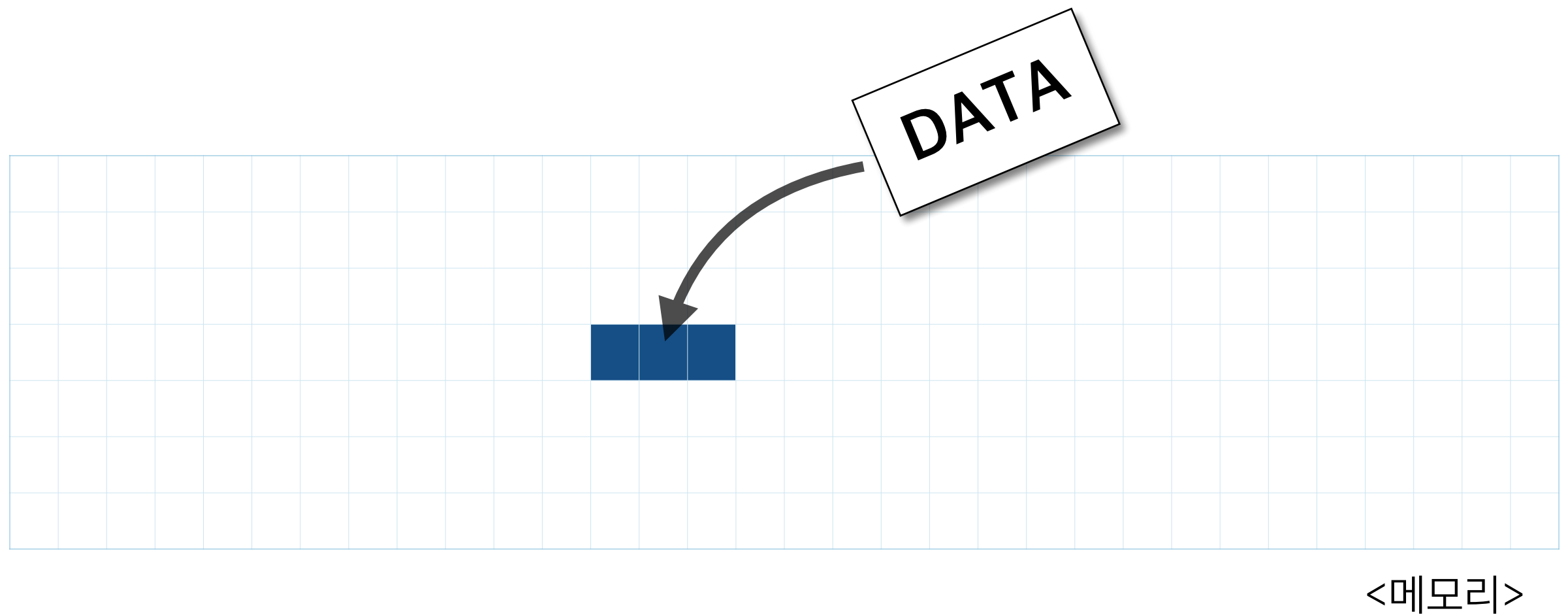

변수 & 함수 기초

변수 & 함수

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

변수

- 변수 : 프로그램에서 데이터의 저장하는 공간



변수의 구성

- 이름(name) : 변수를 구분 짓기 위해 임시로 지정된 이름
- 형(type) : 변수가 가질수 있는 데이터의 자료형
- 메모리주소(address) : 변수가 메모리 상에 위치하고 있는 주소
- 값(value) : 변수에 들어가는 데이터. 대입연산자를 통해 변수에 값을 지정할수 있다.
- 영역(scope) : 변수가 사용가능한 영역. 변수의 사용이 허락되어지는 프로그램 범위를 말한다.

Swift 문법 - 변수

키워드 변수타입

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

변수명 값

명명규칙 : 이름 짓는 방식

- 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 선언

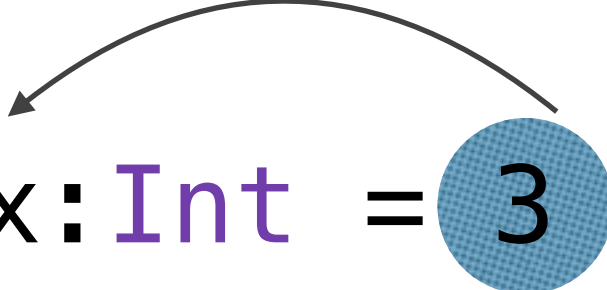
- 선언 : 프로그래밍 언어가 변수를 사용하기 위해 메모리에 할당을 선언
Swift에서는 변수 선언을 위해 var/let 키워드를 사용한다.

```
var x:Int
```

변수 값 할당

- 값 할당 : 대입 연산자 (=)를 통해 값을 할당 한다.

`var x: Int = 3`



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

변수의 사용

- 값 사용 : 변수의 할당된 값을 가져와 사용한다.

```
var x:Int = 3  
var y:Int = x  
var z:Int = x + y
```

변수 자료형

기본형

| 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> |

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

- 참(true), 거짓(false) 두개의 값을 갖는 타입.
- 수식이 참인지 거짓인지 파악할때 사용.
- and, or, not 연산을 통해 논리적 로직 구현 가능

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)

타입 추론

- Swift에서는 선언시 할당되는 값에 의해 타입을 추론하여 가진다.
- 타입 추론시 변수 선언시 type을 작성하지 않아도 된다.

var y = 31 (Int형 타입으로 추론)

변수 VS 상수

- 변수 : 변할수 있는 값

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

변경가능(O)

- 상수 : 변할수 없는 고정 값

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

컴파일 에러(X)

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 배울 내용입니다^^)

```
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.

연산 해보기

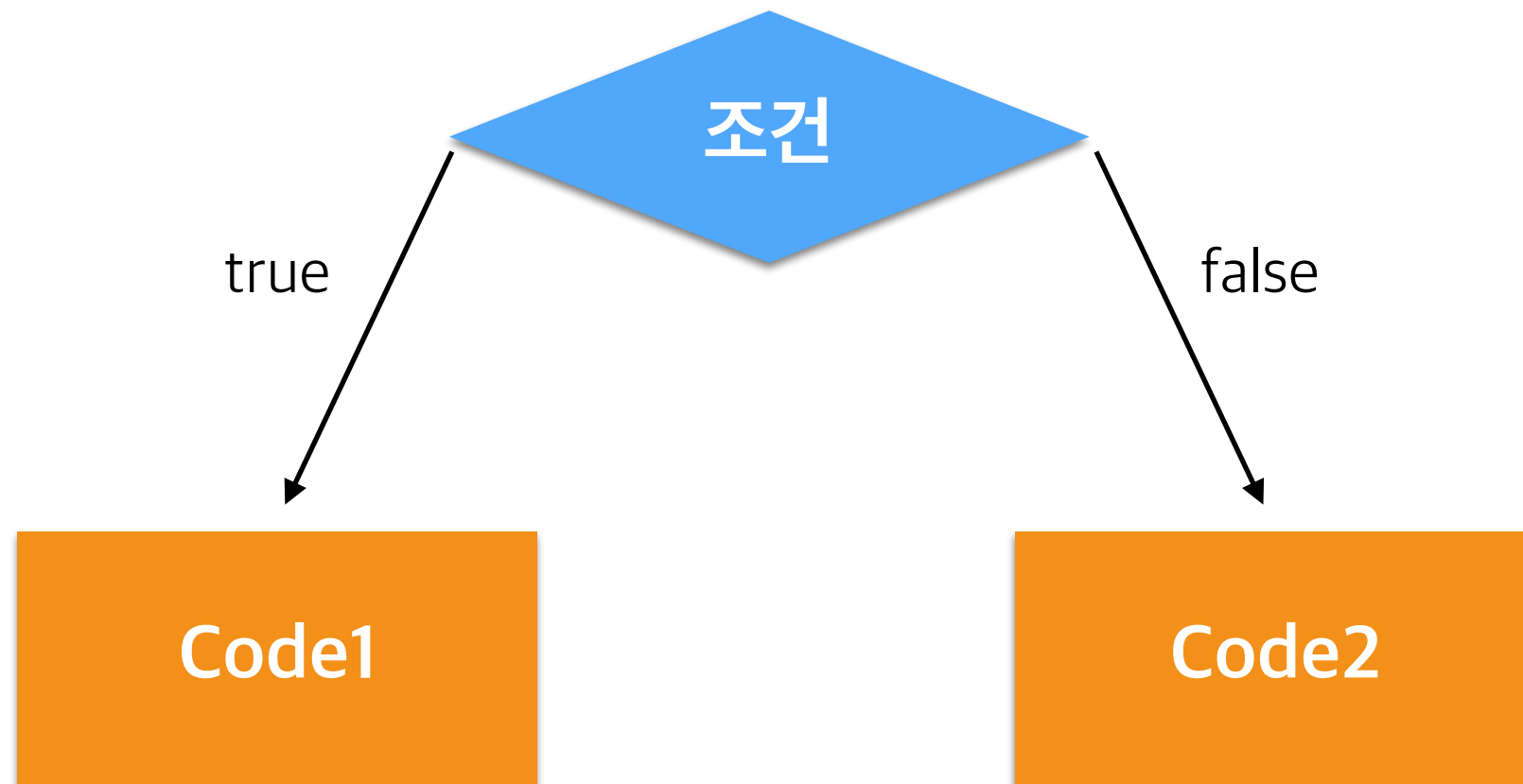
- 변수의 값들은 연산해보는 연습을 해보자.

산술 연산자

| 기호 | 예제 | 설명 |
|----|---------------|-----|
| + | $1 + 2 = 3$ | 더하기 |
| - | $2 - 1 = 1$ | 빼기 |
| * | $2 * 1 = 2$ | 곱하기 |
| / | $10 / 3 = 3$ | 나누기 |
| % | $10 \% 3 = 1$ | 나머지 |

조건문

- 양자 택일 문
- 특정 조건에 따라 선택적으로 코드를 실행시킨다.
- 대표적인 조건문으로 if-else문과 switch-case문이 있다.



if-else문

조건이 참일경우 if문 대괄호 안의 코드가 실행된다.

만약 조건이 거짓인 경우 else문 대괄호 안의 코드가 실행된다.

```
if 조건 {  
    //조건이 만족되면 실행  
}  
else {  
    //조건이 만족되지 않을때 실행  
}
```

***조건값은 참,거짓의 나타나는 Bool값으로 표현.**

else if문

추가 조건 방법으로 반복해서 추가 할수 있다.

```
if 조건1 {  
    //조건1이 만족되면 실행  
} else if 조건2 {  
    //조건1이 만족되지 않을때 실행  
} else {  
    //조건들 모두 만족되지 않을때 실행  
}
```

***조건2는 조건1이 거짓일때 실행된다.**

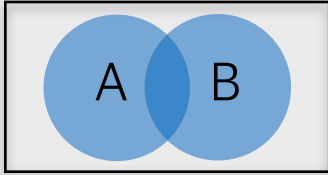
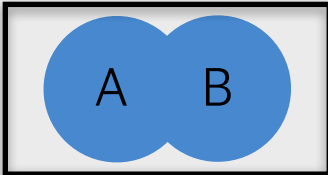
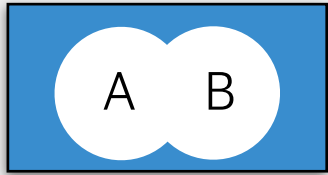
조건 만들기

- 비교연산자를 통해 조건의 결과가 bool값으로 나와야 한다.
- 논리 연산자로 다양한 조건의 조합이 가능하다.

비교 연산자

| 기호 | 예제 | 설명 |
|--------------------|------------------------|------------------------|
| <code>==</code> | <code>A == B</code> | A와 B가 같다 . |
| <code>>=</code> | <code>A >= B</code> | A가 B보다 크거나 같다 |
| <code><=</code> | <code>A <= B</code> | A가 B보다 작거나 같다 . |
| <code>></code> | <code>A > B</code> | A가 B보다 크다 |
| <code><</code> | <code>A < B</code> | A가 B보다 작다 |

논리 연산자

| 기호 | 예제 | 집합 | 설명 |
|----|---------------|---|-----------------------------|
| && | A조건 && B조건 |  | A조건이 참이고, B조건이 참이면 참이다. |
| | A조건 B조건 |  | A조건이나, B조건 둘중에 하나가 참이면 참이다. |
| ! | !(A조건 B조건) |  | A B조건을 반대 |

연습문제

- 홀수 짝수 고르기

반복문

- 반복적으로 실행되는 코드를 만드는 구문
- 대표적인 반복문으로 `while문`과 `for-in문`이 있다.

while문

조건이 참일경우 구문 반복 실행

```
while 조건
```

```
{
```

```
//구문 실행
```

```
}
```

while문 사용예제

```
var index:Int = 0;

while index < 10
{
    print("현재 횟수는 \(index)입니다.")
    index = index + 1;
}
```

*구문안에 조건을 변화시키는 내용이 없으면 무한 반복이 될수 있다.

연습문제

- while문을 이용한 구구단 만들기

범위 연산자

- Int형 데이터의 범위에 해당하는 값을 나타낸다.
- a, b에 변수의 값을 사용 가능

| 범위 연산자 | 예제 | 설명 |
|--------|--------|--------------------|
| a...b | 3...10 | a~b까지의 숫자 |
| a..<b | 0..<10 | a~b까지 숫자중 b는 포함 안함 |

for-in 문

- 스위프트에선 for문 대신 for-in문을 사용한다.
- 배열의 항목, 숫자의 범위 또는 문자열의 문자와 같은 시퀀스를 반복하려면 for-in 반복문을 사용합니다.

```
for element in sequenceData  
{  
  
    //반복될 구문  
  
}
```

for-in 문

```
for index in 1..5 {  
    print("현재 횟수는 \(index)입니다.")  
}
```

```
var x:Int = 0
```

```
for index in x..5 {  
    print("현재 횟수는 \(index)입니다.")  
}
```

연습문제

- for in문을 이용한 구구단 만들기

Subprogram

- 하나의 프로그램을 구성하는 여러 작은 단위의 프로그램
- 프로그램 명령문의 묶음, 일련의 명령문들을 모아두고, 이를 외부에서 호출할 수 있게한 구조
- 대표적으로 함수가 이에 해당된다.

Subprogram

```
var a:Int = 10  
var b:Int = 5  
var c:Int = a + b
```

```
a = 8
```

```
b = 5
```

```
c = a + b
```

```
b = 4
```

```
c = a + b
```

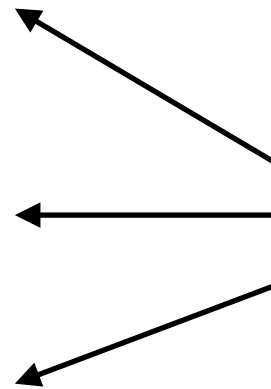
```
.
```

```
.
```

```
.
```

```
.
```

```
.
```



두 수를 더 하는 기능

Subprogram

```
var a:Int = 10
```

```
var b:Int = 5
```

```
var c:Int = A호출
```

```
a = 8
```

```
b = 5
```

```
c = A호출
```

```
b = 4
```

```
c = A호출
```

```
.
```

```
.
```

```
.
```

```
.
```

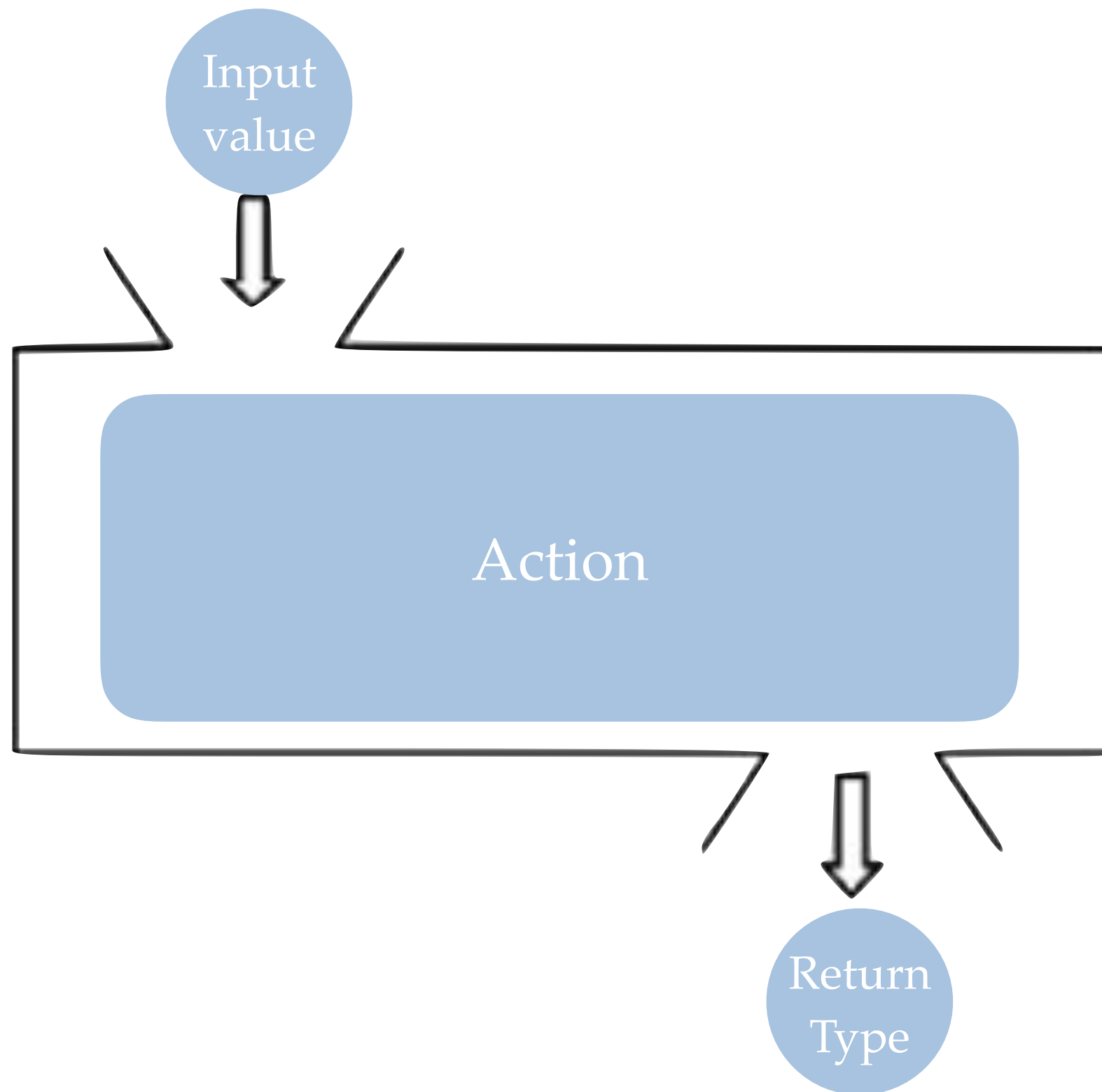
```
.
```

<Subprogram A>
두 수를 더해서
그 값을 반환 한다.

Subprogram 장점

- 프로그램 재사용으로 인한 절약
 - 메모리 공간, 코딩 시간의 절약, 프로그램 크기를 줄임
- 가독성 증가
 - 세부사항을 숨기면서 프로그램의 논리 구조를 강조함
- 유지보수 및 확장성 용이, 문제 해결성 및 분할성 제고 등

함수



함수 문법

함수 이름 매개변수 반환타입

```
func fName(paramName: type) -> type
{
    //실행될 명령어 작성
    return 반환값(은 반환타입과 같아야 된다)
}
```

함수 구현

```
func add(num1:Int, num2:Int) -> Int
{
    return num1 + num2
}
```

함수 사용

add(num1:a, num2:b)

```
var a:Int = 10
var b:Int = 5
var c:Int = add(num1:a, num2:b)
a = 8
b = 5
c = add(num1:a, num2:b)
b = 4
c = add(num1:a, num2:b)
.
.
.
.
.
```

```
func add(num1:Int, num2:Int) -> Int
{
    return num1 + num2
}
```

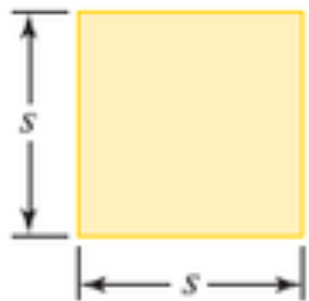
연습문제

- 사칙연산 함수를 만들어 보고 사용해 보세요

Square

$$A = s^2$$

$$P = 4s$$

**Rectangle**

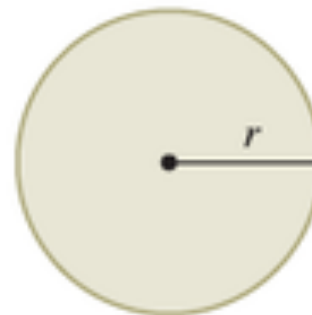
$$A = lw$$

$$P = 2l + 2w$$

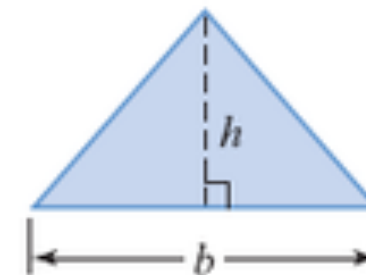
**Circle**

$$A = \pi r^2$$

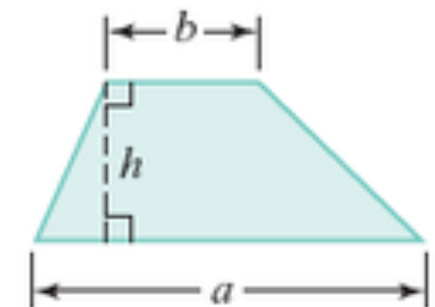
$$C = 2\pi r$$

**Triangle**

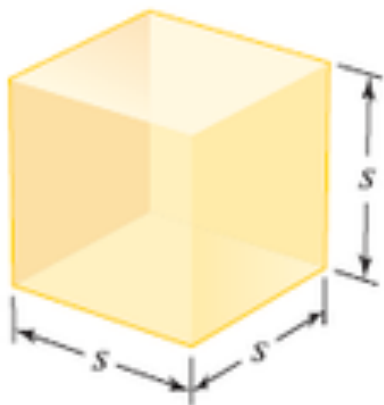
$$A = \frac{1}{2}bh$$

**Trapezoid**

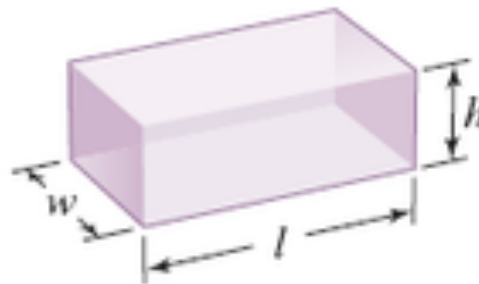
$$A = \frac{1}{2}h(a + b)$$

**Cube**

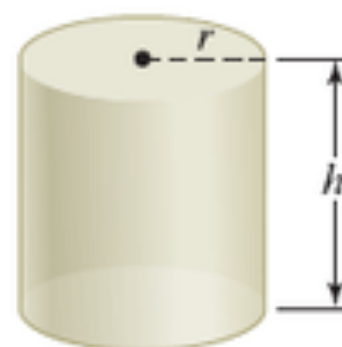
$$V = s^3$$

**Rectangular Solid**

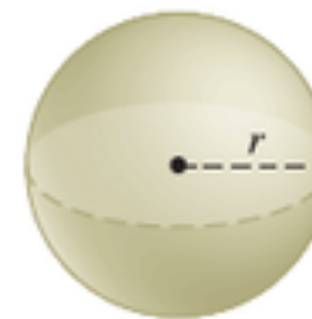
$$V = lwh$$

**Circular Cylinder**

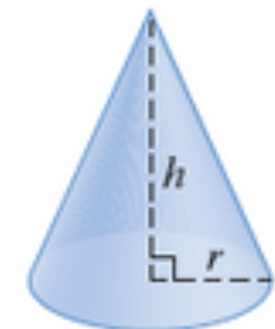
$$V = \pi r^2 h$$

**Sphere**

$$V = \frac{4}{3}\pi r^3$$

**Cone**

$$V = \frac{1}{3}\pi r^2 h$$



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