Swift optional variables

```
/* This is the start of the first multiline comment.
 /* This is the second, nested multiline comment. */
This is the end of the first multiline comment. */
```

```swift
let cat = "🐱 "; print(cat)
// Prints "🐱 "
```

```swift
let decimalInteger = 17
let binaryInteger = 0b10001       // 17 in binary notation
let octalInteger = 0o21           // 17 in octal notation
let hexadecimalInteger = 0x11     // 17 in hexadecimal notation
```

- `1.25e2` means 1.25×10^{2}, or `125.0`.
- `1.25e-2` means 1.25×10^{-2}, or `0.0125`.

```swift
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and equals (404, "Not Found")
```

Optionals

You use *optionals* in situations where a value may be absent. An optional represents two possibilities: Either there *is* a value, and you can unwrap the optional to access that value, or there *isn't* a value at all.

The example below uses the initializer to try to convert a `String` into an `Int`:

```
let possibleNumber = "123"
let convertedNumber = Int(possibleNumber)
// convertedNumber is inferred to be of type "Int?", or "optional Int"
```

Because the initializer might fail, it returns an *optional* `Int`, rather than an `Int`. An optional `Int` is written as `Int?`, not `Int`. The question mark indicates that the value it contains is optional, meaning that it might contain *some* `Int` value, or it might contain *no value at all*. (It can't contain anything else, such as a `Bool` value or a `String` value. It's either an `Int`, or it's nothing at all.)

nil

You set an optional variable to a valueless state by assigning it the special value `nil`:

```
var serverResponseCode: Int? = 404
// serverResponseCode contains an actual Int value of 404
serverResponseCode = nil
// serverResponseCode now contains no value
```

NOTE

You can't use `nil` with non-optional constants and variables. If a constant or variable in your code needs to work with the absence of a value under certain conditions, always declare it as an optional value of the appropriate type.

If you define an optional variable without providing a default value, the variable is automatically set to `nil` for you:

```
var surveyAnswer: String?
// surveyAnswer is automatically set to nil
```

If an optional has a value, it's considered to be "not equal to" `nil`:

```
if convertedNumber != nil {
    print("convertedNumber contains some integer value.")
}
// Prints "convertedNumber contains some integer value."
```

```swift
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
```

```swift
for name in names[2...] {
    print(name)
}
// Brian
// Jack

for name in names[...2] {
    print(name)
}
// Anna
// Alex
// Brian
```

```
for name in names[..<2] {
    print(name)
}
// Anna
// Alex
```

```swift
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
```

```swift
let range = ...5
range.contains(7)   // false
range.contains(4)   // true
range.contains(-1)  // true
```