III / Operators

Arithmetic operators

!! can’t add an **Int** and a **Double**, you can’t multiply a **Float** and an **Int** !!

couple of test variables:

=> add and subtract

Exemple :

let total = firstScore + secondScore
let difference = firstScore - secondScore

=> multiply and divide

Exemple :

let product = firstScore * secondScore
let divided = firstScore / secondScore

=> remainders after division: % (modulo)

Exemple :

let remainder = 13 % secondScore => return one (the left over of the division)

NB: 4 fits into 13 three times with remainder one

———————————

Complex test variables

Exemple 1 :

let weeks = 465 / 7
print("There are \(weeks) weeks until the event.") => weeks = 66

———————————

Exemple 2 :

let weeks: Double = 465 / 7
print("There are \(weeks) weeks until the event.") => weeks = 66.42857142857143

———————————

With remainder operator

Exemple:

let weeks = 465 / 7
let days = 465 % 7
print("There are \(weeks) weeks and \(days) days until the event.")

=> weeks = 66 and days = 3

———————————

With a method is Multiple() does this number divide equally into some other number?

Exemple :

let number = 465
let isMultiple = number.isMultiple(of: 7) => return false

Total score: 6/6 checked

———————————————————————————————————

———————————

Operator overloading

=> Joins strings

Exemple :

let fakers = "Fakers gonna "
let action = fakers + "fake"

=> to join arrays

Exemple :

let firstHalf = ["John", "Paul"]
let secondHalf = ["George", "Ringo"]
let beatles = firstHalf + secondHalf

Total score: 12/12 checked

———————————————————————————————————————
———————————

Compound assignment operators

=> shorthand operators

Exemple :

var score = 95
score -= 5

var quote = "The rain in Spain falls mainly on the "
quote += "Spaniards"

Total score: 6/6 checked

———————————————————————————————————————
———————————

Comparison operators

=> operators that perform comparison

Exemple :

```
let firstScore = 6
let secondScore = 4

firstScore == secondScore => equality => false
firstScore != secondScore => not equals => true

firstScore < secondScore => less than => false
firstScore >= secondScore => greater than => true
```

_____

=> work with strings

Exemple :

```
"Taylor" <= "Swift" => false
```

_____

=> compare many kinds of values out of the box

Exemple:

```
let firstName = "Paul"
let secondName = "Sophie"

let firstAge = 40
let secondAge = 10

print(firstName == secondName)
print(firstName != secondName)
print(firstName < secondName)
print(firstName >= secondName)

print(firstAge == secondAge)
print(firstAge != secondAge)
print(firstAge < secondAge)
print(firstAge >= secondAge)
```

_____

=> make our enums comparable

Exemple :

```
enum Sizes: Comparable {
    case small
    case medium
    case large
}

let first = Sizes.small
let second = Sizes.large
print(first < second) => return true because small comes before large in the enum case list
```

Total score: 6/6 checked