VI / Closures : Part one

Creating basic closures

=> Create a function and assign it to a variable
=> Call that function using that variable

Closures let us wrap up some functionality in a single variable

Exemple :

```
let driving = {
    print("I'm driving in my car")
}
```

=> call driving() as if it were a regular function

```
driving()
```

Reasons why they are used :

1. Running some code after a delay.
2. Running some code after an animation has finished.
3. Running some code when a download has finished.
4. Running some code when a user has selected an option from your menu.

NB: closures put their parameters inside the opening brace

Total score: 12/12 Checked

Accepting parameters in a closure

=> List parameters inside parentheses just after the opening brace, then write in .

Exemple 1 :

```
let driving = { (place: String) in
    print("I'm going to \(place) in my car")
}

driving("London")
```

Exemple 2 :

```
var click = { (button: Int) in
        if button >= 0 {
                print("Button \(button) was clicked.")
        } else {
                print("That button doesn't exist.")
        }
}
```

!! Differences between functions and closures : non labels when running closures !!

Total score: 12/12 Checked

Returning values from a closure

=> Closures can also return values
=> Closure that returns a value rather than printing the message directly, need to use -> returning value before in

Exemple 1 :

```
let drivingWithReturn = { (place: String) -> String in
    return "I'm going to \(place) in my car"
```

```
}

let message = drivingWithReturn("London")
print(message)
```

Exemple 2 :

```
let payment = { (user: String) -> Bool in
    print("Paying \(user)...")
    return true
}
```

Exemple 3 :

```
var flyDrone = { (hasPermit: Bool) -> Bool in
        if hasPermit {
                print("Let's find somewhere safe!")
                return true
        }
        print("That's against the law.")
        return false
}
```

=> return a value without accepting any parameters

Exemple 4 :

```
let payment = { () -> Bool in
    print("Paying an anonymous person...")
    return true
}
```

Total score: 12/12 checked

Closures as parameters (function + closure)

=> Pass that closure into a function which run inside that function, specify the parameter type as () -> Void

returns **Void =** accepts no parameters

Exemple 1 :

1. Basic Closure

```
let driving = {
    print("I'm driving in my car")
}
```

2. Using the closure with the function as parameter

```
func travel(action: () -> Void) {
    print("I'm getting ready to go.")
    action()
    print("I arrived!")
}
```

3. Declaration

```
travel(action: driving)
```

Exemple 2 :

```
let awesomeTalk = {
        print("Here's a great talk!")
}
func deliverTalk(name: String, type: () -> Void) {
        print("My talk is called \(name)")
        type()
}
deliverTalk(name: "My Awesome Talk", type: awesomeTalk)
```

Exemple 3 :

```
let helicopterTravel = {
	print("Get to the chopper!")
}
func travel(by travelMeans: () -> Void) {
	print("Let's go on vacation...")
	travelMeans()
}
travel(by: helicopterTravel) ————> added closure helicopterTravel
```

_____

Exemple 4 :

```
var goOnBike = {
	print("I'll take my bicycle.")
}
func race(using vehicleType: () -> Void) {
	print("Let's race!")
	vehicleType()
}
```

NB : Using a closure to send back data rather than returning a value from the function means it doesn’t need to wait for the function to complete, so it can keep its user interface interactive – it won’t freeze up.

Total score: 12/12 checked

_____

_____

Trailing closure syntax

= > If the last parameter to a function is a closure (return nothing), Swift lets you use special syntax called *trailing closure syntax*

=> Rather than pass in the closure as a parameter, you pass it directly after the function

Exemple 1 : (decomposed in 3 ways)

1.

```
func travel(action: () -> Void) {
    print("I'm getting ready to go.")
    action()
    print("I arrived!")
}
```

2.Because its last parameter is a closure (return nothing, using trailing closure syntax like this:

```
travel() {
    print("I'm driving in my car")
}
```

3.Because there aren’t any other parameters, eliminate the parentheses entirely:

```
travel {
    print("I'm driving in my car")
}
```

Exemple 2 : (decomposed in 3 ways)

1.

```
func animate(duration: Double, animations: () -> Void) {
    print("Starting a \(duration) second animation...")
    animations()
}
```

2.

```
animate(duration: 3, animations: {
    print("Fade out the image")
})
```

3.

```
animate(duration: 3) {
    print("Fade out the image")
```

```
}
```

Exemple 3 :

```
func goOnVacation(to destination: String, _ activities: () -> Void) {
        print("Packing bags...")
        print("Getting on plane to \(destination)...")
        activities() ————————————————————> can add a closure
        print("Time to go home!")
}
goOnVacation(to: "Mexico") {
        print("Go sightseeing")
        print("Relax in sun")
        print("Go hiking")
}
```

NB: Trailing closure syntax is designed to make Swift code easier to read

Total score: 12/12 checked