Commenting and basic types in Swift

Timothy Chong

Commenting in Swift

It is possible that your fellow software engineering co-workers or yourself will revisit some code years after you have written it; therefore, as a good programmer, you should comment your code frequently.

When some lines of codes are commented out, they are completely ignored during compile time. So you can treat it as if it were not there.

There are two ways you can comment your code in Swift.

1. You can comment out a line by beginning it with "//"
2. You can also any length of code by surrounding it with "/*" and "*/"

```
//This is a comment


/*
var comment = "This is a test for comment"

println(comment)

*/
```

You can start your line comment (//) anywhere in a line. You can also start and end your block comment anywhere you desire.

```
var comment = "This is a test for comment" // Comment here

var a = /* Random comment here

*/ 0
```

Constant and variables

Constants or variables are names that are assigned to values of specific types. Constants are declared using the "`let`" keyword, while variables are declared using the "`var`" keyword. The difference between a variable and a constant is that a constant's value cannot be changed once it is initialized; a variable's value, on the other hand, can be changed as frequently as one wants.

```
var aVariable = 0
let aConstant = 1
```

Note: If you have programmed in languages like C or Java, you may wonder if semicolons are needed at the end of each statement. Semicolons are optional in Swift. You can put in semicolons if you want, but it is not required.

The first line declares a variable named “aVariable” to have a value of 0; the second line declares a contant named “aConstant” to have a value of 1.

You can also declare multiple variables in one line by using commas:

```
var a = 8, b = 9, c = 6, d = 4
```

Numeric and logical types in Swift

Each variable or constant is associated with a type. Here we will discuss the few major types that are very frequently used in Swift.

Type `Int`

`Int` are any whole numbers that do not contain a fractional component. One can also specify the number of bits used to store the number. The keywords `Int8, Int16, Int32, Int64` are used. `Int` can be either positive negative. If you want to declare a variable that is not signed (only positive), the keyword `UInt` can be used.

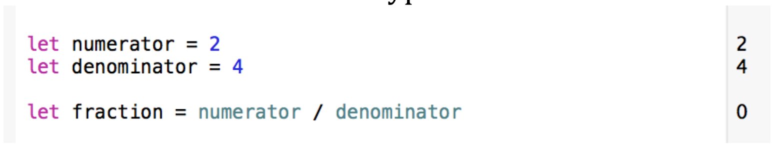
Type `Double`

`Double` are numbers with a fractional component. The major difference between a `Double` and an `Int` is mainly that Int does not have a fractional component. `Double` can also store bigger numbers than `Int`.

Type Float

`Float` are essentially the same as Double. A Double uses 64 bit, while a Float uses 32 bit.

The above three numerical types are used for all numerical calculations.



```
let numerator = 2                                  2
let denominator = 4                                4

let fraction = numerator / denominator             0
```

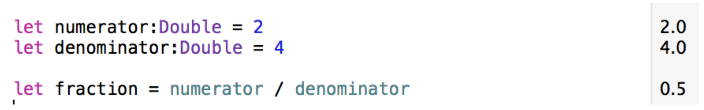
Here, two variables, `numerator` and `denominator` are created using the `let` keyword- their values cannot be altered once they are created. Variable `fraction` is than created; it is assigned to the result of `numerator` divided by `denominator`.

`Numerator` and `Denominator` are both of type Int. Swift is smart enough to infer from the fact that `2` and `4` do not have decimal fractions.

One may wonder why the result of fraction turns out to be 0 (show to the right), but not 0.5. The reason is for most programming languages, an `int` divided by an `int` results in an `int`. Therefore, the fraction in 0.5 is truncated out, leaving 0 as the integer answer.

There are two ways to fix this issue.

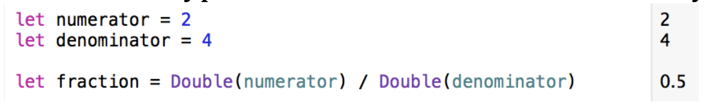
We can either strictly set the type for `numerator` and `denominator` when they are declared.



```
let numerator:Double = 2                          2.0
let denominator:Double = 4                        4.0

let fraction = numerator / denominator            0.5
```

We can also type cast them into `Double` when they are used to calculate `fraction`.



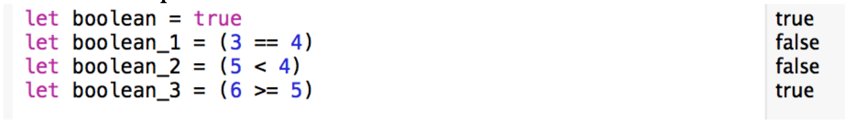
```
let numerator = 2                                           2
let denominator = 4                                         4

let fraction = Double(numerator) / Double(denominator)      0.5
```

Type Boolean

`Boolean` are variables that can only ever be of `true` or `false`.

Some examples can



```
let boolean = true                            true
let boolean_1 = (3 == 4)                      false
let boolean_2 = (5 < 4)                       false
let boolean_3 = (6 >= 5)                      true
```

Note that a == is used instead of = when we are comparing two values.

It is frequently used in an `if` statement, which will be discussed in the future.

The types described in this tutorial is used everywhere inside any program written with Swift. It is important to be very familiar of how these types work because it can become a real headache if one doesn't know 2 / 4 = 0.

It is also important to note that there are other types (such as Arrays and Dictionaries) that are used in Swift. They will be discussed in details in a future tutorial.