```
import UIKit
//: # Basic Operators
//: ## Terminology
//: ### Unary : operates on a single target
//:     -a
//:     !b
//:     c!
//: ### Binary : operates on two targets
//:     2 + 3
//: ### Ternary : operates on three targets
//:     a ? b : c   (ternary conditional operator)
//: ## Assignment Operator
//: ### The assignment operator (a = b) initializes or updates the value of a
 with the value of b:
let b = 10
var a = 5
a = b
// a is now equal to 10
//: ### Assignments in tuples
let (x, y) = (1, 2)
// x is equal to 1, and y is equal to 2
//: ## Arithmetic Operators
/*:
 ### Swift supports the four standard arithmetic operators for all number
  types:
 * 1 + 2       // equals 3
 * 5 - 3       // equals 2
 * 2 * 3       // equals 6
 * 10.0 / 2.5  // equals 4.0
*/
//: ### The addition operator is also supported for String concatenation:
"hello, " + "world"  // equals "hello, world"
//: ## Remainder Operator
//: * a % b
9 % 4    // returns 1
-9 % 4   // equals -1
//: The sign of b is ignored for negative values of b. This means that a % b
 and a % -b always give the same answer.
//: ### Unary Minus Operator
let three = 3
let minusThree = -three       // minusThree equals -3
let plusThree = -minusThree   // plusThree equals 3, or "minus minus three"
//: It is prepended directly before the value it operates on, without any
 white space.
//: ## Unary Plus Operator
let minusSix = -6
let alsoMinusSix = +minusSix  // alsoMinusSix equals -6
//: you can use it to provide symmetry in your code for positive numbers when
 also using the unary minus operator for negative numbers.
//: ## Compound Assignment Operators
//: ### combine assignment (=) with another operation
```

```swift
var d = 1
d += 2
// d is now equal to 3
// The expression d += 2 is shorthand for d = d + 2
//: ## Comparison Operators
/*:
 ### Swift supports all standard C comparison operators:
 * Equal to (a == b)
 * Not equal to (a != b)
 * Greater than (a > b)
 * Less than (a < b)
 * Greater than or equal to (a >= b)
 * Less than or equal to (a <= b)
*/
//: Swift also provides two identity operators (=== and !==), which you use to
 test whether two object references both refer to the same object instance
//: ### Each of the comparison operators returns a Bool value
1 == 1   // true because 1 is equal to 1
2 != 1   // true because 2 is not equal to 1
2 > 1    // true because 2 is greater than 1
1 < 2    // true because 1 is less than 2
1 >= 1   // true because 1 is greater than or equal to 1
2 <= 1   // false because 2 is not less than or equal to 1
//: ### Comparison in tuples:
(1, "zebra") < (2, "apple")   // true because 1 is less than 2; "zebra" and
 "apple" are not compared
(3, "apple") < (3, "bird")    // true because 3 is equal to 3, and "apple" is
 less than "bird"
(4, "dog") == (4, "dog")      // true because 4 is equal to 4, and "dog" is
 equal to "dog"
//: ### the < (>) operator can't be applied to Bool values.
//:     ("blue", false) < ("blue", true)  // Error because < can't compare
 Boolean values

//: ## Ternary Conditional Operator
//: * question ? answer1 : answer2
//: * If question is true, it evaluates answer1 and returns its value;
 otherwise, it evaluates answer2 and returns its value
/*:
 ### The ternary conditional operator is shorthand for the code below:
    if question {
        answer1
    } else {
        answer2
    }
*/
//: * Example:
let contentHeight = 40
let hasHeader = true
let rowHeight = contentHeight + (hasHeader ? 50 : 20)
// rowHeight is equal to 90
```

```
//: ## Nil-Coalescing Operator
//: * The nil-coalescing operator (a ?? b) unwraps an optional a if it
 contains a value, or returns a default value b if a is nil.
//: ### The nil-coalescing operator is shorthand for the code below:
//: * a != nil ? a! : b
//: * Example:
let defaultColorName = "red"
var userDefinedColorName: String?   // defaults to nil

var colorNameToUse = userDefinedColorName ?? defaultColorName
// userDefinedColorName is nil, so colorNameToUse is set to the default of
 "red"

userDefinedColorName = "green"
colorNameToUse = userDefinedColorName ?? defaultColorName
// userDefinedColorName is not nil, so colorNameToUse is set to "green"
//: ## Range Operators
//: ### Closed Range Operator
//: * (a...b) // includes the values a and b
//: * Example:
for index in 1...5 {
    print("\(index) times 5 is \(index * 5)")
}
// 1 times 5 is 5
// 2 times 5 is 10
// 3 times 5 is 15
// 4 times 5 is 20
// 5 times 5 is 25
//: ### Half-Open Range Operator
//: * (a..<b) // does not include b
//: * Example:
let names = ["Anna", "Alex", "Brian", "Jack"]
let count = names.count
for i in 0..<count {
    print("Person \(i + 1) is called \(names[i])")
}
// Person 1 is called Anna
// Person 2 is called Alex
// Person 3 is called Brian
// Person 4 is called Jack
//: ### One-Sided Ranges
//: * Example:
for name in names[2...] {
print(name)
}
// Brian
// Jack

for name in names[...2] {
    print(name)
}
```

```
// Anna
// Alex
// Brian
//: ### One-sided ranges with half-open range operator:
for name in names[..<2] {
    print(name)
}
// Anna
// Alex
//: ## Logical Operators
/*:
 ### Swift supports the three standard logical operators found in C-based
  languages:
 * Logical NOT (!a)
 * Logical AND (a && b)
 * Logical OR (a || b)
*/
//: ### Logical NOT Operator
//: * The logical NOT operator (!a) inverts a Boolean value
//: * The logical NOT operator is a prefix operator
//: * Example:
let allowedEntry = false
if !allowedEntry {
    print("ACCESS DENIED")
}
// Prints "ACCESS DENIED"
//: ### Logical AND Operator
//: * both values must be true for the overall expression to be evaluated as
 true
//: * Example:
let enteredDoorCode = true
let passedRetinaScan = false
if enteredDoorCode && passedRetinaScan {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
}
// Prints "ACCESS DENIED"
//: ### Logical OR Operator
//: * only one of the two values has to be true for the overall expression to
 be true
//: * Example:
let hasDoorKey = false
let knowsOverridePassword = true
if hasDoorKey || knowsOverridePassword {
    print("Welcome!")
} else {
    print("ACCESS DENIED")
}
// Prints "Welcome!"
```