

Tableaux et Tranches

Exercice 1

```
package main

import "fmt"

func main() {
    names := []string{"Einstein", "Tesla", "Shepard"}
    distances := []int{50, 40, 75, 30, 125}
    data := []byte{'H', 'E', 'L', 'L', 'O'}
    ratios := []float64{3.14145}
    alives := []bool{true, false, true, false}
    zero := []byte{}

    fmt.Printf("names      : %[1]T %[1]v\n", names)
    fmt.Printf("distances: %[1]T %[1]d\n", distances)
    fmt.Printf("data       : %[1]T %[1]d\n", data)
    fmt.Printf("ratios    : %[1]T %.2[1]f\n", ratios)
    fmt.Printf("alives   : %[1]T %[1]t\n", alives)
    fmt.Printf("zero     : %[1]T %[1]d\n", zero)
}
```

Exercice 2

```
package main

import (
    "fmt"
    "sort"
    "strings"
)

func main() {
    namesA := "Da Vinci, Wozniak, Carmack"
    namesB := []string{"Wozniak", "Da Vinci", "Carmack"}

    namesC := strings.Split(namesA, ", ")

    sort.Strings(namesC)
    sort.Strings(namesB)

    if len(namesC) == len(namesB) {
        for i := range namesC {
            if namesC[i] != namesB[i] {
                return
            }
        }
        fmt.Println("Ils sont égaux.")
    }
}
```

Exercise 3

```
package main

import (
    "fmt"
    "time"
)

func main() {
    // -----
    // Crée des slices vides
    // -----

    // Pizza toppings
    var pizza []string

    // Departure times
    var departures []time.Time

    // Student graduation years
    var grads []int

    // On/off states of lights in a room
    var lights []bool

    // -----
    // Ajoute des éléments aux slices
    // -----
    pizza = append(pizza, "pepperoni", "onions", "extra cheese")

    now := time.Now()
    departures = append(departures,
        now,
        now.Add(time.Hour*24), // 24 hours after `now`
        now.Add(time.Hour*48)) // 48 hours after `now`

    grads = append(grads, 1998, 2005, 2018)

    lights = append(lights, true, false, true)

    // -----
    // Affiche les slices
    // -----

    fmt.Printf("pizza      : %s\n", pizza)
    fmt.Printf("\ngraduations : %d\n", grads)
    fmt.Printf("\ndepartures  : %s\n", departures)
    fmt.Printf("\nlights     : %t\n", lights)
}
```

Exercise 4

```
package main

import "fmt"

func main() {
    var (
        nums []int
        oldCap float64
    )

    // boucle 10 millions de fois
    for len(nums) < 1e7 {
        // obtient la capacité
        c := float64(cap(nums))

        // imprime seulement lorsque la capacité change
        if c == 0 || c != oldCap {
            // imprime également le ratio de croissance : c/oldCap
            fmt.Printf("len:%-15d cap:%-15g growth:%-15.2f\n",
                len(nums), c, c/oldCap)
        }

        // garde une trace de la capacité précédente
        oldCap = c

        // ajoute un élément arbitraire à la tranche
        nums = append(nums, 1)
    }
}
```