CPSC 357 – iOS Cheat Sheet

Jaewon Park

Strings

* Strings in Swift are fully Unicode-compliant so different languages can be used
* String Interpolation
  + Ex) let firstName = "Tim"

let city = "Cupertino"

let welcomeString = "Hello **\(firstName)**, welcome to **\(city)"**

* What if the string contains quotation marks?
  + Ex) let badString = "He said, "Hi there!" as he passed by." **(WRONG)**
  + let stringWithQuotationMarks = "He said, \"Hi there!\" as he passed by." **(CORRECT)**

Function

* func functionName (parameters) -> ReturnType {// body of the function }

Class

* Class instance are **reference** type
* Reference type instances means that each instance shares data, so, if you change one instance, the other instance will also change.

Struct

* Structure: provide a way to encapsulate data and functionality into re-usable instances
* Structure instance are **value** type
* Value type instances means you’re copying the instance, so, if you change one instance, the other instance will remain unchanged.

“struct Shirt {

var size: Size

var color: Color}

// Defines the attributes of a shirt.

let myShirt = Shirt(size: .xl, color: .blue)

// Creates an instance of an individual shirt.

let yourShirt = Shirt(size: .m, color: .red)

// Creates a separate instance of an individual shirt. ”

* Enumerations
  + The Size and Color types define a group of available options, called an enumeration

Control Flow

* for constant name in collection or range {// code to be executed}
  + Constant name <- the name to be used for a constant that will contain the current item from the collection or range through which the loop is iterating
  + collection or range <- the item through which the loop is iterating.
  + The constant name is not mandatory it can be change for ‘\_’
    - for \_in 1...5 {print(“Hello")}

Guard

* It is similar to the **if statement** with one major difference
  + The if statement runs when a certain condition is met.
  + The guard statement runs when a certain condition is not met.

Ternary Conditional Operator

* This is the if-statement This is using ternary

var largest: Int

let a = 15

let b = 4

largest = a > b ? a : b

“If a > b, assign a to the largest variable; otherwise, assign b.” In this case, a is greater than b, so its value is assigned to largest.”

var largest: Int

let a = 15

let b = 4

if a > b { largest = a}

else {largest = b}

For loop

* Tuple
* What if you need the index of each element in addition to its value? You can use the enumerated() method of an array or string to return a tuple—a special type that can hold an ordered list of values wrapped in parentheses—containing both the index and the value of each item:

for (index, letter) in “ABCD”.enumerated() {

print(”\(index): \(letter)”)}

Console Output:

0: A

1: B

2: C

3: D

Arrays

* Stores an ordered list of same-typed values.

Dictionaries

* This can be represented as either [String: Int] or Dictionary<String, Int>:

  var scores = [”Richard”: 500, “Luke”: 400, “Cheryl”: 800]