DEVI SELVI YANTI

2155201006

**PHP INTRODUCTION**

**PHP STRINGS AND VARIABLES**

**Review**

* Strings are collections of text that the computer treats as a single piece of data.
* A string can be any length and contain any letters, numbers, symbols, or spaces surrounded by quotation marks.
* In order to include certain characters inside strings we have to use escape sequences.
* An *operator* is a character that performs a task in our code.
* We can use the concatenation operator (.) to combine two strings into one.
* [Variables](https://www.codecademy.com/resources/docs/php/variables) are a fundamental programming concept which allow us to easily reuse data in our code.
* We declare a variable using the dollar sign ($) followed by the variable name and then use the assignment operator (=) to give it a value.
* PHP has variable parsing which allows us to include variables directly in our strings.
* Once a variable has been assigned, we can change its value. This is called reassignment.
* We can create an alias for a variable, instead of just a copy, using the reference assignment operator (=&).
* Operations to the right of the assignment operator will be evaluated before assignment takes place.
* The concatenating assignment operator (.=) is a shorthand notation for reassigning a string variable to its current value appended with another string value.

If that was a lot to take in, don’t worry about memorizing everything right away. Remember that when you want to explore more about the language, [the documentation](http://php.net/manual/en/langref.php) is a great place to get comfortable exploring.

**PHP NUMBERS**

**Review**

* PHP has two number data types: integers and floating point numbers
* We can use arithmetic operators for performing math operations:

| **Operation:** |  | **Example:** |
| --- | --- | --- |
| Addition | + | echo 1 + 4.5; // Prints: 5.5 |
| Subtraction | - | echo 9 - 1; // Prints: 8 |
| Multiplication | \* | echo -1.9 \* 2.9; // Prints: -5.51 |
| Division | / | echo 9 / 1; // Prints: 9 |
| Modulo | % | echo 11 % 3; // Prints: 2 |
| Exponentiation | \*\* | echo 8\*\*2; // Prints: 64 |

* Operations have an order of precedence meaning that certain types of operations in a chain will be evaluated before others: first evaluated will be any operation wrapped in **p**arenthesis (()), next **e**xponents (\*\*), then **m**ultiplication (\*) and **d**ivision (/), and finally **a**ddition (+) and **s**ubtraction (-). The acronym PEMDAS can be a helpful way of remembering the order.
* We can assign number values to variables and then perform numerical operations with them.
* We can use mathematical assignment operators as a shorthand when reassigning number variables:

| **Operation:** | **Long Syntax:** | **Short Syntax:** |
| --- | --- | --- |
| Add | $x = $x + $y | $x += $y |
| Subtract | $x = $x - $y | $x -= $y |
| Multiply | $x = $x \* $y | $x \*= $y |
| Divide | $x = $x / $y | $x /= $y |
| Mod | $x = $x % $y | $x %= $y |

**PHP ARRAY AND LOOPS**

**ASSOCIATIVE ARRAYS**

**Review**

You learned so much in this lesson! Let’s review:

* *Associative arrays* are data structures in which string or integer *keys* are associated with *values*.
* We use the => operator to associate a key with its value. $my\_array = ["panda"=>"very cute"]
* To print an array’s keys and their values, we can use the print\_r() function.
* We access the value associated with a given key by using square brackets ([ ]). For example: $my\_array["panda"] will return "very cute".
* We can assign values to keys using this same indexing syntax and the assignment operator (=): $my\_array["dog"] = "good cuteness";
* This same syntax can be used to change existing elements. $my\_array["dog"] = "max cuteness";
* We can remove a *key=>value* pair entirely using the PHP unset() function.
* Keys can be integers. In fact, ordered [arrays](https://www.codecademy.com/resources/docs/php/arrays) are just arrays in which integer keys have been assigned to the values automatically.
* In PHP, associative arrays and ordered arrays are different uses of the same data type.
* The union (+) operator takes two array operands and returns a new array with any unique keys from the second array appended to the first array.
* When writing function with array parameters, we can pass the array by value or by reference depending on our intent.

**LOOPS**

**Review**

Here is a summary of the topics covered in this lesson:

* while [loops](https://www.codecademy.com/resources/docs/php/loops) execute only as long as their conditional evaluates to TRUE.
* do…while loops always execute at least once and then continue executing while their conditional is TRUE.
* for loops contain 3 expressions and are frequently used to execute a code block a specific number of times.
  + The first expression is executed prior to the first iteration.
  + The second expression is evaluated prior to each iteration. If TRUE, the code block executes. Otherwise, the loop terminates.
  + The third expression is evaluated after each iteration.
* foreach loops are used for iterating over the elements of an array. The key and value of each element is available in the code block.
* break is used to end execution of a loop early.
* continue is used to end execution of a loop iteration early and continues to the next iteration.

**PHP CONDITIONLS AND LOGIC**

**LOGICAL OPERATORS AND COMPOUND CONDITIONS**

**Review**

* By nesting [conditionals](https://www.codecademy.com/resources/docs/php/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](https://www.codecademy.com/resources/docs/php/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.

**BOOLEANS AND COMPARISON OPERATORS**

**Review**

* 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.

.