ios 입문 강의

5강 Collection - 딕셔너리와 튜플

1. 배열과 반복문 (지난 수업 복습)

1) Array (배열)

* 메모리 상에 같은 타입의 자료가 **연속적**으로 저장된다.

* 같은 타입의 자료가 저장 / 연속적으로 저장 / 인덱스로 관리가 된다 ⭐

*** 배열(Array) - 문법**

*** 선언**

- var vName:[Type] = [value1, value2…]
- var vName:Array [Type] = [value1, value2…]

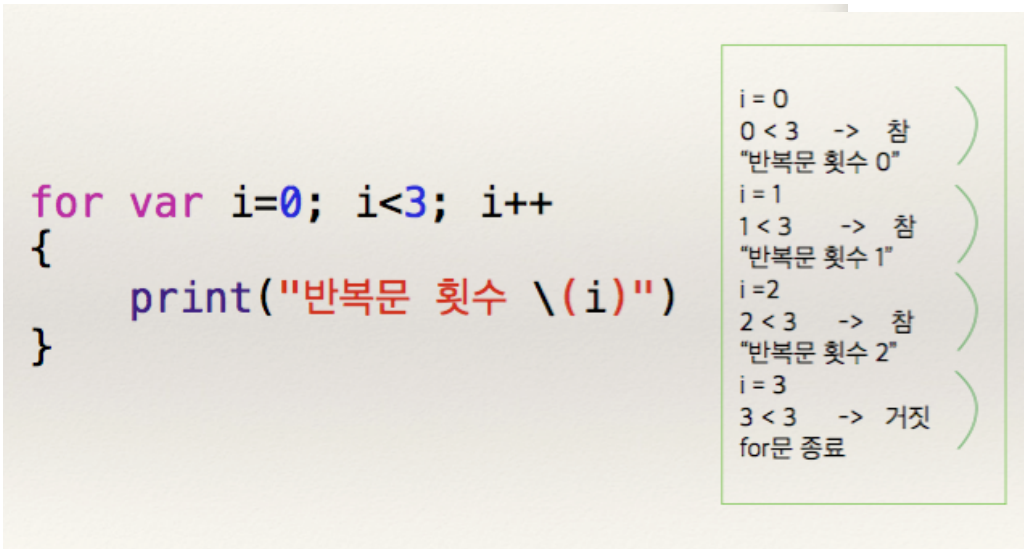
*** 사용**

* vName[index]
  * index 는 0,1,2 순서
* vName.count
  * 배열 내의 자료가 몇 개인지 세어준다.
* vName.append(newValue)
  * 배열의 맨 뒤에 newValue를 추가한다.

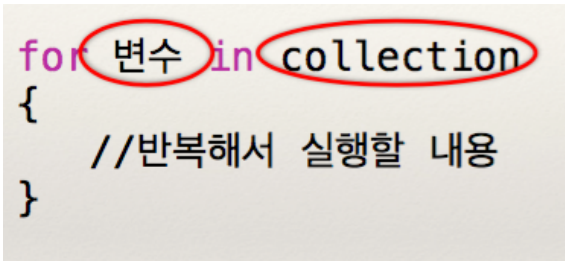
2) 반복문

반복적으로 실행되는 코드를 만드는 구문

* for문 - 문법



* for in 문



* 연습 문제) 소수점 두 번째 자리 반올림하기 / 윤년 구하기 문제

```swift
// 소숫점 두 번째 자리에서 반올림 하기
func roundNum(num:Double) -> Double {
    // 3.12345 + 0.005---> 3.12845 --> 3.12
    // 3.1789 + 0.005 ----> 3.1869 --> 3.18

    var round:Double = num
    round += 0.005
    let bigRound:Int = Int(round * 100)
    round = Double(bigRound)/100
    return round
}
```

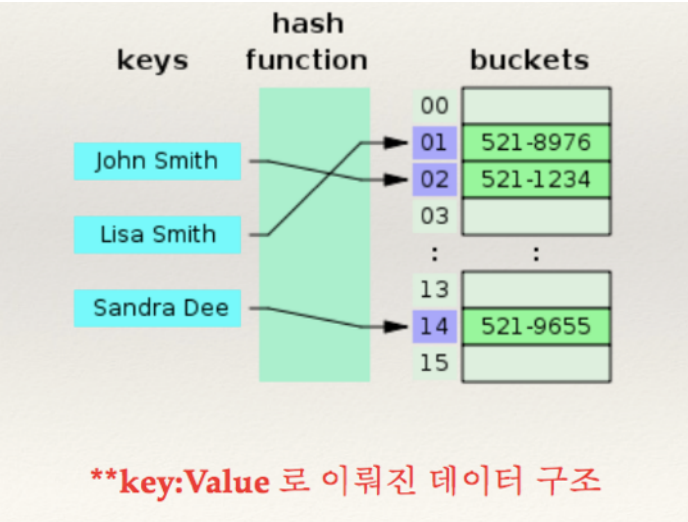
```swift
/*

 조건
 - 4로 나누어 떨어지는 년도는 윤년이다
 - 그 중 100으로 나누어 떨어지는 년도는 윤년이 아니다
 - 400으로 나누어 떨어지는 년도는 무조건 윤년이다

 */

// 윤년 구하기
func leapYear (year:Int)
{
    if 0 == year % 4 && 0 != year%100 || 0 == year % 400 {
        print("윤년입니다")
    }
    else{
        print("윤년이 아닙니다.")
    }
}
```

IOS 입문 5기 화이팅 입니다! :) ❤️

2. 딕셔너리 (Dictionary)

1) Dictionary란?

* 해쉬맵
  * 키를 값에 매핑할 수 있는 구조인, 연관 배열 추가에 사용되는 자료구조이다.
  * 해시 테이블은 해시 함수를 사용하여 색인(index)을 버킷(bucket)이나 슬롯(slot)의 배열로 계산한다.



  * 키를 값(value)에 매핑 할 수 있는 구조
  * 배열은 값을 인덱스로 구별할 수 있었으나, 딕셔너리는 키값을 통해 매칭된 버킷을 가져올 수 있음

	배열	딕셔너리
모양	array = (value1, value2, value3……)	dic = [key1 : value1, key2 : value2, key3 : value3……]
부를 때	array [0] 인덱스 값을 통해 부름 (순서가 있다)	dic [key1] 키 값을 통해 부름 (순서는 보장되지 않음)
불러진 값	value1	value1

2) 딕셔너리 - 문법

* 선언
  * var vName:[keyType:valueType] = [key1:value1, key2:vlaue2 ……]
  * var vName:Dictionary<keyType,VlaueType> = [key1:value1, key2:vlaue2 ……]
    * key의 타입은 보통 이용을 용이하게 하기 위해 String을 쓴다. ex) [**"age"** : 13, **"name"** : "jimin"…]

* 사용
  * 데이터 추가 : vName.updateVlaue("newVale",forkey:"key2") <———> 배열에서는 append

IOS 입문 5기 화이팅 입니다! :) ❤️

* 값을 불러오기 : vName[“key2”]
* var value2 = vName["key2"]
  * 딕셔너리는 순서가 없다. 키값을 통해 정해지기 때문에 순서를 보장할 수 없음.
    * <——> 배열에서는 순차적으로 배열

*예시

```
var dic:[String:AnyObject] = [String:AnyObject]()
*var dic:Dictionary<String,Int> = Dictionary()

dic.updateValue("youngMin", forKey: "name")
dic.updateValue(28, forKey: "age")
dic.updateValue(180, forKey: "height")


var nameStr:String = "my Name is "
var totalAge:Int = 3
nameStr += (dic["name"] as! String)
totalAge += (dic["age"] as! Int)
```

*** as를 하는 이유**

- “name” 키에 나오는 것은 타입은 **Anyobject**
  —> **Anyobject가 가장 상단 타입**인데 string에 넣으려 하면 에러가 난다. 때문에 **다운 캐스팅** 할 때에는 as! 로 원하는 타입으로 강제 지정

※ ! —> 옵셔널

다운 캐스팅과 옵셔널에 대해서는 이후 자세히 배울 예정

📝사전 만들어보기

* 키 값의 타입이 string이고, value는 Anyobject인 dic이라는 딕셔너리를 만들었다.
  * Anyobject란, 모든 타입이 들어가는 것 (Anyobject에 대한 설명은 아래에 보충해놓았습니다.)

📝 사전에 데이터 업데이트 하기

* dic이란 딕셔너리에 **.updatevlaue**를 통해 키값과 value값을 추가 하였다.

📝 만든 사전 데이터 사용해보기 (변수에 사전 데이터를 추가하기)

* 변수 nameStr(string타입)과 변수totalAge(Int타입)를 만든다.
* dic에서 key가 name인 value(“youngMin”)을 변수 nameStr인 “my Name is ”에 더한다.
  * 결과 -> “my Name is youngMin”
* dic에서 key가 age인 value(28)을 변수 tatlaAge값인 3에 더한다.
  * 결과 -> 31

IOS 입문 5기 화이팅 입니다! :) ❤️

참고) Anyobject란 무엇인가?

< 범용타입의 객체 >

- 개발에 있어 편의와 효율성을 높이는 범용타입의 객체가 필요할 때가 있다.
- Objective-c에서는 이를 위해 id 타입을 제공한다.
- id타입은 모든 타입의 값을 저장할 수 있고, 내부적으로 저장된 데이터 형식의 호환성만 보장된다면 어떠한 타입으로든 변환할 수 있는 타입니다.
- 코코아 프레임워크나 코코아 터치 프레임워크에서는 범용타입의 객체를 이용하여 구문없이 객체를 입력받거나 반환하는 API들이 많다.
- 스위프트 또한 코코아 프레임워크, 코코아 터치 프레임워크를 사용하기 위해 범용 타입의 객체를 도입해야했는데, 이를 위해 Any타입과 Anyobject 클래스를 제공한다.
- Any 타입은 구조체, 클래스, 열거형, 함수 등 스위프트에서 제공하는 **모든 타입의 값**을 저장할 수 있는 타입이고,
- Anyobject는 클래스에 한해 데이터만 호환 된다면 **어떤 타입의 클래스로도 변환할 수 있는 범용 타입의 객체**이다.

참고 문헌 - 이재은, 『꼼꼼한 재은 씨의 스위프트 2 프로그래밍 : iOS 앱 개발을 위한 Swift 2

*** 딕셔너리에서 사용 가능한 변수는?**

- 모든 요소의 갯수를 알아내는 변수 -> **count**
- 모든 키 값을 알아내는 변수 -> **keys** -> 리턴은 배열로 준다.
- 모든 값을 나타내는 변수 -> **vlaues**
- 딕셔너리가 현재 비어있는지 확인하는 변수 -> **isEmpty**

(참고) 함수 안에서 dictionary를 치고 command 마우스 커서를 누르면 해당 정보를 얻을 수 있다.
기능 무엇 있는 지 찾을 수 있음

* 딕셔너리에 대한 사용 예시

- 서버와 통신 할 때 가장 많이 쓰인다. 제이슨을 주로 쓰는데, 그 때에 키 구조로 데이터를 불러오는 것

실습 2
딕셔너리(Dictionary) 만들기

1. 사람의 특징을 가지고 있는 딕셔너리 변수를 만드세요
2. 음료수 이름과 가격의 짝을 이룬 딕셔너리를 만드세요
3. 음료수 이름을 받으면 가격을 돌려주는 함수를 만드세요.
   **위에 만든 딕셔너리 변수에서 값을 가져옵니다.
4. 음료수 이름과 가격을 받아서 추가 해주는 함수를 만드세요.
   **위에 만든 딕셔너리 변수에 값 변경

————————예제 답안 ————————

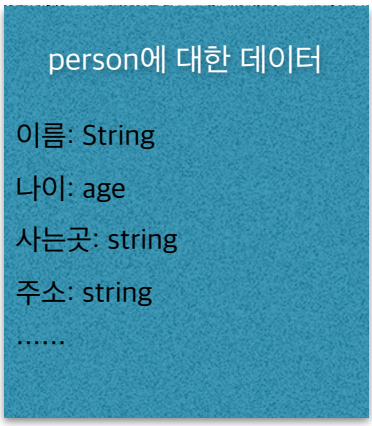
```
// 음료수 이름, 가격 배열
var drink:[String:Int] = ["Coke":1000,"soda":1200,"coffee":
    4000]

// 음료수 이름을 받으면 가격을 알려주는 함수
func costAtDrinkName(name:String) -> Int {
    return drink[name]!//옵셔널
}
// 딕셔너리에 추가하는 함수
func addDrink(drinkName:String, cost:Int)  {
    drink.updateValue(cost, forKey: drinkName)
}
```

실습 3
Data 구조 만들기

* 앨범 데이터 구조 만들기
  * 어떻게 딕셔너리 구조로 구성될지 생각해봅시다.

* 앨범에 있을 내용: 앨범 ID, 앨범 타이틀, 노래제목, 가사, 가수이름, 음원_URL, 앨범 이미지, 노래 ID

** 데이터 구조의 예시



———————— 해설 ————————————-

<앨범 해당>
앨범ID, 앨범타이틀, 가수 이름, 앨범이미지

<상세 내용 해당 - 노래>
노래제목, 가사, 음원_URL, 노래 ID

array dictionary
[앨범ID:String, 앨범타이틀:String, 가수이름: String, “songList”: [[노래1제목:String, 가사:String, 음원_URL:String, 노래 ID:String] , [노래2제목:String, 가사:String, 음원_URL:String, 노래 ID:String]]]

해당 자료는 일정한 가수의 앨범 안에 songList array를 두고, 각각의 노래별 정도를 딕셔너리로 넣어주는 구조를 취한다.

IOS 입문 5기 화이팅 입니다! :) ❤️

3. 튜플 (tuple)

1) 튜플이란?

* **두 개 이상의 자료**를 묶음으로 다루는 구조
* **변경 불가능**한 배열
* **자료형을 여러개 쓸 수 있다 ex 첫 번째는 String 두 번째는 Int …**
  * ex (“토끼”, 10, 100.2, “고양이”……)

    메모리 상에선 이런 모습

string	int	double	string

* 타입설정이 자유롭다. —> 그러나 한 번 정하면 변경 불가능. 대신 다양하게 쓸 수 있으니 좋다.
* 앞에서 썼던 .count 이런거 없음.. 심플하게 사용하고, 심플하게 관리할 수 있는 자료구조

2) 튜플 - 문법

*** 선언**

```
var tName:(Int, String, Int) = (1,"joo",3)
```

*** 사용**

* **인덱스 번호를 통해 사용한다.**

  tName.0

  tName.1

  tName.2

3) 예시

* 예시 1

<Multy Return Function>

```
func multyfunction() -> (Int,Int) {

    let firstNum : Int = 1
    let secondNum : Int = 3
    return (firstNum, secondNum)
}
```

<사용>

```
let tupleValue = (Int, Int) = multyfunction( )
tupleValue.0
tupleValue.1
```

IOS 입문 5기 화이팅 입니다! :) ❤️

    * return 타입을 여러개로 지정할 수 있음 (리턴 타입이 튜플인 것)
    * tupleVlaue.0은 fristNum 일 것

* 예시2

<Use Switch문>

```
let point:(Int,Int) = (1,1)

switch point
{
case (let x, 0) :
 print("좌표가 (\(x),0)인 x좌표 위의 점입니다.")
case (0, let y):
 print("좌표가 (0,\(y))인 y좌표 위의 점입니다.")
case let(x, y):
 print("좌표가 (\(x),\(y))인 점입니다.")
}
```

    * switch문의 응용

* 예시3

<Use For-in문>

```
let drinkData:[String:Int] = ["drink1":1000,"drink2":
800,"drink3":1300,"drink4":900]

for (key, value) in drinkData
{
    print("음료수 이름은 \(key) 이고 가격은 \(value) 입니다.")
}
```

    * for in문
        * 딕셔너리에서 값을 가져올 때
            * 딕셔너리는 key와 value 두 가지니까 for in문을 사용할 때 (key, value)로 넣으면 될 것
            * for in 반복문을 튜플로 적용하여 더 많은 딕셔너리 데이터들을 쉽게 가져 올 수 있다.

IOS 입문 5기 화이팅 입니다! :) ❤️

실습 3

튜플 만들어보기

- string, int 조합의 튜플을 만들어 보세요.
- string, bool, string 조합의 튜플을 만들어 보세요.
- 두 수를 받아서 두수의 덧셈과, 두수의 뺄셈값을 동시에 반환해주는 함수를 만들어 보세요.
- 음료수 딕셔너리에 있는 모든 음료이름과 가격 반환하는 함수

  *실습 2에서 만든 음료수 딕셔너리 사용

——— 예제 답안 ———

```
43      //- string, int 조합의 튜플을 만들어 보세요.
44      var tuple1:(Int,Int) = (1,2)
45
46      //string, bool, string 조합의 튜플을 만들어 보세요
47      var person:(String,Bool, String) = ("주영민",true,"강사")
48
49      //두 수를 받아서 두수의 덧셈과, 두수의 뺄셈값을 동시에 반환해주는 함수를 만들어 보세요.
50      func calculator(num1:Int, num2:Int) -> (Int,Int){
51      let plus:Int = num1 + num2
52      let minus:Int = num1 - num2
53      return (plus,minus)
54      }
```

```
// 음료수 딕셔너리에 있는 모든 음료이름과 가격 반환하는 함수
func printAllBeverage()
{
    for (key,value) in drink
    {
        print(key,value)
    }
}

// 이름과 가격을 따로 배열로 반환
func nameAndCost() -> ([String], [Int])
{
    var allName:[String] = []
    var allCost:[Int] = []

    for (name, cost) in drink
    {
        allName.append(name)
        allCost.append(cost)
    }
    return (allName, allCost)
}
```

IOS 입문 5기 화이팅 입니다! :) ❤️