
변수 & 함수

Swift Class Architecture

*오디에나
가장유용한
언어!*

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

    func fName1() - > Any
    {

    }

    func fName2(_ ani:Bool)
    {

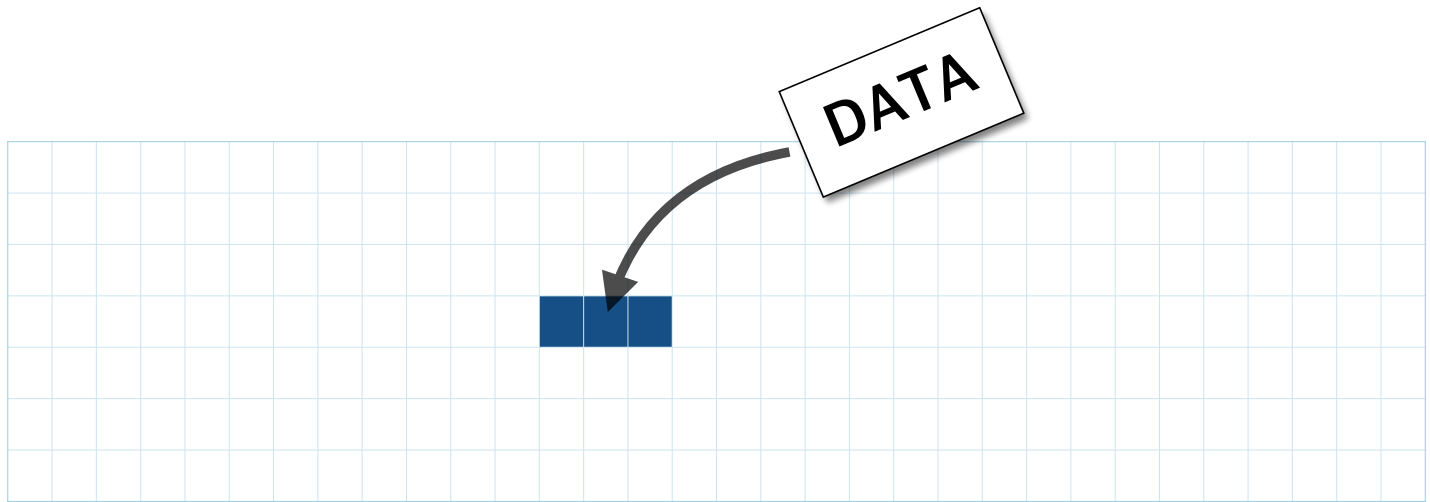
    }
}
```

<CalssName.swift>

변수 & 함수

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

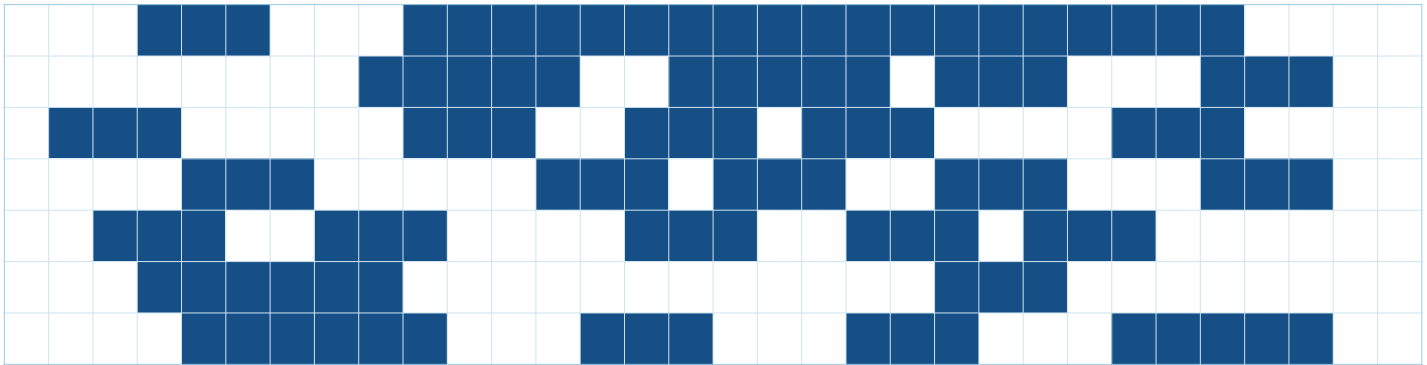
변수



<메모리>

각 메모리 안에는 어떤 데이터가 들어있을까요?
조금 전 넣은 데이터는 어디 일까요?

약속 있음.



<메모리>

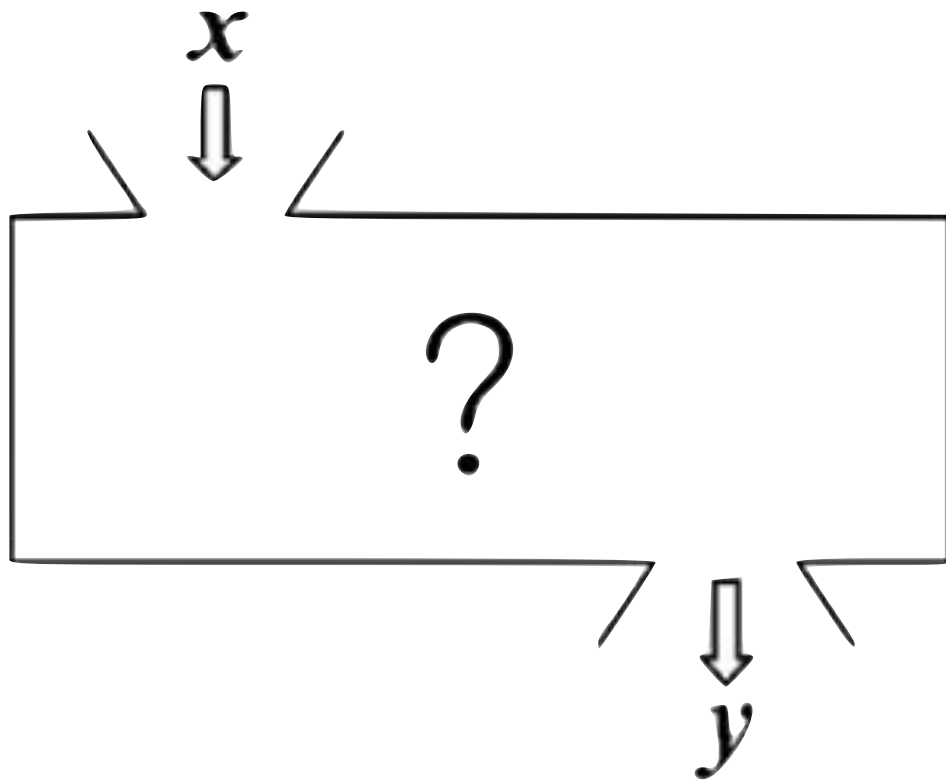
- 변수를 만드는데 있어 필요한 것은?

키워드 + 변수 명(Name) + 변수 타입(Type)

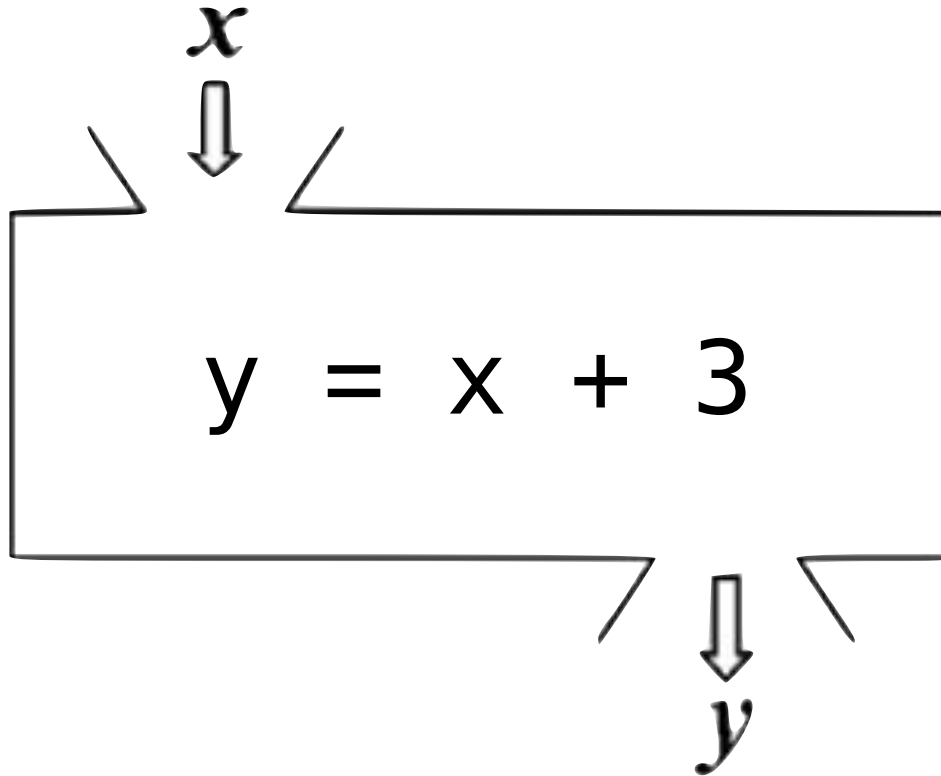
문법 : `var` vName:Any

↓ variable의
약자.

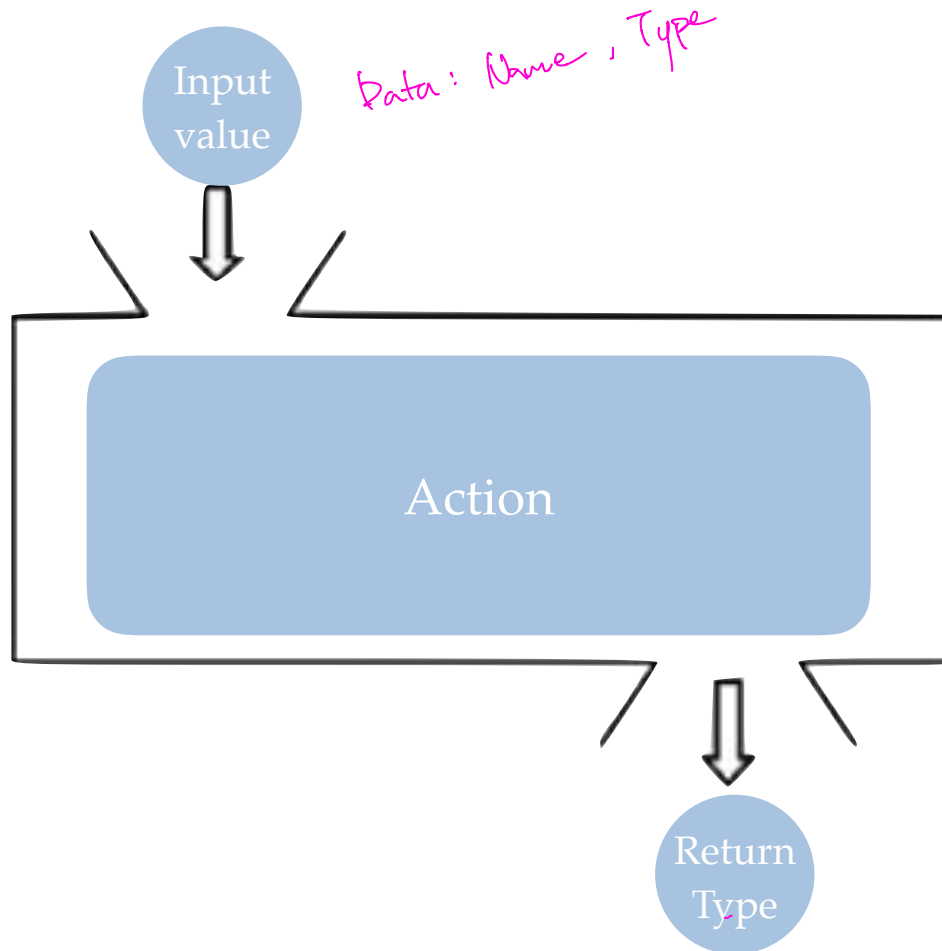
함수



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
 }

정리 해보아요

- 변수 만들기 위해 필요한것?

키워드 + 변수 명(Name) + 변수 타입(Type)

- 함수 만들기 위해 필요한것?

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



Inner Peace

Swift 문법 - 변수

키워드 변수타입

var **name** **:** **Type** = **value**

변수명 값

∴ name 이라는 것은 "Type" 이라는 타입의 피아일고
value 는 name 에 넣는다.

다양한 형태의 변수 (일단 보고 가실게요)

//일반 변수 선언

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

//변수 값 재정의

```
var number:Int = 50  
number = 100
```

Int : integer 정수

→ Java 의 final

//상수 선언

```
let PI = 3.14
```

↳ constants

//옵셔널 변수 선언 (나중에 배울꺼예요)

```
var address:String?  
address = "서울시 신사동"
```

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
```

print(name) → iOS 개발 스쿨.

- 상수 : 변할수 없는 고정 값

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

print(name) → error !

변수명

변수 & 상수 모두 이름 변경 X.

- 명명규칙에 따라 작성
- 유니 코드 문자를 포함한 거의 모든 문자가 포함될 수 있다.(한글 가능) + emoji, 한자 등등.
한글 24성시 pros. 알아보기 좋다.
cons. 자동완성 기능이 느려짐. Objective-C와 혼란
클로변을 지칭하지 않음. 어려워
- 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.

← camel casing

↑ Pascal Casing .

변수 타입

기본형 . primitive type 과 non primitive type 으로 나눌 수 있다.

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

참조형 ?

타입 이름	타입	설명	Swift 문법 예제
Custom Type	ClassName	클래스 객체를 다른곳에서 사용할 경우	<code>let customView: UIView</code>
			<code>let timer: Timer</code>

Int & Uint

- 정수형 타입 (Integer)

- Int : +/- 부호를 포함한 정수이다.

↳ integer

-9 ~ +9

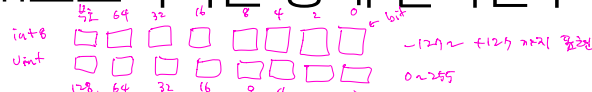
- Uint : - 부호를 포함하지 않은(0은 포함) 정수

↳ Unsigned integer

0 ~ 9

- 최대값과 최소값은 max, min프로퍼티를 통해 알아볼수 있다.

- Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64의 타입으로 나뉘어져 있는데 시스템 아키텍처에 따라서 달라진다.



for performance.
메모리 할당을 효율적으로
하기 위함.

- 접두어에 따라 진수를 표현할수 있다. (2진법 0b, 8진법 0o, 16진법 0x)

윈도우 32 bit, 윈도우 64 bit와 같음. 아이폰 4 부터 64bit → int 를 사용하면 자동적으로 UInt64 사용.

Bool

- 불리언 타입 (true, false)

1 0
Var B: Bool = 1 → true.

Float & Double

- 부동 소수점을 사용하는 실수형 타입
- 64비트의 부동소수점은 Double, 32비트 부동 소수점은 Float으로 표현한다. *아키텍처에 따라 32/64 결정됨.*
- Double은 15자리, Float은 6자리의 숫자를 표현가능
0.1은 어떻게 표현? 0.1에서 $\times 10$ 하고 데이터로 저장.
- 상황에 맞는 타입을 사용하는것이 좋으나 불확실할때는 Double을 사용하는 것을 권장. *일반적으로 64 bit Double은*

• CG Float \rightarrow 자동적으로 비트검정.

*검독어
(22비트
만정)*

Character

- 단어나 문장이 아닌 문자 하나! *flag 등에 사용.*
- 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.

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) ")
```

*in Java .
println("my name is "
+ name)*

\n

*Regular Expression.
etc.*

\ ()가 interpolation

튜플

사용과 지정 타입.

- 정해지지 않은 데이터 타입의 묶음
- 소괄호 () 안에 타입을 묶음으로 새로운 튜플타입을 만들수 있다. ex) (Int, Int) // (String, Int, String) → 타입 배열 순서만 같은 밀치해야 함.
- 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)")
```

4개씩 0, 1, 2, 3

타입이
이름 지정.

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

↓

weight

Any, AnyObject, nil

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



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	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.

다양한 형태의 변수

//일반 변수 선언

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

//변수 값 재정의

```
var number:Int = 50  
number = 100
```

//상수 선언

```
let PI = 3.14
```

//옵셔널 변수 선언(나중에 배울꺼예요)

```
var address:String?  
address = "서울시 신사동"
```

놀이터에서 문법 익히기

다양한 변수를 만들어 봅시다.

이름, 나이, 성별, 학교, 직업, 연봉 등
다른 타입으로 30개의 변수(상수) 작성하기.



Inner Peace

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 above the code identify parts: '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. A bracket below the opening curly brace is labeled '함수 이름' (function name), pointing to 'fName'. A bracket below the closing curly brace is labeled '함수 내용' (function body), pointing to the code inside the braces. Red circles highlight the keywords 'func', 'return', and the types 'Int' and 'Int'.

Argument Labels and Parameter Names

인수레이블 명

매개변수명

매개변수타입

```
func fName(agumentName paramName: Int) -> Int  
{
```

```
    return paramName + 3
```

```
}fName(agumentName: 10) ← 함수호출
```

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

argumentName은 안써도 되지만 매개변수를 설명 할 수 있기 때문에
사용하는게 좋다.

Default Parameter Values

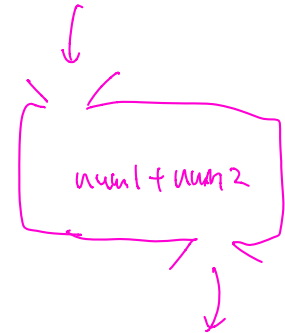
```
func number(num1:Int, num2:Int = 10) -> Int {  
    return num1 + num2  
}
```

number(num1: 10)

number(num1: 10, num2: 5)

← 20

← 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  
}
```

parameter를 통해서 들어온 값은 상수.

- 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

wildCard. : 이름을 쓰지않겠다.

```
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)
```



swap 함수. 값을 바꾸.

여러가지 함수 - 매개변수

argument Name
parameter Name
parameter Type

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

기반형.
Return type.

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

argument X

```
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  
}
```

Default X.

공수는 class 내에서 behavior 를 정의하는 것.



Inner Peace

반환타입

반환타입

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

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

반환타입 예제

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

반환값이 있으면
변수에 넣어야 함.

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

반환값 X

프린트

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

자

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

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

함수 꾸미기

- 단순 명령어의 순서가 아닌 복잡한 명령을 내리기 위해선 프로그램 명령어 컨트롤 방법(Controll Flow)이 필요하다.
- 조건문(while, for-in) & 선택문(if, guard, switch)을 통해 함수를 컨트롤 할수 있다.