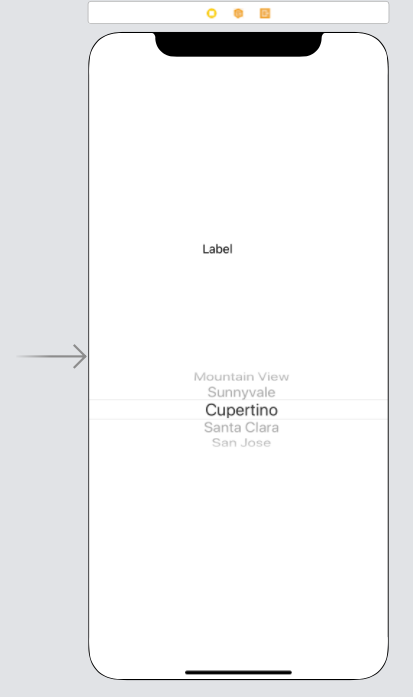
# UIPickerView

In this worksheet we will look at one of the other interface widgets that are available in the library – namely the Picker View. A picker is a scrollable list of values from which the user can select one value.

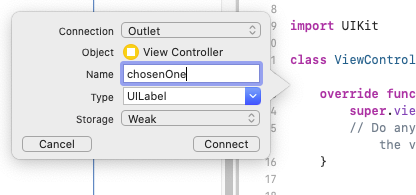
## Single Component Picker Example

In this example, we will create a simple picker with just a single list of items to select from.

1. Create a new Single View Application
2. Add a Label to the Main storyboard
3. Add a Picker View to the Main storyboard



1. Add an Outlet for the label, named chosenOne



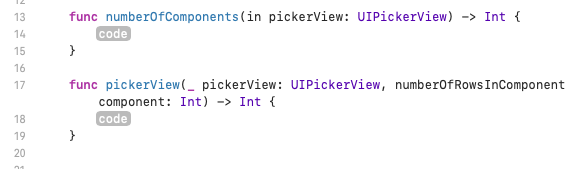
1. Add an Outlet for the Picker View called myPickerView



1. Add UIPickerViewDataSource and UIPickerViewDelegate to the ViewController class



1. This should pop up an error – if you pick fix, it will add in the code stubs that we need. Otherwise you will need to add these yourself



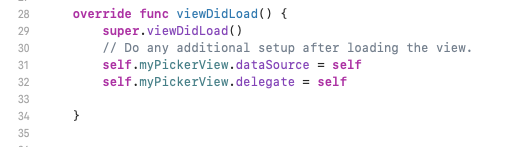
We will compete those functions in a moment. First, we will add the data we want to view in the picker.

1. Add the following array to the ViewController class



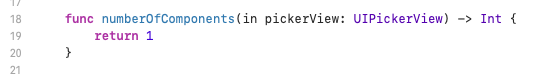
Next we need to link the picker view to the delegates.

1. Add the following code to the viewDidLoad override function

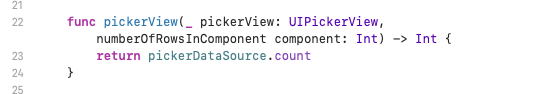


Now we can complete the stub functions

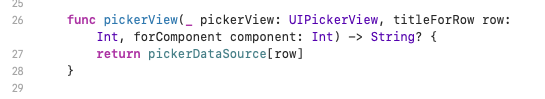
The numberOfComponents function needs to return the number of components (different spinners) in our picker. We only have one component, so this is trivial, complete the code in the following way



The numberOfRowsInComponent function needs to return the number of rows (different choices) are available on each component (different spinners). We only have one component, so again this is pretty trivial. Complete the code the following way



We now need to add a function, titleForRow to return the actual data to be displayed for each row of each component. Add the following code:



We now have all the code necessary to pick an item from the picker, we just need to do something with the picked item. In our case we will display it in the chosenOne label.

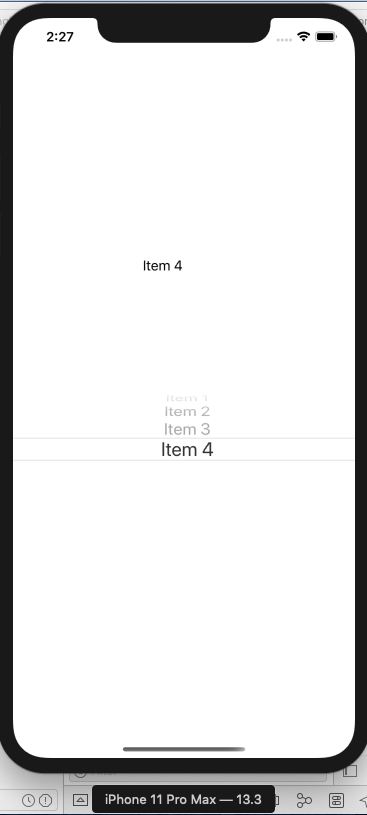
1. Add the following function:



The function is passed both the row and component number. Because we only have one component, we can pick the data to display based only on the row number. We are using a switch statement to achieve this.

In most of the cases, we are displaying a text string in the label; however, in case 3 we are setting the label to the value of the picked item (just to show how we would do it).

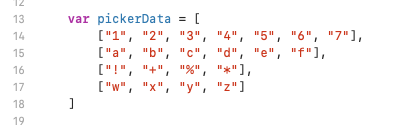
1. Run the simulator



## Multi-component Picker Example

In this example, we will include more than one component in the picker – i.e. the user can pick values from more than one spinner. Here the user can pick four values from four spinners to make a code.

1. Create a new Single View Application
2. Add a Label to the Main storyboard
3. Add a Picker View to the Main storyboard
4. Add an Outlet for the label, named selectedCode
5. Add an Outlet for the Picker View called codePicker
6. Add UIPickerViewDataSource and UIPickerViewDelegate to the ViewController class
7. Add the following data to View Controller class

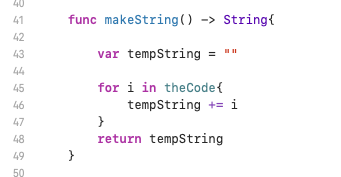


Here we can see the data for each spinner (component) is in a separate array. The spinners to not need to have the same number of values (rows) as each other. The picker has four components, the first component has seven rows, both components and rows are indexed from 0, so pickerData[3][2] refers to the third row of the fourth component, i.e. "y"

1. Add the following variable to hold the current code. We can see we are storing the code as an array of strings, one string for each component.

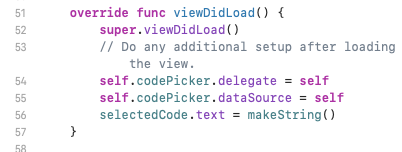


1. Add the following function to convert the code array (of strings) into a single string. We need this to convert our theCode array into a string which we can display in the label. We don't have to pass the array as a parameter as the theCode variable is global to the ViewController class and everything it contains. The function returns a string.



Next we need to link the picker view to the delegates and initialise the label

1. Add the following code to the viewDidLoad override function

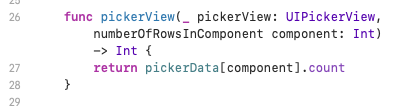


See how we are using the makeString function to convert the array to a string so that it can be displayed in the label.

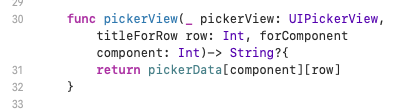
The numberOfComponents function needs to return the number of components (different spinners) in our picker. We have more than one component, so complete the code in the following way



The numberOfRowsInComponent function needs to return the number of rows (different choices) are available on each component (different spinners). We have more than one component. Complete the code the following way

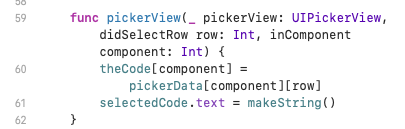


We now need to add a function, titleForRow to return the actual data to be displayed for each row of each component. Add the following code:



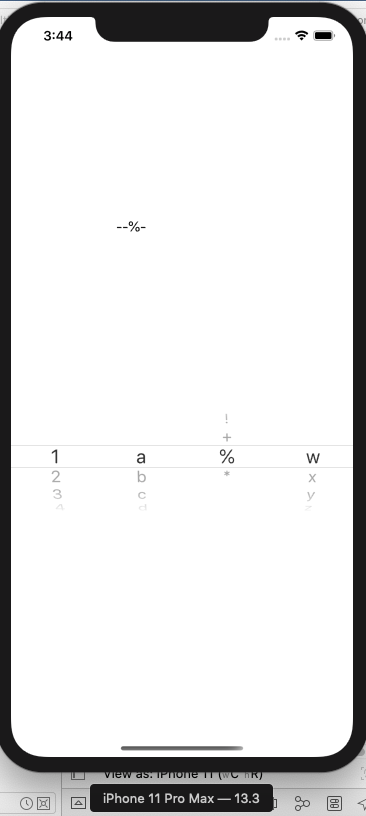
We now have all the code necessary to pick an item from the picker, we just need to do something with the picked item. In our case, we will display it in the selectedCode label.

1. Add the following function which will fire when the user picks from the picker:



First, we are changing the relevant value in the theCode array. Then we are converting the array to a string and setting the selectedCode to this string.

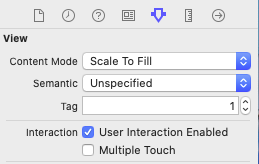
1. Run the simulator



## Multiple Picker Example

In this example, we will include more than one picker. We will have one picker for size and one for colour.

1. Create a new Single View Application
2. Add a Label to the Main storyboard
3. Add a Picker View to the Main storyboard
4. Change the Tag number for the Picker View to 1



1. Add an Outlet for the label, named sizeLabel
2. Add a second Label to the Main storyboard
3. Add a second Picker View to the Main storyboard
4. Change the Tag number for the Picker View to 2
5. Add an Outlet for the label, named colourLabel
6. Click and drag from the first Picker View to the View Controller icon at the top left of the storyboard, select dataSource
7. Do the same again, but select delegate
8. Repeat steps 33 and 34 for the second Picker View
9. Add UIPickerViewDataSource and UIPickerViewDelegate to the View Controller class



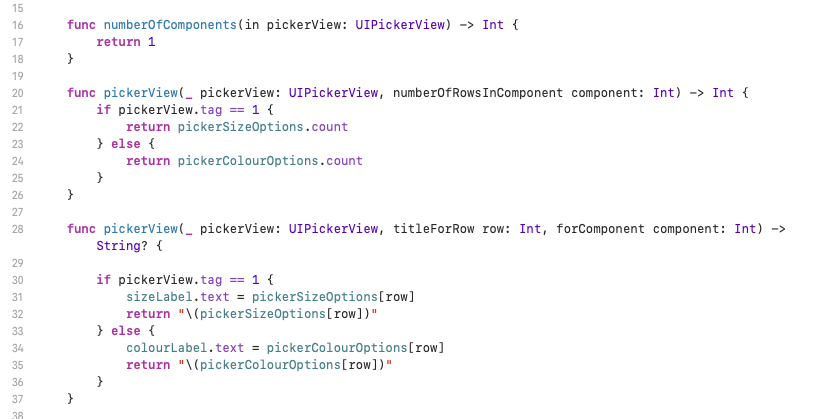
1. In the View Controller class add two variables to hold the options for the pickers



1. In the viewDidLoad function, set your options for the pickers



1. Implement the following code for the delegate and data source methods



1. Note how we are using the Tag value to determine which Picker View is being referred to (line 21 and 30)
2. Run the simulator

