Declaration, PrintRune, if and else

Declaration

On va voir aujourd'hui les bases du GO et de la programmation : La notion de variable et comment les déclarer.

Qu'est-ce qu'une variable ?

Une variable est un élément qui va pouvoir stocker des valeurs, on pourra récupérer ces valeurs en appelant le nom de la variable. En GO, une variable est liée à un type qui va définir les valeurs qu'elle peut contenir.

Qu'est ce qu'un type ?

Un type permet de catégoriser les différentes valeurs qui sont possible, le type `int` correspond à toutes la valeurs numériques entière positive ou négative. (0, 1, -10, 42, 579135, ...)

Il existe beaucoup de types différents, voici une liste des quelques types courants en GO :

- `int` : Les nombres entiers négatif ou positif (0, 1, -10, 42, ...)
- `string` : Les chaines de caractères ("toto", "lamentin", Colorado, ...)
- `boolean ou bool` : true ou false
- `rune` : Des caractères unique, ils correspondent aux caractères de la table ASCII : https://upload.wikimedia.org/wikipedia/commons/d/dd/ASCII-Table.svg

Il en existe bien d'autres que vous aurez l'occasion de découvrir.

Déclarer une variable

En GO, une variable peut se déclarer de 3 manières, cependant les 3 manières répondent à des logiques similaires. Voyons d'abord la méthode classique :

```
Déclaration classique :
    var toto int
```

Ici, le mot-clé **var** permet de préciser que ce que nous déclarons est une variable.
toto quant à lui correspond au nom de notre variable, c'est par ce nom que nous pourrons l'utiliser dans nos codes.
Pour finir, **int** correspond au type de notre variable.

Notre variable est maintenant déclaré, elle ne contient pour l'instant pas de valeurs, remédions à cela :

```
var toto int
toto = 42

// Si j'affiche le contenue de ma variable toto, j'obtiendrais 42

// Je peux bien entendu toujours modifier la valeur de ma variable.
toto = 0
// Si j'affiche le contenue de ma variable toto, j'obtiendrais 0
```

Il est aussi possible de déclarer une variable et de directement lui affecter sa valeur :

```
Déclaration rapide :
    var toto = 42

Déclaration très rapide :
    toto := 42
```

Lorsqu'on donne une valeur à notre variable dés sa déclaration, le type donné à notre variable sera le type de la valeur qu'on lui a donné. (42 est un nombre entier --> la variable toto est de type `int`)

Dans la déclaration très rapide, le mot-clé **var** est substitué par les `:` qui indiquent une déclaration, ils ne sont pas à remettre lorsqu'on souhaite modifier plus tard la valeur de la variable.

PrintRune

L'une des fonction qui vous est fourni est la fonction `PrintRune`, elle permet d'afficher dans votre console une `rune`.

Etant une librairie externe, vous devez d'abord la récupérer pour pouvoir l'utiliser.

```
Dans votre invite de commande :
> go get github.com/01-edu/z01
```

Ensuite, dans votre fichier .go vous devez importer la librairie.

```
import "github.com/01-edu/z01"
```

Vous êtes maintenant capable d'utiliser la fonction z01.PrintRune()

```
import "github.com/01-edu/z01"

func main() {
    z01.PrintRune()
}
```

`z01` correspond à la librairie utilisé (on le précise car il s'agit d'une librairie externe).
`PrintRune` correspond au nom de notre fonction.
Les `()` contiennent les paramètres de notre fonction, ce qu'elle va utiliser pour s'exécuter.

Pour afficher la rune que l'on souhaite, il suffit d'indiquer dans les paramètre de z01.PrintRune quelle rune on souhaite afficher.

```
import "github.com/01-edu/z01"

func main() {
    z01.PrintRune('A')
}

// Cette instruction affichera dans la console : A
```

If and Else

En programmation, une des notion les plus importante et les plus utilisé est la notion de **condition**.
Elle permet d'effectuer certaines opérations seulement sous certaines conditions.

IF

Le `if` est le mot-clé de la condition, il se présente sous cette forme :

```
if condition {
    instruction
}
```

`condition` correspond comme son nom l'indique à la condition pour rentrer dans le `if`, si elle n'est pas respecté, le programme passera à la suite du code.
`{ instruction }` désigne les instructions à faire si la condition est remplie.

```
Les conditions sont des instructions dont le résultat est soit vrai soit faux.
on a alors souvent recours à ce que l'on appel des opérateurs de comparaisons
```

- `A == B` Qui signifie : A strictement égal à B
- `A != B` Qui signifie : A pas égal à B
- `A > B` Qui signifie : A strictement suppérieur à B
- `A >= B` Qui signifie : A suppérieur ou égal à B
- `A < B` Qui signifie : A strictement inférieur à B
- `A <= B` Qui signifie : A inférieur ou égal à B

Un exemple concret sera plus parlant :

```
import "github.com/01-edu/z01"

func main() {
    var toto = 42
    if toto  == 42 {
        z01.PrintRune('0')
    }
    if toto > 42 {
        z01.PrintRune('1')
    }
    if toto <= 42 {
        z01.PrintRune('2')
    }
}

// Le programme affichera les runes '0' et '2'
```

ELSE

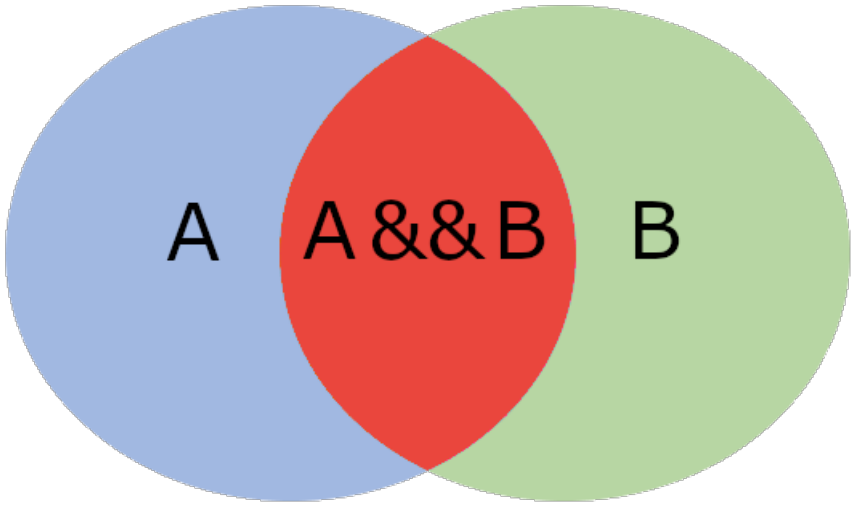
`else` est un mot-clé qui ne peut pas exister sans un if, il fonction à la manière d'un "sinon".
Le programme ne peut rentrer dans un `else` (ou un `else if`) que si la condition précédente n'est pas valide.

```
import "github.com/01-edu/z01"

func main() {
    var toto = 42
    if toto  == 42 {
        z01.PrintRune('0')
    } else if toto > 42 {
        z01.PrintRune('1')
    } else toto <= 42 {
        z01.PrintRune('2')
    }
}

// Ici, le programme n'affichera que la rune '0' et ne regardera pas les else
```

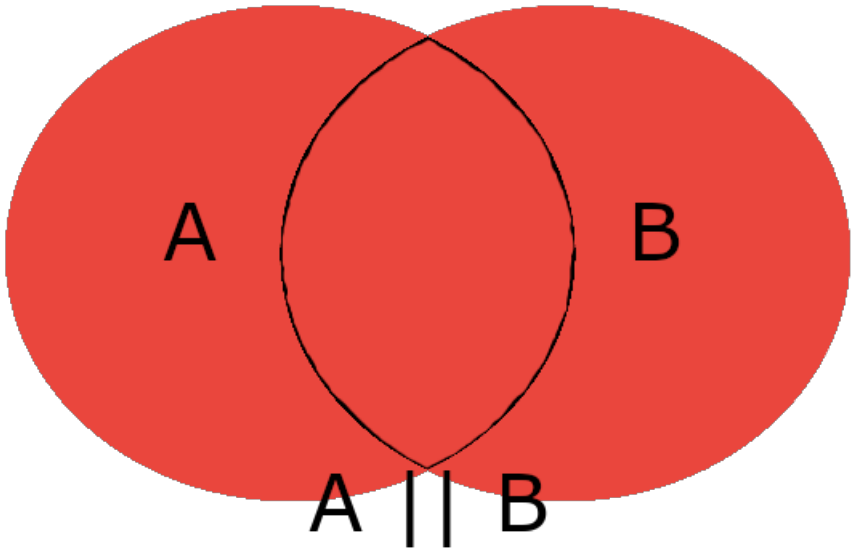
Les opérateurs logique

Pour terminer avec les conditions, il est possible de les combiner par le biais de ce qu'on appel des **Opérateurs logique**

- AND (ET) : `&&`



- OR (OU) : `||`




```
import "github.com/01-edu/z01"

func main() {
    var toto = 42
    if toto > 40 && toto < 50 {
        z01.PrintRune('0')
    }
    if toto < 40 || < 50 {
        z01.PrintRune('1')
    }
    if toto > 40 && toto == 0 {
        z01.PrintRune('2')
    }
}

// Le programme affichera la rune '0' et la rune '1'
```