Intro

We're going to build out our activity array and start loading activities in to our performance scene, just so we can see the actual activities in the array coming up into the UI.

Adding an activity array

The first thing we want to do is go to our performance view. Now let's add in our activity array to the controller behind this particular scene. Click on the View Controller icon. Then in the Identity Inspector rename it to `PerformanceView-Controller`. Change the file name as well.

Loading Data

Let's declare a variable for an array:

```
var activities = ["Running", "Swimming", "Jogging"]
```

This declares an array with three elements. There are a few things we need to handle to make sure that our Table View is going to be bound to this controller. Right now our scene is bound to a view controller. But the Table View is just bound to a UI Table View, it's not bound to the scene. So go back to the `PerformanceViewController`.

We are going to have `TableViewDataSource` and `UITableViewDelegate`. The data source is saying that this view controller is going to be where the data for this table is coming from. The delegate will allow us to make use of a few functions that, in turn, will be called by the Table View. Therefore let's add these functions now.

```
func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return activities.count
}
```

This way we're allowing three rows to display, and then we want to return a cell that's going to have our item in it. To do this, declare a cell as `UITableViewCell`.

```
let cell:UITableViewCell = UITableViewCell(style: UITableViewCellStyle.Default, reuseIdentifier: "
```

This is actually a constant, because we are using `let`. If you are using `var`, you can change a variable later, but a constant cannot be reassigned.

Now grab an object out of our array. This is going to be the index path, and item, so that gives me the actual index number.

```
cell.textLabel!.text = activities[indexPath.item]
```

By inserting the `!` symbol I'm saying that this is never going to be null, it's always going to have a value.

Now we need to return the cell, so here is the final version of the function:

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableVie
    let cell:UITableViewCell = UITableViewCell(style: UITableViewCellStyle.Default, reuseIdentifie
    cell.textLabel!.text = activities[indexPath.item]
    return cell
}
```

Now let's bind the Table View. Highlight it, hold down the `Ctrl` key and drag it to the View Controller icon. Select Data Source, do the same thing and select the Delegate.

We also want to access the activities array from all of our scenes. As we move through the course, you're going to see how we'll take the activities array and move it into the model with the `Activity` class. An activity manager will be utilized as well - it will handle passing of the array from scene to scene, because we're not going to access these activities directly.