# Part 1: the driver code

* The ‘entity’ part of the code is for declaring input and output variables of the module (mux)
* All SIGNALS and CONSTANTS are stored between the ‘architecture is’ and ‘begin’ lines
* std\_logic is a data type which can have either logic 1 or logic 0 values (it can also have other data values, but this is outside the scope of this video)
* i\_trafficlights: this is a naming convention; it can have any other name. we reference the entity in our module using the ‘.’ Operator, since we didn’t give our library a name, its default name becomes ‘work’; here we are creating an instance of the module in the test bench work.trafficlights(rtl)
* Generic maps and port maps are ways to assign values in the module to local values, module values on the right, local values on the left; values can mean signals or variables
* Clocked logic: AKA sequential logic, a type of logic where everything happens in sync with the clock signal, where a master clock produces a zero or one signal n number of times depending on the clock frequency or speed, and all changes occur at the edges of the clock signal; the clock period is the length of time for which the signal is either zero or one. “clock <= not clock after clock\_duration / 2;”, ‘clock’ here is a std\_logic signal, and it switches between zero and one every clock\_duration/2 seconds

# Part 2: the module

* The port is a way of connecting the inputs and outputs of a module to the test bench
* The generic keyword is similar to the #define in c
* The new data type allows us to determine the state of both traffic lights