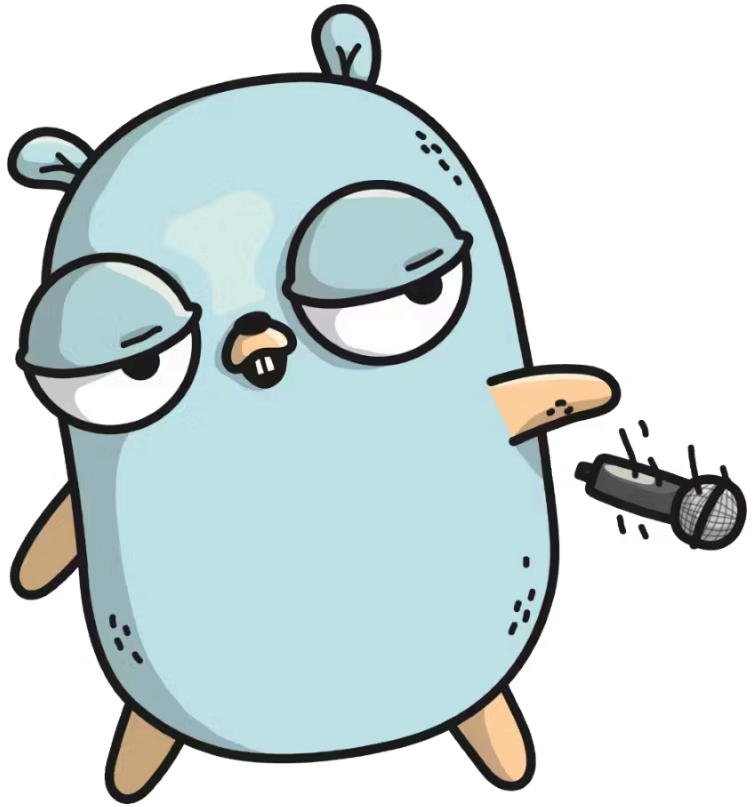
> main.go

```
os.Getenv("PORT")
```

checklist: config

- ✅ env configuration
- ☐ graceful shutdown
- ☐ golangci-lint
- ☐ Dockerfile
- ☐ gitlab-ci
- ☐ testable
- ☐ git-hook

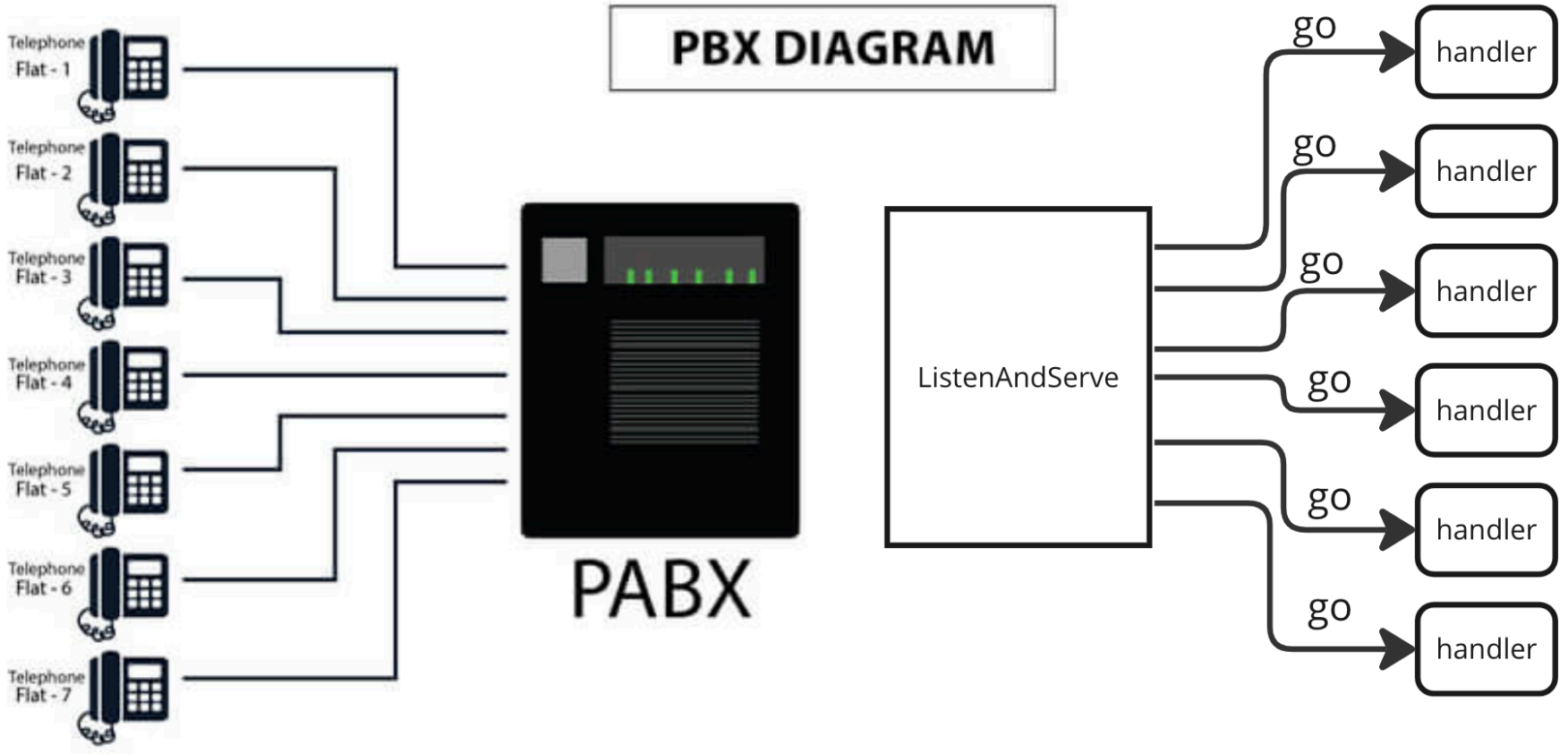
Gracefully shutting down



Call Center

☎ 1234

ListenAndServe



The Kubernetes termination lifecycle

https://cloud.google.com/blog/products/containers-kubernetes/kubernetes-best-practices-terminating-with-grace

> In practice, this means your application needs to handle the SIGTERM message and begin shutting down when it receives it. This means saving all data that needs to be saved, closing down network connections, finishing any work that is left, and other similar tasks.

graceful example

```
ctx, stop := signal.NotifyContext(context.Background(), syscall.SIGINT, syscall.SIGTERM)
defer stop()

idleConnsClosed := make(chan struct{})

mux := http.NewServeMux()

srv := http.Server{
    Addr:              ":" + os.Getenv("PORT"),
    Handler:           mux,
    ReadHeaderTimeout: 5 * time.Second,
}

go func() {
    <-ctx.Done()
    fmt.Println("shutting down...")
    if err := srv.Shutdown(context.Background()); err != nil {
        log.Printf("HTTP server Shutdown: %v", err)
    }
    close(idleConnsClosed)
}()

if err := srv.ListenAndServe(); err != http.ErrServerClosed {
    log.Fatalf("HTTP server ListenAndServe: %v", err)
}

<-idleConnsClosed
fmt.Println("bye")
```

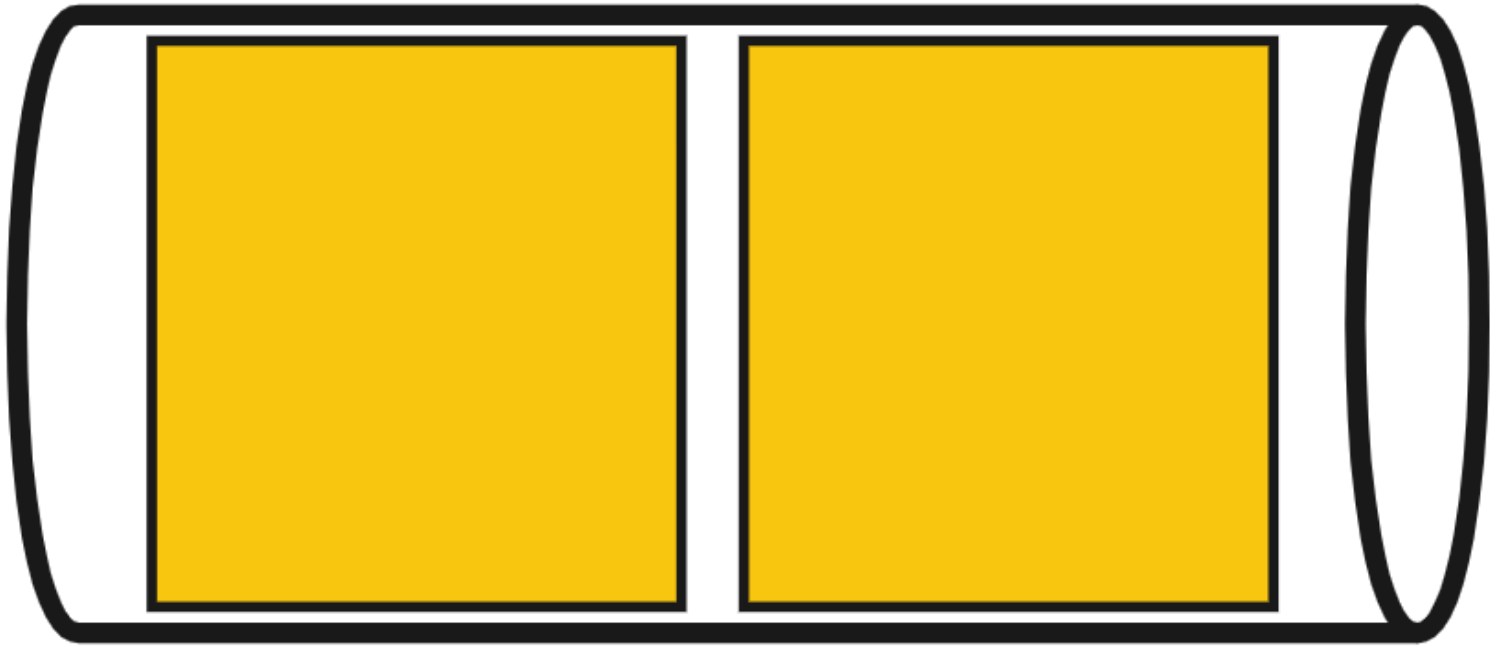
channel

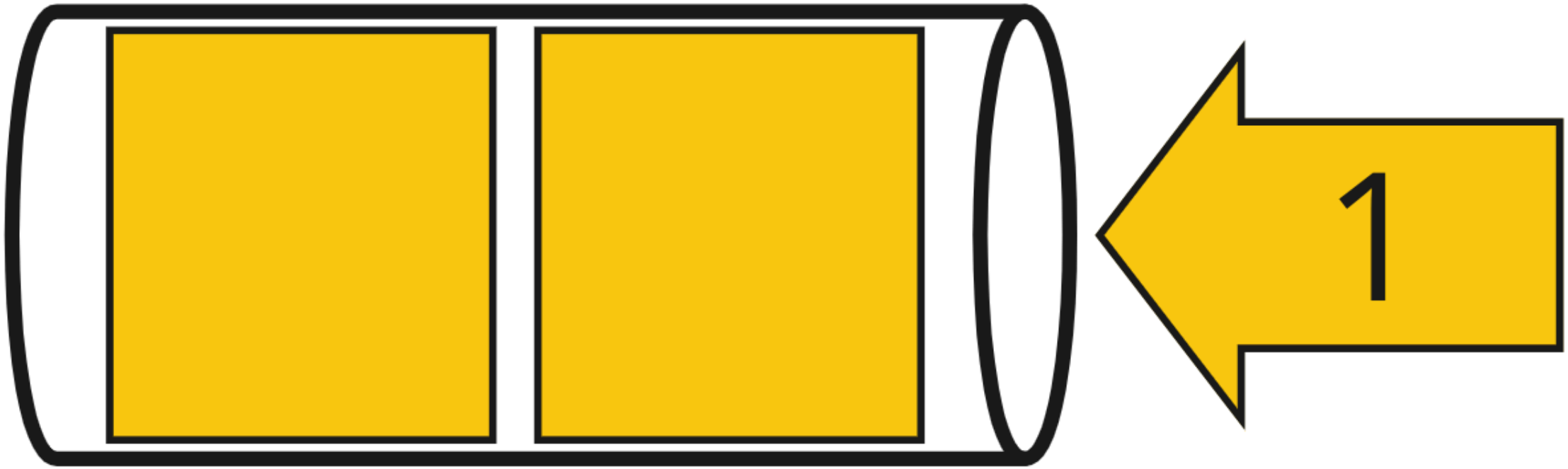
keyword `chan`

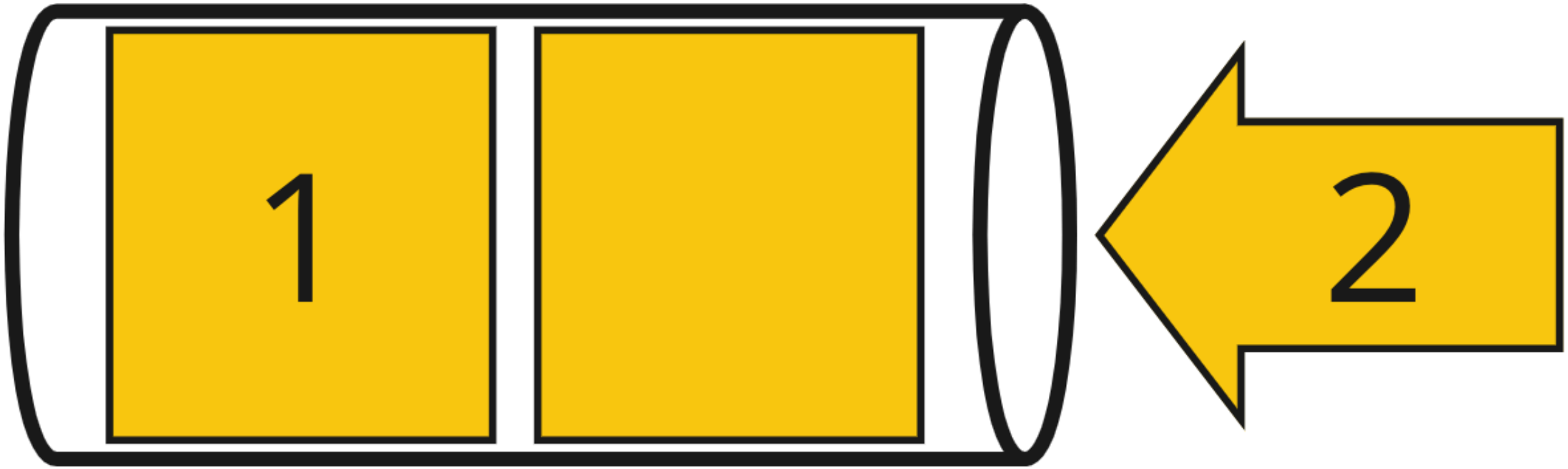
- no buffered channel
- buffered channel

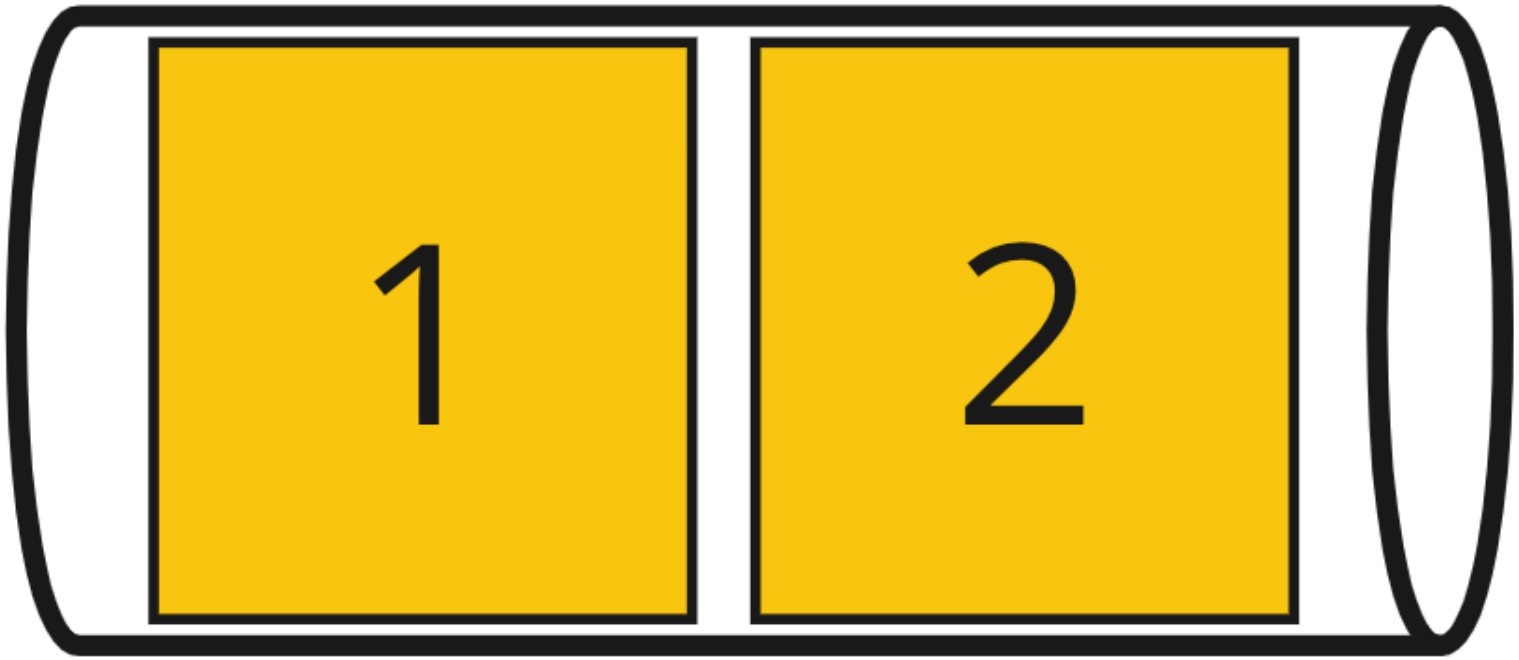
buffered channel simple

```go
func main() {
    ch := make(chan int, 2)
    ch <- 1
    ch <- 2
    fmt.Println(<-ch)
    fmt.Println(<-ch)
}
```



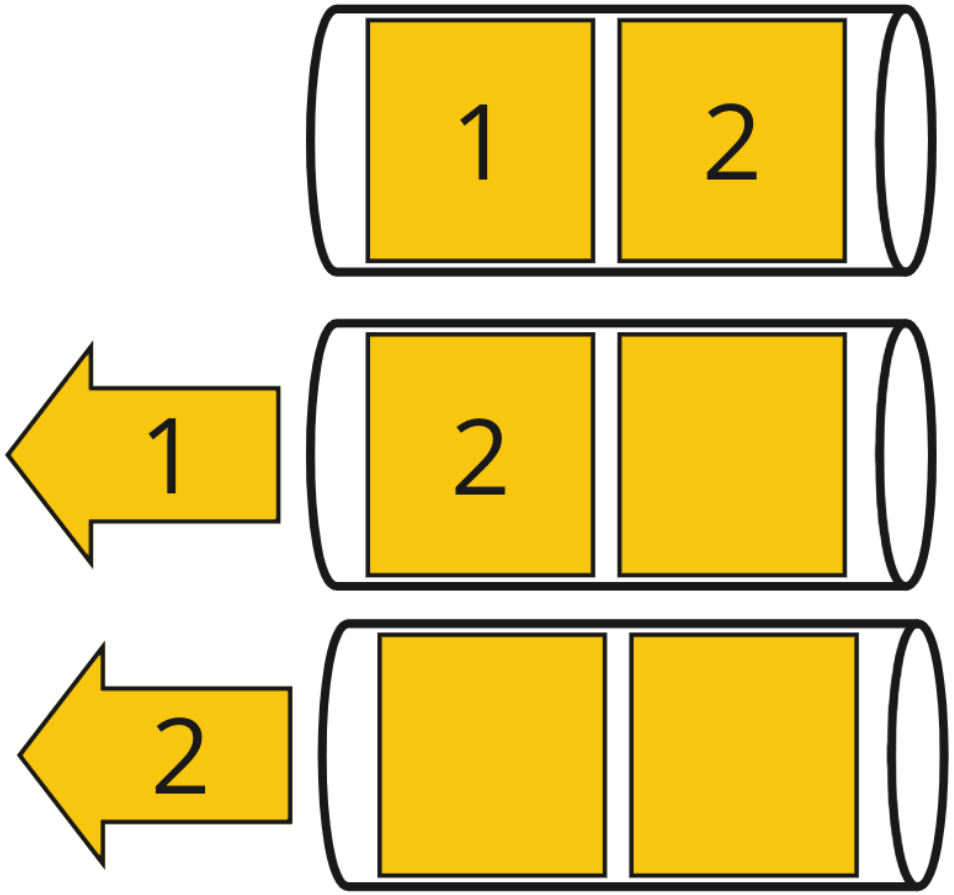

1


1
2


1
2

buffered channel simple

```go
func main() {
    ch := make(chan int, 2)
    ch <- 1
    ch <- 2
    fmt.Println(<-ch)
    fmt.Println(<-ch)
}
```

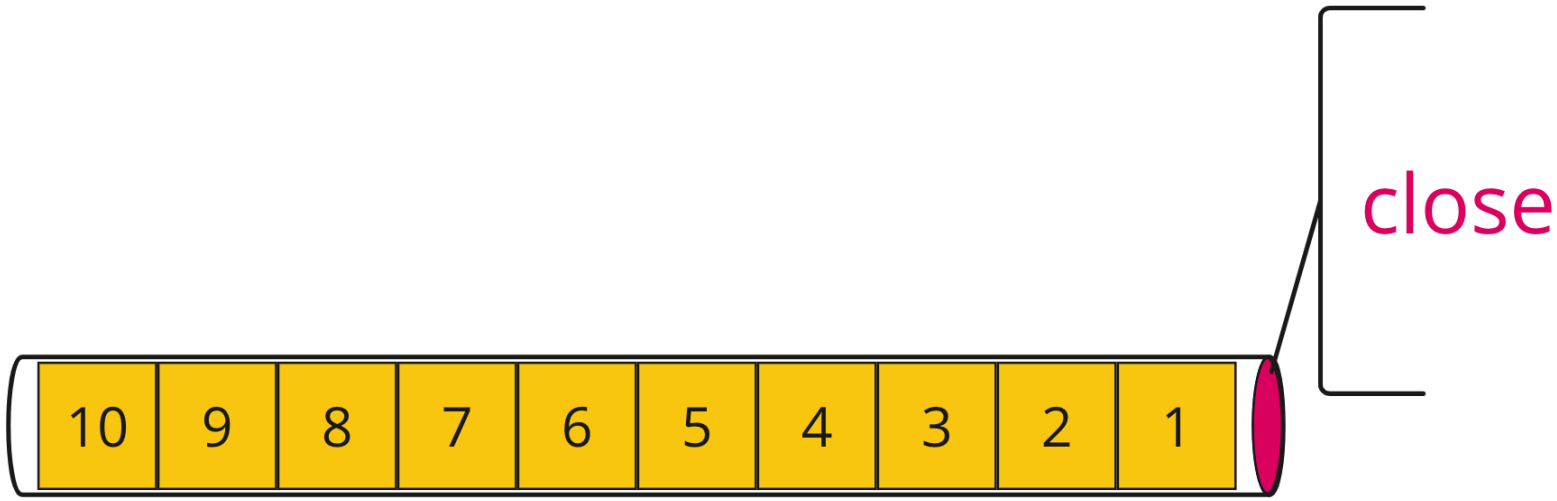

1
2
1
2
2

buffered channel simple

```
func main() {
    ch := make(chan int, 2)
    ch <- 1
    ch <- 2
    fmt.Println(<-ch)
    fmt.Println(<-ch)
}
```

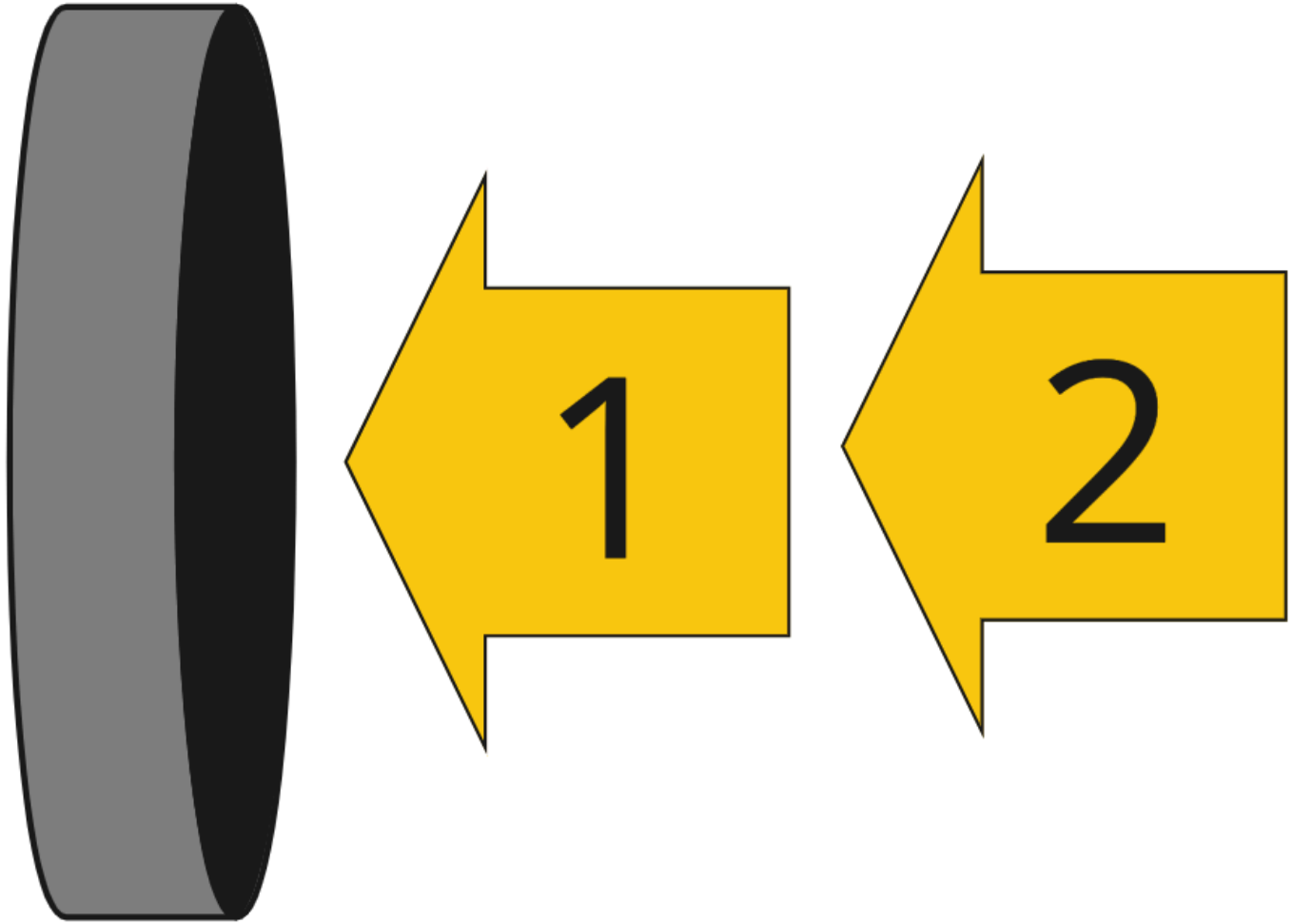
buffered channel

```
total := 10
ch := make(chan int, total)
for i := total; i > 0; i-- {
    ch <- i
}
close(ch)

for i := range ch {
    fmt.Println(i)
}
```


close
10
9
8
7
6
5
4
3
2
1

Demo: buffered channel

no buffered channel

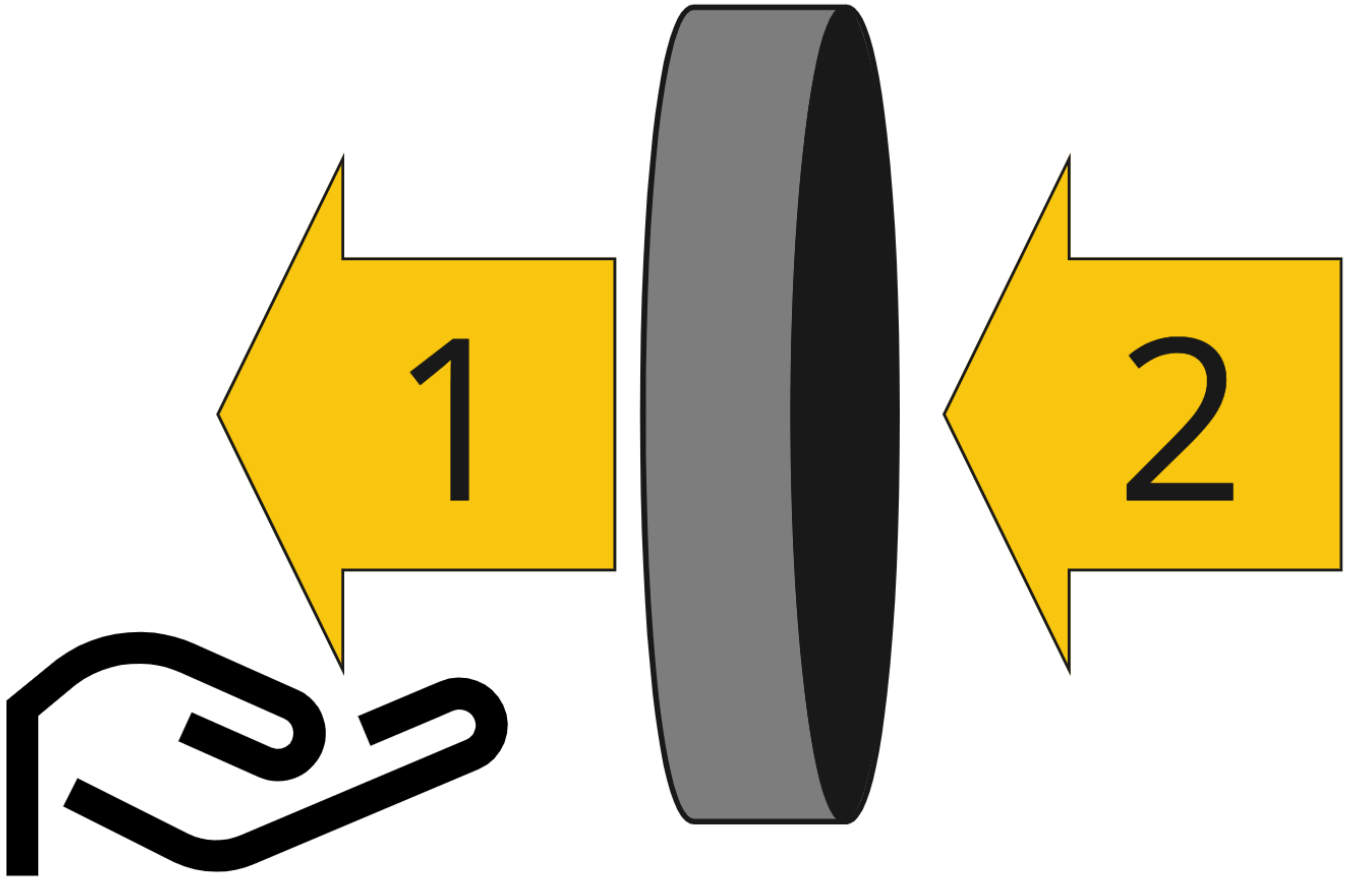


dead lock if no one request

```
ch := make(chan int)
ch <- 1
```

dead lock if no one request

```
ch := make(chan int)
go func() {
    ch <- 1
}()
<-ch
```


1
2

Demo: no buffered channel

test shutdown

```
sudo lsof -i :8080
kill -15 [PID] || kill -SIGINT [PID]
kill -SIGTERM [PID]
```

checklist: graceful

- ✅ env configuration
- ✅ graceful shutdown
- ⬜ golangci-lint
- ⬜ Dockerfile
- ⬜ git-hook
- ⬜ gitlab-ci
- ⬜ testable

golangci-lint

```
go install github.com/golangci/golangci-lint/cmd/golangci-lint@v1.55.2

golangci-lint run ./...
```

checklist: golangci-lint

- ✅ env configuration
- ✅ graceful shutdown
- ✅ golangci-lint
- ⬜ Dockerfile
- ⬜ git-hook
- ⬜ gitlab-ci
- ⬜ testable

Dockerfile

https://twitter.com/sidpalas/status/1637839918448754688?s=20

Docker run

```
docker run --init
```

https://www.baeldung.com/ops/docker-init-parameter

checklist: golangci-lint

- ✅ env configuration
- ✅ graceful shutdown
- ✅ golangci-lint
- ✅ Dockerfile
- ⬜ git-hook
- ⬜ gitlab-ci
- ⬜ testable

git-hook

```
brew install pre-commit
pre-commit install
```

checklist: git-hook

- ✅ env configuration
- ✅ graceful shutdown
- ✅ golangci-lint
- ✅ Dockerfile
- ✅ git-hook
- ⬛ gitlab-ci
- ⬛ testable

Programming Languages API

Create Table `laguages`

```
CREATE TABLE laguages (
    id INTEGER primary key autoincrement,
    name TEXT,
    imageUrl TEXT,
    created_at TEXT, -- ISO8601: YYYY-MM-DD HH:MM:SS.SSS
    updated_at TEXT,
    deleted_at TEXT
);
```

```
sqlite3 languages.sqlite < ./create_table.sql
```

Lang: list

GET /laguages


```
[
    {
        "name": "Java",
        "imageUrl": "https://logos-world.net/wp-content/uploads/2022/07/Java-Logo-700x394.png"
    },
    {
        "name": "Python",
        "imageUrl": "https://logos-world.net/wp-content/uploads/2021/10/Python-Emblem.png"
    },
    {
        "name": "Go",
        "imageUrl": "https://go.dev/blog/go-brand/Go-Logo/PNG/Go-Logo_Aqua.png"
    }
]
```

Start with Simple GIN

```go
package main

import (
  "net/http"

  "github.com/gin-gonic/gin"
)

func main() {
  r := gin.Default()
  r.GET("/ping", func(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
      "message": "pong",
    })
  })
  r.Run() // listen and serve on 0.0.0.0:8080 (for windows "localhost:8080")
}
```

Cross-Functional

- [ ] env configuration
- [ ] graceful shutdown
- [ ] golangci-lint
- [ ] Dockerfile
- [ ] git-hook

Cross-Functional: done

- ✅ env configuration
- ✅ graceful shutdown
- ✅ golangci-lint
- ✅ Dockerfile
- ✅ git-hook

GET /laguages

Code Structure

- [ ] Testable?
- [ ] Changeable?

interface

interface defines a set of method signatures

```
type I interface{
     M()
}
```

zero value of interface{} is nil

```
type I interface{
    M()
}

func main() {
    var i I
    fmt.Println(i)
}
```

Interfaces are implemented implicitly

```
type I interface {
    M()
}

type S struct {}
func (S) M() {
    fmt.Println("S implements the interface I")
}
```

```
func main() {
    var i I
    var s S
    i = s
    i.M()
}
```

Interfaces are implemented implicitly

```
type I interface {
    M()
}

type S struct {}
func (S) M() {
    fmt.Println("S implements the interface I")
}
```

```
func main() {
    var i I
    var s S
    i = s
    i.M()
}
```

Stringer

```
type Stringer interface {
    String() string
}
```

https://go.dev/doc/effective_go#interface-names

Interface: names

By convention, one-method interfaces are named by the method name plus an -er suffix or similar modification to construct an agent noun: Reader, Writer, Formatter, CloseNotifier etc.

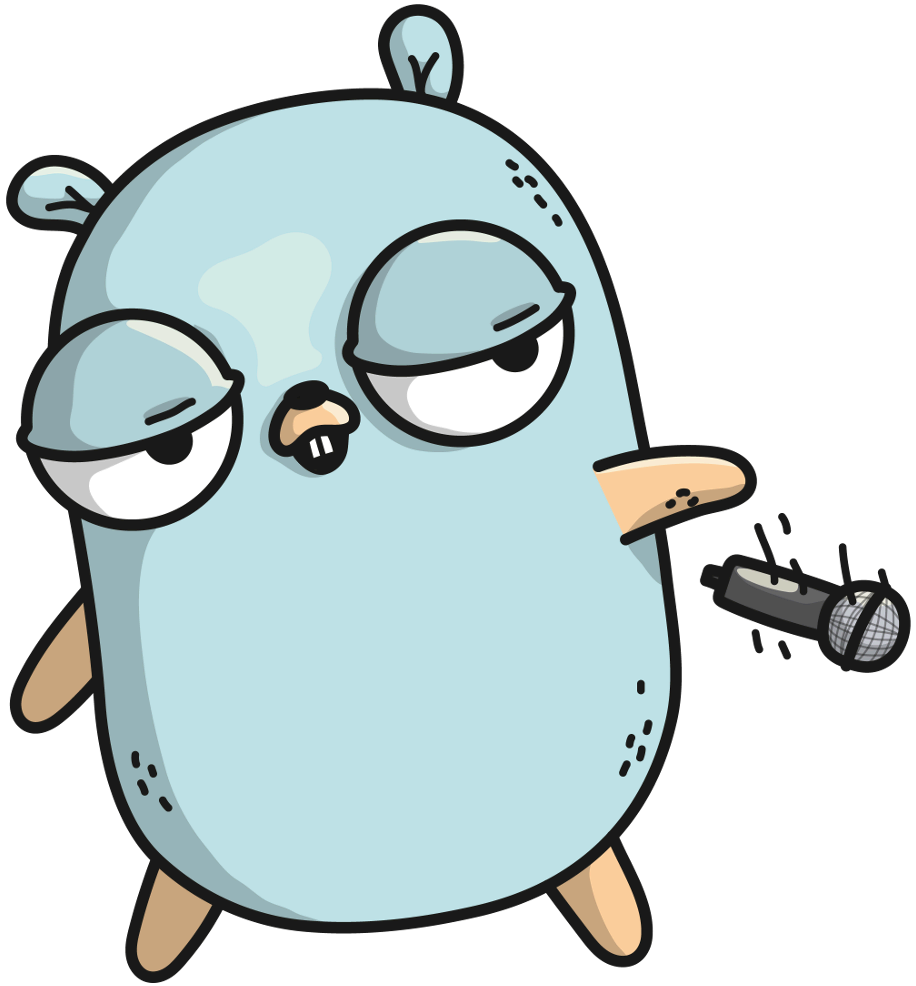
```
type Reader interface {
    Read(p []byte) (n int, err error)
}

type Writer interface {
    Write(p []byte) (n int, err error)
}
```

error type

```
type error interface {
    Error() string
}
```

Play with interface

```go
func AnimalSound(specie string) string {
    switch specie {
        case "cat":
            return "Meow"
        case "dog":
            return "Woof"
        case "lion":
            return "Roar"
        case "owl":
            return "Hoot"
        default:
            return "Grr"
    }
}

type Animal interface {
    Sound() string
}

func MakeSound(a Animal) {
    fmt.Println(a.Sound())
}
```




แบบประเมินผลหลังเข้ารับการฝึกอบรม
Go Basic Workshop Round 1
22-23 April 2024 I 09:30: - 17:30 I The PARQ, 5th Floor - Incubator Area
Anuchito