📱

Curren's Swift Cheat Sheet

Source Code: https://github.com/currentaber/CPSC_357

Constants

Declaring and assigning a constant:

```
let myConstant: String = "Curren"
```

Variables

Declaring and assigning a variable:

```
var myVariable: Int = 10
```

Functions

Sample form for writing functions:

```
// Takes two Int paremeters (x and y)
// Returns an Int
func add(_ x: Int, _ y: Int) -> Int {
  return x + y
}
```

Calling a function:

```
add(2, 2) // Returns 4
```

Operators

```
// Addition (+)
let sum = 2 + 1 // sum = 3

// Subtraction (-)
let diff = 4.2 - 2.0 // diff = 2.2

// Multiplication (*):
let prod = 3 * 5 // prod = 15

// Division (/):
let quotInt = 5 / 4 // quot = 1
let quotFloat = 5.0 / 4.0 // quot = 1.25

// Modulo (%):
let mod = 7 % 4 // mod = 3, AKA the remainder
```

Classes

Sample form for writing classes:

```
class Dog {
    let breed: String

    init (breed: String) {
        self.breed = breed
    }

    func sound() {
        print("Bark, Bark!")
    }
}
```

Instances

Creating a class instance:

```
let milo = Dog(breed: "Schnauzer")
print(milo.breed) // prints "Schnauzer\n"
milo.sound() // prints "Bark, Bark!\n"
```

Control Flow

Logical and Comparison Operators

Name	Type	Plain English
==	Comparison	"is equal to"
!=	Comparison	"is not equal to"
>	Comparison	"is greater than"
>=	Comparison	"is greater than or equal to"
<	Comparison	"is less than"
<=	Comparison	"is less than or equal to"
&&	Logical	AND
||	Logical	OR
!	Logical	NOT

If-else statements:

```
let rainfall = 10

if rainfall < 10 {
    print("Must be the desert...")
} else if rainfall < 30 {
    print("Not a lot of rain.")
} else if rainfall < 50 {
    print("We got a little rain.")
} else {
    print("Woah, that's a lot of rain.")
}
```

Switch statement:

```
let grade = 100

switch grade {
case 0...60:
    print("F")
case 61...69:
    print("D")
case 71...79:
    print("C")
case 80...89:
    print("B")
case 90...100:
    print("A")
default:
    print("Error")
}
```

Ternary conditional operator:

```
let x = 2
let y = 1
let bigger = x > y ? x : y // bigger = x
```

Strings

Declaring and initializing a string constant:

```
let message = "I told her, \"My name is Curren\""
print(message) // prints "I told her, "My name is Curren\n"
```

String concatenation:

```
var response = ""
if response.isEmpty {
    response += "Welcome to CPSC 357!"
}
print(response) // prints "Welcome to CPSC 357!\n"
```

String methods:

```
let student1 = "Curren"
let student2 = "Elon Musk"
let student3 = "curren"

if student1.lowercased() == student2.lowercased() {
    print("S1 and S2 names are the same") // doesn't print
}
```

```
if student1.lowercased() == student3.lowercased() {
    print("S1 and S3 names are the same") // prints
}

if student2.contains("Elon") {
    print("That's a cool name!") // prints
}

// String length
print(student1.count) // prints "6\n"
```

Arrays

Making an array:

```
var arrEmpty: [Int] = []
var arrFull: [Int] = [1, 2, 3, 4]
```

Array methods:

```
arrFull.count // 4
arrFull.isEmpty // false
arrFull.contains(2) // true
arrFull[0] = 2 // arrFull = [2, 2, 3, 4]
arrFull.append(5) // arrFull = [2, 2, 3, 4, 5]
arrFull.remove(at: 1) // arrFull = [2, 3, 4, 5]
arrFull.insert(10, at: 2) // arrFull = [2, 3, 10, 4, 5]
```

Dictionaries

Making a dictionary:

```
var dictEmpty: [Int: String] = [:]
var dictFull: [Int: String] = [1: "Bob", 2: "Steve"]
```

Dictionary methods:

```
dictFull[3] = "Claire" // dictFull = [1: "Bob", 2: "Steve", 3: "Claire"]
dictFull.updateValue("Steven", forKey: 2) // dictFull = [1: "Bob", 2: "Steven", 3: "Claire"]
dictFull[1] = nil // dictFull = [2: "Steve", 3: "Claire"]
```

```
dictFull.removeValue(forKey: 2) // dictFull = [3: "Claire"]
dictFull.keys // [3]
dictFull.values // ["Claire"]
if let person3 = dictFull[3] {
    print(person3) // prints "Claire\n"
}
```

Closures

Sample form for writing a closure:

```
let sayTheDate = { (date: String) -> String in
    return "Today is \(date)."
}
```

Calling a closure:

```
sayTheDate("September 30th") // "Today is September 30th"
```

Passing a closure into a function:

```
func notify(date: String, message: (String) -> String) {
    message(date)
}

notify(date: "September 30th", message: sayTheDate) // "Today is September 30th"
```

Trailing closures:

```
notify(date: "September 30th") { (date) -> String in
    return "Another way to say \(date)" // "Today is September 30th"
}
```

Guards

Sample form for writing a guard in a function:

```
func ageCheck(_ age: Int) {
    guard age >= 21 else { // Executes block if false
        print("You're not old enough to buy alcohol.")
        return
    }
    print("You can drink!")
}

ageCheck(19) // prints "You're not old enough to buy alcohol.\n"
ageCheck(21) // prints "You can drink!\n"
```

Tuples

Declaring and initializing a tuple:

```
var form = (name: "Curren", phone: 1234567890, email: "curren@email.com")
```

Getting and setting a tuple's values:

```
form.name // "Curren"
form.phone = 0987654321 // changes phone value
```

Enumerations

Sample form for writing an enumeration:

```
enum LightColor {
    case green
    case yellow
    case red
}
```

Using enumerations in switch statements:

```
var color: LightColor = .green

switch color {
case .green:
    print("Go") // prints
case .yellow:
    print("Slow")
case .red:
    print("Stop")
}
```