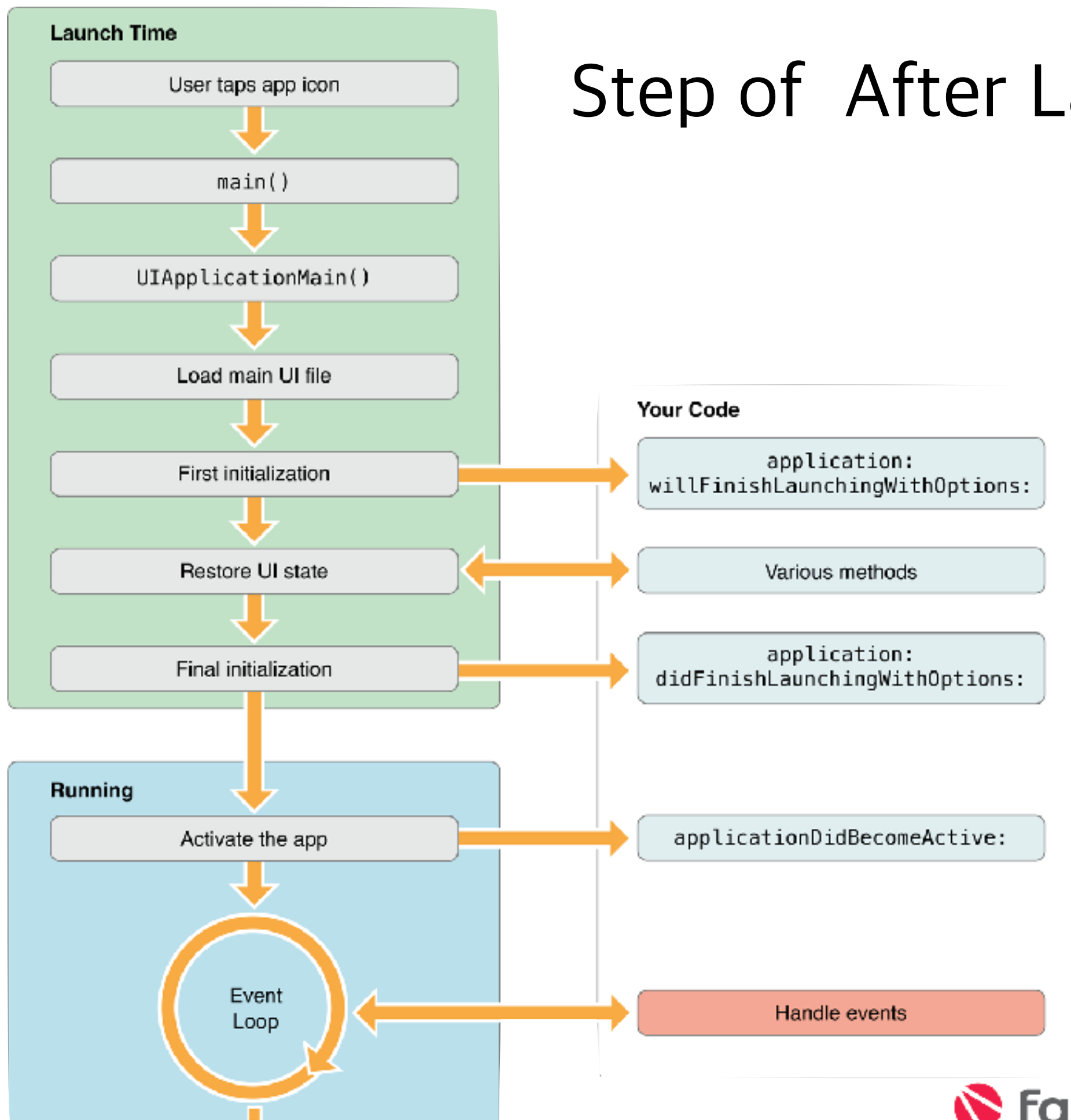


Step of After Launch



변수

Swift Class Architecture

```
class ClassName : superClass
{
    var vName1 = "1"
    var vName2 = 4

    func fName1() - > Any
    {

    }

    func fName2(_ ani:Bool)
    {

    }
}
```

<ClassName.swift>

변수 & 함수

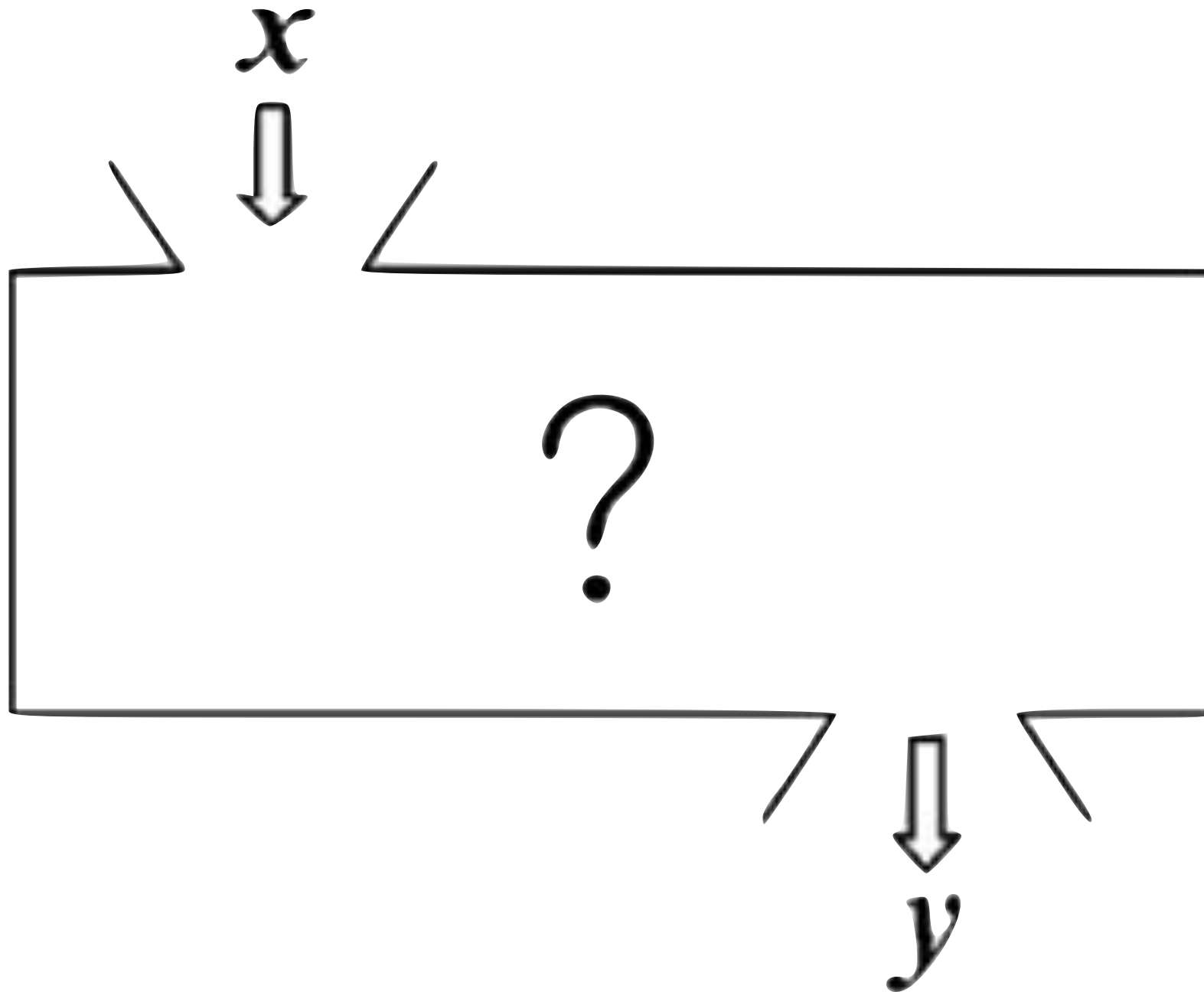
- 변수 : 프로그램에서 데이터의 저장공간을 담당
- 함수 : 프로그램이 실행되는 행동을 담당

- 변수를 만드는데 있어 필요한 것은?

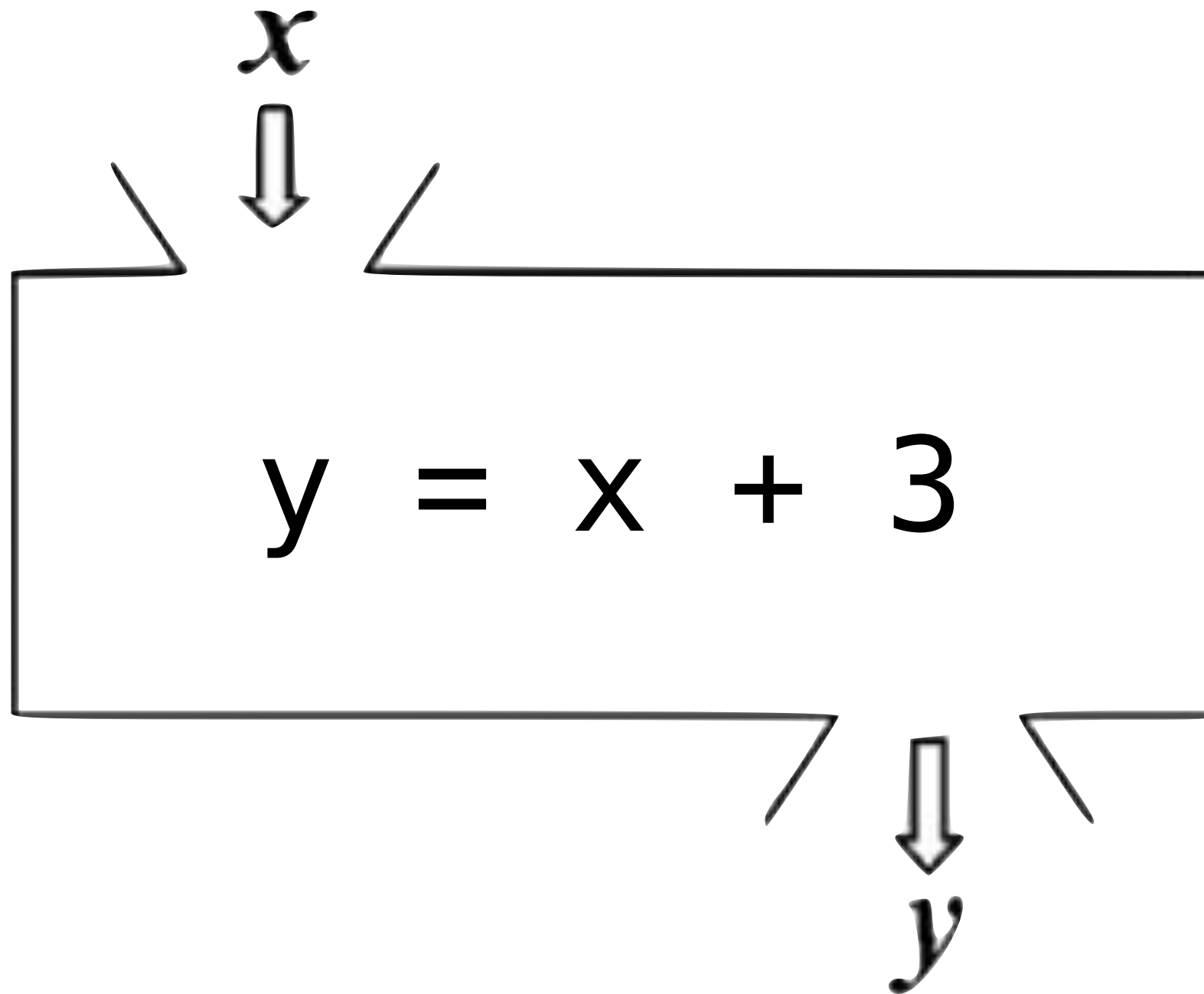
키워드 + 변수 명(Name) + 변수 타입(Type)

문법 : `var vName:Any`

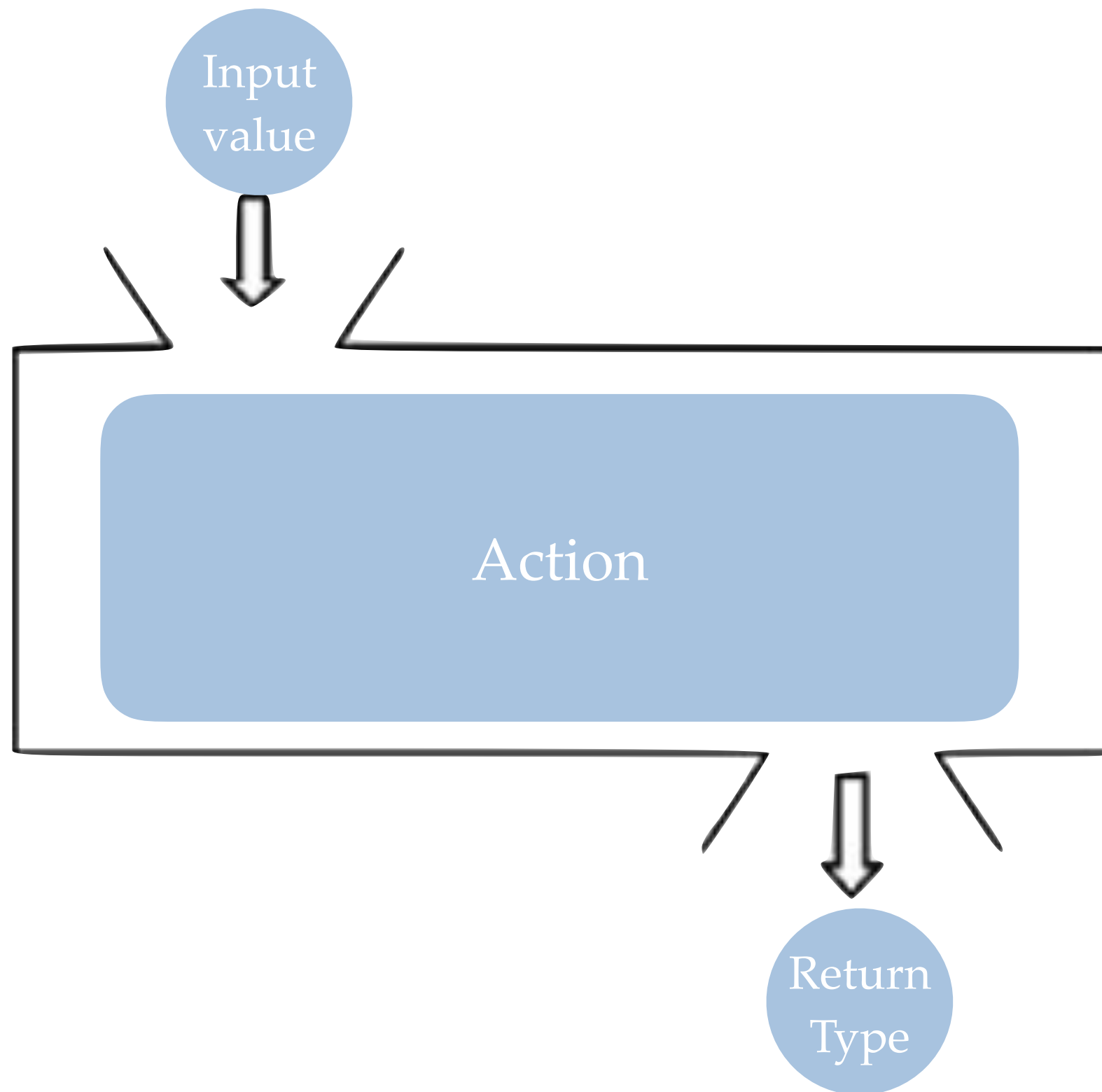
함수



함수



함수



- 함수 만들기 위해 필요한것?

키워드 + 함수명(Name) + 입력값(Input Value) +
함수 내용(Action) + 결과타입(Return Type)

문법 : `func vName(_ parameter: Any) -> Any`
`{`
`//함수 내용`
`}`

Swift 문법 - 변수

키워드 변수타입

`var` `name` `:` `Type` `=` `value`

변수명 값

키워드

- 변수 : 변할수 있는 값

```
var name:String = "joo"
```

- 상수 : 변할수 없는 고정 값

```
let name:String = "joo"
```

키워드

- 변수 : 변할수 있는 값

```
var name:String = "joo"  
name = "iOS개발 스쿨" ———— O
```

- 상수 : 변할수 없는 고정값

```
let name:String = "joo"  
name = "iOS개발 스쿨" ———— X
```

변수명

- 명명규칙에 따라 작성
- 유니 코드 문자를 포함한 거의 모든 문자가 포함될 수 있다.(한글 가능)
- 변수안에 들어있는 데이터를 표현해 주는 이름으로 작성
- 중복작성 불가 (한 클래스, 함수, 구문 안에서)

명명규칙

- 시스템 예약어는 사용할 수 없다.
- 숫자는 이름으로 시작될 수는 없지만 이름에 포함될 수 있다.
- 공백을 포함 할 수 없다.
- 변수 & 함수명을 lowerCamelCase,
클래스 명은 UpperCamelCase로 작성한다.

변수 타입

기본형

타입이름	타입	설명	Swift 문법 예제
정수	Int	1, 2, 3, 10, 100	var intName: Int
실수	Double	1.1, 2.35, 3.2	var doubleName: Double
문자열	String	"this is string"	var stringName: String
불리언	Bool	true or false	var boolName: Bool

참조형

타입이름	타입	설명	Swift 문법 예제
Custom Type	ClassName	클래스 객체를 다른곳에서 사용할 경우	let customView: UIView
			let timer: Timer

Int & Uint

- 정수형 타입 (Integer)
- Int : +/- 부호를 포함한 정수이다.
- Uint : - 부호를 포함하지 않은(0은 포함) 정수
- 최대값과 최소값은 max, min프로퍼티를 통해 알아볼수 있다.
- Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64의 타입으로 나뉘져 있는데 시스템 아키텍처에 따라서 달라진다.
- 접두어에 따라 진수를 표현할수 있다. (2진법 0b, 8진법0o, 16진법 0x)

Bool

- 불리언 타입 (true, false)

Float & Double

- 부동 소수점을 사용하는 실수형 타입
- 64비트의 부동소수점은 Double, 32비트 부동 소수점은 Float으로 표현한다.
- Double은 15자리,Float은 6자리의 숫자를 표현가능
- 상황에 맞는 타입을 사용하는것이 좋으나 불확실할때는 Double을 사용하는 것을 권장.

Character

- 단어나 문장이 아닌 문자 하나!
- 스위프트는 유니코드 문자를 사용함으로, 영어는 물론, 유니코드 지원 언어, 특수기호등을 모두 사용 할 수 있다.
- 문자를 표현하기 위해서는 앞뒤에 쌍 따옴표(“ ”)를 붙여야 한다.

String

- 문자의 나열, 문자열이라고 한다.
- Character와 마찬가지로 유니코드로 이뤄져 있다.
- 문자열을 다루기 위한 다양한 기능이 제공된다.
(hasPrefix, uppercased, isEmpty등)

String 조합

1. string 병합: + 기호를 사용

```
var name:String  
name = "주" + "영민"
```

2. interpolation(삽입): \ (참조값)

```
var name:String = "주영민"  
print("my name is \ (name) ")
```

\ ()가 interpolation

튜플

- 정해지지 않은 데이터 타입의 묶음
- 소괄호 () 안에 타입을 묶음으로 새로운 튜플타입을 만들수 있다. ex) (Int, Int) // (String, Int, String)
- 각 타입마다 이름을 지정해 줄수도 있다.
ex) (name:String, age:Int)

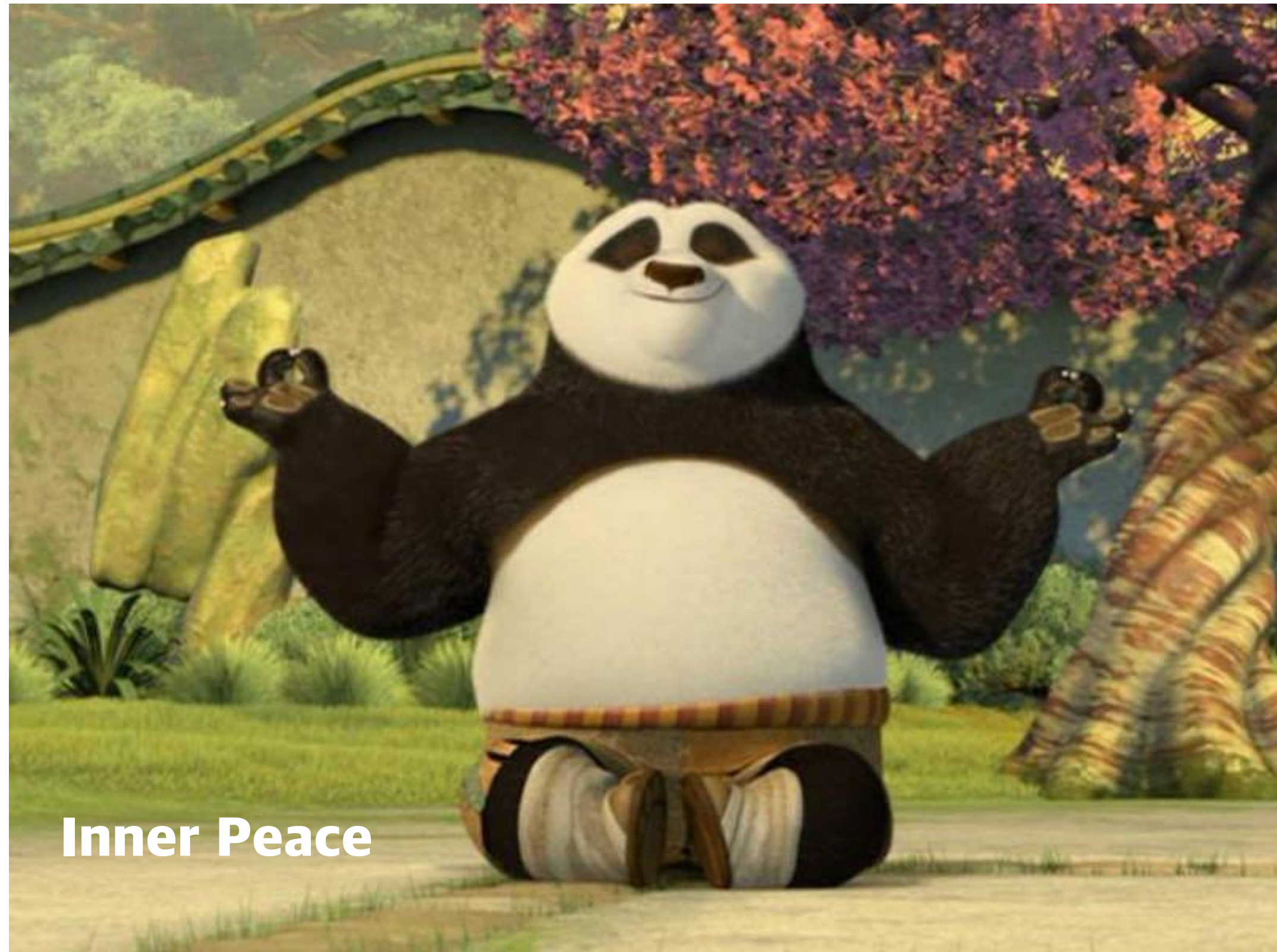
튜플 예시

```
var coin:(Int,Int,Int,Int) = (3,1,5,3)
print("10원짜리 : \" + coin.0)
print("50원짜리 : \" + coin.1)
print("100원짜리 : \" + coin.2)
print("500원짜리 : \" + coin.3)
```

```
var person:(name:String, age:Int, weight:Double)
           = ("joo", 30, 180.2)
print("이름 : \" + person.name)
print("나이 : \" + person.age)
print("몸무게 : \" + person.age)
```

Any, AnyObject, nil

- Any : 스위프트 내의 모든 타입을 나타냄
- AnyObject : 스위프트 내의 모든 객체 타입을 나타낸다.(클래스)
- nil : 데이터가 없음 을 나타내는 키워드



Inner Peace

캐스팅(형변환)

```
var total:Int = 107
```

```
var average:Double
```

```
average = total/5
```

← type Error

캐스팅을 해야하는 이유

실수 : 107.0

1	1	0	0	1	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

정수 : 107

1	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

캐스팅(형변환)

```
var total:Int = 107
```

```
var average:Double
```

```
average = total/5 ← type Error
```

```
average = Double(total)/5 ← casting
```


캐스팅(형변환)

```
var stringNum:String  
var doubleNum:Double  
let intNum:Int = 3
```


```
stringNum = String(intNum) ← int to string  
doubleNum = Double(intNum) ← int to double
```

변수 값 지정

`var number: Int = 3`



`var age = 31` (타입 추론)



대입연산자	예제	설명
=	number = 4	number변수에 숫자 4를 넣는다.

접근 제어

접근 수준

- 외무 모듈에서의 접근을 제어하는 수단.
- 캡슐화, 은닉화를 위해 사용

모듈 & 소스파일

- 모듈 : 배포할 코드의 묶음 단위, 통상 프레임워크나 라이브러리, 어플리케이션이 모듈의 단위가 될수 있다.
- 소스파일 : 하나의 스위프트 소스코드 파일

접근제어

- Open (개방 접근수준) : 모듈 외부까지 접근 가능
- public (공개 접근수준) : 모듈 외부까지 접근 가능
- internal (내부 접근수준) : 모듈 내부에서 접근가능, 기본 지정값
- fileprivate (파일외 비공개) : 파일 내부에서만 접근가능
- private (비공개) : 기능 정의 내부에서만 가능

Open VS Public

- Open을 제외한 다른 모든 접근수준의 클래스는 그 클래스가 정의된 모듈 안에서만 상속될 수 있다.
- Open을 제외한 다른 모든 접근수준의 클래스 멤버는 그 멤버가 정의된 모듈 안에서만 재정의 될 수 있다.
- Open 수준의 클래스는 그 클래스가 정의된 모듈 밖의 다른 모듈에서도 상속되고, 재정의 될수 있다.
- 클래스를 Open으로 명시하는 것은 그 클래스를 다른 모듈에서도 부모클래스로 사용할수 있다는 얘기

예시

```
public class SomePublicClass {  
    public var somePublicProperty = 0  
    var someInternalProperty = 0  
    fileprivate func someFilePrivateMethod() {}  
    private func somePrivateMethod() {}  
}
```

```
class SomeInternalClass {  
    var someInternalProperty = 0  
    fileprivate func someFilePrivateMethod() {}  
    private func somePrivateMethod() {}  
}
```

```
fileprivate class SomeFilePrivateClass {  
    func someFilePrivateMethod() {}  
    private func somePrivateMethod() {}  
}
```

```
private class SomePrivateClass {  
    func somePrivateMethod() {}  
}
```

접근수준 확인하기

- Test클래스 생성
- Public, internal, fileprivate, private 접근 수준을 포함한 메소드 만들기
- 각각의 상황에서 메소드 호출 해보기

함수

Swift 문법 - 함수

```
func fName(agumentName paramName:Int) -> Int
{
    return paramName + 3
}
```

Swift 문법 - 함수

키워드 인수명 매개변수명 반환타입

함수 이름 매개변수타입

```
func fName(argumentName paramName: Int) -> Int  
{  
    return paramName + 3  
}
```

함수 내용

The diagram illustrates the syntax of a Swift function. The code is: `func fName(argumentName paramName: Int) -> Int { return paramName + 3 }`. Annotations in blue text identify parts of the code: '키워드' (keyword) points to 'func'; '인수명' (argument name) points to 'argumentName'; '매개변수명' (parameter name) points to 'paramName'; '매개변수타입' (parameter type) points to 'Int' in the parameter list; '반환타입' (return type) points to 'Int' after the arrow; '함수 이름' (function name) points to 'fName'; and '함수 내용' (function body) points to the code inside the curly braces. Red circles highlight the keyword 'func', the function name 'fName', the parameter name 'paramName', the parameter type 'Int', the return type 'Int', and the opening and closing curly braces of the function body.

Argument Labels and Parameter Names

인수레이블 명

매개변수명

매개변수타입

```
func fName(argumentName paramName: Int) -> Int  
{  
    return paramName + 3  
}
```

- 인수레이블은 함수 호출시 사용 되는 이름표.
- 매개변수는 함수 내부에서 사용 되는 변수명
- 인수레이블은 생략가능하며 없을때는 매개변수명이 인수레이블로 사용된다.

Default Parameter Values

```
func number(num1:Int, num2:Int = 10) -> Int {  
    return num1 + num2  
}
```

```
number(num1: 10)           ← 20  
number(num1: 10, num2: 5) ← 15
```

- 매개변수에는 기본값을 설정할수 있다.
- 기본값은 인자로 값이 들어오지 않을때 사용된다.

In-Out Parameter Keyword

inout Keyword

```
func swapTwoInts(_ a: inout Int, _ b: inout Int) {  
    let temporaryA = a  
    a = b  
    b = temporaryA  
}
```

- 매개변수는 기본 상수값이다.
- 만약 매개변수의 값을 변경해야 한다면 inout 키워드를 사용하여 inout 변수로 지정해야만 한다.
- inout 변수 지정은 타입 앞에 inout keyword를 작성해준다.
- inout 변수가 지정된 함수의 인수앞에서 & 가 붙어야 한다.

In-Out Parameter Keyword

```
func swapTwoInts(_ a: inout Int, _ b: inout Int) {  
    let temporaryA = a  
    a = b  
    b = temporaryA  
}
```

```
var someInt = 3  
var anotherInt = 107  
swapTwoInts(&someInt, &anotherInt)
```



```
swapTwoInts(3, 107)  
swapTwoInts(&3, &107)
```



여러가지 함수 - 매개변수

```
func getNumber(firstNum num1:Int) -> Int {  
    return num1  
}
```

```
func getNumber(num1:Int) -> Int {  
    return num1  
}
```

```
func getNumber() -> Int {  
    var num1:Int = 22  
    return num1  
}
```

```
func getNumber(firstNum num1:Int, secondNum num2:Int) -> Int {  
    return num1 + num2  
}
```

```
func sumNumber(num1:Int, num2:Int = 5) -> Int {  
    return num1 + num2  
}
```

반환타입

```
func fName(agumentName paramName: Int) -> Int
{
    return paramName + 3
}
```

반환타입

- 함수 실행 결과의 타입을 명시 해준다. (Return Type)
- **return** 키워드를 사용하여 함수 결과 반환.
반환 타입과 같은 타입의 데이터를 반환 해야 한다.
- 한개의 값만 반환 할수 있다.
- 반환값이 없는 경우는 Return Type을 작성하지 않고(-> 제거)
return 키워드를 사용할 필요가 없다.(반환값이 없기때문)

반환타입 예제

```
func printName() -> String{  
    return "my name is youngmin"  
}
```

```
func printName(){  
    print("my name is youngmin")  
}
```

```
func printName(name:String = "youngmin"){  
    print("my name is \(name)")  
}
```

```
func printName(explain str:String, name str2:String) -> String{  
    return str + str2  
}
```

```
func printName(explain str: inout String) -> String{  
    str += "joo"  
    return str  
}
```

Step 3. 버튼 액션

