함수 꾸미기

- 단순 명령어의 순서가 아닌 복잡한 명령을 내리기 위해선 프로그램 명령어 컨트롤 방법(Controll Flow)이 필요하다.

- 조건문(while, for-in) & 선택문(if, guard, switch)을 통해 함수를 컨트롤 할수 있다.

연산자

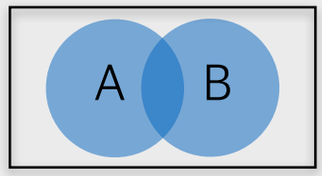
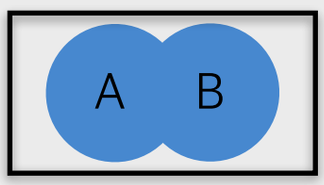
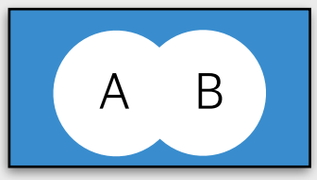
산술 연산자

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

비교 연산자

기호	예제	설명
==	A == B	A와 B가 같다.
>=	A >= B	A가 B보다 크거나 같다
<=	A <= B	A가 B보다 작거나 같다.
>	A > B	A가 B보다 크다
<	A < B	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 || B조건의 반대

추가연산자

복합연산자	예제	설명
+=	a += 1	a에 값을 더하기
-=	b -= 2	b에 값을 빼기

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

Identity 연산자	예제	설명
===	person2 === person1	person1과 person2는 같은 인스턴스를 참조하고 있다.
!==	person2 !== person1	person1과 person2는 다른 인스턴스를 참조하고 있다.

삼항연산자

삼항연산자	설명
question ? answer1 : answer2.	question이 참이면 answer1값을 거짓이면 answer2값은 지정한다.

```
let age =  20
var result:String = age > 19 ? "성년" : "미성년"
```

조건문

강사 주영민

조건문 개론

- 함수 내부에서 실행되는 선택문
- 특정 조건에 따라 선택적으로 코드를 실행시킨다.
- 대표적인 조건문으로 if-else문과 switch-case문이 있다.

Switch문

- 패턴 비교문
- 가장 첫번째 매칭되는 패턴의 구문이 실행된다.

switch 문법

```
switch some value to consider {
case value 1:
    respond to value 1
case value 2,
     value 3:
    respond to value 2 or 3
default:
    otherwise, do something else
}
```

switch 문법

- 각의 상태는 case 키워드를 통해 나타낼수 있다.
- 각 case 상태 끝에는 콜론(:)을 작성해서 매칭 패턴을 종료한다.
- 여러개의 case가 필요하면 콤마(,)를 통해 상태를 추가 할수 있다.
- 각 case는 분기로 실행되며 매칭된 해당 case문이 종료되면 switch 문을 종료시킨다.
- 각 case의 상태는 switch 문의 value값에 매칭된 지점을 결정하며, 모든 value에 대해 매칭 되어야 한다. 만약 매칭되는 값이 없다면 default 키워드를 통해 기본 매칭값을 설정하며, default키워드는 마지막에 배치된다.

switch 예제

```
func sampleSwitch(someCharacter:Character)
{
    switch someCharacter {
    case "a":
        print("The first letter of the alphabet")
    case "z":
        print("The last letter of the alphabet")
    default:
        print("Some other character")
    }
}
```

문제

1. 시험 점수를 받아서 해당 점수의 등급**(Grade)**을 반환해주는 함수
   ***Grade = A+, A, B+, B, C+…**

2. **Grade**받아서 **Point**로 변환해주는 함수
   Point = 4.5, 4.0, 3.5, 3.0…

Interval Matching

- Switch 문의 상태는 단순 value매칭을 넘어 좀더 다양한 패턴을 통해 매칭이 가능하다.

- interval matching은 범위 연산자를 통해 해당 범위에 해당하는 value를 매칭 시킬수 있다.

Interval Matching 예제

```
func interSwitch(count:Int)
{
    let countedThings = "moons orbiting Saturn"
    let naturalCount: String
    switch count {
    case 0:
        naturalCount = "no"
    case 1..<5:
        naturalCount = "a few"
    case 5..<12:
        naturalCount = "several"
    case 12..<100:
        naturalCount = "dozens of"
    case 100..<1000:
        naturalCount = "hundreds of"
    default:
        naturalCount = "many"
    }
    print("There are \(naturalCount) \(countedThings).")
}
```

튜플매칭

- 튜플을 사용해서 여러개의 value를 동시에 확인 할수 있습니다.
- 사용 가능한 모든 값에 대한 매칭은 와일드 카드 (_)를 통해서 매칭 가능합니다.

튜플 예제

```
func getPoint(somePoint:(Int,Int))
{
    switch somePoint {
    case (0, 0):
        print("\(somePoint) is at the origin")
    case (_, 0):
        print("\(somePoint) is on the x-axis")
    case (0, _):
        print("\(somePoint) is on the y-axis")
    case (-2...2, -2...2):
        print("\(somePoint) is inside the box")
    default:
        print("\(somePoint) is outside of the box")
    }

}
```

값 바인딩

- case 내부에서 사용되는 임시 값으로 매칭 시킬수 있다.

값 바인딩 예제

```swift
func getPoint(somePoint:(Int,Int))
{
    switch somePoint {
    case (0, 0):
        print("\(somePoint) is at the origin")
    case (let x, 0):
        print("on the x-axis with an x value of \(x)")
    case (0, let y):
        print("on the y-axis with an y value of \(y)")
    case (-2...2, -2...2):
        print("\(somePoint) is inside the box")
    default:
        print("\(somePoint) is outside of the box")
    }

}
```

where문

- where 문의 추가로 추가 조건을 넣을수 있다.

where문 예제

```
func wherePoint(point:(int,Int))
{
    switch point {
    case let (x, y) where x == y:
        print("(\(x), \(y)) is on the line x == y")
    case let (x, y) where x == -y:
        print("(\(x), \(y)) is on the line x == -y")
    case let (x, y):
        print("(\(x), \(y)) is just some arbitrary point")
    }
}
```