***SWIFT***

Notes came from this youtube video
Link: https://www.youtube.com/watch?v=FcsY1YPBwzQ

## Variables and Immutables

- mutables: var <variableName> : <dataType> = <assignment>
- immutable: let <varName> : <dataType> = <assignment>

Scala is statically typed. Think of the environment that swift supports as a chain of declarations and each new declaration is appended to the head of the linked list

*List of Datatypes*

1. _ : Int
2. _ : Float
3. _ : Double
4. _ : String
5. _ : Character
6. _ : Bool
7. all other extraneous datatypes, Long Double

## Conditionals / Loops / Operations

Syntax conditionals is the same as c++

```
if (condition1){
}else if{
}else{
}
```

*List of operations*

- &&, ||, !, <>, %, == , !=
- then you have all the bitwise operations

*Syntax for Switch statements*, no paraenthesis

```
switch <value> {
case <val1> :
case <val2> :
default :
}
```

**For loop syntax**

for <counter> in <lower> … <upper> { logic }  // standard way to write it
for _ in <lower> … <upper> { logic } // do this if you don't plan on utilizing the counter
for item in <array> // array iteration, but array is in immutable state

While loops are the same as in every other language just remember that you have to keep track of the counter manually or whatever it is.

## Functions in Swift

// basic structure of a function

```
func printMe(){
    print("hello world")
}
    // side note, \n is the newline character like in C
```

// function that takes in no parameter, but specifies a return type

```
func sum(){
    let a: Int = 5
    let b: Int = 5
    return a + b
}
```

// structure of a function that takes arguments

```
func add(_ arg1: Int, _ arg2: Int) -> Int {
    return arg1 + arg2
}
```

*Notes*

- You can use argument labels if you need too they are optional and just make the code easier to read when you start calling functions
- Swift does support tail recursion so whenever you can do it, do it.
-

## Classes and Objects in Swift

// basic definition of a class

```
class Person{
private
var name: String
var age: Int
var gender: String
public
init(){}
```

**<methods and such>**

## The UIKit

- Its apple's framework that lets us create ios apps, its the library of essential tools. A lot of user interface elements, UI view - stuff you can put on the screen
- it provides the window and view architecture for implementing your interface, the event handling interface, the event handling infrastructure for delivering multi-touch and other types of input to your app. The main run loop needed to manage interactions among the user, the system, and your loop.
    - Animation support
    - drawing / printing support, etc.

## Optionals

- In essence its shorthand for assignments based on condition.
    - Example
        - **let a: Int? = nil** // a can be an integer or a nil
- Before you can access the object of an optional variable , you must unbox it first.
        - <optional>**!** ← this unboxes the object
- Its safer practice to define optionals explicit and manually unbox them because it allows for type checking to catch any errors
        - **so** , try not to pre-unbox things like this
            - **let a = Int! = nil**

Things you want to go over in the official swift documentation

1. Optionals
2. Inheritance
3. the UIKit and its important features
4. **Closures** <- are these just functions passed in as parameters?