

Go 程式設計課 01

Go 與其他語言



關於我

- ❖ 林俊佑 Leon Lin
- ❖ Mail : a0970785699@gmail.com
- ❖ <https://github.com/leon123858/go-tutorial>
- ❖ 經歷
 - 國立臺灣大學 資訊工程研究所 碩士
 - 國立成功大學 機械工程學系 學士



關於我 林俊佑 Leon Lin

- ❖ 證照
 - google cloud
Professional Cloud Architect
- ❖ 專長領域
 - 網頁全端／雲端 Infra／系統軟體開發
- ❖ 程式語言
 - C/C++
 - JS/TS
 - GO



課程將會教你什麼

- ❖ 興趣取向
 - 盡可能多、深入介紹 Go 語言以及網頁開發
 - 較少的基本語法教學
 - 重於**實際應用及背後原理**
- ❖ 滿足大家對這個領域的求知慾
- ❖ 盡量不教Google或 ChatGPT 就能告訴你的東西



課程介紹與注意事項

- ❖ 前置課程
 - 熟悉至少一種程式語言的語法與使用
- ❖ 計分標準
 - 共 8 個章節，每章節各 1 份作業，各占 10 分 (80%)
 - 每次上課，課後回饋，一次 2 分 (20%)
 - 超過 60 分視為通過，會在最後一堂課結束後一週統計作業與回饋



課程大綱

- Go 語言
 - Go與其他語言
 - Go 與設計模式
 - Go 與併發設計 *溝通*
 - Go 與作業系統
- 網頁設計
 - Go 與資料庫
 - Go 與後端開發
 - Go 與雲原生
 - Go 與 Kubernetes(K8s)

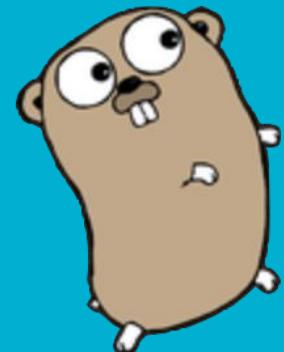
兩單元,八小章,前半描述 Go 的原理與使用,後半講解 Go 實際進行的網頁開發

Lab

Go環境設置

Labo1 Go 環境設置

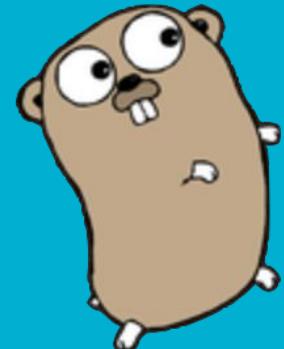
- 1) Go 安裝
- 2) docker-desktop
- 3) git
- 4) vscode
 - a) 插件: golang.go



Go語言安裝

官網安裝 : <https://go.dev/dl/>

- 下載安裝器
 - Mac -> pkg
 - Windows -> msi

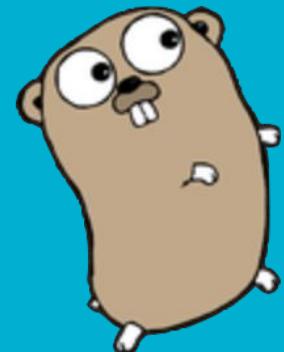


Docker Desktop安裝

官方安裝：<https://www.docker.com/products/docker-desktop/>

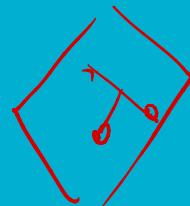
- 下載安裝器
 - Mac -> pkg
 - Windows -> msi
- ❖ Simple to maintain
- ❖ Secure from the start
- ❖ Easy to scale

2周後

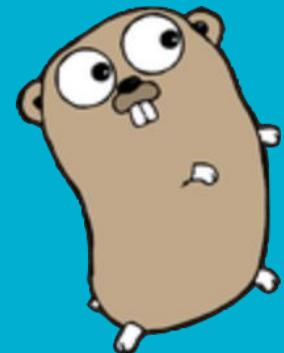


Git安裝

官方安裝 : <http://git-scm.com/download>



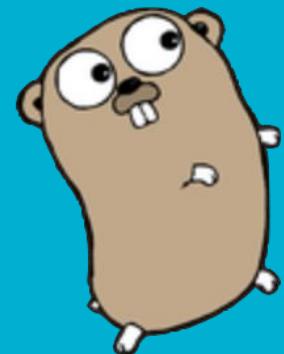
suggest



Vscode安裝

官方安裝：<http://git-scm.com/download/mac>

✓ now



了解 Go 的歷史

- ❖ 由Google開發，誕生於2007年
 - Robert Griesemer, Rob Pike, 和 Ken Thompson等人設計
 - 希望能解決其他語言缺點，並同時保留其優點(C的執行速度,Python的快速開發)
- ❖ 2009年11月正式宣布推出
- ❖ 成為開源項目，並於2012年公開發布

了解 Go 的哲學

- ❖ Go=C+Python
- ❖ Simple is the best
 - 一個目的, 一種做法
 - 宣告式/三元判斷/語法糖



JavaScript

```
// 指令式  
for (let i in arr) {}  
// 告白式  
arr.map(v => {})  
arr.reduce(v => {})
```

↳ [0, 1, 2, 3, 4].map(v => v²)

Go

```
// 指令式  
for _, v := range arr {}
```

≡

C++

auto $x = b ? \text{"yes"} : \text{"no"}$

(
 auto $x = a || b$
 $x = a \& b$)

auto $x = (a || b) \&& c$

syntax (and)

C#
using (resource)
{
 op resource
}

if err != nil
 GO
 err := getResource()
if err != nil
 err = rebus /
 return

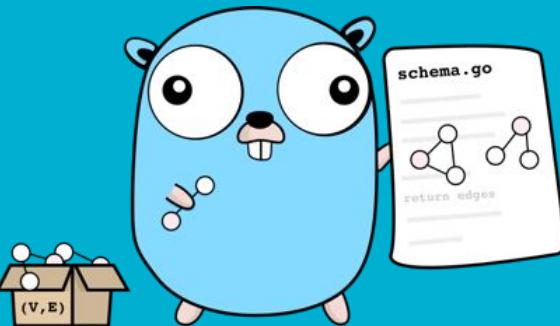
var bunny string = "rabbit"
型別推定 → bunny := "rabbit"

try {
 // ...
 // ...
}
catch (e1) {}
catch (e2) {}
catch (e3) {}

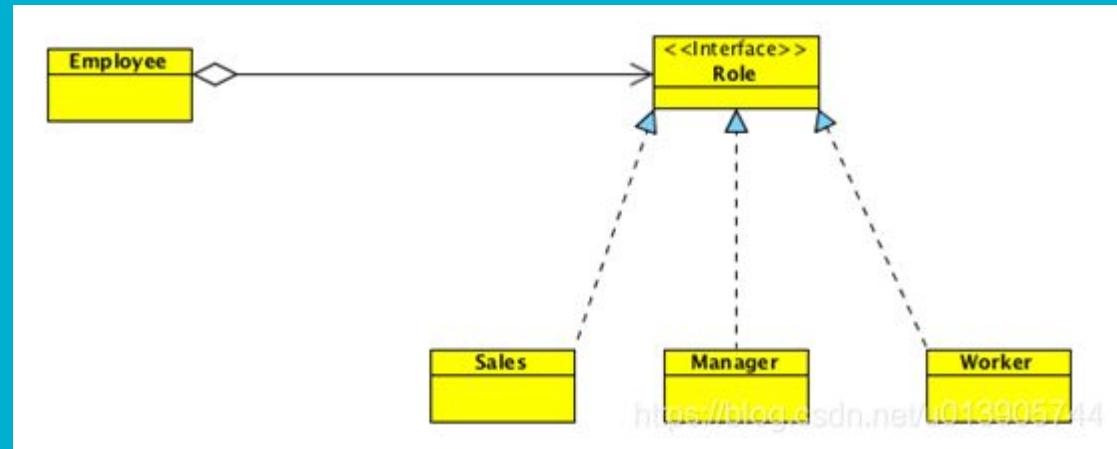
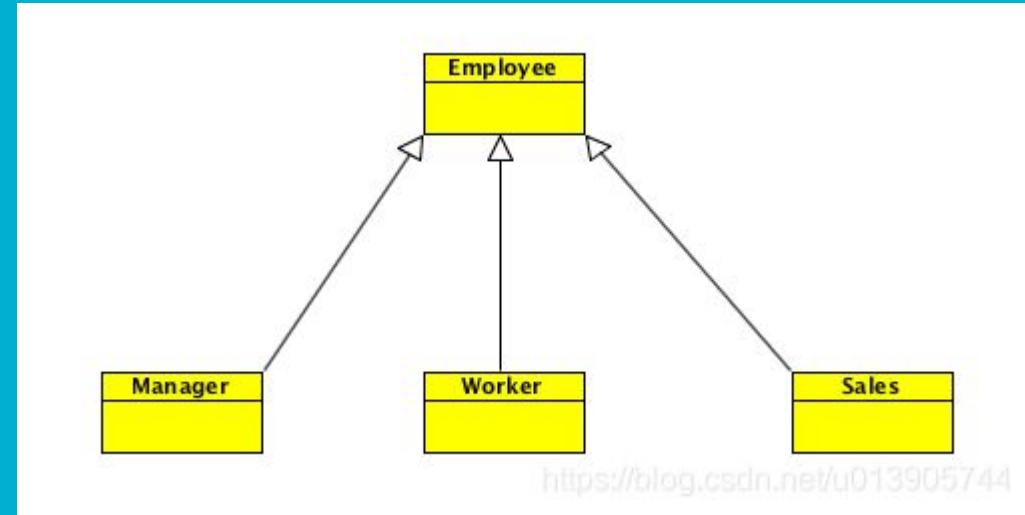
了解 Go 的哲學

❖ 正交性

- 組合大於繼承
- 介面即為泛型
- 模組優於物件



law couple



了解 Go 的特性

❖ 編譯式語言 (\hookrightarrow JAVA, C# = 中間語言)

- 強型別和靜態型別語言(編譯時期會知道變數型別)

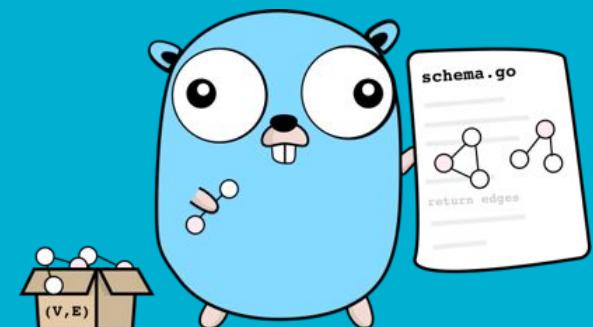
❖ 原生併發語法

- Goroutine

- 基於CSP模型(Communicating Sequential Processes)的實現

- 管道通信機制(Channel)

❖ Garbage Collection



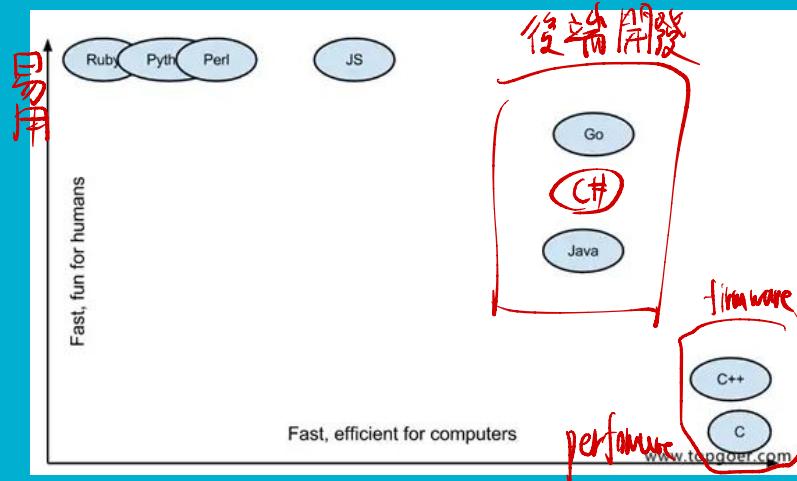
了解 Go 和其他語言的比較

❖ C/C++

- Go增加了切片(Slice)型、併行、管道(Channel)、垃圾回收(GC)、介面等特性的語言級支援
- 但是, Go並不包括如列舉、例外處理、繼承、泛型(此功能在go1.18中加入)等功能

❖ Java

- Go原生提供了:關聯陣列(又稱雜湊表(Hashes)或字典(Dictionaries))



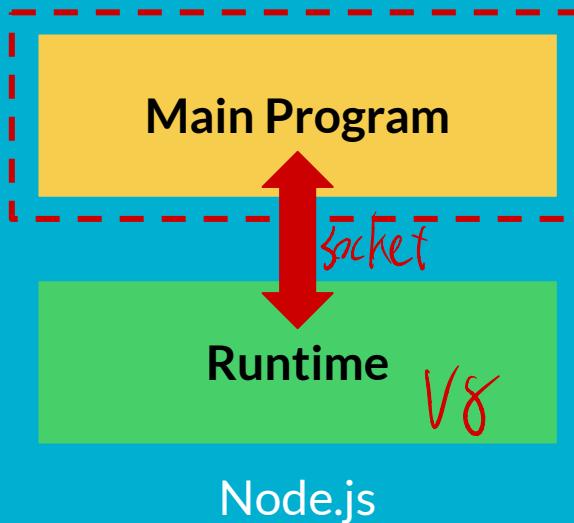
了解 Go 和其他語言的比較

Object.SetString(object) object.SetString

編譯式語言,

而非直譯式語言(含中間語言), 却保留了如反射、GC、Portable 等直譯式語言的功能

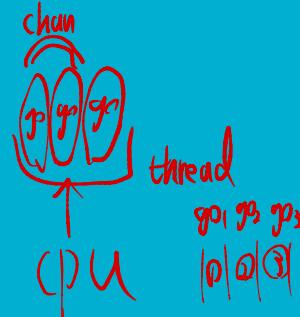
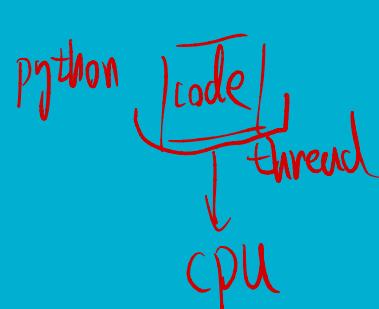
meta data + data



Go的優點

- ❖ 簡單性&易讀性, 容易學習及維護
- ❖ 執行速度快, 故適合用來開發
- ❖ 執行時期資源耗用極低
- ❖ 編譯快速, 能高效開發
- ❖ 簡單的高併發=>透過Goroutine & Channel 的機制實現

Goroutine



- ❖ 為多個並發控制執行序的概念，稱為goroutines，在一個共享位址空間運行，並有效地重複使用到作業系統執行緒上。
 - 對阻塞操作的調用，如從檔案或網路中讀取，只阻塞進行該操作的goroutine
 - 該線程上的其他goroutine可能會被移到另一個線程，以便在呼叫者被阻塞時繼續執行。
- ❖ goroutine可以根據需要調整大小。
- ❖ 對於一個伺服器程式來說，擁有數千甚至數百萬個goroutines是很平常的，因為它們的成本要遠低於執行緒。

Channel

- ❖ Go 提供了通道(Channel), 用於在不同的goroutine之間傳遞資料和同步。
- ❖ 它們提供了一種通訊的機制, 可以讓goroutines之間安全地交換信息, 而不需要額外的互斥鎖或信號量。
- ❖ 通道是一個單向的、尺寸有限的管道
- ❖ Go 還提供了一個多向 select, 可以根據通信的進行來控制執行。

Go的優點

- ❖ 垃圾回收機制GC(Garbage Collection)
 - 可以自動管理內存，減輕開發負擔
- ❖ 部署方便、開發環境單純
 - go build -x file ⇒ 原則上只用靜態鏈接
- ❖ 強大的標準函式庫、工具生態系統
 - go get
 - gofmt
- ❖ 跨平台支援

1. runtime

2. ↗

Garbage Collection

GC(Garbage Collection), 也稱為垃圾回收, 是一種自動記憶體管理機制, GC 嘗試回收不再被程式所需的物件所佔用的記憶體。

GC 最早起源於 LISP 語言, 1959 年左右由 John McCarthy 創造用以簡化 LISP 中的記憶體管理。所以 GC 可以說是一項"古老"的技術, 但是直到 20 世紀 90 年代 Java 的出現並流行, 才被人們得以知曉。

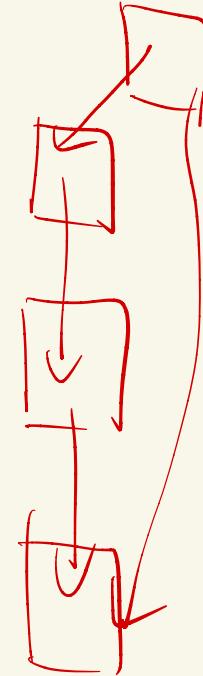
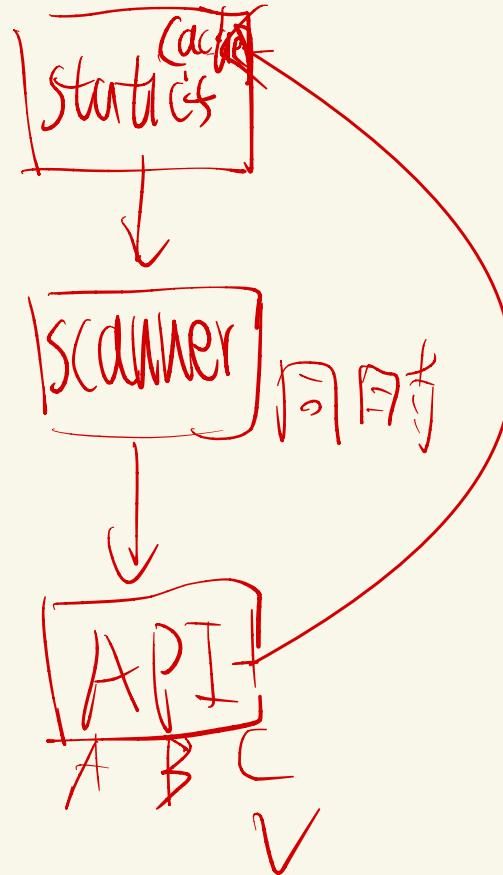
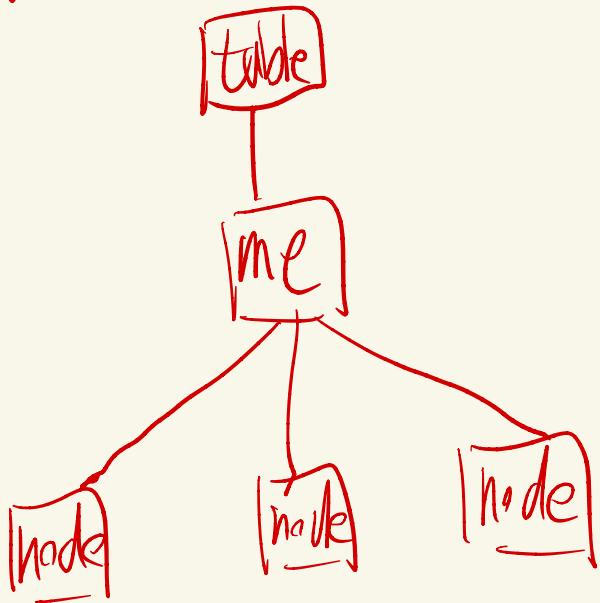
除 Go 以外許多語言如 Java, C#, JS 和 Python 等都支援 GC。

Go的缺點

- ❖ 為了簡單產生的問題 (過於追求簡單,, 而缺乏彈性)
 - 不支援類別Class、繼承
 - 相依限制大 (不接受循環相依), 因為組合優於繼承
 - 錯誤處理繁瑣
- ❖ 為了正交產生的問題 (期望能使組件天然解耦, 導致大型程式易混亂)
 - 接口濫用
 - 泛型支援不足, 需仰賴反射
- ❖ GUI庫不好用
- ❖ 缺乏穩定、功能豐富的Web框架(Ex: Django, Ruby On Rails, Spring Boot, ...)

偏科, goLang 只會他會的

相依



Go的應用

- ❖ 系統層級應用程式
- ❖ Web 應用程式
- ❖ DevOps
- ❖ 雲端原生(cloud-based)應用程式、伺服器端(server-side)應用程式
- ❖ 公用程式和命令列(CLI)工具
- ❖ 資料庫實作

➤ 2018年1月 Google 公佈了去年底統計的問卷結果，發現61% 用 Go 來寫網站，37% 用來開發系統程式，36% 用來做 DevOps (多重選擇)

參考來源

: <https://www.pluralsight.com/resources/blog/cloud/what-is-go-an-intro-to-googles-go-programming-language-aka-golang>

2024 年 4 月編程語言排行榜				
排名	編程語言	流行度	對比上月	年度明星
1	Python	16.41%	▲ 0.78%	2021, 2020
2	C	10.21%	▼ -0.96%	2019, 2017
3	C++	9.76%	▼ -0.94%	2022, 2003
4	Java	8.94%	▼ -0.01%	2015, 2005
5	C#	6.77%	▼ -0.77%	2023
6	JavaScript	2.89%	▼ -0.49%	2014
7	Go	1.85%	▲ 0.29%	2016, 2009
8	Visual Basic	1.70%	▲ 0.28%	-
9	SQL	1.61%	▼ -0.31%	-

Go Case Study

- Dropbox
- Trivago
- Riot Games

Dropbox

將高效能需求後端服務，從Python遷移至Go。

理由：

- 更佳的併發支援
- 更快的處理速度

團隊開發程式庫(open sourced)：<https://github.com/dropbox/godropbox>



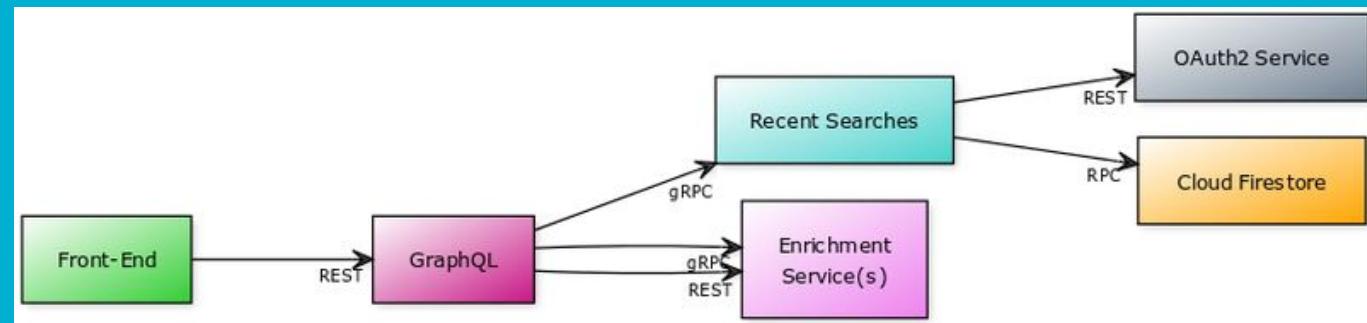
參考：<https://dropbox.tech/infrastructure/open-sourcing-our-go-libraries>

Trivago



理由：

- 簡易開發
- 豐富的函式庫
- go fmt 工具豐富



參考：<https://tech.trivago.com/post/2020-03-02-whywechosego>

Trivago

案例: Race Detector

- 編譯器使用記錄訪問記憶體的時間和方式的程式碼來檢測所有記憶體訪問而運行時庫則監視對共享變數的非同步存取

```
type Service struct {
    value string
}

func (s *Service) handle(req Request) {
    s.value = req.stringField
    fmt.Println(s.value)
}
```

參考: <https://tech.trivago.com/post/2020-03-02-whywechosego>

Riot Games

理由：

- 容易封裝、部署
- 建置速度很快
- 龐大且強大的標準函式庫
- goroutine和通道Channel的形式
 - 提供強大的平行化和並發原語



參考：<https://technology.riotgames.com/news/leveraging-golang-game-development-and-operations>

Riot Games

案例: 營運監控



```
select {
    case workQueue <- requestData:
        return nil
    case <-time.After(timeout):
        return errors.New("queue is full")
}
```

參考: <https://technology.riotgames.com/news/leveraging-golang-game-development-and-operations>

Riot Games - Valorant

遊戲整個後端微服務架構皆用Go建構

- 並發型原生語言
- 隱式介面
- 封裝模組化

參考 : <https://technology.riotgames.com/news/leveraging-golang-game-development-and-operations>

Lab

Go 基本語法

Go 基本語法 Lab

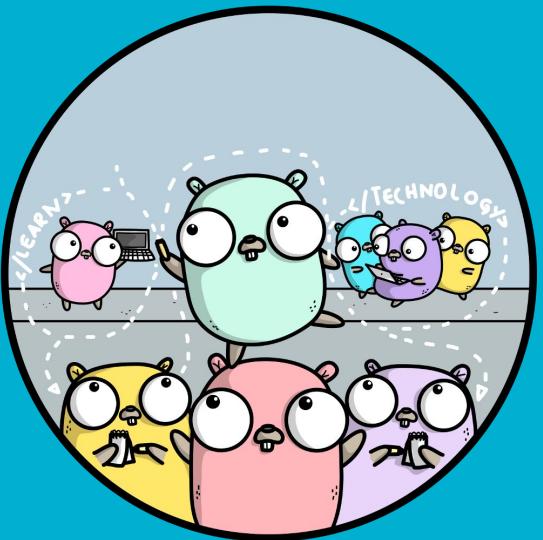
```
// install 教學
go install golang.org/x/website/tour@latest
// run
tour
// find gp path
go env GOPATH
// remove bin
cd $(go env GOPATH)
cd bin
rm tour
```

Go 基本命令

1. go help
2. go mod init <package>
3. go run <file>
4. go get -u <package>
5. go install <package>
6. go build <package>
7. go fmt <file>
8. go version

小試身手

- 1) welcome
- 2) basic
- 3) flow
- 4) type



Q&A 時間

pointer (ptr)

$x := 5$

$y := \&x$

x

55688	57
址	值

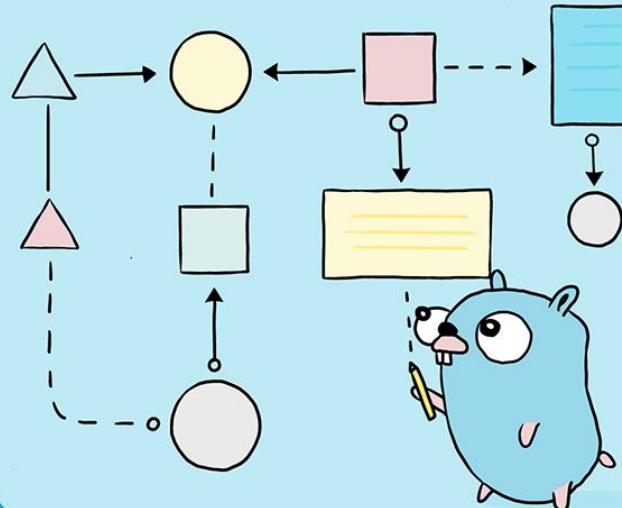
y

77539	55688
*	$y = 7$

$\&$: 取址
 $*$: 取值

bigbrother { 10MB }

func (bigbrother)

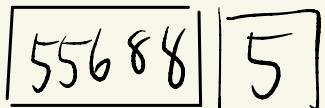


$x := v \dots ,$ 複製
func(str string) (
 println(str))

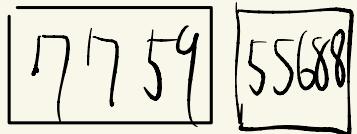
$y := \&x$ 指向
func(str *string) (
 > println(*str))

ptr (pointer)

$x := 5$



$y := \underline{x}$



\ast : 取值

$\&$: 取址

$(\ast \text{Var}) = \text{值}$

$(\& \text{Var}) = \text{址}$

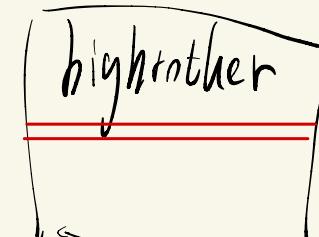
$(\& y) = 7759$

struct bigbrother

记得体位置

func ($\ast \text{bigbrother}$) $\ast \text{bigbrother}$

$b := \text{bigbrother} \rightarrow$



$\&b = 1111$

func(11111)

$\&b$

call by ref

[slice

chan

map]

func (strs []string)

slice

header + data

value

[string

struct

int

:]

func (a int)

obj → copy .

ptr

[*任何]

func (a *int)

(b *slice)

(c *string)

(d *struct)

本節作業 - A simple CLI

實作一個 CLI tool 可以完成以下三個任務

1. 打招呼
2. 產生名言佳句
3. 印出任意網址原始碼
4. 計算每個人的回饋次數

Link: <https://github.com/leon123858/go-tutorial/blob/main/cli-sample/cli.go>