Les types

Golang est un langage de programmation à typage statique, ce qui signifie que chaque variable a un type. Go a plusieurs types intégrés de base.

Strings

Une string est une chaîne de caractères. En Go, une chaîne de caractères est une séquence de byte.

Les chaînes en Golang sont déclarées soit à l'aide de guillemets doubles comme dans `"Hello World"` soit de backticks comme dans `Hello World`.

```
package main

func main() {
    var ynov string
    ynov = "Une école"

    uneAutrePhrase := `Ici on peut écrire
                       sur plusieurs lignes`
}
```

Integers

Les entiers sont utilisés pour stocker des nombres entiers. Go a plusieurs types d'entiers intégrés de taille variable (en fonction du nombre de bytes) pour stocker des entiers signés (signed) et non signés (unsigned).

Signed :

- `int` : nombre négatif, zéro, positif, la taille dépend du type de votre OS
- `int8` : nombre compris entre -128 et 128
- `int16` : nombre compris entre -2^15 et 2^15 - 1
- `int32` : nombre compris -2^31 et 2^31 - 1
- `int64` : nombre compris -2^63 et 2^63 - 1

Unsigned :

- `uint` : zéro, nombre positif, la taille dépend du type de votre OS
- `uint8` : nombre compris entre 0 et 128
- `uint16` : nombre compris entre 0 et 2^15 - 1
- `uint32` : nombre compris 0 et 2^31 - 1
- `uint64` : nombre compris 0 et 2^63 - 1

```
package main

import "fmt"

func main() {
    var a int
    a = -899

    var b uint
    b = 32

    var c int8
    c = 127
}
```

Runes

Une rune en Go est un alias du type int32, ce qui signifie qu'il s'agit d'une valeur entière. Cette valeur entière doit normalement représenter un point Unicode. Pour comprendre les runes, vous devez savoir ce qu'est Unicode :

Unicode est un sur-ensemble de caractères ASCII qui attribue un numéro unique à chaque caractère existant.

Vous pouvez trouver la table Ascii ici

```
package main

import "fmt"

func main() {
    a := 'A'
    b := 66
    c := 'C'

    fmt.Println(a)
    fmt.Println(string(rune(b)))
    fmt.Println(string(c))
}
```

Floats

Les types à virgule flottante sont utilisés pour stocker des nombres avec une composante décimale (ex -1.24, 4.50000). Go a deux types de virgule flottante : `float32` et `float64`.

- `float32` : les nombres à virgule qui occupent 32-bits dans la mémoire
- `float64` : les nombres à virgule qui occupent 64-bits dans la mémoire

```
package main

func main() {
    var a float32
    var b float64

    a = 2.8384
    b = 19299.3903
}
```

Booleans

Go fournit un type de données appelé `bool` pour stocker les valeurs booléennes. Il peut avoir deux valeurs possibles : vrai et faux.

```
package main

func main() {
    var isGood bool

    isGood = true
    isNotGood := false
}
```

Arrays

Les tableaux en Go sont des valeurs. Ce sont des séquences de longueur fixe du même type.

La syntaxe est la suivante : `[longueur]type{valeurs}`

```
package main

import "fmt"

func main() {
    monTableauDeQuatreElements := [4]int{2, 3, 4, 5}
    fmt.Println(monTableauDeQuatreElements) // [2 3 4 5]
}
```

Slices

Les slices sont comme des tableaux mais ont une taille dynamique. Vous pouvez alors manipuler de façon plus flexible les tableaux, notamment grâce à la méthode `append` pour ajouter des valeurs.

```
package main

import "fmt"

func main() {
    mySlice := make([]int, 2, 5)
    fmt.Println(mySlice) // [0 0]

    mySlice = append(mySlice, 3)
    fmt.Println(mySlice) // [0 0 3]

    alphabet := []string{"a", "b", "c"}
    fmt.Println(alphabet) // [a b c]

    alphabet = append(alphabet, "c", "d")
    fmt.Println(alphabet) // [a b c c d]
}
```

Pointers

Un pointeur est une variable qui contient l'adresse mémoire d'une autre variable. La valeur zéro d'un pointeur est nulle.

```
package main

import "fmt"

func main() {
    var b *int
    a := 2
    b = &a

    fmt.Println(b)  // 0xc0000160a0
    fmt.Println(*b) // 2
}
```