CS121 Tutorial 1
This is the first ios dev tutorial.

Purple bubbles give you information you’ll need to know.

Yellow Bubbles tell you what to do.

Orange bubbles tell you what you’re not expected to understand yet. 😊
Launch Xcode then click here to start a new project.
1. We are going to build an iOS application so click here.

2. We’ll build a single view application, so click here.

Xcode creates template code based on the type of project you select.

Here is the definition of a single view app.

3. Click on the next button.
1. Name your product HW 1.

2. Use com. followed by your first initial and last name.

3. Use HW1 as your “class prefix”.

Reversing the company’s DNS lookup string is a common convention for producing this ID. It won’t cause any problems if you don’t own your own domain name.
1. Choose iPhone as the device.

We can create apps for iPhone, iPad or both. We won’t need much screen space for this app so iPhone is good enough.

2. Make sure that Automatic Reference Counting (ARC) is the only block checked.

iOS uses ARC in lieu of garbage collection. We’ll talk more about this later.

3. Click next.
1. Navigate to the desktop to save your project folder.

2. Make sure this is unchecked.

3. Click create.

WARNING: You should move your project folder to your Charlie drive before logging off. Using a network drive while you’re working can be problematic. You are welcome to experiment though.
Woohoo! You’ve created your first iOS app.

The left panel is the navigator area.

The center panel is the editor area.

The right panel is the utility area.
1. Use this button to hide/reveal the navigator area.

2. The editor area has a console window that can be hidden/revealed with the center button.

3. Use this button to hide/reveal the utility area.
Xcode created these template files for us. Note that they all have a prefix HW1 as we requested!

This is the navigator toolbar.

This icon selects the project folder for navigation.
Click on the HW1ViewController.h file and it will appear in the editor area.
Click on the HW1ViewController.xib file to display the interface builder (IB) editor.

A xib file is an XML file that describes the user interface. Initially the file is empty but we can add to it through the IB editor.
2. Drag a "label" onto the iPhone screen mockup in the IB editor.

1. Click on the cube to see the object library.
The editor provides a variety of tools to position/format elements. Feel free to play around with it. (Make sure the Label in the IB editor is selected.)

Click here for quick help.

This describes how UILabels functions.
Click here for the attribute editor.

Various attributes can be edited here.
Click here for the size editor.

Size attributes can be edited here or simply by resizing/moving the label field on the iPhone screen mockup.
Before we go any further, let’s build our app! Click on the run button.

You’ll see various build messages here. Eventually you should get a popup message that says build succeeded. Sometime later a simulator will appear on your desktop.
The app is now running in the iPhone simulator and appears on your desktop.

2. Now drag a button onto the iPhone screen mockup in the IB editor.
1. Double click in the button to add text. Type PRESS.

2. Run the app.
Press the button!

I know you are thinking this is awesome. 😊 But just wait. Next we’ll make the app count how many times the button is pressed!
iOS uses a Model-View-Controller (MVC) architectural pattern. We’ll talk more about it in the next lecture. For now, all you need to know is that the HW1ViewController class is going to do the work of counting the button presses.

Class declarations have the following format: @interface newClassName: parentClassName
{
    // ClassMembers
}
// ClassMethods
@end

The parentheses are optional when there are no class members.

1. Stop the simulation.

2. Click on HW1ViewController.h to display the header file.
Add these two lines of code.

```swift
@property int numPresses;
-(IBAction) buttonPressed:(id) sender;
@end
```

This is the method that will get called when the button is pressed.

@property is a shortcut to declare a class member as well as accessor (getter and setter) methods or it.
Method declarations are preceded by + or - symbols. A + signifies a class (static) method; a – signifies an instance method.

The method type is IBAction. The IB prefix stands for Interface Builder.

The method name is buttonPressed.

The method takes one parameter, sender, which has type id.
Click on HW1ViewController.m to display the source file.

Xcode creates lots of template code. We’ll see what all of this does later.
@synthesize causes the accessor code to be automatically generated. @property and @synthesize must always be used together.
Implement the `buttonPressed` method.
1. Add this line to initialize the number of presses.

```c
numPresses = 0;
```

Compilation warnings and errors will be highlighted in the source code.

2. Run the app. Make sure it builds successfully and you have no warnings before moving on.
Next we need to “connect” our PRESS button in the UI to our buttonPressed method.
1. Click on `.xib` file to display the IB editor.

2. Click on the button.

3. Click the Connections Inspector icon.
The action we want to recognize is when the user lifts her/his finger off the button.
Make sure you can see the placeholders section before proceeding.

The Xcode window can get really cramped. Feel free to close the navigator panel for the next few steps.

Check here to hide/reveal the placeholder section.
1. Click on the “touch up inside” action.

2. Holding down the mouse button, drag the cursor over to the box labeled File’s Owner. A blue line should trail the cursor.
Let go of the mouse button and a bubble will appear. Click on buttonPressed.

At this point we see all possible functions that can be connected to the action. In our case we’ve only written one: buttonPressed.
The connection inspector now shows that buttonPressed is connected to the action.

The xib file is “owned by” the HW1ViewController object. We’ll talk more about the system architecture later.
At this point the easiest way to check that we are counting button presses is to add a “print statement.”
1. Open the HW1ViewController.m file.

2. Add this line of code.

In C we’d use
printf("%d",numPresses);

Objective-C strings (NSStrings to be precise) are preceded by an @ symbol.

We will talk more about NSStrings next time.
1. Click here to reveal the console.

2. Run the app.
Press the button!
Press the button again.
Here is `numPresses` after the first and second time you pressed the button!
Now we are going to display the number of button presses in the label field.
1. Open the HW1ViewController.h file.

2. Add this line.
We can connect interface elements to our code through *actions* and *outlets*.

The `IBOutlet` keyword is ignored by the compiler. It is used to make the code more readable by indicating the class member `numPressesLabel` is connected to a UI element.

This is related to ARC. We’ll talk more about it later.
1. Open the HW1ViewController.m file.

2. Synthesize our new member.

3. Add these lines of code. (You can get rid of the NSLog line.)
Here is another `NSString`.

Objective-c does not have constructors. Constructing an object is a 2-step process. First we allocate space for it and then we initialize it.

This is a method call *a la* Objective-c. We’ll cover the details later.
numPressesLabel is a UILabel. It has a text field to store an NSString.

Next we have to connect our numPressesLabel member to the label field in our UI.
1. Open the HW1ViewController.xib file.

2. Select File’s Owner.
1. Holding down the control and command keys, drag the cursor to the label field in the IB editor. A blue line should trail the cursor.

2. A bubble will appear. Let go of the mouse and another bubble will appear. Click on `numPressesLabel`. 
The connection editor now shows that the label field is connected to our `numPressesLabel` variable.

Run the app.
This isn’t good! We’ll fix it in a minute.

Click on PRESS
Now change your app so it starts with a 0 rather than the word Label. (There are several ways to do this. Take your pick.)
Zip up your project folder and move it to your wiki page for cs121.