# Booleans

Swift has a basic *Boolean* type, called `Bool`. Boolean values are referred to as *logical*, because they can only ever be true or false. Swift provides two Boolean constant values, `true` and `false`:

```swift
let orangesAreOrange = true
let turnipsAreDelicious = false
```

The types of `orangesAreOrange` and `turnipsAreDelicious` have been inferred as `Bool` from the fact that they were initialized with Boolean literal values. As with `Int` and `Double` above, you don't need to declare constants or variables as `Bool` if you set them to `true` or `false` as soon as you create them. Type inference helps make Swift code more concise 간결한 and readable when it initializes constants or variables with other values whose type is already known.

Boolean values are particularly useful when you work with conditional statements such as the `if` statement:

```swift
if turnipsAreDelicious {
    print("Mmm, tasty turnips!")
} else {
    print("Eww, turnips are horrible.")
}
// Prints "Eww, turnips are horrible."
```

Conditional statements such as the `if` statement are covered in more detail in Control Flow.

바꾸다, 역할을 대신하다

Swift's type safety prevents non-Boolean values from being substituted for `Bool`. The following example reports a compile-time error:

```
let i = 1
if i {
    // this example will not compile, and will report an error
}
```

However, the alternative example below is valid:

```
let i = 1
if i == 1 {
    // this example will compile successfully
}
```

The result of the `i == 1` comparison is of type `Bool`, and so this second example passes the type-check. Comparisons like `i == 1` are discussed in Basic Operators.

의도

As with other examples of type safety in Swift, this approach avoids accidental errors and ensures that the intention of a particular section of code is always clear.