정의

- 단일 값이나 복합 값의 구조로 나타낼 수 있는 유형.
- 패턴의 일부 값을 일치시키거나, 일부 또는 전체를 다른 변수의 이름으로 바인딩 할 수 있음

```
/*
        pattern (x, y)
        tuple (1, 2)
        일치함

        let (x, y) = (1, 2)
        print(x)
        print(y)
*/
```

종류

- 어떠한 값이든 성공적으로 일치시키는 패턴
  - WildCard 패턴
  - Indentifier 패턴
  - Value-Binding 패턴
  - Tuple 패턴
- 런타임에 매칭에 실패할 수 있는 패턴
  - EnumCase
  - Optional
  - Type-Casting
  - Expression

와일드 카드 패턴(Wildcard)

- 와일드 카드 패턴은 모든 값과 일치되며, 개발자가 매치된 값에 신경을 쓰지 않을때 사용함
- 아래의 예시는 3번의 반복문을 진행 중 현재 값을 신경쓰지 않을때 사용됨

```
for _ in 1...3{
        print("단순 반복")
}
```

식별자 패턴 (Identifier)

- 어떠한 값이든 매칭이 가능한 패턴이다
- Value-binding Pattern 의 subpattern 이다

```
let someValue = 10

/*
someValue : Int
someValue value : 10

match succeeds
someValue = 10 binding

*/
```

Value-Binding 패턴

- 상수와 변수에 값을 매칭 시키는 패턴이다.
- 상수에는 let, 변수에는 var 키워드를 사용한다.
- 식별자 패턴은 값 바인딩 패턴의 서브패턴으로 새로운 변수나 상수를 매칭 시키는 형태다.

```
let point = (3,2)
switch point{
case let (x , y) // var 도 가능하며, 해당값은 (let x, let y) 와 같음
        print(x, y)
        fallthrough
case let (x, _) where x < 5 :
        print(x)
}
```

Tuple 패턴

- 튜플패턴은 0개 이상의 패턴 리스트 집합이다.
- 튜블 패턴은 for 문에서 매치할 패턴은 where 절을 사용한다.

```
let points = [(0,0), (1,0), (1,1), (2, 0)]
```

```
for(x, y) in points {
        print(x, y, separator: " ")
}

for (x, y) in points where y == 1 {
        print(x, y, separator: " ")
}
```

Optional 패턴

- 옵셔널 패턴은 Optional<\Wrapped> 의 some(Wrapped) 와 매칭된다.

```
let someOptional : Int? = 42

// enumeration case pattern
if case .some(let x) = someOptional {
         print(x)
}

// Match using optional pattern
if case let v? = someOptional {
         print(v)
}
```

- 반복문에서의 옵셔널 패턴

```
let arrayOfOptionalInts: [Int?] = [nil, 2, 3, nil, 5]
 // Match only non-nil values.

 for case let number? in arrayOfOptionalInts {
          print("Found a \(number)")
 }
```