



iOS/Swift GameApp yagom

Created	@October 26, 2020 1:06 AM
Created By	Kevin
Last Edited Time	@August 1, 2021 4:50 PM
Participants	
Property	Yagom
Type	언어

▼ 완성된 코드

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
    @IBOutlet weak var sliderValueLabel: UILabel!
    @IBOutlet weak var minimumValueLabel: UILabel!
    @IBOutlet weak var maximumValueLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
        reset()
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){
        print(sender.value)
        let integerValue: Int = Int(sender.value)
        sliderValueLabel.text = String(integerValue)
    }

    func showAlert(myMessage: String){ // nil은 없다라는 의미임
        let alert = UIAlertController(title: nil,
                                     message: myMessage,
                                     preferredStyle: .alert)
        let okAction = UIAlertAction(title: "OK",
                                     style: .default) { (action) in
            self.reset()
        }
        alert.addAction(okAction)
        present(alert,
                animated: true,
                completion: nil)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){
        print(slider.value)
        let hitValue: Int = Int(slider.value)
        slider.value = Float(hitValue)
        tryCount = tryCount + 1
        // tryCountLabel.text = String(tryCount) + " / 5"
        tryCountLabel.text = "\((tryCount) / 5)"

        if randomValue == hitValue {
            print("YOU HIT!")
            showAlert(myMessage: "YOU HIT!")
            reset()
            // return
        } else if tryCount >= 5 {
            print("YOU LOSE...")
            showAlert(myMessage: "YOU LOSE...")
            reset()
            // return
        }
    }
}
```

```

    } else if randomValue > hitValue {
        slider.minimumValue = Float(hitValue)
        // minValueLabel.text = String(slider.minimumValue)
        minValueLabel.text = "\u2022(slider.minimumValue)"
    } else {
        slider.maximumValue = Float(hitValue)
        maxValueLabel.text = String(slider.maximumValue)
    }
}

@IBAction func touchUpResetButton(_ sender:UIButton){
    print("Touch Up Reset Button!")
    reset()
}

func reset(){
    print("Reset!")
    randomValue = Int.random(in: 0...30)
    print(randomValue)
    tryCount = 0
    tryCountLabel.text = "0 / 5"
    slider.minimumValue = 0
    slider.maximumValue = 30
    slider.value = 15
    minValueLabel.text = "0"
    maxValueLabel.text = "30"
    sliderValueLabel.text = "15"
}
}

```

▼ Contents

- [1. Project 개요](#)
- [2. Setting](#)
- [3. Main Story Board](#)
- [4. ViewController.swift](#)
- [5. Event Driven Programming](#)
- [6. Asset Catalog](#)
- [7. Styling UI](#)
- [8. Function](#)
- [9. Variables & Constants 선언](#)
- [10. Comparing values / Terminating functions](#)
- [11. Conditional Execution](#)
- [12. Showing Alerts](#)
- [13. Creating Credit Scene](#)
- [14. Storyboard Segue \(scene에서 scene으로 넘기기\)](#)

1. Project 개요

- To-Do List

1. Add Slider
2. Add 'HIT Button'
3. Receive value changed events from the slider
4. Add 'RESET Button'
5. Add labels presenting information
6. Generate the random number
7. Compare the random number with input number
8. Show alerts
9. Implement 'reset' feature
10. Add 'Credit' view

2. Setting

- Single View App → App 으로 업데이트하면서 변경된 듯
- Orientation : Landscape

3. Main Story Board

- 아이폰 상 (기종 선택 가능)의 화면 미리보기 및 UX Flow 확인

4. ViewController.swift

- **IBOutlet** : interface builder에 올라와 있는 UI 요소의 value를 가져오거나 또는 그것을 코드로 가져와 활용하고 싶을 때 쓴다.
- **IBAction** : interface builder에 어떤 요소들이 이벤트를 받았을 때, 그것에 반응하기 위한 Action 코드를 작성하기 위해 쓴다.
- *IB : Interface Builder (storyboard에서 UI와 연결시킬 내용)

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func sliderValueChanged(_ sender: UISlider){ // 이벤트에 반응하는 Action이다. (slider를 움직이면 value가 바뀌는 기능을 구현하는 함수이다)
        print(sender.value)
    }
}
```

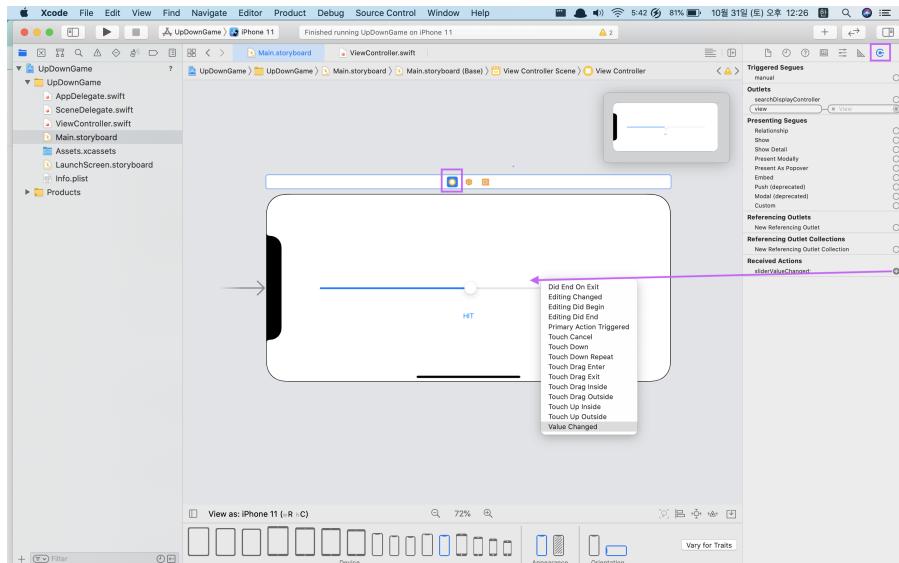
▼ viewDidLoad() method 란?

뷰의 상태변화 감지 화면에 보여지는 뷰가 바뀌면, 뷰 컨트롤러는 자동으로 특정한 메서드를 호출해서 이 변화에 다른 클래스들이 반응할 수 있도록 합니다

<https://etst.tistory.com/90>

▼ 해당 코드를 slider에 연결하는 방법

(view controller > connections inspector) 드래그 하면 된다!



slider를 움직이면 console 화면에서 value가 바뀌는 것을 확인할 수 있다.

• (스토리보드의 UI를 사용하는) 변수 선언 - Syntax

```
var randomVariables: Int = 0

var slider: UISlider!

@IBOutlet weak var slider: UISlider!
```

```
@IBOutlet weak var slider: UISlider!
1) slider라는 변수를 선언하고
2) 스토리보드 위의 UISlider의 값을 가져온다.
```

*참고

```
일반적인 변수 선언 (data type: Int, 초기값은 0이다)
var randomVariables: Int = 0
```

▼ slider의 value를 button을 통해 출력하는 기능

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider! // 4. Outlet은 스토리보드 위의 특정 요소를 코드에 연결해준다. 여기서 변수 slider를 선언한다.

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

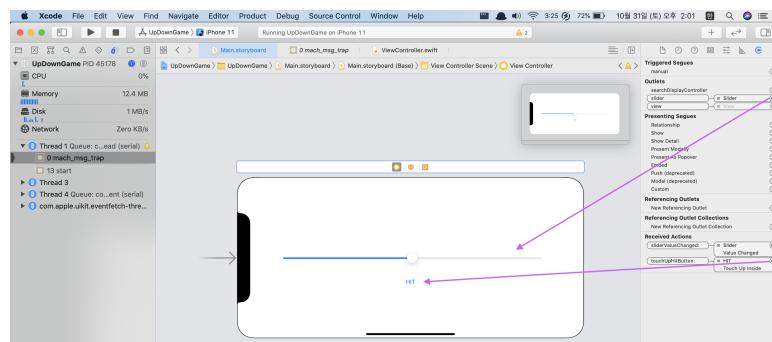
    @IBAction func sliderValueChanged(_ sender:UISlider){ // 1. sender인 slider가 움직이면서 함수가 실행되고, sender에 넘어온 value를 이용해
        print(sender.value)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){ // 2. 반면, 이건 sender가 button이라서 value 활용이 불가하다. 화면의 slider를 코드
        // print(sender.value)
        print(slider.value) // 3. slider라는 변수의 값을 출력하겠다. (이후에 스토리보드에서 이 코드와 연결시키면 된다.)
    }
}
```

connections inspector에서

위 Outlets의 slider 변수와 스토리보드 위의 Slider을 드래그하여 연결시켜주고,

아래 Action의 touchUpHitButton도 마찬가지로 Button에 연결 해준다. (touchup inside 선택)



slider를 움직이면 아래 console 창에서 value가 변하는 것을 확인할 수 있다.

HIT button을 누르면 현재 value가 출력된다.

▼ 동일한 방법으로 Reset button 생성

1. 코딩

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){
        print(sender.value)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){
        // print(sender.value)
        print(slider.value)
    }
}
```

```

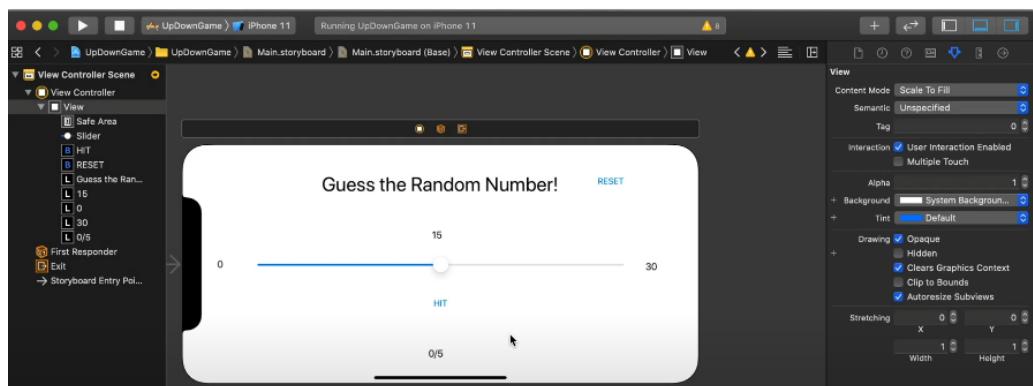
    @IBAction func touchUpResetButton(_ sender:UIButton){
        print("Touch up reset button") // reset button을 누르면 출력
    }
}

```

2. storyboard로 이동해서 우측상단에 RESET button을 생성한다.
3. connection inspector에서 해당 Action과 RESET button을 Drag&Drop으로 연결시킨다.
(*함수명 수정 시 참고
ViewController에서 함수명을 임의로 수정하면 [연결 오류](#)가 발생하므로
[함수 우클릭>Refactor>Rename](#)에서 수정하면 일괄 적용된다.)

▼ Label 생성

- Object library - Label (read-only text 즉, 읽기전용 텍스트)
- Label1 - Attribute inspector
 - Text : Title을 입력한다. "Guess the Random Number!"
 - Font : system 17.0으로 고정되어 있으면 사용자가 시스템 설정에서 글자크기를 바꿔도 적용되지 않는다. 따라서 [Text style>Large Title](#)로 지정한다.
- Label2,3,4,5
 - slider의 최소값, 최대값 (body) 그리고 현재값 (headline)을 보여준다.
 - 게임을 몇 회 진행했는지 (headline) 보여준다.



5. Event Driven Programming

유저로부터 어떤 이벤트가 발생했을 때 이런 코드를 실행하겠다고 미리 코드를 짜놓는 방식이다.

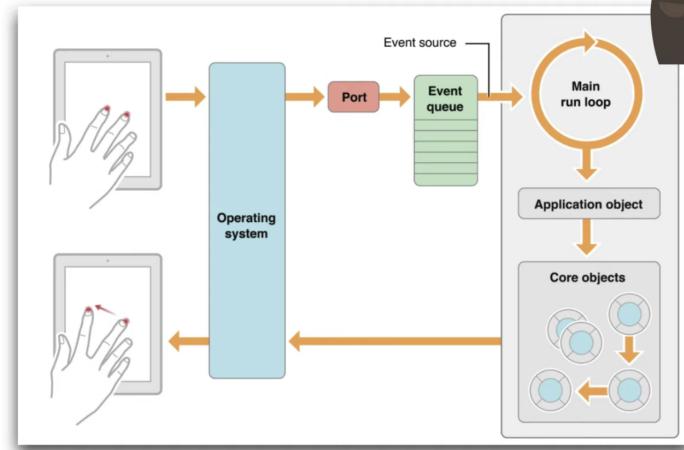
▼ Concept

- 사용자가 event를 주면 Core objects에서 작성한 코드가 실행된다.
그 전까지는 대기 상태이다.

Event Driven Programming

yagom

A category of programming
in which the flow of the app is determined by events:
system events and user actions.



6. Asset Catalog

▼ Concept

Asset을 모아둔 Catalog로 이미지 등을 세트로 관리할 수 있다.

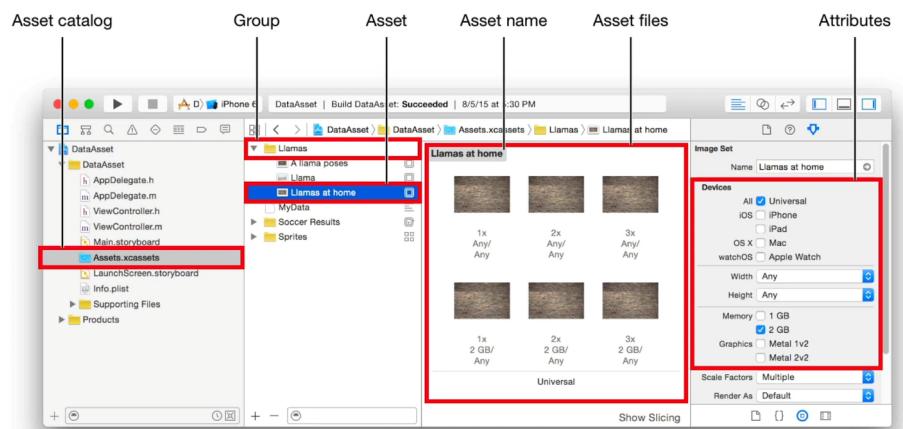
Asset의 종류는 App Icon 외에도 Color set, Data Set (Sound, Docs, Videos 등), Image Set, Launch image, Sticker, Watch complication 등이 있다.

Asset Catalog

yagom

A tool to manage assets like images that are used by your app as part of its user interface (UI).

Asset catalogs simplify access to app resources by mapping between named assets and one or more files targeted for different device attributes



▼ Importing Assets

- Project>Targets>하단 App Icons Source를 Do not use asset catalogs로 변경한다.
- Assets.xcassets>AppIcon (앱의 아이콘을 표현하는 기능)
Yagom의 Asset파일3 추가 (Import)
- 다시 Project>Targets>하단 App Icons Source를 Use Asset Catalogs-Asset migrate로 변경한다.
App Icons Source에 자동지정된 AppIcon-2를 AppIcon으로 변경한다.

7. Styling UI

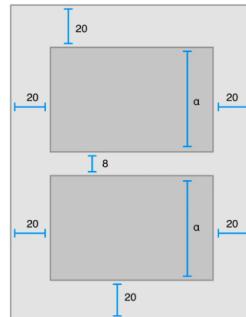
- storyboard>View (전체화면이 자동선택됨)>Attribute inspector에서 Background 색상을 변경한다.
custom-HEX #7FB8A0 (RGB 127, 184, 160)
- Label text color-White로 변경한다.
- HIT button을 이미지로 변경한다.
Attribute inspector에서 Image>list 중에 check mark를 검색하여 checkmark.seal.fill을 클릭한다.
체크 표시 옆의 text를 지워준다.
- 이미지 크기 변경을 위해 Configuration>Font (Font size에 비례하게 button 크기를 조정하는 기능)>Large Title 지정, Scale>Large 지정한다.
이미지 색상 변경을 위해 Attribute inspector 하단 View>Tint>System Yellow Color 지정한다.
- Slider 색상을 변경한다.
Min Track, Max Track>Place Holder Text Color 지정한다.
- RESET button 및 INFO button을 이미지로 변경한다.
1) RESET : Image>list 중에 memories 지정하고 text 삭제한다.
INFO : Image>list 중에 info.circle로 지정하고 text 삭제한다.
- 두 button을 일괄선택하여 설정을 변경한다. Configuration>Font>Font Style-Title3, Scale-Large, Weight-Bold,
Tint>Yellow 지정한다.
- Layout을 설정한다.
Slider는 Vertically, Horizontally in Container 지정한다. 기타 생략

▼ Auto Layout - Concept

애플에서 통용되는 엔진. UI 구현을 위해 사용한다.
Constrains (제약)을 통해 margin, block size 등을 지정한다.
기기가 다양하기 때문에 비율 지정을 통해 여러 화면크기에 유동인 layout을 설정하게 된다.

Auto Layout

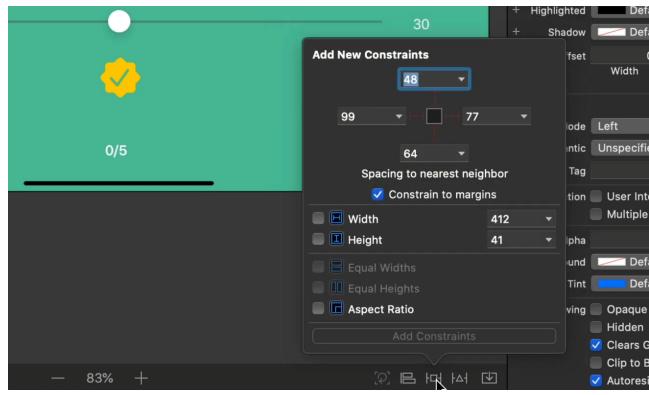
A layout engine
that helps lay out
user interface (UI)
based on the
constraints you specify.



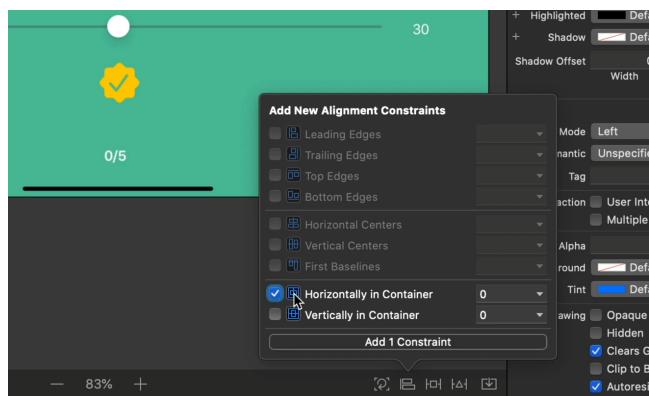
▼ Auto Layout - 적용하기

다른 view 와의 거리 및 정렬

- 우측 하단의 Add Constraints>화면 상단으로부터의 거리 지정한다.



- 우측 하단의 Alignment Constraints>Horizontally in Container 지정한다.
(전체화면에서 수평으로 가운데 정렬. view들 간의 중앙이 아님)



*실수로 button을 이동해도 Constraints에서 지정한대로 화면에서 실행이 되며,
우측 하단의 [Update Frame](#)을 클릭하면 Constraints에 맞게 자동으로 이동한다.
*Size inspector>Constraints>Vertical에서 지정 조건을 확인 가능하다.
*우측 하단의 resolve autolayout issues 항목에서 [clear constraints](#)로 전체 조건을 해제할 수 있다.

▼ Slider 이미지 추가하기 (storyboard의 한계로 코딩이 필요)

`setThumbImage` 함수를 활용한다.

storyboard에서 모든 Attribute를 구현할 수는 없다.

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    override func viewDidLoad() {
        super.viewDidLoad() // viewDidLoad라는 method에 입력한다.
        // Do any additional setup after loading the view.

        slider.setThumbImage(image, for: .normal) // image에서 Image Literal을 선택하면 Asset 이미지를 넣어줄 수 있다. !!
    } // for : slider가 어떤 상태일 때 이 이미지를 표시할 것인가? .normal 상태일 때

    @IBAction func sliderValueChanged(_ sender: UISlider){
        print(sender.value)
    }

    @IBAction func touchUpHitButton(_ sender: UIButton){
        // print(sender.value)
        print(slider.value)
    }

    @IBAction func touchUpResetButton(_ sender: UIButton){
        print("Touch up reset button")
    }
}
```

```
    }
}
```

storyboard에서 Slider Drag 버튼에 이미지가 들어간 것을 확인할 수 있다.

8. Function

- Swift도 마찬가지로 function(**parameter**){ }의 구조이다.
즉, 아래에서 _ sender: UIButton는 매개변수이다.

```
@IBAction func touchUpHitButton(_ sender: UIButton){
    print(sender.value)
}
```

- @IBAction func**는 <Interface Builder>에서 (특정 이벤트에) 연결할 Action함수에 속한다.

▼ class {} 안에 속한 함수는 **method**라고 한다.

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        slider.setThumbImage(image, for: .normal)

    @IBAction func sliderValueChanged(_ sender: UISlider){
        print(sender.value)
    }

    @IBAction func touchUpHitButton(_ sender: UIButton){
        // print(sender.value)
        print(slider.value)
    }

    @IBAction func touchUpResetButton(_ sender: UIButton){
        print("Touch up reset button")
    }
}
```

```
func whatever(x,y){ // whatever라는 함수를 선언하겠다. *실행하는 구문보다 아래에 있어도 알아서 찾아 쓴다.
    print("x+y")
}
```

```
whatever(1,2) // whatever라는 함수를 호출/실행하겠다.
```

□ 왜 2번에서 reset()을 추가하지? 화면 변화 감지하는 것과 reset이 무슨 상관?

▼ Implementing Function - reset 함수 구현하기

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        slider.setThumbImage(image, for: .normal)
        reset() // 2. 추가 - 함수 실행 *reset()을 만나면 reset 정의 부분을 찾아가서 실행한다. (reset 정의가 아래에 있어도 가능)

    @IBAction func sliderValueChanged(_ sender: UISlider){
        print(sender.value)
    }
}
```

```

    }

    @IBAction func touchUpHitButton(_ sender: UIButton){
        // print(sender.value)
        print(slidervalue)
    }

    @IBAction func touchUpResetButton(_ sender: UIButton){
        print("Touch up reset button")
        reset() // 3. 추가 - 함수 실행
    }

    func reset(){ // 1. 함수 정의
        print("reset!")
    }
}

```

9. Variables & Constants 선언

▼ 변수 선언 - Syntax 😊😊😊

```

변수 선언 (data type은 Int, 초기값은 0이다)
var randomVariables: Int = 0

상수 선언 (data type은 Int, 초기값은 0이다)
let randomConstants: Int = 0

```

- 변수의 활용 범위 (range of Variables) 는 [중괄호 {braces pair}](#) 이내이다.
- 즉, 변수가 선언되면 해당 중괄호 안에서 유효하다.
(function 안에서 선언됐다면 해당 function 중괄호 이내에서, class 안에서 선언됐다면 해당 class 중괄호 이내에서)

▼ Label에 변수 선언하기 - HIT 버튼 클릭하면 바뀌는 항목들

- 변경할 UI 요소
 1. 상 : 현재 slider value
 2. 좌 : HIT 버튼 클릭 이후 변경된 최소 value
 3. 우 : HIT 버튼 클릭 이후 변경된 최대 value
 4. 하 : 시도한 횟수

```

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0 // 1. 게임 시작과 동시에 할당할 변수 randomValue 를 선언하고, 초기값을 0으로 지정한다.
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel! // 2. 필요한 UI 요소 중 (4)번에 해당하는 변수를 선언한다. *해당 값을 Label UI에다가 보여줄 3
    @IBOutlet weak var slidervalueLabel: UILabel! // 3. 필요한 UI 요소 중 (1)번
    @IBOutlet weak var minimumValueLabel: UILabel! // 4. 필요한 UI 요소 중 (2)번
    @IBOutlet weak var maximumValueLabel: UILabel! // 5. 필요한 UI 요소 중 (3)번

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
        reset()
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){
        print(sender.value)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){
        print(slidervalue)
    }

    @IBAction func touchUpResetButton(_ sender:UIButton){
        print("Touch Up Reset Button!")
        reset()
    }

    func reset(){

```

```
        print("Reset!")
    }
}
```

▼ Random Number 부여하기

- Reset 버튼 클릭하면 Random Number를 부여한다.

▼ 범위 연산자 (Range Operators)

- Closed 폐쇄 : 0...30 - 처음과 끝 포함 (0 이상 30 이하)
- Half-Closed 반폐쇄 : 0..<30 - 한쪽 끝만 포함 (0 이상 30 미만)
- One-Sided 단방향 : 0... (0 이상) / ...30 (30 이하) / ..<30 (30 미만)

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0 // 1. 정답값을 저장할 변수를 여기에서 선언했다.
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
    @IBOutlet weak var sliderValueLabel: UILabel!
    @IBOutlet weak var minValueLabel: UILabel!
    @IBOutlet weak var maxValueLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
        reset()
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){
        print(sender.value)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){
        print(slider.value)
    }

    @IBAction func touchUpResetButton(_ sender:UIButton){
        print("Touch Up Reset Button!")
        reset()
    }

    func reset(){ // 2. 처음 게임을 시작하거나 reset 했을 때 정답값을 주니까 reset함수에서 정답값을 임의로 지정한다.
        print("Reset!")
        randomValue = Int.random(in: 0...30) // 3. 임의의 Int(정수)를 변수 randomValue로 할당한다. 0에서 30의 범위 내에서.
    }
}
```

▼ RESET 버튼 클릭하면 바뀌는 항목들 (초기화하기)

- 변경할 UI

1. 상 : 초기 slider value
2. 좌 : 최소 value
3. 우 : 최대 value
4. 하 : 시도한 횟수
5. slider : 드래그 버튼 중앙으로 이동

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!
```

```

var randomValue: Int = 0
var tryCount: Int = 0
@IBOutlet weak var tryCountLabel: UILabel!
@IBOutlet weak var sliderValueLabel: UILabel!
@IBOutlet weak var minValueLabel: UILabel!
@IBOutlet weak var maxValueLabel: UILabel!

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.

    slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
    reset() // 참고 - viewDidLoad 함수 안에 이미 reset 함수를 호출하고 있기 때문에 -> 파일을 Run 하면, console 첫 줄에서 Reset!이 출력되는 것
}

@IBAction func sliderValueChanged(_ sender:UISlider){
    print(sender.value)
}

@IBAction func touchUpHitButton(_ sender:UIButton){
    print(slider.value)
}

@IBAction func touchUpResetButton(_ sender:UIButton){
    print("Touch Up Reset Button!")
    reset() // 참고 - 여기서 호출될 것이다. (RESET 버튼을 누르는 Action이 들어왔을 때)
}

func reset(){
    print("Reset!")
    randomValue = Int.random(in: 0...30)
    tryCount = 0 // RESET 버튼 클릭했을 때 바꿔는 것들을 자정해준다.
    tryCountLabel.text = "0 / 5" // Label에다가 text 형태로 이렇게 입력하겠다.

    slider.minimumValue = 0 // slider 기능에 설정된 attribute값을 최소 0, 최대 30, 현재 15로 재설정하겠다는 의미이다.
    slider.maximumValue = 30
    slider.value = 15

    minValueLabel.text = "0" // Label에다가 text 형태로 이렇게 입력하겠다.
    maxValueLabel.text = "30"
    sliderValueLabel.text = "15"
}
}

```

- Variables Data Type 변경하기

▼ 변경된 slideValue를 사용자에게 보여주기 (label에 slider값을 표시하기) 😊😊😊 Quiz

- slider의 data type은 float (실수)이고,
label의 data type은 string(문자열)이다.
- [문자열 보관법](#)

▼ "“(이곳에 문자열을 넣으면 된다)”

```

tryCountLabel.text = String(tryCount) + " / 5" // tryCount를 String으로 변환하여 label에 입력해준다.

tryCountLabel.text = "\((tryCount) / 5" // 보다 가독성 있다. *///slash가 아니라 delete 아래 버튼\\\

```

- 순서

1. slider를 드래그 했을 때 변경사항 반영하기
2. HIT 버튼 클릭했을 때 변경사항 반영하기

▼ Quiz - 먼저 풀어봐!

```

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
    @IBOutlet weak var sliderValueLabel: UILabel!
}

```

```

@IBOutlet weak var minValueLabel: UILabel!
@IBOutlet weak var maxValueLabel: UILabel!

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.

    slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
    reset()
}

@IBAction func sliderValueChanged(_ sender:UISlider){ (1) 사용자가 만든 slider값을 Int화하고 -> 상단 label에 보여줌
    print(sender.value)
}

@IBAction func touchUpHitButton(_ sender:UIButton){ (2-1) 사용자가 HIT 버튼 눌렀을 때 slider값을 Int화하고 -> slider# 
    print(slider.value)

// (2-2) 하단 label에 tryCount를 1씩 더해서 보여줌
}

@IBAction func touchUpResetButton(_ sender:UIButton){
    print("Touch Up Reset Button!")
    reset()
}

func reset(){
    print("Reset!")
    randomValue = Int.random(in: 0...30)
    print(randomValue)
    tryCount = 0
    tryCountLabel.text = "0 / 5"
    slider.minimumValue = 0
    slider.maximumValue = 30
    slider.value = 15
    minValueLabel.text = "0"
    maxValueLabel.text = "30"
    sliderValueLabel.text = "15"
}
}

```

▼ 답

```

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
    @IBOutlet weak var sliderValueLabel: UILabel!
    @IBOutlet weak var minValueLabel: UILabel!
    @IBOutlet weak var maxValueLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
        reset()
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){ (1) 사용자가 만든 slider값을 Int화하고 -> 상단 label에 보여줌 / 왜
        print(sender.value)
        let integerValue: Int = Int(sender.value) // 실수가 아닌 정수로 보여주려고 한번 가공한다. (소수점 버림) integerValue라는 :
        sliderValueLabel.text = String(integerValue) // sliderValueLabel에 넣을 문자열을 입력하기 위해 <- integerValue를 St
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){ (2-1) 사용자가 HIT 버튼 눌렀을 때 slider값을 Int화하고 -> slider#
        print(slider.value)
        let hitValue: Int = Int(slider.value) // HIT 버튼을 눌렀을 때 slider값을 정수로 변환하여 -> data type Int 형식의 상수?
        slider.value = Float(hitValue) // hitValue를 *실수로 변환하여 -> slider값으로 입력해준다. slider가 해당 정수 위치로 미세

        tryCount = tryCount + 1 // (2-2) 하단 label에 tryCount를 1씩 더해서 보여줌
        // tryCountLabel.text = String(tryCount) + " / 5" // tryCount를 String으로 변환하여 label에 입력해준다. *문자열 보관
        tryCountLabel.text = "/(tryCount) / 5"
    }

    @IBAction func touchUpResetButton(_ sender:UIButton){
}

```

```

        print("Touch Up Reset Button!")
        reset()
    }

    func reset(){
        print("Reset!")
        randomValue = Int.random(in: 0...30)
        print(randomValue)
        tryCount = 0
        tryCountLabel.text = "0 / 5"
        slider.minimumValue = 0
        slider.maximumValue = 30
        slider.value = 15
        minValueLabel.text = "0"
        maxValueLabel.text = "30"
        sliderValueLabel.text = "15"
    }
}

```

10. Comparing values / Terminating functions

- 비교연산자

▼ Concept

```

if A == B {}
A와 B가 같을 경우 {}를 실행한다.

*구분하기 위해 사용한다
= 대입 연산자
== 비교 연산자

*기타 비교연산자
>= 크거나 같다
<= 작거나 같다
!= 같지 않다

```

▼ HIT 버튼 눌렀을 때의 slider값과 정답값 비교하기 / 함수 종료하기

- 일치하면 → YOU HIT! 출력하고, 게임을 초기화하고 함수를 종료
- 5번 이상 불일치하면 → YOU LOSE! 출력하고, 게임을 초기화하고 함수를 종료

```

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
    @IBOutlet weak var sliderValueLabel: UILabel!
    @IBOutlet weak var minValueLabel: UILabel!
    @IBOutlet weak var maxValueLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
        reset()
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){
        print(sender.value)
        let integerValue: Int = Int(sender.value)
        sliderValueLabel.text = String(integerValue)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){
        print(slider.value)
        let hitValue: Int = Int(slider.value)
        slider.value = Float(hitValue)
        tryCount = tryCount + 1
        // tryCountLabel.text = String(tryCount) + " / 5"
        tryCountLabel.text = "\((tryCount)) / 5"

        if randomValue == hitValue { // 1. 일치할 경우
            print("YOU HIT!")
        }
    }
}

```

```

        reset() // 4. 그 전에 게임을 초기화 한다.
        return // 3. 여기서 함수를 종료한다. (return 아래로는 실행되지 않음)
    }
    if tryCount >= 5 { // 2. 5번 이상 불일치할 경우 -> 그런데 5번째에 정답을 맞추는 경우 논리적 오류가 생긴다. 따라서 일치할 경우 훈
        print("YOU LOSE...")
        reset() // 5. 여기도 마찬가지로 처리한다.
        return
    }

    @IBAction func touchUpResetButton(_ sender:UIButton){
        print("Touch Up Reset Button!")
        reset()
    }

    func reset(){
        print("Reset!")
        randomValue = Int.random(in: 0...30)
        print(randomValue)
        tryCount = 0
        tryCountLabel.text = "0 / 5"
        slider.minimumValue = 0
        slider.maximumValue = 30
        slider.value = 15
        minValueLabel.text = "0"
        maxValueLabel.text = "30"
        sliderValueLabel.text = "15"
    }
}

```

11. Conditional Execution

- 조건문 (else if)

▼ Concept

```

if A {
    print("A")
} else if B {
    print("B")
}
만약 A가 참이면 {} 하고, 거짓인 경우 만약 B가 참이면 {} 한다.

*A가 거짓인 경우만 B로 넘어갈 수 있다.
*장점 - return이 따로 필요 없다. 왜 그런지 알겠어?

```

▼ slider값을 minimum / maximum value - 좌우 Label에 반영하기

- 만약 slider값이 정답값보다 작다면 → slider값을 minimum value에 대입한다.
- 만약 slider값이 정답값보다 크다면 → slider값을 maximum value에 대입한다.

▼ 구조 개선 (else if)

```

if 일치 {출력 return}
if 불일치 5번 {출력 return}
if randomValue>hitValue {minimum 조정}
if randomValue<hitValue {maximum 조정}

```

```

// else if 를 통해 개선해보면

if 일치 {출력 return}
else if 불일치 5번 {출력 return}
else if randomValue>hitValue {minimum 조정}
else randomValue<hitValue {maximum 조정}

```

```

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
}

```

```

@IBOutlet weak var sliderValueLabel: UILabel!
@IBOutlet weak var minValueLabel: UILabel!
@IBOutlet weak var maxValueLabel: UILabel!

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.

    slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
    reset()
}

@IBAction func sliderValueChanged(_ sender: UISlider) {
    print(sender.value)
    let integerValue: Int = Int(sender.value)
    sliderValueLabel.text = String(integerValue)
}

@IBAction func touchUpHitButton(_ sender: UIButton) {
    print(slider.value)
    let hitValue: Int = Int(slider.value)
    slider.value = Float(hitValue)
    tryCount = tryCount + 1
    // tryCountLabel.text = String(tryCount) + " / 5"
    tryCountLabel.text = "\(tryCount) / 5"

    if randomValue == hitValue { // 1. 정답값을 맞추면 끝나니까 가장 상위 조건
        print("YOU HIT!")
        reset()
        // return
    } else if tryCount >= 5 {
        print("YOU LOSE...")
        reset()
        // return
    } else if randomValue > hitValue { // 2. 값 비교 하고나서 minimum/maximum value 조정
        slider.minimumValue = Float(hitValue) // *slider값은 문법상 Float이므로 data type을 변환해줌
        minValueLabel.text = String(slider.minimumValue)
    } else {
        slider.maximumValue = Float(hitValue)
        maxValueLabel.text = String(slider.maximumValue)
    }
}

@IBAction func touchUpResetButton(_ sender: UIButton) {
    print("Touch Up Reset Button!")
    reset()
}

func reset() {
    print("Reset!")
    randomValue = Int.random(in: 0...30)
    print(randomValue)
    tryCount = 0
    tryCountLabel.text = "0 / 5"
    slider.minimumValue = 0
    slider.maximumValue = 30
    slider.value = 15
    minValueLabel.text = "0"
    maxValueLabel.text = "30"
    sliderValueLabel.text = "15"
}
}

```

12. Showing Alerts

▼ 알림창 - Alert Syntax

- alert 관련 method 확인하기

present - method (현재 화면 위에 창을 올려준다)

자동완성기능을 활용한다.

```

func showAlert(myMessage: String){
    let alert = UIAlertController(title: <#T##String?#>, message: <#T##String?#>, preferredStyle: <#T##UIAlertController.Style#>
        let okAction = UIAlertAction(title: <#T##String?#>, style: <#T##UIAlertAction.Style#>, handler: <#T##((UIAlertAction) -> Void)>
    }

=> 입력

func showAlert(myMessage: String){
    let alert = UIAlertController(title: nil, // *nil : 없다라는 의미임
        message: myMessage,

```

```

        preferredStyle: .alert)
let okAction = UIAlertAction(title: "OK",
                           style: .default) { (action) in
    self.reset()
}
alert.addAction(okAction)
present(alert,           // method를 활용한다....? 나중에 검색해보자
       animated: true,
       completion: nil) // alert가 완료 (completion)된 이후에 추가적으로 할 action이 없으므로 nil
}

```

▼ showAlert 함수를 생성하고 필요한 곳에 호출하기

```

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var slider: UISlider!

    var randomValue: Int = 0
    var tryCount: Int = 0
    @IBOutlet weak var tryCountLabel: UILabel!
    @IBOutlet weak var sliderValueLabel: UILabel!
    @IBOutlet weak var minimumValueLabel: UILabel!
    @IBOutlet weak var maximumValueLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        slider.setThumbImage(#imageLiteral(resourceName: "slider_thumb"), for: .normal)
        reset()
    }

    @IBAction func sliderValueChanged(_ sender:UISlider){
        print(sender.value)
        let integerValue: Int = Int(sender.value)
        sliderValueLabel.text = String(integerValue)
    }

    func showAlert(myMessage: String){ // 1. showAlert 함수를 생성한다. (여러움)
        let alert = UIAlertController(title: nil, // nil은 없다라는 의미임
                                     message: myMessage,
                                     preferredStyle: .alert)
        let okAction = UIAlertAction(title: "OK",
                                   style: .default) { (action) in
            self.reset()
        }
        alert.addAction(okAction)
        present(alert,
               animated: true,
               completion: nil)
    }

    @IBAction func touchUpHitButton(_ sender:UIButton){
        print(slider.value)
        let hitValue: Int = Int(slider.value)
        slider.value = Float(hitValue)
        tryCount = tryCount + 1
        // tryCountLabel.text = String(tryCount) + " / 5"
        tryCountLabel.text = "\(tryCount) / 5"

        if randomValue == hitValue {
            print("YOU HIT!")
            showAlert(myMessage: "YOU HIT!") // 2. showAlert 함수를 실행시켜서 alert를 보여준다.
            reset()
            // return
        } else if tryCount >= 5 {
            print("YOU LOSE...")
            showAlert(myMessage: "YOU LOSE...") // 여기도 마찬가지
            reset()
            // return
        } else if randomValue > hitValue {
            slider.minimumValue = Float(hitValue)
            minimumValueLabel.text = String(slider.minimumValue)
        } else {
            slider.maximumValue = Float(hitValue)
            maximumValueLabel.text = String(slider.maximumValue)
        }
    }

    @IBAction func touchUpResetButton(_ sender:UIButton){
        print("Touch Up Reset Button!")
        reset()
    }
}

```

```

    }

    func reset(){
        print("Reset!")
        randomValue = Int.random(in: 0...30)
        print(randomValue)
        tryCount = 0
        tryCountLabel.text = "0 / 5"
        slider.minimumValue = 0
        slider.maximumValue = 30
        slider.value = 15
        minValueLabel.text = "0"
        maxValueLabel.text = "30"
        sliderValueLabel.text = "15"
    }
}

```

13. Creating Credit Scene

- new scene (storyboard 상에서 1화면 단위) 생성
 - file>new file>cocoa touch>CreditViewController 입력, UIViewController 선택
 - storyboard에서 library-view controller를 추가 (view는 사용자가 보는 화면, view controller는 개발자가 작업하는 화면)
 - view controller에서 attribute inspector>Class에서 CreditViewController 선택
- credit view에 요소 올리기
 - library 또는 Human interface guidelines for iOS를 참고한다.
 - library - image view, text view를 추가한다.
 - image view
 - attribute inspector에서 symbol image>asset의 해당 이미지 선택
 - text view 특징
 - label과 달리 사용자 interaction이 가능하다. (복붙, 입력 등)
 - 글의 길이가 길어지면 자동 스크롤이 된다.
 - 줄바꿈 : option 키 사용
 - attribute inspector>data detector에서 이메일, 웹주소 등 하이퍼링크 처리가 가능하다. (폰넘버, 링크, 캘린더 이벤트, 항공편 번호 등)
 - *editable : 사용자가 해당 정보를 수정하지 않게 하려면 해제
 - *selectable : 사용자가 텍스트를 드래그하고 복붙 가능하게 하려면 체크
 - text field 특징
 - 사용자 interaction이 가능하고, 사용자로부터 한 줄 짜리 텍스트를 입력받을 때 사용한다.
 - 창닫기 이미지 삽입
 - attribute inspector>image>xmark 입력

14. Storyboard Segue (scene에서 scene으로 넘기기)

- segue #1 (화면 전환 : game 화면 info 버튼 ⇒ credit view)
 - info 버튼 +control & 드래그해서 credit view에 드롭>action segue에서 present modally 선택
 - scene 사이의 연결 버튼을 누르고 attribute inspector>transition에서 cross dissolve 등 애니메이션 설정
- segue #2 (화면 전환 : credit view X버튼 ⇒ game 화면)
 - 절대 X버튼에서 새로운 segue를 생성하여 game view로 연결하면 안된다.
 - 이유 : 이전 화면으로 돌아가는 것이 아니라 새로운 화면을 생성하게 되므로 사용자가 play 하던 game이 날아감
 - CreditViewController 코딩 화면에서 @IBAction을 추가한다.
 - Alert 화면을 만들 때 사용한 present - method (화면에 창을 올려주는 기능)를 동일하게 활용한다.
 즉, present와 반대 기능인 dismiss - method (화면에서 창을 내리는 기능)을 사용한다.

```

import UIKit

class CreditViewController: UIViewController {

```

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Do any additional setup after loading the view.
}

@IBAction func touchUpDismissButton(_ sender: UIButton){
    dismiss(animated: true,
           completion: nil)
}
```