화살이(가) 표시된 사진

자동 생성된 설명

iOS 4주차 레포트

|  |  |
| --- | --- |
| 과목명 : iOS 프로그래밍 | 담당교수 : 한성현 교수님 |
| 학과 : 컴퓨터소프트웨어학과 | 학년 : 2학년 |
| 학번 : 201912047 | 이름 : 정정욱 |
| 제출일 : 2022.09.27 |  |

Int **형의 크기**

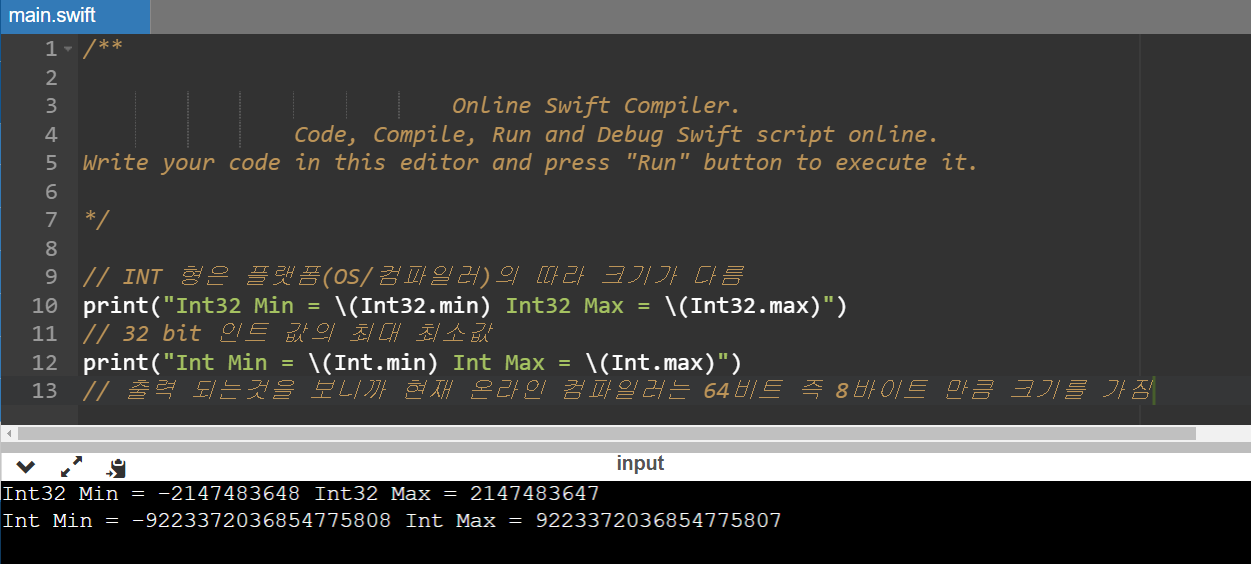
*/컴파일러)의 따라 크기가 다름*

print("Int32 Min = **\(**Int32.min**)** Int32 Max = **\(**Int32.max**)**")

*// 32 bit 인트 값의 최대 최소값*

print("Int Min = **\(**Int.min**)** Int Max = **\(**Int.max**)**")

*// 출력 되는것을 보니까 현재 온라인 컴파일러는 64비트 즉 8바이트 만큼 크기를 가짐*



**문자 데이터 타입 : Character 선언 시 중요 포인트**

**var** myChar2 = ":"

**var** myChar3 : Character = "X" *//:Character 생략불가,생략하면 String형임*

*// 그래서 Character을 써야하는 상황일때는 꼭 : Character 써줘야함*

print(type(of: myChar2))

print(type(of: myChar3))

*//유니코드를 이용하여 변수에 문자 'X'를 할당 \u 다음에 코드 값 넣으면 됨*

**var** myChar4 = "\u{0058}"

print(myChar4)

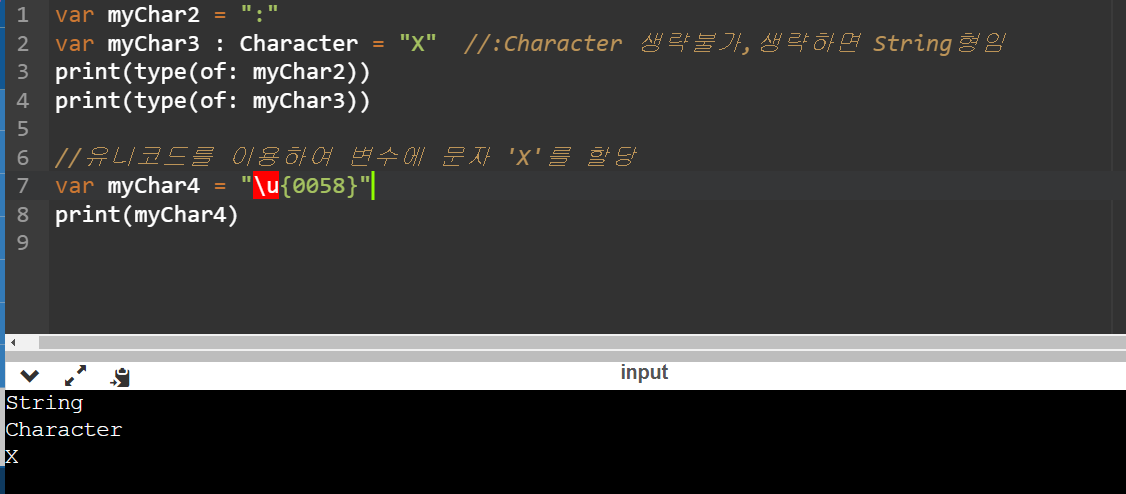
*/\**

*String*

*Character*

*X*

*\*/*



**문자열 데이터 타입 : String 이용시 문자열 보간을 사용하여 문자열과 변수, 상수, 표현식, 함수 호출의 조합으로 만들 수도 있음.**

**var** userName = "Kim" *// : String 생략하는 것이 일반적임*

**var** age = 20

**var** message = "**\(**userName**)**의 나이는 **\(**age**)**입니다."

print(message) *// Kim의 나이는 20입니다.*

*// Kim의 나이는 20입니다*

텍스트이(가) 표시된 사진

자동 생성된 설명

**타입 어노테이션과 타입 추론**

변수 또는 상수가 코드 내에서 선언되는 시점에 타입 어노테이션(type annotation)을 사용하는 것

* + 변수 또는 상수 이름 다음에 타입 선언을 두면 됨
  + Int 타입의 userCount 라는 이름의 변수를 선언

var userCount : Int = 10 // : Int가 type annotation

즉 직접 선언할 때 타입 명시하는 것을 타입 어노 테이션이라고함 (: int) 직접 쓰는 것

위에 : Int 를 생략하는게 타입 타입 추론(type inference) 이라고 함

* + var signalStrength = 2.231 // var signalStrength : Double = 2.231
  + let companyName = "My Company"
  + signalStrength라는 변수를 Double 타입(스위프트의 타입 추론에서 모든 부동 소수점 수는 Double 타입)
  + companyName이라는 상수는 String 타입으로 간주

**상수와 변수 선언하고 값 수정하기**

애플은 코드의 효율성과 실행 성능을 높이기 위해서 변수(var)보다는 상수(let)를 사용하라고 권장

*// let으로 만든 상수는 초기값에서 변경이 불가능함*

**let** x = 10

print(x)

x = 20

print(x)

*/\**

*main.swift:4:1: error: cannot assign to value: 'x' is a 'let' constant*

*x = 20*

*^*

*main.swift:2:1: note: change 'let' to 'var' to make it mutable*

*let x = 10*

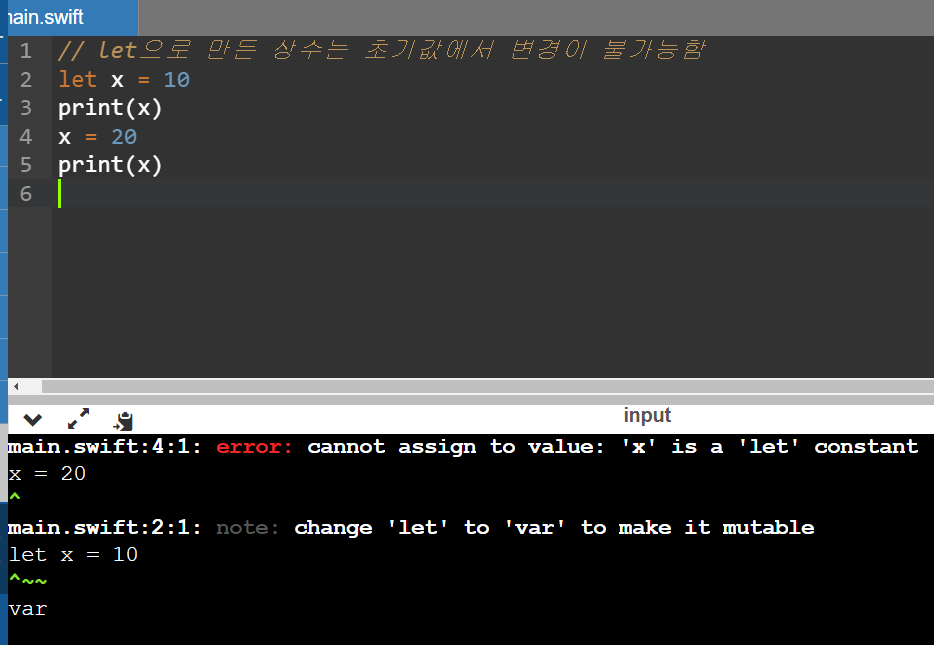
*^~~*

*var*

*오류 내용을 보면 constant라 변경이 불가능해 let 키워드를 var 변수로*

*(Mutable 값을 변경 가능하게)바꾸라고 나와있음*

*\*/*



**튜플(Tuple)**

* 튜플은 스위프트 프로그래밍 언어에서 가장 강력한 기능 중 하나
* 여러 값을 하나의 개체에 일시적으로 묶는 방법
* 튜플에 저장되는 항목들은 어떠한 타입도 될 수 있으며, 저장된 값들이 모두 동일한 타입이어야 한다는 제약도 없음

**let** myTuple = (10, 12.1, "Hi")

**var** myString = myTuple.2

print(myString) *//출력되는 값은?*

*// Hi 인덱스 값을 사용하여 값을 추출함*

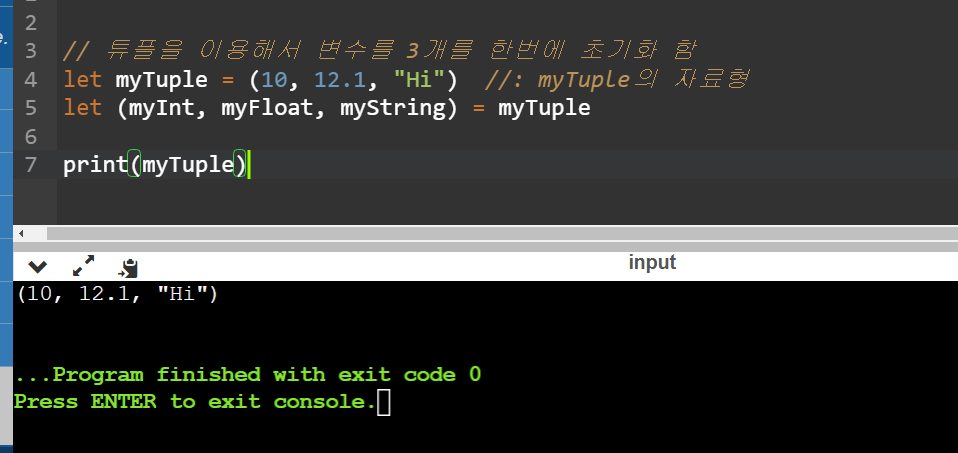
**단 한 줄의 코드로 튜플의 모든 값을 추출하여 변수 또는 상수에 할당**

*// 튜플을 이용해서 변수를 3개를 한번에 초기화 함*

**let** myTuple = (10, 12.1, "Hi") *//: myTuple의 자료형*

**let** (myInt, myFloat, myString) = myTuple

print(myTuple)



**튜플의 값을 선택적으로 추출하는 데 사용될 수 있으며,**

**무시하고 싶은 값에 밑줄을 사용하면 그 값은 무시**

**let** myTuple = (10, 12.1, "Hi")

**var** (myInt, **\_**, myString) = myTuple *//부동 소수점 수는 무시*

**튜플을 생성할 때 각 값에 이름을 할당할 수도 있음**

**let** myTuple2 = (count: 10, length: 12.1, message: "Hi")

print(myTuple2.message)

// Hi

**튜플 정리 전체적인 코드**

**let** myTuple = (10, 12.1, "Hi")

print(myTuple.0)

print(type(of:myTuple)) *// 자료형은 튜플안에 선언 한 값의 자료형과*

*튜플 자료형을 나타내는 ()로 감쌓아 나옴*

*// let (myInt, MyFloat, myString) = myTuple*

*// var (myInt, \_, myString) = myTuple //부동 소수점 수는 무시*

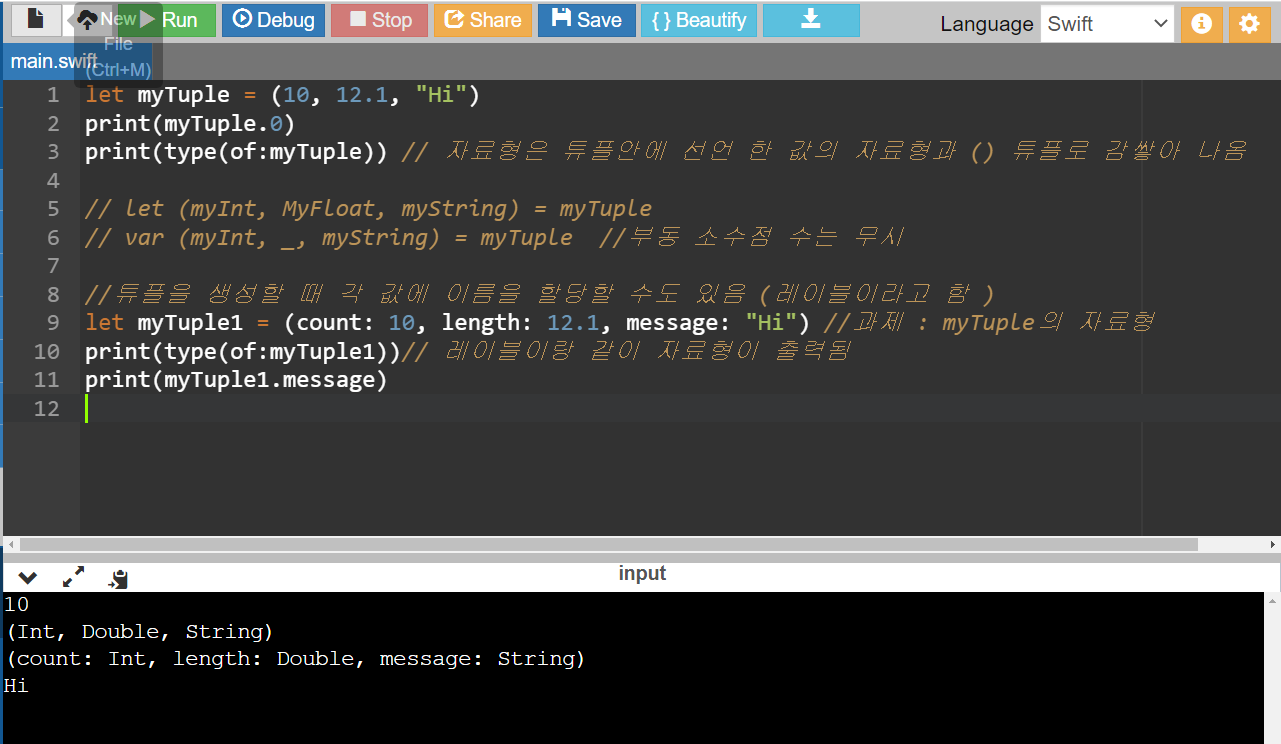
*//튜플을 생성할 때 각 값에 이름을 할당할 수도 있음 (레이블이라고 함 )*

**let** myTuple1 = (count: 10, length: 12.1, message: "Hi") *//과제 : myTuple의 자료형*

print(type(of:myTuple1))*// 레이블이랑 같이 자료형이 출력됨*

print(myTuple1.message)

*// 즉 인덱스 번호를 사용할수 있지만 값 별로 이름을 지정해서 이름을 불러 사용할수 있는것임*



\*튜플 시험\*

**스위프트의 가장 어렵고 막강한 기능 중 하나인 옵셔널**

* **옵셔널 타입만이 값을 갖지 않는다는 의미의 nil 값을 가질 수 있음**

*// 옵셔널*

*// Int형 값을 저장 or 값이 없음(nil)*

*// nil 앱 개발 하거나 값을 처리하는데 있어서 값이 없을 때를 처리하기 위해 고안된 자료형*

print(10)

print(Int("12"))

print(Int("hi"))

*// 10*

*// Optional(12)*

*// nil*

*Null과 nil 의 차이점*

*텍스트이(가) 표시된 사진

자동 생성된 설명*

[*https://seolhee2750.tistory.com/10*](https://seolhee2750.tistory.com/10) *참조 문헌*

*내가 정리한 null 과 nil 개념*

*Null은 값을 아직 가지지 않는 상태 즉 할당 받지 않는 상태*

*Nil은 값이 없는 정말* (無: 없음)무 *를 나타낸다.*

**본격 옵셔널 값 추출 방법들 : 옵셔널 을 푸는 방법들**

***옵셔널 타입 강제 언래핑(forced unwrapping) 1***

**var** x : Int? *//옵셔널 정수형 변수 x 선언*

**var** y : Int = 0

x = 10 *// 주석처리하면?*

print(x) *// Optional(10)*

print(x!) *// forced unwrapping해서 10이 나옴*

print(y)

*//x = x+2 //가능? X*

x = x! + 2 *// 옵셔널 값은 그냥연산 못함 풀어서 사용해야함 ! 사용하는forced unwrapping 사용*

print(x)

*//y = x //가능? x*

y = x! *// 일반 변수에 넣을때도 값을 풀어서 넣어줘야함 일반 자료형 변수에*

*// 옵셔널값 그냥 못 넣음*

print(y)

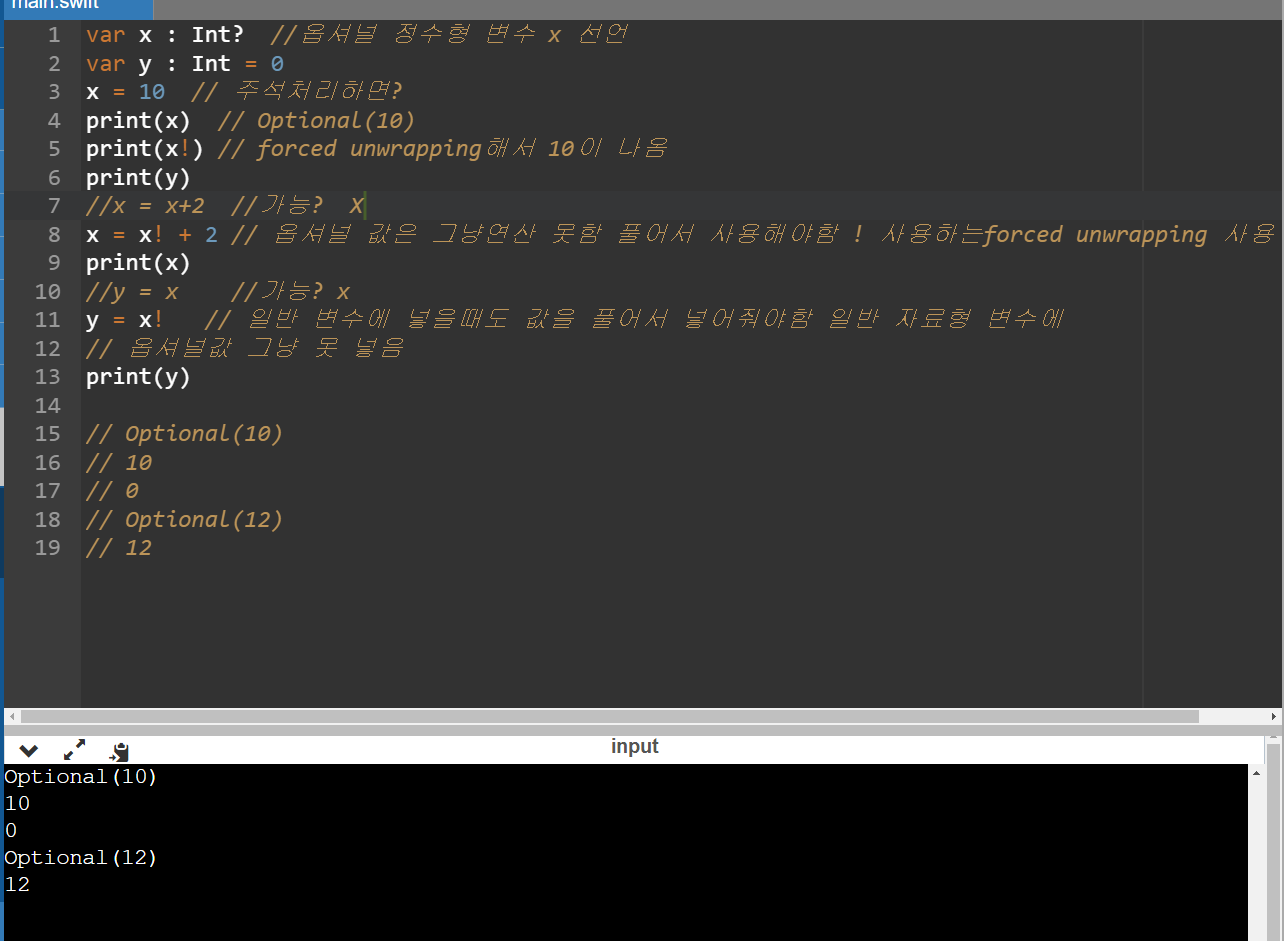
*// Optional(10)*

*// 10*

*// 0*

*// Optional(12)*

*// 12*



**옵셔널 값을 풀 때 주의 할점!**

**var** x : Int? *//옵셔널 변수는 초기값을 안주면 nil이 들어간다*

*//x = 10 // 주석처리하면?*

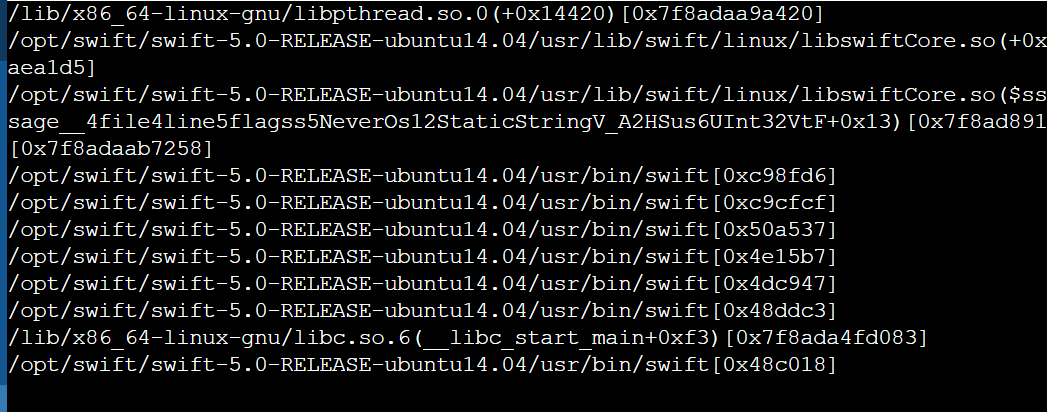
print(x)

*//print(x!) // nil 값이 들어간 옵셔널 변수를 강제로 풀면 Crash가 난다*

Crash란?

<https://ko.wikipedia.org/wiki/%EC%B6%A9%EB%8F%8C_(%EC%BB%B4%ED%93%A8%ED%8C%85)>





**스위프트에서 옵셔널을 풀때 가장 많이 쓰는 방법1**

**If-else 문과 *forced unwrapping***

**var** x : Int?

x = 10

**if** x != **nil** { *// 옵셔널은 닐이 아닐때만 풀어야 하기때문에 이렇게 if문에 의해서*

print(x!) *// nil이 아닌지 체크하고 풀어야함*

*// 주의 : if x!=nil 이라고 쓰면 안됨*

*// 위와 같이 쓰면 x변수의 옵셔널을 풀고 nil 값을 넣어라가 되기 때문이고*

*// 심지어 x가 옵셔널이 풀려서 nil 값도 못넣기 때문임*

}

**else** {

print("nil")

}

**var** x1 : Int? *//nil*

**if** x1 != **nil** {

print(x1!)

}

**else** {

print("nil")

}



**스위프트에서 옵셔널을 풀때 가장 많이 쓰는 방법2**

**옵셔널 바인딩(optional binding)을 이용하여 옵셔널에 할당된 값을 임시 변수 또는 상수에 할당**

**var** x : Int?

x = 10

**if** **let** xx = x { *//옵셔널 변수 x가 값(10)이 있으므로 언래핑해서 일반 상수 xx에 대입하고 if문 실행*

print(x, xx) *// 키워드가 let이 아니고 var면 안에서 값을 가지고 놀겠다는 뜻임*

}

**else** {

print("nil")

}

**var** x1 : Int?

**if** **let** xx = x1 { *//옵셔널 변수 x1이 값이 없어서 if문의 조건이 거짓이 되어 if문 실행하지 않고 else로 감*

print(xx)

}

**else** {

print("nil")

}

*//Optional(10) 10*

*//nil*

**스위프트에서 옵셔널을 풀때 가장 많이 쓰는 방법2 응용1**

**var** x : Int?

x = 10

**if** **let** x = x { *// 옵셔널 바인딩시 이렇게 같은 이름을 줘도 상관 없음*

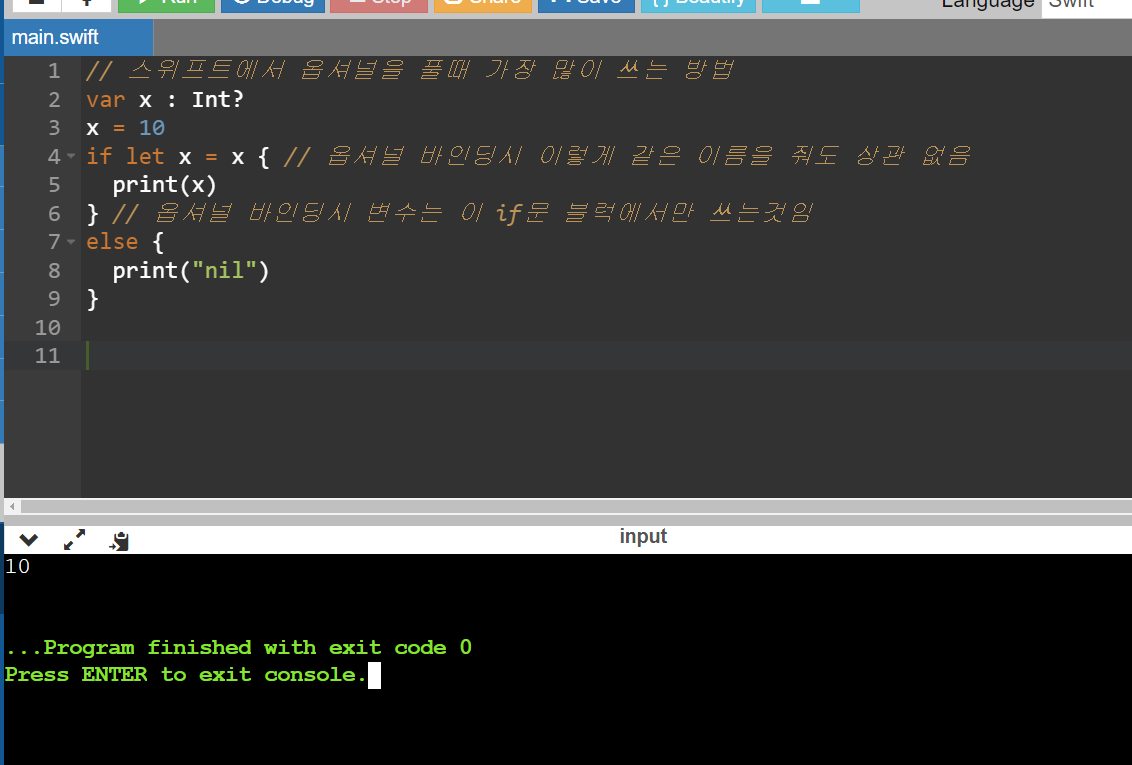
print(x)

} *// 옵셔널 바인딩시 변수는 이 if문 블럭에서만 쓰는것임*

**else** {

print("nil")

}



**스위프트에서 옵셔널을 풀때 가장 많이 쓰는 방법2 응용2**

이번 9월 12일 Swift 5.7버전업데이트 하면 이제 이런식으로 작성 가능함

<https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html>

텍스트이(가) 표시된 사진

자동 생성된 설명

**var** x : Int?

x = 10

**if** **let** x { *// if let x 이걸로 옵셔널 바인딩을 한줄에 끝내버림*

print(x) *// 옵셔널 바인딩시 변수는 이 if문 블럭에서만 쓰는것임*

}

**else** {

print("nil")

}

5.7 버전 사용 가능한 온라인 컴파일러 주소

<https://swiftfiddle.com/?c=H4sIAAAAAAAAA32Q3UrDQBCFX2XYKwti9UKEgnjtO%2ByNSAqBMEgNJRAKka4%2FxKBFtzRCt3pRLQWFECPkmTKTd3DXH7zzauAMc75zJhbHoie6XeB0yXPVatWmNc8mrObA%2BQefa1ITNgrau4SmGTRFwotnoNUNmwr4vqBUAxVrKqcSh0cDiKAHhxgeSIxgH3a2Jfp9CLzQLmJwnF9Te6XZ1KTXfG1ZpqKny6Z0hAc2yZfwohyZlznd2jE%2Bayqrzy7YZBIBTgY%2BhhtRBySO%2FnGm94SvchfTJfb79FoD1Zoe375r0ir76dGUY14oa%2BcFpx7Efwwp0A%2Bk6FiQRIliUwzt03a39sToE6QOsK1AAQAA>

**여러 옵셔널을 언래핑**

//여러 옵셔널 변수를 한번에 언래핑하는 방법은 콤마 사용

+ let 키워드를 둘 다 붙여줘야함

**var** pet1: String?

**var** pet2: String?

pet1 = "cat"

pet2 = "dog"

**if** **let** firstPet = pet1, **let** secondPet = pet2 {

*// 스위프트에서 ,는 and 임 즉 두개다 참이여야함 하나라도 nill이면 else쪽으로감*

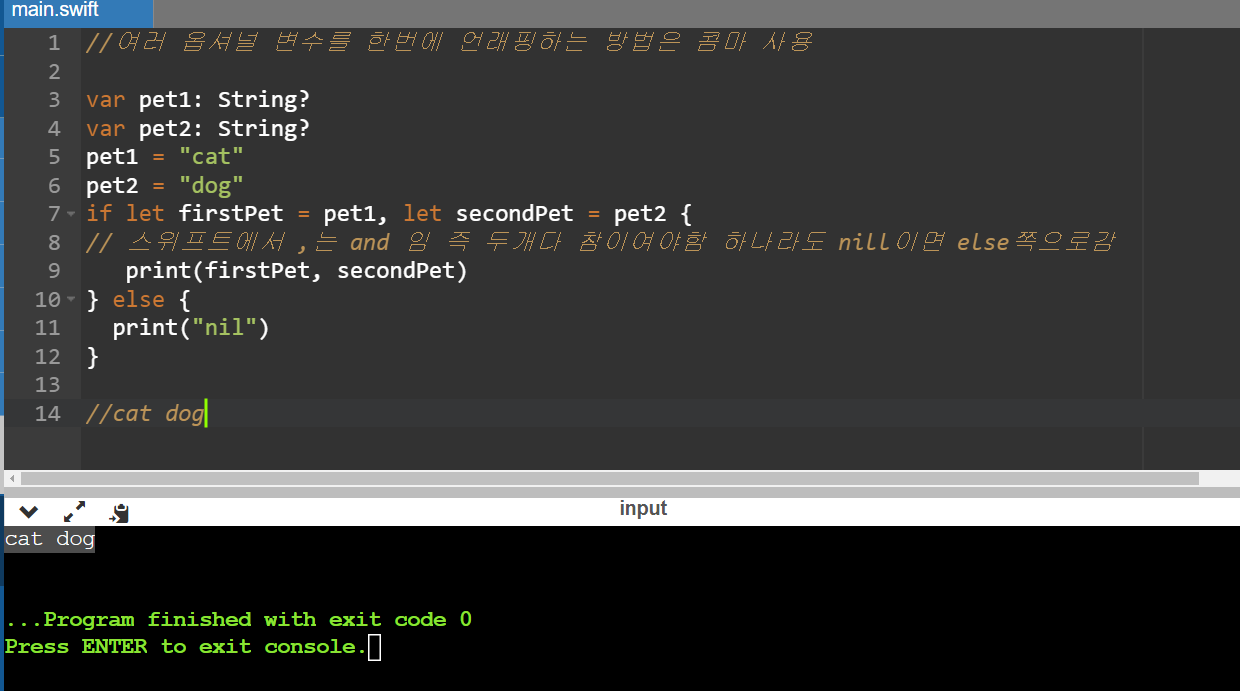
print(firstPet, secondPet)

} **else** {

print("nil")

}

*//cat dog*



즉 둘다 nil이 아니여야 참이 됨

* Any
  + Any can represent an instance of any type at all, including function types.
  + 클래스, 구조체, 열거형, 함수타입 즉 어떤 것이라도 저장 가능함

**var** x: Any = "Hi"

print(x, type(of:x))

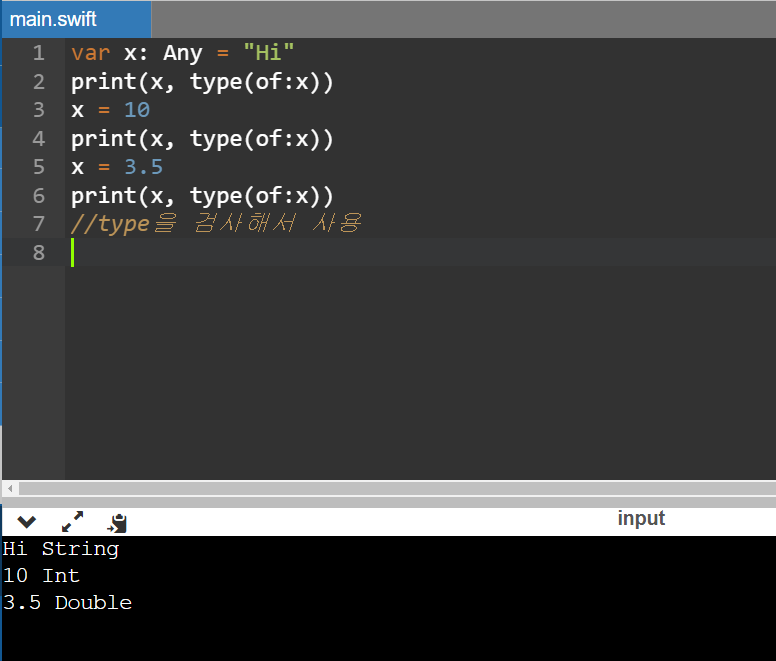
x = 10

print(x, type(of:x))

x = 3.5

print(x, type(of:x))

*//type을 검사해서 사용*



**대부분의 Swift 연산자는 다른 언어들과 비슷하지만 그중 몇가지 특징을 살펴보겠다.**

**증감 연산자 유무**

* + x++ // x를 1 증가시킴, Swift 3에서 없어짐, x+=1
  + x-- // x를 1 감소시킴, Swift 3에서 없어짐, x-=1

Swift에서는 증감 연산자가 Swift3 버전에서 사라졌다.

**범위 연산자 실습**

*//One-Sided Ranges*

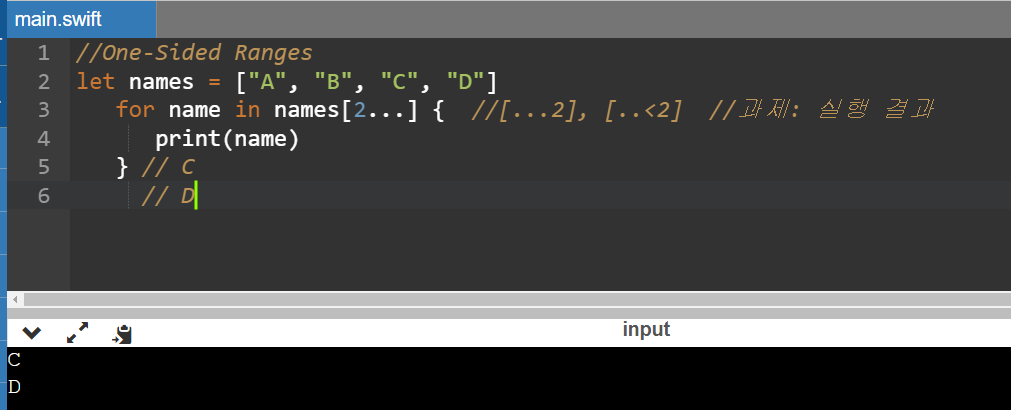
**let** names = ["A", "B", "C", "D"]

**for** name **in** names[2...] { *//[...2], [..<2] //과제: 실행 결과*

print(name)

} *// C*

*// D*



*// 반 열린 범위 연산자(half-open range operator)*

*// x..<y*

*// x부터 시작하여 y가 포함되지 않는 모든 숫자*

**let** names = ["A", "B", "C", "D"]

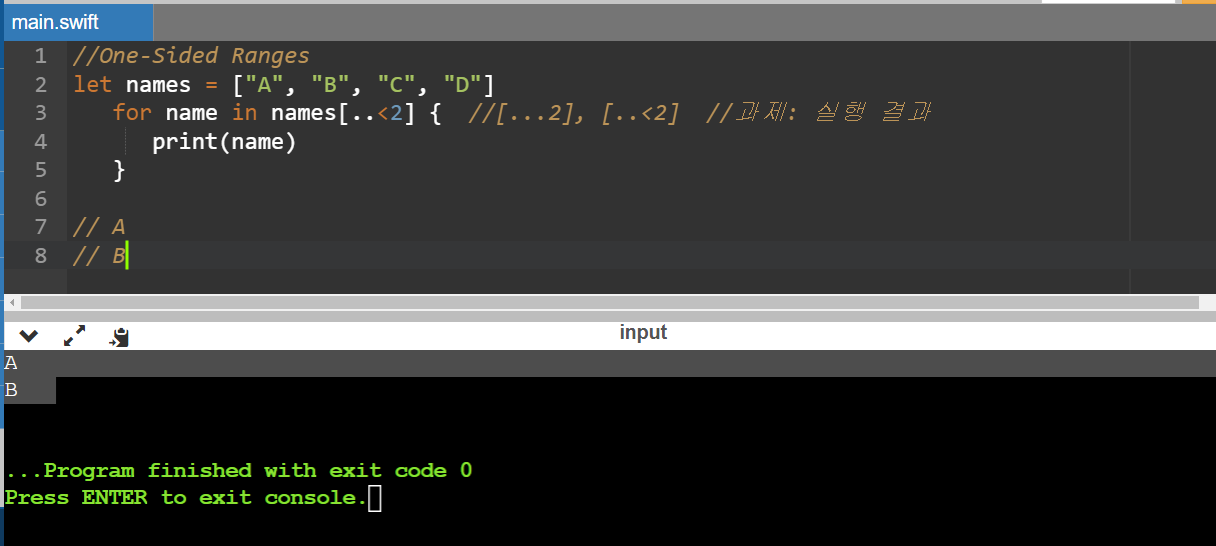
**for** name **in** names[..<2] { *//[...2], [..<2] //과제: 실행 결과*

print(name)

}

*// A*

*// B*



//닫힌 범위 연산자(closed range operator)

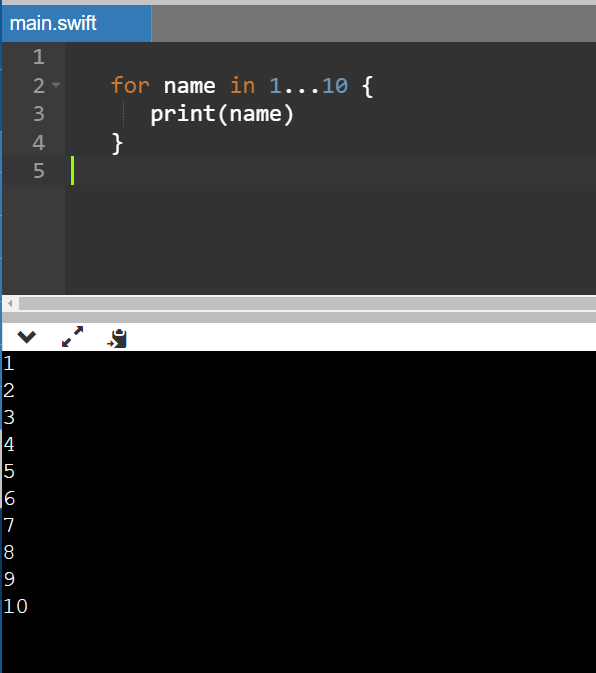
//x...y

//x에서 시작하여 y로 끝나는 범위에 포함된 숫자

**for** name **in** 1...10 {

print(name)

}

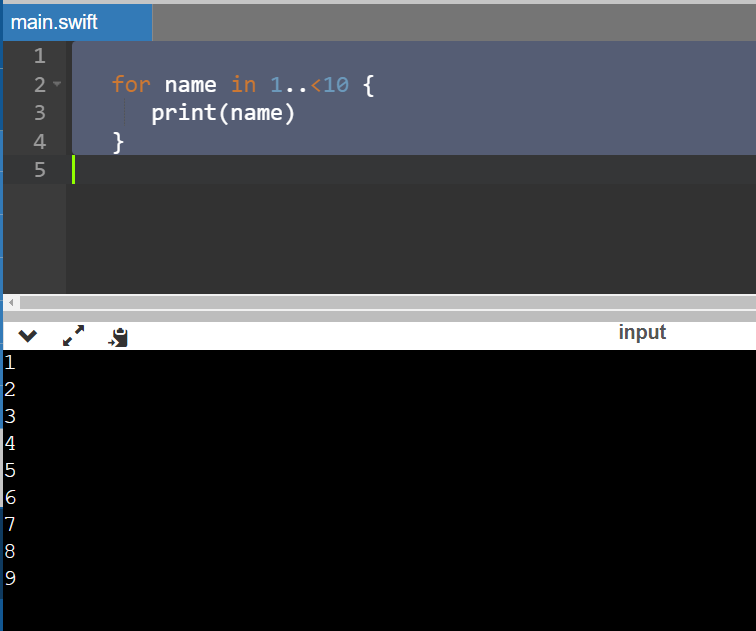


반 열림 범위 연산자 for in 문 사용시

**for** name **in** 1..<10 {

print(name)

}



**Nil-Coalescing Operator (Nil합병연산자) ??**

**let** defaultAge = 1

**var** age : Int?

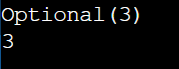
age = 3

print(age) *//과제:값은?*

**var** myAge = age ?? defaultAge *// 닐을 풀때 사용하는 연산자임 닐이라면 우쯕 값이 들어감*

*//age가 nil이 아니므로 언래핑된 값이 나옴*

print(myAge) *//과제: 값은?*



**장점 옵셔널 값을 따로 (• Optional Binding - 옵셔널 바인딩, • Force Unwrapping - 강제 추출)할 필요 없이 연산자로 한번에 풀 수 있음**

시험!!!!

// 옵셔널변수 ?? nil일 때 할당되는 값

// 옵셔널 변수의 값이 nil이면 ?? 다음 값으로 할당됨

// 옵셔널 변수의 값이 nil이 아니면 언래핑된 값이 나옴

**let** defaultColor = "black"

**var** userDefinedColor: String? *// defaults to nil*

**var** myColor = userDefinedColor ?? defaultColor

*//nil이므로 defaultColor인 black으로 할당됨*

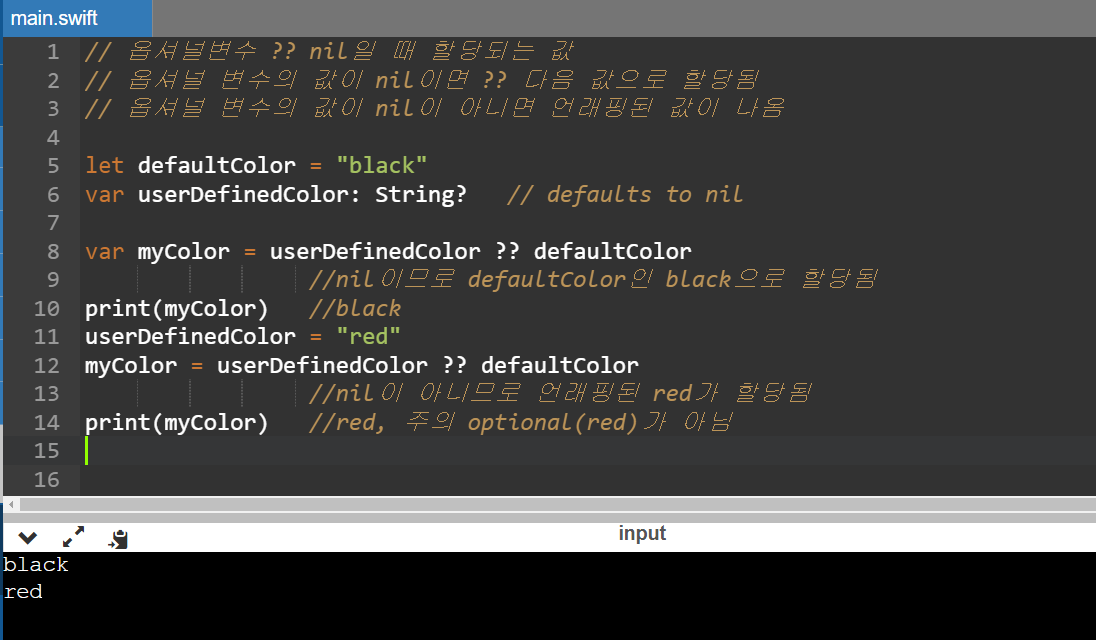
print(myColor) *//black*

userDefinedColor = "red"

myColor = userDefinedColor ?? defaultColor

*//nil이 아니므로 언래핑된 red가 할당됨*

print(myColor) *//red, 주의 optional(red)가 아님*



For-in 반복문 사용과 \_ 참조체

**for** **\_** **in** 1...5 {

print("안녕")

}

**for** i **in** 1...5 {

print("**\(**i**)** 안녕") *//\_로 참조체(i) 생략 가능*

}

*// 안녕*

*// 안녕*

*// 안녕*

*// 안녕*

*// 안녕*

*// 1 안녕*

*// 2 안녕*

*// 3 안녕*

*// 4 안녕*

*// 5 안녕*

**반복문에서 빠져나오기(break) 주의**

**for** i **in** 1..<10 {

**if** i > 5 {**break**} *//if 조건 다음에 실행줄이 한줄이라도 중괄호 사용해 줘야함*

*//중요!! 에러 수정 : error: expected '{' after 'if' condition*

print(i)

}



* 다른 프로그래밍 언어들과는 다르게, Swift에서는 if 문 다음의 실행 코드가 한 줄이라도 중괄호({})를 필수적으로 사용해야 한다.

**if문 조건에서 콤마의 의미(중요)**

**var** a = 1

**var** b = 2

**var** c = 3

**var** d = 4

**if** a < b && d > c {

print("yes")

}

**if** a < b, d > c { *//과제, if문에서 콤마는 AND(&&)의 의미*

print("yes")

}

*// yes*

*// yes*

**다중 if-else문 : BMI 계산 결과 판정**

**let** weight = 60.0

**let** height = 170.0

**let** bmi = weight / (height\*height\*0.0001) *// kg/m\*m*

**var** body = ""

**if** bmi >= 40 {

body = "3단계 비만"

} **else** **if** bmi >= 30 && bmi < 40 {

body = "2단계 비만"

} **else** **if** bmi >= 25 && bmi < 30 {

body = "1단계 비만"

} **else** **if** bmi >= 18.5 && bmi < 25 {

body = "정상"

} **else** {

body = "저체중"

}

print("BMI:**\(**bmi**)**, 판정:**\(**body**)**")

텍스트이(가) 표시된 사진

자동 생성된 설명