
변수 & 함수

Review

Swift Class Architecture

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

    func fName1() - > Any
    {

    }

    func fName2(_ ani:Bool)
    {

    }
}
```

<CalssName.swift>

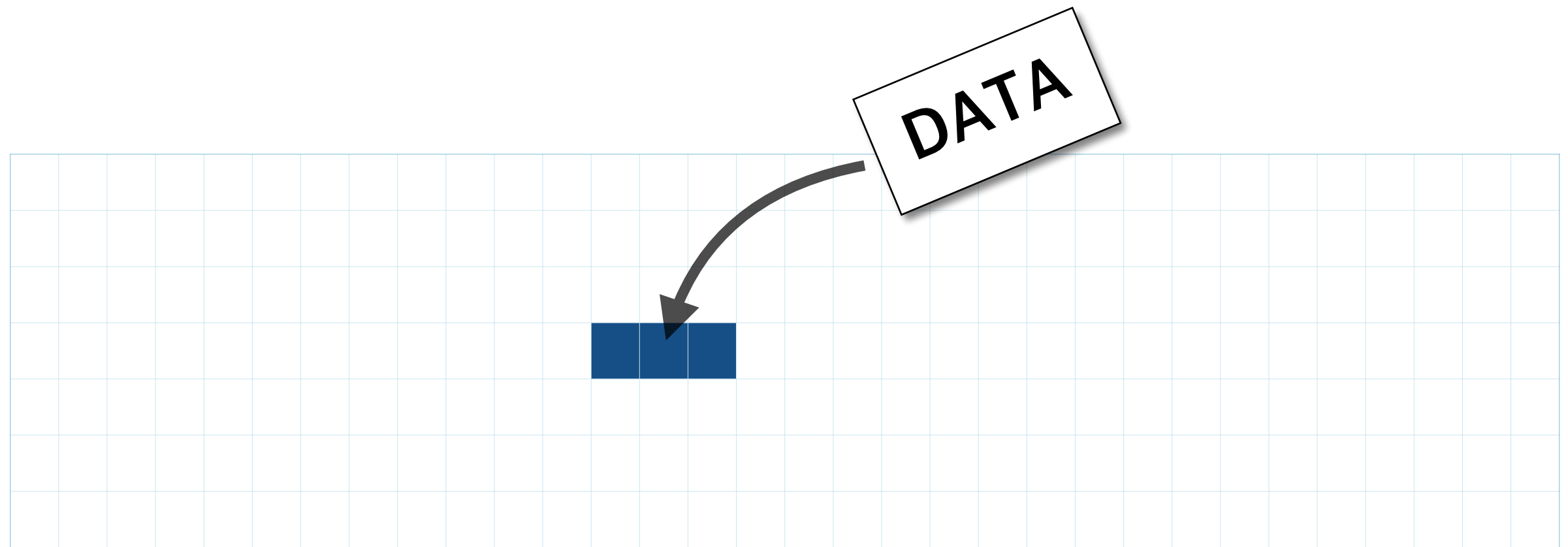
[Review](#)

변수 & 함수

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

Review

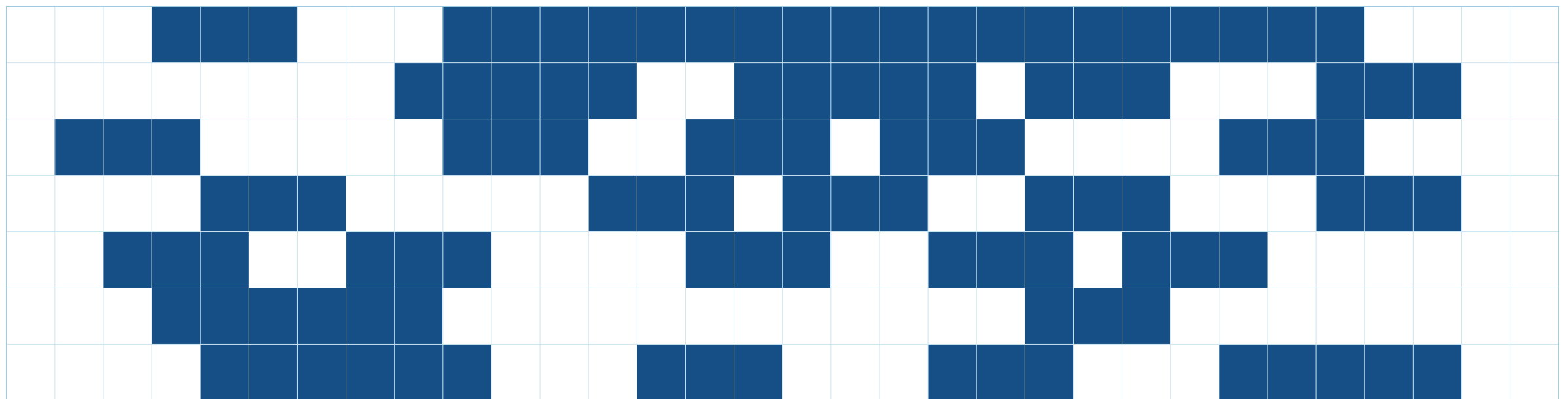
변수



<메모리>

Review

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



<메모리>

Review

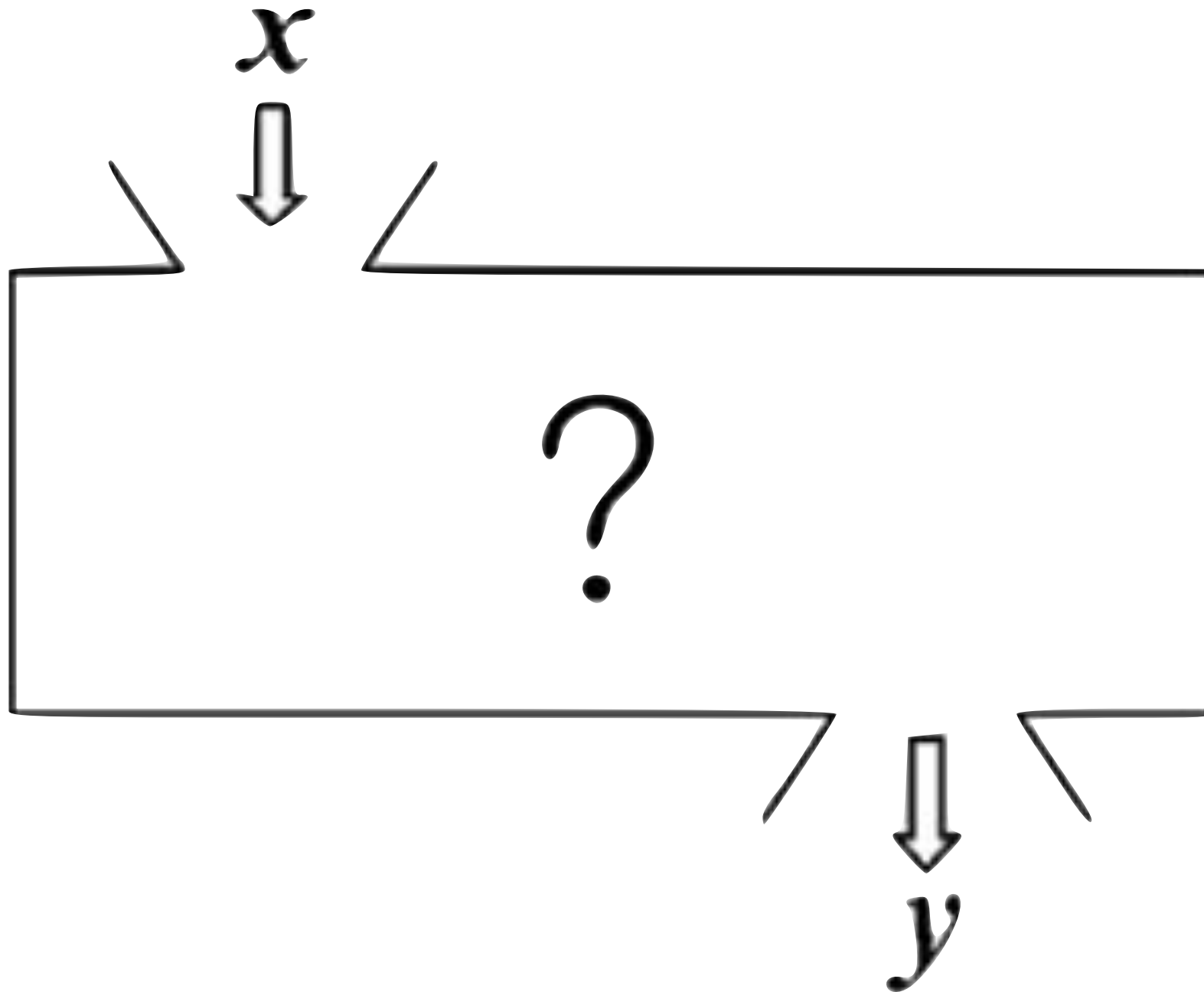
- 변수를 만드는데 있어 필요한 것은?

키워드 + 변수 명(Name) + 변수 타입(Type)

문법 : `var` vName:Any

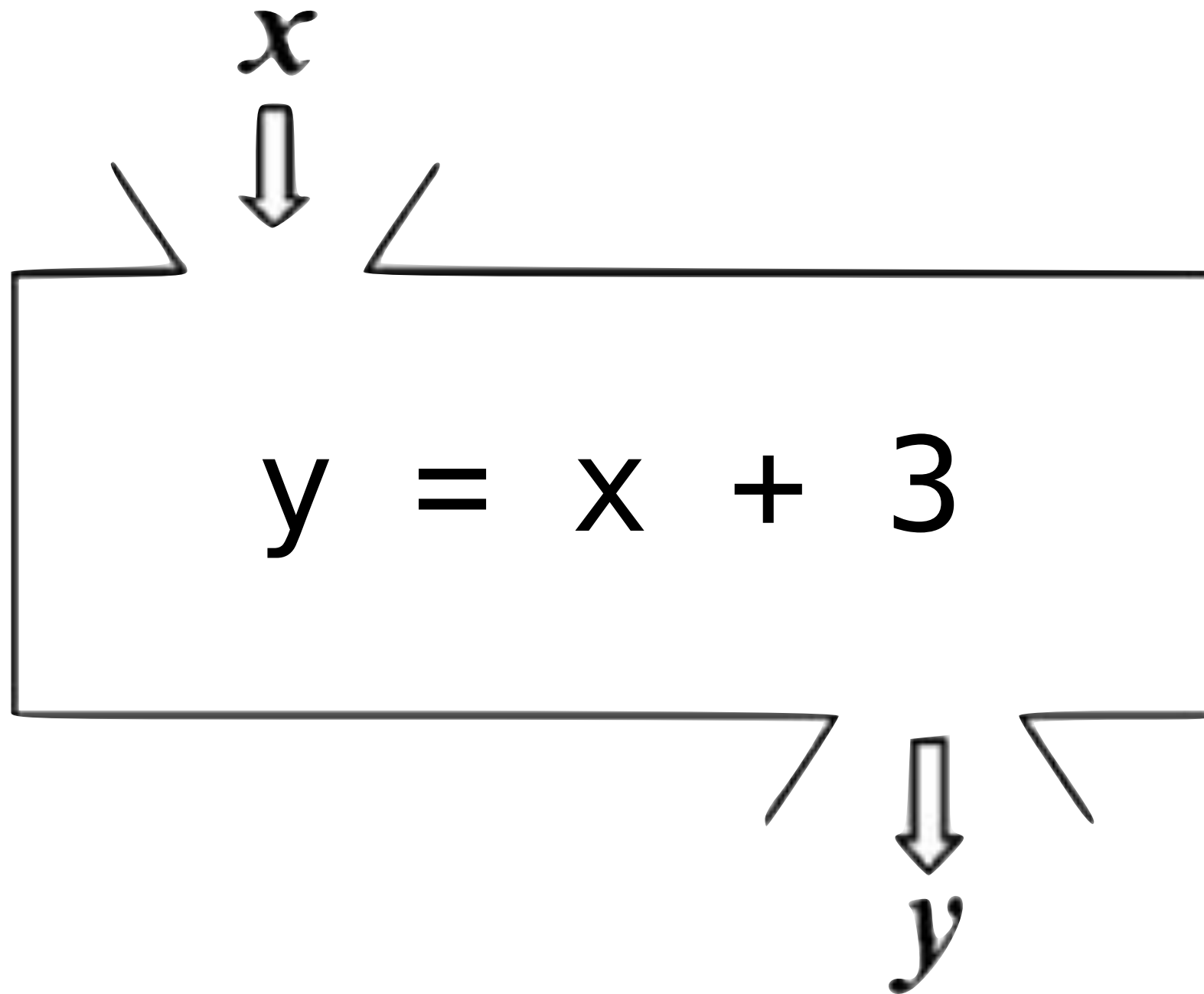
Review

함수



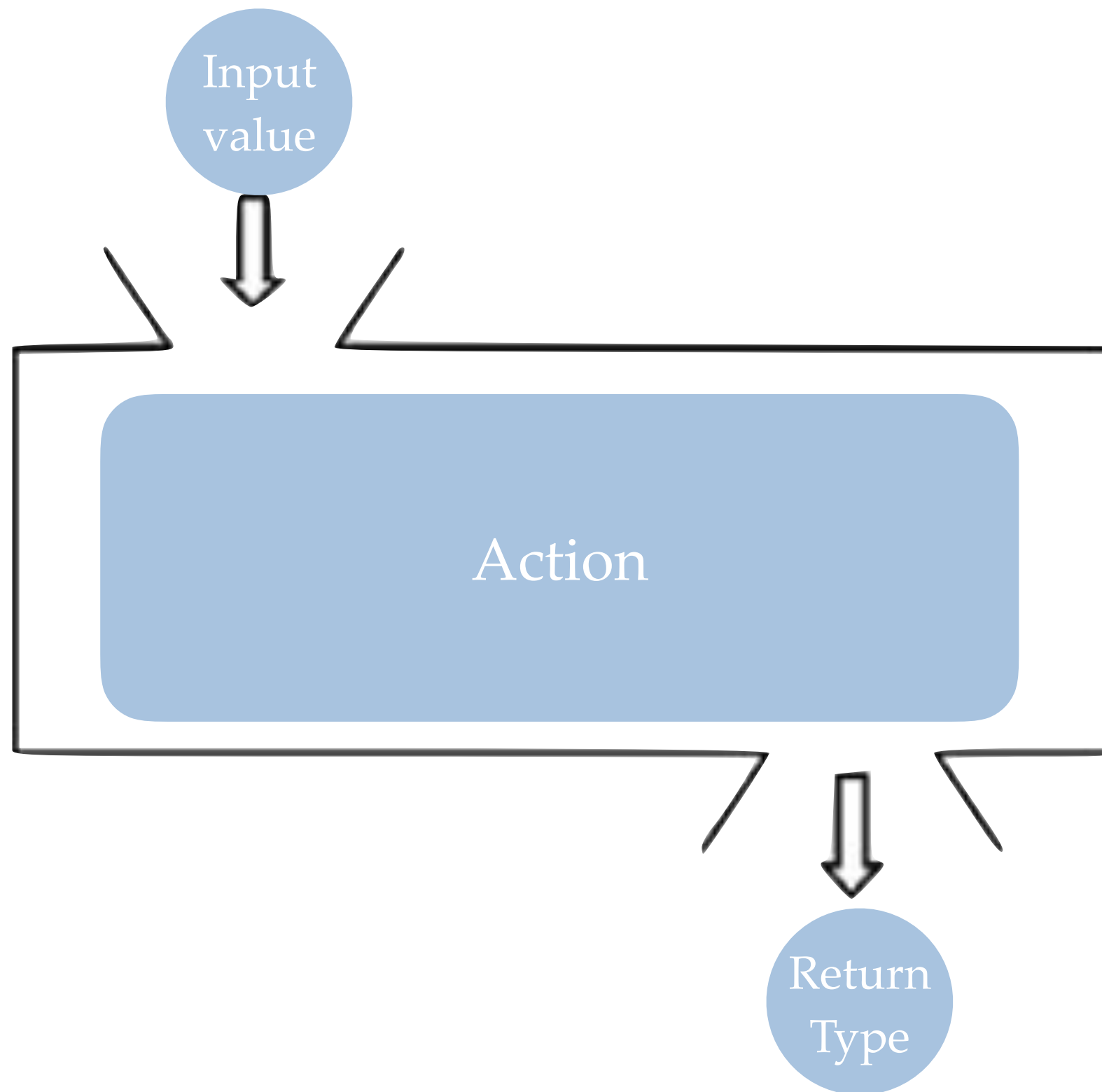
Review

함수



Review

함수



Review

- 함수 만들기 위해 필요한것?

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

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

Review

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)

Review

변수문법

키워드 변수타입
var **name** : **Type** = **value**
 변수명 값

다양한 형태의 변수 (일단 보고 가실게요)

//일반 변수 선언

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

//변수 값 재정의

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

//상수 선언

```
let PI = 3.14
```

```
var address:String?
```

```
address = "서울시 신사동"
```

키워드

- 변수 : 변할수 있는 값

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

- 상수 : 변할수 없는 고정 값

```
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 *value type, reference type, 거의 만들어져 있음*

타입이름	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 *class - memory 안에 유동적인 data 공간이 만들어 진다. 그래서 주소 값이 들어간다.*

타입이름	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.
- Uint : - 부호를 포함하지 않은(0은 포함) 정수
u : unsigned -> 부호가 없는
- 최대값과 최소값은 max, min 프로퍼티를 통해 알아볼수 있다.
Int.max
- Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64의 타입으로 나뉘져 있는데 기본은 시스템 아키텍처에 따라서 달라진다.
- 접두어에 따라 진수를 표현할수 있다. (2진법 0b, 8진법0o, 16진법 0x)

Bool

- 불리언 타입 (true, false)

논리형 type,

논리의 조합이 있어야 한다.

집합을 생각해 보라. AND, OR, NOT 등

Not -> !

!true => false

&& -> and

|| -> or

&, | => bit 연산에 사용

실수, 부동소수점 표현

Double : Float의 2배 => 표현할 수 있는 자릿수의 2배

정수와 실수의 표현 방식 자체가 달라서 바로 연산이 안되고 casting 하여 계산한다.

실수는 오차가 생긴다.

Double & Float

소수점, 어느 정도 자리에서 반올림

실질적 애플리케이션에서 나눌 수 있는 문제!

Graphic 이 수치화 되면서 나타나는 문제 -

retina display -> 4 pixel -> 1 point,

* PIXEL, POINT

- 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

단 하나의 문자

character란 type을 지정해야 함

“ ” 사용

문자열 -> character의 모음

- 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.
- 문자열이 존재하기 때문에 거의 사용하지 않음

String

- 문자의 나열, 문자열이라고 한다.
- Character와 마찬가지로 유니코드로 이뤄져 있다.
- 함수를 제공해줌- 문자열을 다루기 위한 다양한 기능이 제공된다.
(hasPrefix, uppercased, isEmpty등)
“http”등 구분할 때 사용
- 문자열을 Character로 분해하여 꺼낼 수 있다.

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

SubString

Swift4에 추가된 부분

String의 part

내부적으로 메모리를 효율적으로 관리하기 위해 사용

String을 가공하면 substring으로 나오고, 이것을 다시 String으로 casting 하여 사용

google 검색 : swift4 string slice

- String을 나누면 SubString 타입으로 반환된다.

```
let sampleText = "<<<Hello>>>"
let startIndex = sampleText.index(sampleText.startIndex, offsetBy: 3)
let endIndex = sampleText.index(sampleText.endIndex, offsetBy: -3)
```

```
let substring1: SubString = sampleText[startIndex... endIndex]
let resultStr:String = String(substring1) // "Hello"
```

```
let sampleText = "<<<Hello>>>"
let startIndex = sampleText.index(sampleText.startIndex, offsetBy: 3)
let endIndex = sampleText.index(sampleText.endIndex, offsetBy: -4)
print(startIndex)
```

ℳ

```
let substring1: SubString = sampleText[startIndex ... endIndex]
let resultStr:String = String(substring1) // "Hello"
print(resultStr)
```

튜플

Type이다.
collection이 아니다.
Type의 조합

- 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)")
```

```
func calculator(num1: Int, num2: Int) -> (Int, String)
{
    let sum: Int = num1 + num2
    return (sum, String(sum))
}
```

```
let result: (intSum: Int, strSum: String) = calculator(num1: 10, num2: 3)
let intResult = result.intSum
let stringResult = result.strSum
```

Any, AnyObject, nil

Any : 모든 type

AnyObject: 모든 객체 type

일반적으로 'Any'를 많이 사용함

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

변수 안에 아무것도 없음, 비어 있는 방, 값이 아예 없는 상태

var a: Int —> nil

var p: Person() —> nil

Optional —> 이 변수는 nil일 수도 있다.

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
```

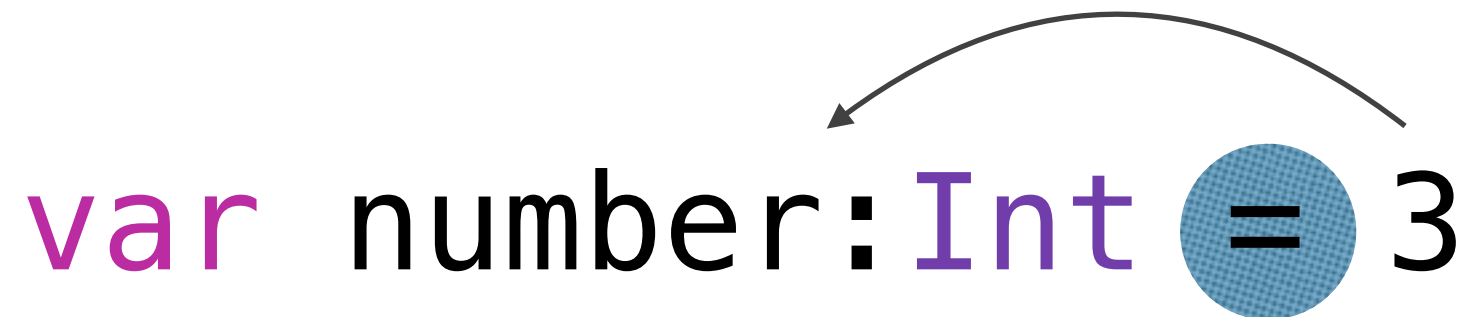
캐스팅(형변환)

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

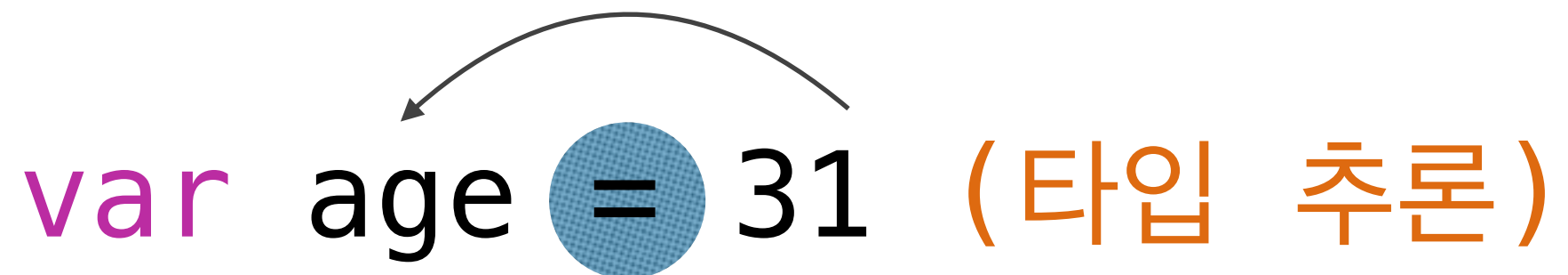
```
stringNum = String(intNum) ← int to string  
doubleNum = Double(intNum) ← int to double
```


변수 값 지정

`var number: Int = 3`



`var age = 31` (타입 추론)



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

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

```
var address:String?
```

```
address = "서울시 신사동"
```

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 Swift function syntax with the following components and annotations:

- func**: Keyword (키워드), highlighted with a red circle.
- fName**: Function name (함수 이름), highlighted with a red circle.
- argumentName**: Parameter name (인수명), highlighted with a red circle.
- paramName**: Parameter name (매개변수명), highlighted with a red circle.
- Int**: Parameter type (매개변수타입), highlighted with a red circle.
- >**: Return type arrow, highlighted with a red circle.
- Int**: Return type (반환타입), highlighted with a red circle.
- {**: Opening curly brace, part of the function body.
- return paramName + 3**: Function body content (함수 내용).
- }**: Closing curly brace, part of the function body.

Argument Name and Parameter Names

<함수구현시>

외부에서 보이는 이름 내부에서 보이는 이름 swift에 메소드는 서술형으로 짜져있다
인수명 매개변수명 매개변수타입 message 를 보낸다 (함수 실행)
method 실행

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

함수 내에서 매개변수(parameter)로 사용

.....

<함수구현시>

외부에서 보이는 이름을 사용하기 원치 않으면
_ 내부이름 -> underbar 내부변수

```
fName(argumentName: 10)
```

함수 밖에선 인수명(argument)로 사용

Argument Name and Parameter Names

- 인수명은 함수 호출시 사용 되는 이름. (Argument-아규먼트)
- 매개변수는 함수 내부에서 사용 되는 변수명.(Parameter-파라미터)
- 인수명은 생략가능하며, 생략하면 매개변수명이 인수명으로 사용된다.
- 인수명을 제거하고 싶으면 와일드카드 (_)를 사용한다.

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 설정할 수 있다.
- 매개변수로 값이 들어오지 않을때 기본값이 사용된다.

In-Out Parameter Keyword

inout Keyword

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

기본적인 함수의 파라미터는 상수임.

inout을 통해 변수처럼, 원래의 상수를 가져올 수 있게 함. reference참조

- 매개변수는 상수값이다.
- 만약 매개변수의 값을 변경해야 한다면 inout 키워드를 사용하여 inout매개변수로 지정 해야 한다.
- inout매개변수 지정은 타입 앞에 inout keyword를 작성해준다.
- inout 변수가 지정된 함수의 인수앞에서 &가 붙어야 한다.
(단! 변수를 사용해서 대입할 수 있다. 직접 값 대입 불가)

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)
```



반환타입

반환타입

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

return : 함수를 종료

- 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)

여러가지 함수 - 예제

```
func pass(num:Int) -> Int {  
    return num  
}
```

```
func getPI() -> Double {  
    let pi: Double = 3.141592  
    return pi  
}
```

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

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

여러가지 함수 - 예제

```
func passNum(_ num:Int) -> Int {  
    return num  
}
```

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

```
func calculator(num1:Int, num2:Int) -> (Int, Int) {  
  
    let plus = num1 + num2  
    let minus = num1 - num2  
  
    return (plus, minus)  
}
```

여러가지 함수 - 예제

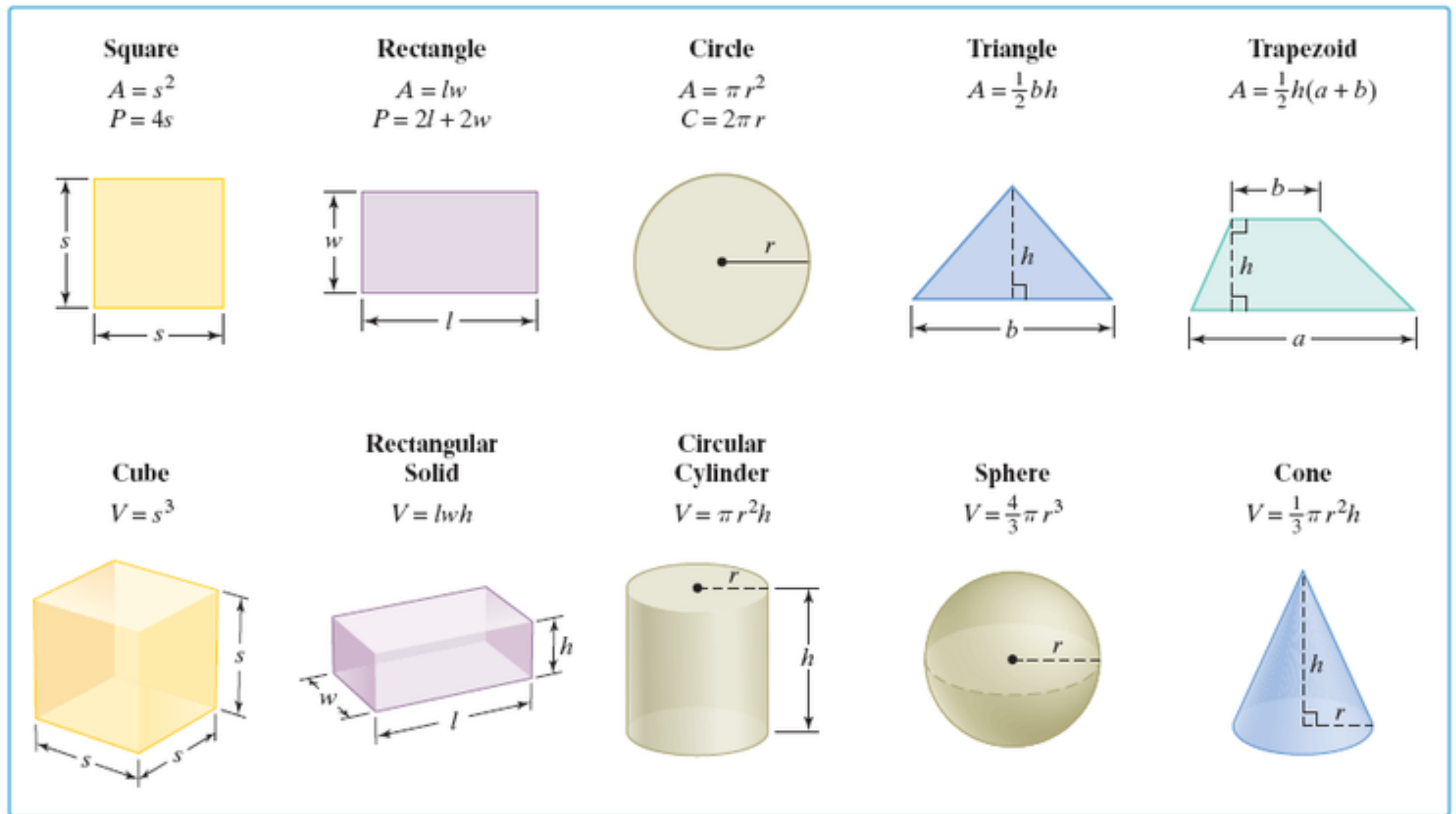
```
func printName() -> String{  
    return "my name is youngmin"  
}
```

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

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

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

함수 만들기 실습!



A(Area)넓이, P(Perimeter) 직사각형 둘레, C(Circumference)원의 둘레, V(Volume) 부피

함수 만들기 실습!

```
func squareArea(length: Int) -> Int
{
    return length * length
}
```

```
func squarPerimeter(length: Int) -> Int
{
    return 4 * length
}
```

