Cous Go du jour :

La structure la plus basique de Go (en respectant la casse) :

```
package main

func main() {
}
```

Un fichier Go doit se construire de cette manière là. Avec “package main” et “func main() {}”

Package => permet d’importer certaines fonctions d’un fichier dans un autre fichier (comme z01).

Ensuite, on doit avoir tout le temps une fonction main (c’est le point d’entrée). Pour que le compilateur sache par où commencer.
On l’a met où on veut, mais le prof le met tout en bas car dans certains langages il faut le mettre tout en bas ou ça le reconnaîtra pas.
C’est le basique du basique car on l’utilisera tout le temps.

Une fonction count qui prend en paramètre deux chiffres :

```
package main

func main() {

}

func count(a int, b int) int {

return
}
```

On peut écrire « func count(a, b int) int { », c’est exactement la même chose que « a int, b int ». Il faut toujours typer ces variables.

Il faut mettre une variable de retour : on doit mettre « return ». On va retourner un entier avec « return ».

Si dans la fonction on additionne a et b et qu’on appelle le résultat c :

```
package main

func main() {

}

func count(a int, b int) int {

var c int
c = a + b
return c
}
```

Il y a trois manières de faire :
- La première on déclare directement une variable : « var c int ».
- La seconde c'est de mettre « := ». Ca permet d'automatiquement typer une variable.

Ou on peut aussi faire ça (3eme méthode) :

```
package main

func main() {

}

func count(a int, b int) int {

return a + b
}
```

On "return" directement le résultat car on a typer "a" et "b" dans les paramètres de la fonction "count".

Les bibliothèques se font avec « import ». On met des parenthèses quand il y en a plusieurs. Mais pas besoin de parenthèses s'il y en a qu'une.

```
package main

import "fmt"

func main() {

}

}
```

On peut utiliser if (c'est un booléen), ça permet de rajouter une condition.
Exemples : « Si ma condition est juste, alors ... » « Si ma condition n'est pas juste, alors… »

```
package main

import “fmt”

func main() {

}
z := 5
if z < 5
}
```

Ici, on a décidé de mettre que z est un int, et est égal à 5, « si z est inférieur à 5 » alors on fera quelque chose.

Pour les noms de fonction, si on ne l'utilise que dans son fichier, on commencera toujours par une minuscule, puis s'il y a plusieurs mots on colle tous les mots en ajoutant une majuscule au début des autres. Par exemple :

```
package main

import “fmt”

func c’estDeLaMerde() {

}
fmt.Println(“test\n”)
}
```

On a commencé par une minuscule, et puis tous les autres mots commencent par une majuscule. C'est le principe du camelCase.

Mais si on décide d'exporter notre fonction, alors on le fera commencer par une majuscule dès le début, et à chacun des mots.

```
package module

func TestUp(a int, b int) bool {

res := false
if a > b {
res = true
  }
return res
}
```

Ensuite, pour l'ajouter au fichier main (fichier principal) on retourne sur le fichier main et on fait :

```
package main

import (
"./module"
"fmt"
)

func main() {
fmt.Println(module.TestUp(5, 10))
}
```

"./module" est le chemin que va faire le compilateur pour retrouver le fichier. Puis va l'ajouter au fichier "main". C'est très pratique pour les gros programmes, et afin de ne pas s'y perdre.