Introduction - Variables & Opérateurs arithmétiques

Exercice 1

- Déclarez et affichez une variable de type int.
- Le nom de la variable déclarée devrait être : `height` .

```
package main

func main() {
    // Déclarez la variable `height` ici

    // Affichez la variable `height` ici
}
```

Résultat attendu

```
0
```

Exercice 2

Déclarez puis affichez quatre variables en utilisant l'instruction de déclaration courte.

```
package main

func main() {
    // AJOUTEZ VOS DÉCLARATIONS ICI
    //

    // DÉCOMMENTEZ ENSUITE LE CODE CI-DESSOUS

    // fmt.Println(
    //  "i:", i,
    //  "f:", f,
    //  "s:", s,
    //  "b:", b,
    // )
}
```

Résultat attendu

```
i: 314 f: 3.14 s: Hello b: true
```

Exercice 3

- Déclarez deux variables int : age et yourAge en utilisant la syntaxe de déclaration courte (short declaration) multiple.
- Déclarez ensuite une nouvelle variable float : ratio, et changez la valeur de la variable 'age' à 42. (! Vous devriez utiliser la redéclaration)
- Affichez toutes les variables.

Résultat attendu

```
42 20 3.14
```

```go
package main

func main() {
    // AJOUTEZ VOS DÉCLARATIONS ICI
    //

    // DÉCOMMENTEZ ENSUITE LE CODE CI-DESSOUS

    // fmt.Println(age, yourAge, ratio)
}
```

Exercice 4

- Changez `color` en "orange" et `color2` en "green" en même temps en utilisant une affectation multiple.
- Affichez les variables.

```go
package main

func main() {
    // DÉCOMMENTEZ LE CODE CI-DESSOUS :

    // color, color2 := "red", "blue"

    // Effectuez l'affectation multiple ici pour changer les valeurs de `color` et `color2`.

    // Affichez les variables ici.
}
```

Résultat attendu

```
orange green
```

Exercice 5

Corrigez le code en utilisant une expression de conversion.

```go
package main

func main() {
    // a, b := 10, 5.5
    // Utilisez une expression de conversion pour corriger l'opération ci-dessous.
    // fmt.Println(a + b)
}
```

Résultat attendu

```
15.5
```

Exercice 6

Simplifiez le code (refactorisez).

RESTRICTION Utilisez uniquement les opérations d'incrémentation, de décrémentation et d'affectation.

```
package main

import "fmt"

func main() {
    width, height := 10, 2

    // Simplifiez le code en utilisant uniquement les opérations d'incrémentation, de
décrémentation et d'affectation.
    width++
    width += height
    width--
    width -= height
    width *= 20
    width /= 25
    width %= 5

    fmt.Println(width)
}
```

Résultat attendu

```
3
```

Exercice 7

Calculer le volume d'une sphère, avec un rayon de 10.

FORMULE DU VOLUME D'UNE SPHÈRE https://fr.wikipedia.org/wiki/Volume_d%27une_boule

RESTRICTION Utilisez `math.Pow` pour calculer le volume.

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var radius, vol float64

    // Fixez la valeur du rayon à 10
    radius = 10

    // Calculez le volume de la sphère en utilisant la formule avec math.Pow

    // NE MODIFIEZ PAS CE CODE
    fmt.Printf("radius: %g -> volume: %.2f\n", radius, vol)
}
```

Résultat attendu

radius: 10 -> volume: 4188.79

Exercice 8

Écrivez un programme en Go qui calcule le montant final à payer pour un achat avec une remise et une taxe.

Étapes :

- Déclarez une constante `taxe` avec la valeur **0.08**.
- Déclarez les variables `montantTotal` (type **float64**) et `pourcentageRemise` (type **float64**).
- Demandez à l'utilisateur d'entrer le montant total de l'achat en utilisant `fmt.Print` et `fmt.Scan`.
- Demandez à l'utilisateur d'entrer le pourcentage de remise en utilisant `fmt.Print` et `fmt.Scan`.
- Calculez le montant de la remise en multipliant `montantTotal` par `pourcentageRemise` divisé par 100.
- Calculez le montant final à payer en soustrayant la remise et en ajoutant la taxe au montant total.
- Affichez le montant final à payer en utilisant `fmt.Println`.

Utilisation de `fmt.Scan`

```
package main

import "fmt"

func main() {
    var valeurEntiere int

    fmt.Print("Entrez un nombre entier : ")
    fmt.Scan(&valeurEntiere)

    fmt.Println("Vous avez entré :", valeurEntiere)
}
```

Bonus

Refaite le même programme en utilisant `os.Args` pour lire les arguments de la ligne de commande.

```
package main

import (
    "fmt"
    "os"
    "strconv"
)

func main() {
    // Vérifiez si le bon nombre d'arguments a été fourni
    if len(os.Args) != 3 {
        fmt.Println("Veuillez fournir deux arguments numériques.")
        os.Exit(1)
    }

    // Convertir le premier argument en float64
    premier, err1 := strconv.ParseFloat(os.Args[1], 64)
    if err1 != nil {
        fmt.Printf("Erreur lors de la conversion du premier argument : %v\n", err1)
        os.Exit(1)
    }

    // Convertir le deuxième argument en float64
```

```
    deuxieme, err2 := strconv.ParseFloat(os.Args[2], 64)
    if err2 != nil {
        fmt.Printf("Erreur lors de la conversion du deuxième argument : %v\n", err2)
        os.Exit(1)
    }

    // Affichez les résultats
    fmt.Println("Le premier argument converti en float64 :", premier)
    fmt.Println("Le deuxième argument converti en float64 :", deuxieme)
}
```

Cette application attend deux arguments sur la ligne de commande. Elle les convertit en float64 et les affiche à l'utilisateur. Si la conversion échoue pour l'un des deux arguments, le programme affiche une erreur et se termine.

Exemple d'exécution :

```
$ go run main.go 10 20
Le premier argument converti en float64 : 10
Le deuxième argument converti en float64 : 20
```