

GOLANG



Info en vrac.

La première version date de 2009.

Initialement conçu par les équipes de Google.

La mascotte est une marmotte (gopher).

Inspiré de C et du Pascal.

Le go se trouve dans énormément de produits tel que Docker, Kubernetes et bien d'autres.

C'est un langage compilé.

Le binaire de Go est cross-plateforme ! mais dans certains cas il faudra lui préciser la plateforme de destination.

>_

```
go help
```

```
go run hello.go
```

```
go build
```

```
go fmt
```

```
go get
```

```
go mod
```

```
go test
```

Commandes à connaître

→ Obtenir de l'aide !

→ Executer du code

→ Construire le binaire

→ Formater le code

→ Obtenir une librairie

→ Installer des librairies

→ Executer les tests



Nom du package de l'application

Dissection d'un morceau de code

```

package main

import "fmt"

func main() {
    message := greetMe("world")
    fmt.Println(message)
}

func greetMe(name string) string {
    return "Hello, " + name + "!"
}

```

Les différentes librairies qui seront importées se trouvent ici !

Les fonctions se déclarent grâce à "func" et peuvent contenir des arguments. A noter que le retour est typé.

Pour affecter une variable on utilise ":", une fois celle ci affectée plus besoin de ":".

On exécute les commandes "Println" du package "fmt" afin d'afficher un message en console !

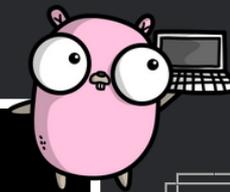
Les particularités

Les structures

Le Golang n'est pas un langage orienté objet mais permet de structurer son code à l'aide des "Struct" qui ont un comportement assez similaire aux classes.

Les GoRoutines

L'une des particularité majeure du Go, c'est l'utilisation des goroutines. C'est un thread léger géré par le runtime Go qui permet d'exécuter des tâches en concurrence.



Les ressources

- Documentation: <https://golang.org/doc/>
- Installation: <https://golang.org/doc/install>
- A Tour of Go: <https://tour.golang.org/list>
- Go by example: <https://gobyexample.com/>
- Go cheatsheet - DevHints: <https://devhints.io/go>



```
// Ligne simple

/*
  Commentaire comportant
  plusieurs lignes
*/
```

GOLANG

Les fonctions

Fonction lambda

```
myfunc := func() bool {
  return x > 10000
}
```

Fonction retours multiples

```
a, b := getMessage()

func getMessage() (a string, b string) {
  return "Hello", "World"
}
```

Les variables et les types

Déclarer une variable:

```
var hello string
hello = "Hello, World!"
```

OU

```
hello := "Hello, World!"
```

Déclarer une constante:

```
const Phi = 1.618
```

Les types

Les nombres:

```
num := 3                int
num := 3.               float64
num := 3 + 4i           complex128
num := byte('a')      byte
```

Les array:

```
var numbers [5]int
numbers = [...]int{0, 0, 0, 0, 0}
```

Les slices:

```
slice := []int{2, 3, 4}
```

Les conditions

Structure if classique

```
if day == "sunday" || day == "saturday" {
  rest()
} else if day == "monday" && isTired() {
  groan()
} else {
  work()
}
```

Structure if contenant une déclaration

```
if _, err := doThing(); err != nil {
  fmt.Println("Uh oh")
}
```

Structure Switch/Case

```
switch day {
case "sunday":
  fallthrough
case "saturday":
  rest()
default:
  work()
}
```

Les boucles

Boucle for

```
for count := 0; count <= 10; count++ {
  fmt.Println("My counter is at", count)
}
```

Boucle for range

```
entry := []string{"Jack", "John", "Jones"}
for i, val := range entry {
  fmt.Printf("At position %d, the character %s is present\n", i, val)
}
```

Pour une fiche un poil plus complète:
<https://devhints.io/go>



Happy Coding!

