**Solution: Traffic light**

(upbeat music) - [Instructor] To build my state machine for the traffic light challenge, the first step is to create an enum for the three possible states. I'll right click on the block diagram, go to the numeric palette and select enum constant. I'll right click on that and go to edit items and then click the insert button. Now I can type out my three states, green, press enter, yellow, enter, and red, and click okay. It's always good practice to turn the enum for a state machine into a type def, so I'll right click on it and choose make type def. Next, I'll build the framework for my state machine by adding a while loop from the functions palette structures menu, and then I'll add a case structure inside of that. Again from the structures menu, case structure. Next, to implement the transition between states, I'll right click on the while loop and add a shift register, and then I'll wire the enum value for the first state into that, which is the green state, and I'll wire the output from that shift register to the selector terminal on the case structure. That populates the case structure with two of the three states, so I'll need to right click on it and select add case for every value. Now I have cases for all three states. Now to implement the transitions, I'll hold the control key and drag the enum into here to make a copy of it in the yellow state. The yellow state always transitions to red, so I'll change that to red and then wire it over to the right shift register. I'll do the same thing for the other two states. In the green state, I'll always transition to yellow, so I'll wire that across and then for the red state, I'll always transition to green next and so I'll wire that through. That completes the transition logic for all three of these states, so now let's implement the functionality. I'll select the three indicators for the different light colors and drag those into the while loop to the right of the case structure. Inside of the red case, I'll add a Boolean true constant from the functions palette, Boolean menu, and I'll wire that to the red light. Switching over to the yellow case, I'll do the same thing for the yellow light and then in the green case, I'll do the same thing. By true constant wired to the green light. So that configures which light will be on in each of these states, but I also need to pass a Boolean false value to each of the other lights that will be off in each state. I could do that by adding Boolean false constants to each of the cases and wiring them to the remaining terminals, or I can use a shortcut. If I right click on the terminal, I can choose the option to use default if unwired. Selecting that option partially fills in the terminal, so it has a tiny little white square inside of it. In the case of Boolean terminals, the default value they pass if left unwired is false. So I'll right click on the other two terminals and select use default if unwired, as well. That finishes configuring which light will be on in each state. Now all that's left to control is the amount of time each light will be on. I'll open the functions palette, go to the timing menu, and select the wait milliseconds function, and I'll put that to the right of the case structure. I'll configure that amount of time to wait during each state in a similar way to how I configured the lights, so I'll add a numeric constant inside of the case structure and since the green case should wait for three seconds, I'll give it a value of 3000 or 3000 milliseconds and wire that to the wait function. I'll do the same for the yellow case, create a numeric constant, and the yellow case should wait for one second, so I'll give it a value of 1000, and then for the red case, since that should wait for two seconds, I'll give it a value of 2000. And that completes the state machine. I see LabVIEW's still giving me a broken run arrow because I forgot to connect a stop terminal on the while loop, so I'll right click to create control and that resolves the issue. Now I can run this VI. I see that it starts in the green state for three seconds then transitions to yellow for one second, red for two seconds, and then back to green. This was a very simple state machine, but if you enjoyed this challenge, you can build on it by expanding your own state machine to handle multiple lights in a traffic intersection or add a pedestrian walk button to practice handling user input.