Woah! We covered a lot in this lesson. Good work. Let’s review what we learned:

* Conditionals make it possible for programs to decide how to react to a wide variety of situations.
* if statements allow us to run a block of code **if** a condition is met.
* The boolean data type is either the value TRUE or FALSE and is the foundation of programmatic decision making.
* We use else to include a block of code to run when the condition is not met.
* Comparison operators evaluate a relationship between two operands and return a boolean value.
  + The less than operator (<)
  + The less than or equal to operator (<=)
  + The greater than operator (>)
  + The greater than or equal to operator (>=)
  + The Identical operator (===)
  + The not identical operator (!==)
* We can write conditionals with multiple if statements using the elseif construction.
* Instead of using a series of if statements when we want to compare a value, expression, or variable against many different possible values and run different code depending on which it matches, we can use a switch statement.
* The keyword break tells the computer to break out of the switch statement, without it, it will *fall through* the rest of the switch executing all the code until it reaches a break or the end of the statement.
* A ternary operator (?:) is shorthand conditional operator. It takes three operands (a condition to check, an expression to return if the condition is TRUE, and an expression to return if the condition is FALSE).
* Any value or expression inside a condition will be converted to TRUE or FALSE. We consider values that will convert to TRUE to be *truthy* and values that will convert to FALSE to be *falsy*.
* We can get user input from the terminal with the readline() function.

That really is a lot… Take some time to practice and review! You’re doing great.

Great job! You’ve learned the tools needed to craft programs with powerful decision making capabilities. Let’s review what we covered:

* By nesting conditionals within one another, we can create branching decisions.
* The logical operator || takes two different boolean values or expressions as its operands and returns a single boolean value. It returns TRUE if either its left operand or its right operand evaluate to TRUE.
* The logical && operator returns TRUE only if both of its operands evaluate to TRUE. It returns FALSE if either or both of its operands evaluate to FALSE.
* The logical not operator (!) takes only a right operand. It reverses the boolean value of its operand.
* The logical exclusive or operator (xor) returns TRUE only if either its left operand or its right operand evaluate to TRUE, but **not both** or **neither**.
* PHP includes alternate syntax for the || and && operators: we can use or in place of ||, and we can use and in place of &&. These operators work much the same way but have different [operator precedence](https://www.php.net/manual/en/language.operators.precedence.php).
* We can **include** code from one file inside another with include which allows us to write mode *modular* programs.