# 클로저(Closure)

> 다른언어에서는 익명함수라 부르지만
> Swift 만큼에서는 익명함수라고 사용되지 않는다.

함수는 클로저의 일종으로 이름있는 클로저라고 생각할 수 있다.

## 클로저 문법

```
{ ( Parameter1, Parameter2, ... )-> return Type in
        Excute Code
}
```

클로저 사용법

```
let sum: (Int, Int)-> Int = { (a : Int, b = Int) in
        return a + b
}

let sumResult: Int = sum(1,2)
print(sumResult) // 3
```

-> 클로저를 사용하여 sumResult변수에 할당하면 Return 값이 변수에 담긴다.

## 전달인자로서의 클로저

클로저는 함수의 전달인자로 쓰이기도 한다.
즉 함수에 함수를 전달해줄 수 있다.

```
let add: (Int, Int) -> Int
add = { (a: Int, b: Int) in
    return a + b
}
let substract: (Int, Int) -> Int
substract = { (a: Int, b: Int) in
    return a - b
}
```

```swift
let divide: (Int, Int) -> Int
divide = { (a: Int, b: Int) in
    return a / b
}
// calculate 함수에서는 method parameter를 사용하여 클로저는 입력받고 있다.
func calculate(a: Int, b: Int, method: (Int, Int) -> Int) -> Int {
    return method(a, b)
}
var calculated: Int
calculated = calculate(a: 50, b: 10, method: add)
print(calculated) // 60
calculated = calculate(a: 50, b: 10, method: substract)
print(calculated) // 40
calculated = calculate(a: 50, b: 10, method: divide)
print(calculated) // 5
//따로 클로저를 상수/변수에 넣어 전달하지 않고,
//함수를 호출할 때 클로저를 작성하여 전달할 수도 있습니다.
calculated = calculate(a: 50, b: 10, method: { (left: Int, right: Int) ->
Int in
    return left * right
})
print(calculated) // 500
```