functions → Tuple type : returning multiple value.

```
class Main {
    func findMinMaxValue(arr: [Int]) -> (minVal:Int, maxVal:Int) {
        if(arr.isEmpty) {
            return (0,0)
        }

        var currMax = arr[0];
        var currMin = arr[0];

        for val in arr[1..<arr.count] {
            if val < currMin {
            currMin = val
        } else if val > currMax {
            currMax = val
        }
    }

    return (currMin, currMax)

    }
}

let objMain =  Main()
var array:[Int] = [14,112,13,10,78]
 let result = objMain.findMinMaxValue(arr: array)
print(result.minVal)
print(result.maxVal)
```

→ Multiple values are returned.

} — Printing them.

Optional and Required type in swift.

```swift
class Person {
    var firstName: String = String()
    var middleName: String?
    var lastName: String = String()
}
```

Required type → `var firstName: String = String()`

default value or " ".

optional type → `var middleName: String?`

Note:- Required type variables need a default value.

* what is optionals → Basically you can define a variable saying this is optional i.e. not required.

```swift
7  */
8  class Person {
9      var firstName: String = String()
10     var middleName: String?
11     var lastName: String = String()
12
13     func printPersonInfo() -> () {
14         print("\(firstName) , \(middleName!) , \(lastName)") // force exec
15     }
16 }
17
18 let objPerson = Person()
19
20 objPerson.firstName = "Clark"
21 objPerson.lastName = "Kent"
22 objPerson.printPersonInfo()
```

force execution

```
input
main/main.swift:14: Fatal error: Unexpectedly found nil while unwrapping an
Optional value
```

Here, we are getting error because we are enforcing swift to read this variable no matter there is any value inside in it or not.

```
20  objPerson.firstName = "Clark"
21  objPerson.middleName = "Super"
22  objPerson.lastName = "Kent"
23  objPerson.printPersonInfo()
```

→ after providing middle name NO ERROR. ☺

```
Clark , Super , Kent
```

Now, what to do for Handling if NO middlename is provided —

```
8  class Person {
9      var firstName: String = String()
10     var middleName: String?
11     var lastName: String = String()
12
13     func printPersonInfo() -> () {
14         if(middleName != nil){
15             print("\(firstName) , \(middleName!) , \(lastName)")
16         }else{
17             print("\(firstName) , \(lastName)") // force execution
18         }
19     }
20 }
21 let objPerson = Person()
22 objPerson.firstName = "Clark"
23 objPerson.lastName = "Kent"
24 objPerson.printPersonInfo()
```

This is a one way to do it.

```
Clark , Kent
```

Swift call this UNWRAPPPING optionals

```swift
class Person {
    var firstName: String = String()
    var middleName: String?
    var lastName: String = String()

    func printPersonInfo() -> () {

        if let midName = middleName {
            print("\(firstName) , \(midName) , \(lastName)")
        }else{
            print("\(firstName) , \(lastName)")
        }
    }
}
let objPerson = Person()
objPerson.firstName = "Clark"
objPerson.lastName = "Kent"
objPerson.printPersonInfo()
```

unwrapping the middlename.

using if let

input

```
Clark , Kent
```

```swift
class Person {
    var firstName: String = String()
    var middleName: String?
    var lastName: String = String()

    func printPersonInfo() -> () {

        guard let midName = middleName else {
            // if middleName is nil, early exit from this block
            print("\(firstName) , \(lastName)")
            return
        }
        // if middleName is not nil, use midName safely here
        print("\(firstName) , \(midName) , \(lastName)")
    }
}
let objPerson = Person()
objPerson.firstName = "Clark"
objPerson.lastName = "Kent"
objPerson.printPersonInfo()
```

doing same

returning if no middlename

using guard let

input

```
Clark , Kent
```

```swift
class Person {
    var firstName: String = String()
    var middleName: String?
    var lastName: String = String()

    func printPersonInfo() {
        // Optional chaining -> safely check length of middleName
        let midLength = middleName?.count
        print("Middle name length: \(midLength ?? 0)")
        // use ?? to give default if nil

        // Nil-coalescing -> provide "No Middle Name" if nil
        let midName = middleName ?? "No Middle Name"

        print("\(firstName) , \(midName) , \(lastName)")
    }
}
let objPerson = Person()
objPerson.firstName = "Clark"
objPerson.middleName="superman"
objPerson.lastName = "Kent"
objPerson.printPersonInfo()
```


input

```
Middle name length: 8
Clark , superman , Kent
```

CLOSURES

* Functions are First-class citizens.

that means we can assign a function to a variable.

example:

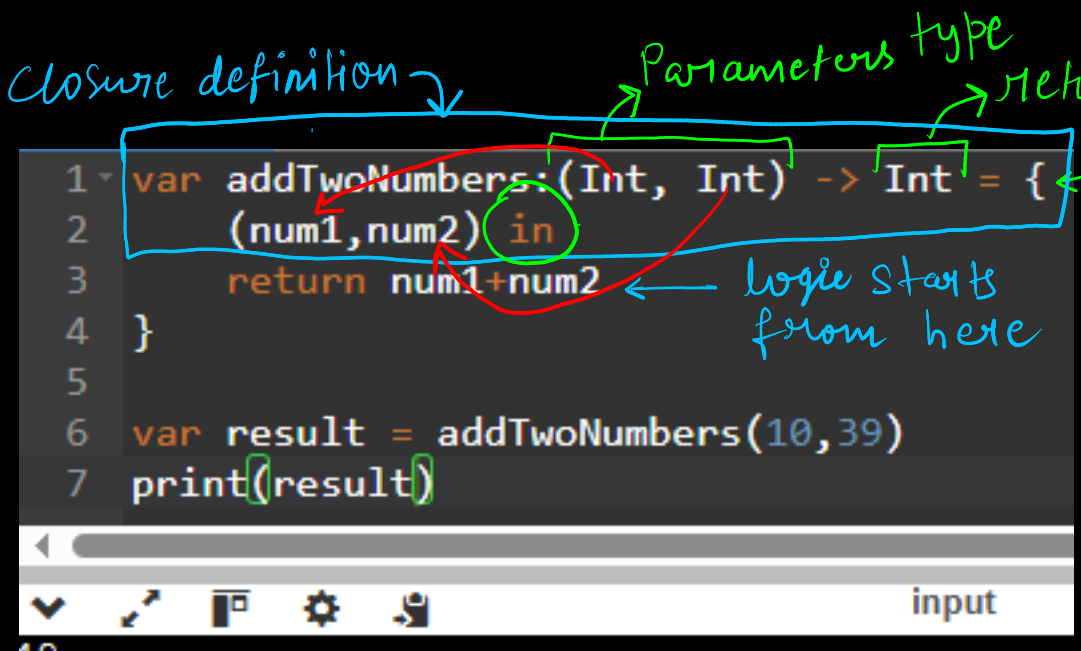
```
1  func addTwoNumbers(num1: Int, num2: Int) -> Int {
2      return num1+num2
3  }
4  var result = addTwoNumbers(num1: 10, num2: 10)
5  var addNumberVar = addTwoNumbers
6  var resultVar = addNumberVar(10,20)
7  print(result)
8  print(resultVar)
```

input

```
20
30
```

→ Here a func. is assigned to a variable.

SYNTAX



when to use closures ??