CS193p - Lecture 1

- Model: what our program does that is UI independent
- View: UI - user interacts with
- Controller: Glue between Model and View.

MVC - way of dividing up all the classes in our object oriented app into 3 different groups, model, view and controller group

Single inheritance in swift
All controllers or MVC controllers either directly or indirectly inherit from UIViewController

Properties(instance variables) can have behavior associated with them

Action - when something happens in UI, you want to invoke a method in your controller

Outlet - instead of creating a method create a property. Used when you want to change in UI from controller

let - used to declare constant. Why

1. user will know it won't change and doesn't have to worry about it
2. useful to create readonly arrays and dictionaries

Optionals:

- is a type like int, bool, dictionary, array
- It can have only 2 values
  - Not set expressed with keyword nil. Nil only means this optional is not set
  - Set. Means associated value, which can be of any other type. Actually it can also be an optional
- var currentTitle: String?
  - means its an optional and it's associated value is of type string. It's optional string
- Anything can be optional
- Optional is question mark at the end of declaration

Type Inference
let digit = sender.currentTitle

- Swift will infer the type all the time.
- Bad coding style if type declared like this let digit:String = sender.currentTitle
- print("\(digit)") gives Optional("5"). So how to get associated value of the optional string. Answer is put exclamation point in the end
- let digit = sender.currentTitle!
- Exclamation is unwrap this and give me the associated value

- What if optional was not set and exclamation tried to get the associated value. It would crash the app. Crashing is good for finding bugs before shipping to customers

@IBOutlet weak var display: UILabel! & @IBOutlet weak var display: UILabel? both works

- When declaring exclamation and question mark both mean optional
- display!.text = nil is how you set optional's associated value to not set

var userIsInMiddleOfTyping: Bool gives Viewcontoller has no initializers
- this is coz in swift all variable need to have an initial value. So either create an initializer or else say = false
- what about this: @IBOutlet weak var display: UILabel!
- optionals are treated differently, they are initially set to nil

Instead of unwrapping display so many times, can unwrap it in declaration and get rid of exclamation at other places. @IBOutlet weak var display: UILabel! called implicitly unwrapped optional