Lesson 03 - Cheat Sheet for Advanced Students

Reference:
https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

Available primitive types

- Integers (Int) : -3, -2, -1, 0, 1, 2, 3, …
- Floats (Float) : 3.14159, 2.71828
- Doubles (Double): just like Float
- Strings: Swift Strings are always unicode, so you can't index directly into them, because you don't know how many bytes a character may be.
  Only uses double-quotes (no singles)
  "Hello, World!"
- Booleans (Bool) : true/false
  boolean operators:
- nil : value for null
- Optional: Can be nil or a value from another specified type, e.g. Optional(Double)

Numeric Types (Integers, Floats, Doubles)

- The arithmetic operators are the ones you're used to:
  - `+, -, *, /, %`
- Use parentheses to control order-of-operations:
  - `(1 + 2) * 3`
- Binary comparison operators
  - `3 < 1, 3 <= 1`
  - `3 > 1, 3 <= 1`
  - `3 == 3`
  - `3 != 3.1`

Booleans

- Boolean constants
  - `true`
  - `false`
- Boolean operators
  - `&& (and)`

    - || (or)
    - ! (not)

Variables

```
var x = 1       // implicit type declaration and initialization
                // x will always be an Int hereafter
var x = 1.0     // the same, but will be a Double
var x : Int = 1 // type annotation and initialization as an Integer
var x : Int     // variable declared but not initialized
                // This will throw an error.
```

Constants

Constants, same as variables, but can't change, and uses "let" keyword to declare:

```
let y = 1
let y : Int = 1
```

Notes about variables and constants

- Note that you can't declare a variable twice in the same scope.

Control Flow - Conditionals (if/else statements)

if-else statements look this:

```
if temp <=0 {
    println("freezing")
} else if temp >= 100 {
    println("boiling")
} else {
    println("liquid")
}
```

You can add as many "else if" clauses as you like.

Ternary operator

The ternary operator looks like this:

```
let x = 123
```

```
let evenOrOdd : String = x % 2 == 0 ? "even" : "odd"
```

Control Flow - Loops

While loops. Using a conditional:

```
var s = 0
var i = 0
while i < 10 {
    s = s + i
}
println(s)
```

For-loops come in two varieties. One uses a conditional, and the other a data structure. We'll go over the first now:

```
var sum = 0
for (var i = 0; i < 10; i++ ) {
    sum += i
}
println(sum)
```

Use "break" to stop a loop prematurely.

```
while i < 10 {
    if i == 4 { break }
    i = i + 1
}
```

Use "continue" to skip the rest of the block, but continue the loop.

```
while i < 10 {
    if i % 2 == 0 { continue }
    i = i + 1
}
```

Optionals + Nil

Variables can be given a special type that wraps another type, called Optional. It's either nil or a value of the type it wraps.

```
var opt : Int?   // declares opt but is initialized as nil
                 // The type is Optional(Int)
opt              // opt will equal nil
```

Optionals can be initialized at declaration time or afterwards with a real value:

```
opt = 1
```

However! To get to the non-nil value, you have to "unwrap" it using !:

```
opt!
```

Note that you can't unwrap an Optional if it equals nil! In order to properly determine whether we have to unwrap or not, use the following if-let syntax:

```
if let _opt = opt {
    // Note the assignment syntax, =, and not ==.

    // We're not checking for equality, we're checking for
    // non-nilness by assigning the contained non-Optional value,
    // if there is one.

    // In this block, _opt is the unwrapped value (i.e. non-nil)
} else {
    // In this block, _opt is not available, and we know opt is nil.
}
```

Note that I like to use the "_opt" syntax to keep track of the unwrapped value in the true-clause, but it's neither a Swift convention nor a semantic notation. (It's really a Python convention for private variables.) It also makes it easier to pick a new variable name; you don't have to think about it anymore.

We can also provide a default value instead of checking with the if-let syntax above, using the "nil coalescing operator" (similar to || in Ruby or Javascript):

```
var optionalString : String?
let valueToDisplay = optionalString ?? default
println(valueToDisplay)
```