

ios 입문 강의

2강. 정보를 담는 공간 “변수&상수”

행동에 대한 명령 “함수”

1. 프로그래밍과 변수, 함수

* What is Programing?

* 프로그램

- * 프로그래밍을 통해 만들어진 것
- * 사용자의 입력에 반응하도록 구현된 명령어의 집합

* 프로그래밍

- * 프로그래밍 언어를 이용해 구체적인 컴퓨터 프로그램으로 구현하는 기술

* 정보의 절차



* Swift Class Architecture (그림참고)

- * Class 형태로 크게 이루어짐
 - * 그 안에는 변수와 함수로 이루어짐
 - * 그 함수 안에는 조건문, 반복문, 변수 등 여러가지 요소들이 존재
- * Class들의 모음으로 프로그램이 구성

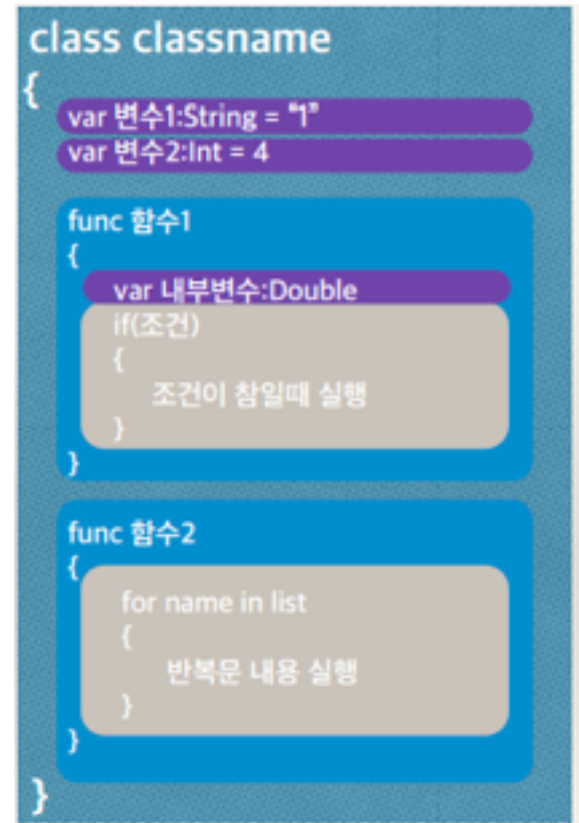
* 프로그래밍에서 가장 중요한 것

* 변수(Variable)

- * 데이터가공
- * 저장소 개념
- * 메모리에 저장됨

* 함수(Method)

- * 행동지시
- * 액션을 지칭
- * CPU가 연산 후, 변수에 집어넣는다 (메모리에 집어 넣는다)
 - * 일정한 행동을 끝내면 변수에 집어넣고
 - * 행동이 끝일 때에는 CPU에서 액션이 끝난다.



* 변수에 대하여

- * 변수란, 데이터를 담는 박스

* 변수 - Type

- * 변수 내용물의 형태
 - * 변수 타입에 따라 변수에 저장할 수 있는 값의 종류와 범위가 달라진다.

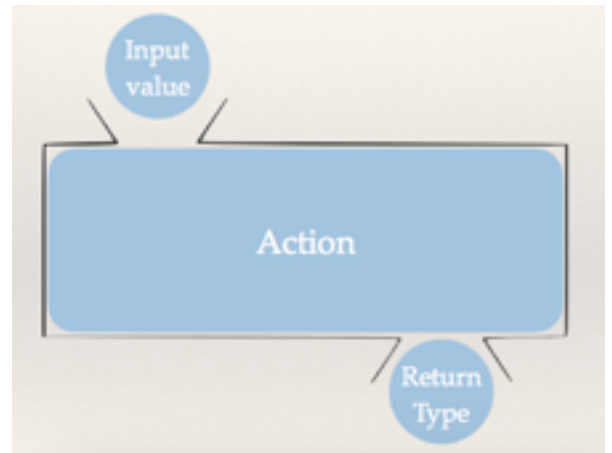
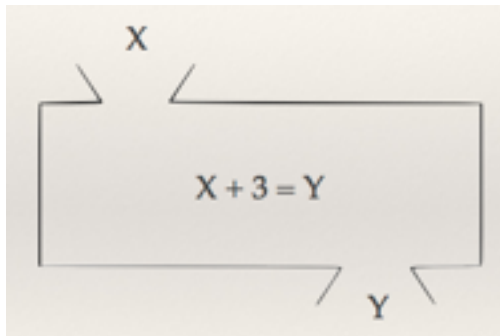
* 변수 - Name

- * 조금 전에 저장한 데이터는 어느 상자에 있는가?
- * 변수명을 지정 -> 컴퓨터와 사용자간의 데이터 사용을 위한 네이밍

* 변수 만드는데 필요한 것

- * 변수 명 (Name) + 변수 타입 (Type)

* 함수에 대하여



- * $x+3 = y$ → 일정한 값에 대한 결과값
- * 함수란, 입력을 통해 어떠한 일을 수행한 다음 결과물을 내어놓는 것

* 함수를 만들기 위해 필요한 것

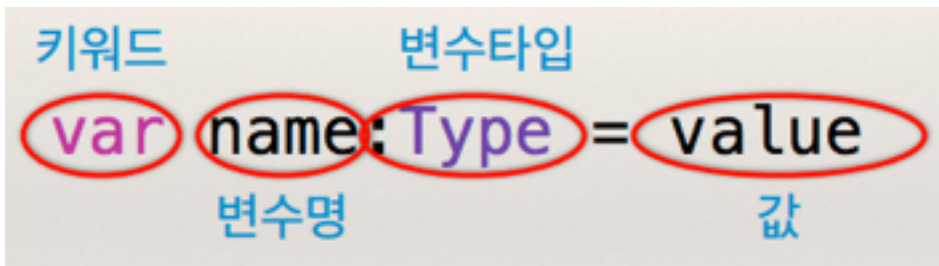
- * 함수명(Name) + 입력값(Input Value) + 함수내용(Action) + 결과타입(Return Type)
 - * input값에 따라 output 값이 정해짐
 - * 결과물에는 Type 을 정해줘야한다.

2. Swift 문법 - 변수

* 프로그래밍과 언어의 문법

- * 사용자가 만든 코드 —컴파일—> 컴퓨터가 이해할 수 있는 코드
- * 문법 : 컴파일러가 이해하기 위한 프로그래밍 규칙
 - * 우리는 swift에 맞는 문법을 배울 것

* Swift에서 변수를 만드는 문법



- * `var name:Type = value`
- * 키워드(var) / 변수명(name) / 변수타입(type) / 값 (value)

* 변수를 만드는 키워드

```
var num:Int = 3
num = 4 — O
```

- *변수: 변할 수 있는 값 (var)
- *`var name:Type`
- *name을 나중에 바꿀 수 있음

```
let num:Int = 3
num = 4 — X
```

- *상수: 변할 수 없는 고정값(let)
- *`let name:Type`
- *name변경 불가. 삭제하거나 새로 만들어야 한다.

* 이름을 만드는 규칙 (명명 지침)

- * 시스템 예약어를 사용할 수 없다. (컴파일러가 원래 사용하는 명령어들)
- * 시작글자에 숫자를 쓸 수 없다.
- * 띄어쓰기(공백)를 할 수 없다. (띄어쓰기 자체를 문법의 핵심으로 취급)
- * 구분을 위한 약속 방법 ——> 개발자간의 권고사항
 - * 변수&함수명은 camelBackCase로 작성
 - * 첫번째 글자는 소문자로 쓰고, 그 다음에 바뀌는 단어는 대문자로 끊어 읽기
 - * 클래스명은 CamelCase로 작성
 - * 첫 시작을 대문자, 단어가 바뀔 때마다 대문자

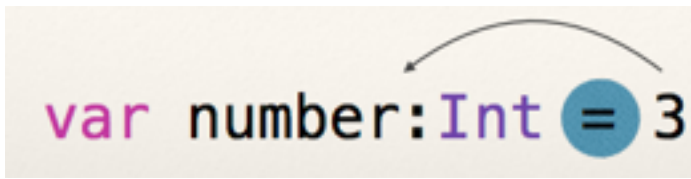
* 변수 타입

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
클래스명	ClassName	클래스 객체를 다른곳에서 사용할 경우	var customView:UIView var timer:NSTimer

- * 기본형
 - * 정수 / 실수 / 문자열 / 불리언
 - * 맨 앞글자는 다 대문자로 써야 한다
 - * 불리언
 - * True and False -> 선택에 기로에 서있을 때 사용
 - * 정수는 일정한 범위가 있음 (확인해보면 될 것)
- * 참조형
 - * 기본형 외에는 다 참조형

* 값 지정하기



```
var number: Int = 3
```

* 우측에 있는 값을 좌측 변수에 대입

- 예제 -

* 변수와 상수

```
class ViewController: UIViewController {  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        // 1. 변수와 상수 연습 //  
        var age: Int = 6  
        var weight: Double = 8.2  
        let name: String = "cheese"  
        var isCat: Bool  
        isCat = true  
  
        print(name)  
        print(weight)  
        print(isCat)  
        print("변경 전 나이: \(age)")  
        age = 7  
        print("변경 후 나이: \(age)")  
    }  
}
```

* var -> 변수라 이후 변경 가능

* 왼쪽 예제 age와 같은 변수

* let -> 이후 변경 불가

* 왼쪽 예제에서 name

3. Swift 문법 - 함수

```
키워드   함수 이름   매개변수   Return Type
func functionName(paramName: Int) -> Int {
    var returnNum : Int
    returnNum = paramName + 3
    return returnNum
}
```

함수 내용(구현부)

- * 중괄호 {} 를 통해 구현부 영역을 정해준다.
- 함수의 특정한 성질을 나타내는 변수를 말한다.
- 매개변수는 () 안에 변수로 작성된다.
- 매개변수는 해당 함수 안에서만 사용된다

* 매개변수 예제

• No Input값 예제

```
func getAge() -> Int {
    var age: Int = 22
    return age
}
```

• 다중 input값 예제

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

* No Input값 예제

- * input이 없으면 () 빈 괄호

* 다중 Input값의 예제

- * input이 여러 개 이면 콤마(,) 로 구분

* Swift 문법 - Return 값 (반환값)

```
func functionName(paramName:Int) -> Int {  
    var returnNum : Int  
    returnNum = paramName + 3  
    return returnNum  
}
```

Return Type
키워드 반환값

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

* 반환값 예제

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

```
func sumNumber(num1:Int, num2:Int){  
    var returnNum:Int  
    returnNum = num1 + num2  
}
```

- * printName 함수는 return값 없이 print만 해주는 함수
- * sumNumber 함수는 returnNum이라는 변수에 매개변수 num1, num2를 더해주는 함수
(반환값 없음)

* Swift 문법 - 함수의 종류

* 입력값 ○ 반환값 ○

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

* 두 번째 파라미터에 secondNum이라고 써줌 (objective-c의 잔해)

* ---> 안해도 됨 (취향 차이)

* 입력값 ○ 반환값 X

```
func setAge(newAge:Int){  
  
    self.age = newAge  
  
}
```

*다른곳에 정의된 변수 age에 파라미터로 받은 새로운 age를 넣는다.

* 함수 밖에 있는 변수에 역할을 넣어주는 것

* self.age : setAge함수 밖에 있지만, 같은 클래스 안에 있는 변수 age를 불러온다.

* 입력값 X 반환값 ○

```
func getAge() -> Int {  
  
    return self.age  
}
```

*다른곳에 정의된 변수 age에 값을 돌려주는 함수

* 입력값이 없다. -> 빈 괄호 ()

* self.age -> 다른곳의 변수 age의 값을 돌려줌

* 입력값 X 반환값 X

```
func stopMusic(){  
    //현재 재생중이 음악 멈춤  
}
```

- 예제 -

```
// 2. 이름을 출력하는 함수의 값(?)  
self.printName("치쭈")  
// self.__ : 같은 클래스 내의 __ 함수 실행!  
}  
  
override func didReceiveMemoryWarning() {  
    super.didReceiveMemoryWarning()  
    // Dispose of any resources that can be recreated.  
}
```

함수 호출

```
// 2. 이름을 출력하는 함수  
func printName(newname:String)  
{  
    print(newname)  
}
```

함수 구현

* 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

* 변수의 생명주기

- * 변수(객체)가 생성되면, 메모리에 적재된다.
- * 프로그램을 만들 때, 너무 많은 양의 변수가 쌓이게 되면 메모리에 지나치게 많이 쌓임
- * ARC가 메모리 자동 관리 해줌
- * 기본적으로는 { } 중괄호 안에서만 유지
 - * 함수 안에서 만들면 함수가 끝날 때 메모리 해지
 - * ex) A 함수의 변수는 B 함수의 변수에서는 사라짐