Swift Study #5

#이상한모임 @신촌

Subscripts

```
someArray[0]

someDictionary[“apple”]
```

Subscript Syntax

```
subscript (index: Int) -> Int {
    get {
      ...
    }
    set(newValue) {
      ...
    }
}
```

Read-Only

```
subscript (index: Int) -> Int {

}
```

```
struct TimesTable {
    let multiplier: Int
    subscript(index:Int) -> Int {
        return multiplier * index
    }
}

let three = TimesTable(multiplier: 3)
println(three[3])
```

9

Subscript Usage

```
var numberOfLegs
      = ["spider": 8, "ant": 6, "cat": 4]

numberOfLegs["bird"] = 2
```

```
numberOfLegs["ant"] = nil
```

Subscript Options

```
struct Matrix {
    let rows: Int, columns: Int
    var grid: [Double]

    init(rows: Int, columns: Int) {
        self.rows = rows
        self.columns = columns
        grid = Array(count: rows * columns,
repeatedValue: 0.0)
    }
    func indexIsValidForRow(row: Int, column: Int) ->
Bool {}

    subscript(row: Int, column: Int) -> Double {}
}
```

```
subscript(row: Int, column: Int) -> Double {

    get {
        assert(indexIsValidForRow(row, column: column),
"Index out of range")
        return grid[(row * columns) + column]
    }

    set {
        assert(indexIsValidForRow(row, column: column),
"Index out of range")
        grid[(row * columns) + column] = newValue
    }
}
```

```
func indexIsValidForRow(row: Int, column: Int) ->
Bool {
    return row >= 0 && row < rows && column >= 0 &&
column < columns
}
```

```
var matrix = Matrix(rows: 2, columns: 2)
matrix[0, 1] = 1.5
matrix[1, 0] = 3.2

println(matrix[0,1])  //1.5
println(matrix[1,0])  //3.2
```