Learn PHP Variables, Strings, and Numbers

Parsing Variables within PHP Strings

In PHP, variables can be parsed within strings specified with double quotes (`"`).
This means that within the string, the computer will replace an occurence of a variable with that variable's value.
When additional valid identifier characters (ie. characters that could be included in a variable name) are intended to appear adjacent to the variable's value, the variable name can be wrapped in curly braces `{}` , thus avoiding confusion as to the variable's name.

```
$my_var = "cat";

echo "There is one $my_var on the mat";

/* If there were three cats, then you can
type: */
echo "There are three {$my_var}s on the
mat ";
/* The curly braces help to avoid
confusion between the variable name and
the letter s, so PHP does not consider the
variable name as my_vars */
```

Reassignment of PHP Variables

In PHP, variables are assigned values with the *assignment* operator (`=`). The same variable can later be reassigned a new value using the same operator.
This process is known as *reassignment*.

```
$var1 = "Bob";
echo $var1;
// var1 holds the value "Bob"

$var1 = "John";
echo $var1;
// var1 now holds the value "John"
```

Concatenating Strings in PHP

In PHP, if you want to join two strings together, you need to use the `.` operator.
This process is called *concatenation*. Put the `.` operator between the two strings in order to join them.
Note that strings are joined as-is, without inserting a whitespace character. So if you need to put spaces, you need to incorporate the whitespace manually within the string.

```
echo "Hello,"." welcome to Codecademy!";
// prints - Hello, welcome to Codecademy!
```

Appending a String in PHP

In PHP, there is a shortcut for appending a new string to the end of another string. This can be easily done with the *string concatenation assignment operator* (`.=`).
This operator will append the value on its right to the value on its left and then reassign the result to the variable on its left.

```
$str = "Hello, ";
$str .= "World!";

echo $str;
// Output: Hello, World!
```

PHP Strings

In PHP, a *string* is a sequence of characters surrounded by double quotation marks. It can be as long as you want and contain any letters, numbers, symbols, and spaces.

```
echo "Hello 123"; // prints Hello 123
```

PHP copying variables

In PHP, one variable's value can be assigned to another variable.
This creates a copy of that variable's value and assigns the new variable name to it.
Changes to the original variable will not affect the copy and changes to the copy will not affect the original. These variables are entirely separate entities.

```
$original = "Ice T";
$copy = $original;
$orginal = "Iced Tea";
echo $copy; // "Ice T";
```

PHP String Escape Sequences

In PHP, sometimes special characters must be escaped in order to include them in a string. *Escape sequences* start with a backslash character (`\`).
There are a variety of escape sequences that can be used to accomplish different tasks. For example, to include a new line within a string, the sequence `\n` can be used. To include double quotation marks, the sequence `\"` can be used. Similarly, to include single quotes, the sequence `\'` can be used.

```
echo "Hello, World!\nThis is a String!";
/* prints-
Hello, World!
This is a String!
*/
```

PHP Evaluation Order during Assignment

In PHP, when an assignment takes place, operations to the right of the assignment operator (`=`) will be evaluated to a single value first. The result of these operations will then be assigned to the variable.

```
$var1 = 5 + 6 / 2;
/* Here, the operations to the right of
the assignment operator will be carried
out first. So first 6 will be divided by 2
(6 / 2 = 3). Then 3 will be added to 5 (5
+ 3 = 8). Finally, the value 8 will be
assigned to $var1. */

echo $var1;
// Output: 8
```

PHP Variables

In PHP, variables are assigned values with the assignment operator (`=`).
Variable names can contain numbers, letters, and underscores (`_`). A *sigil* (`$`) must always precede a variable name. They cannot start with a number and they cannot have spaces or any special characters.
The convention in PHP is to use *snake case* for variable naming; this means that lowercase words are delimited with an underscore character (`_`). Variable names are case-sensitive.

```
$my_variable = "Hello";

$another_cool_variable = 25;
```

PHP Reference Assignment Operator

In PHP, the *reference assignment operator* (`=&`) is used to create a new variable as an alias to an existing spot in memory.
In other words, the reference assignment operator (`=&`) creates two variable names which point to the same value. So, changes to one variable will affect the other, without having to copy the existing data.

```
$var1 = 5;
$var2 =& $var1;

$var1 = 6;
echo $var2;
// Output: 6
```

Integer Values in PHP

PHP supports *integer* values for numbers.
Integers are the set of all whole numbers, their negative counterparts, and zero. In other words, an *integer* is a number of the set $\mathbb{Z}$ = {..., -2, -1, 0, 1, 2, ...}.

Exponentiation Operator in PHP

PHP supports an arithmetic operator for *exponentiation* (`**`).
This operator gives the result of raising the value on the left to the power of the value on the right.

```
$n = 4;
echo 2 ** 4;
// Output: 16
```

Arithmetic Operators in PHP

PHP supports *arithmetic operators* for addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`).
PHP operators will return *integers* whenever the result of an operation evaluates to a whole number. If the result evaluates to a fraction or decimal, then it will return a *floating point number*.

```
$a = 6 / 3;
// The variable $a will hold an integer
value, since the operation evaluates to a
whole number.

$b = 7 / 3;
// The variable $b will hold a floating
point value, since the operation evaluates
to a decimal number.
```

codecademy

The Modulo Operator

PHP supports a *modulo operator* (`%`). The *modulo operator* returns the remainder of the left operand divided by the right operand. Operands of a modulo operation are converted to integers prior to performing the operation. The operation returns an integer with the same sign as the dividend.

```
$a = 9 % 2.3;
//2.3 is converted to the integer, 2. The
remainder of 9 % 2 is 1. So the variable
$a will hold the integer value 1.

$b = -19 % 4;
//The remainder of this operation is -3.
So the variable $b will hold the integer
value -3.

$c = 20 % 2;
//The remainder of this operation is 0. So
the variable $c will hold the integer
value 0.
```

Floating Point Numbers in PHP

PHP supports *floating-point* (decimal) numbers. They can be used to represent fractional quantities as well as precise measurements. Some examples of floating point numbers are `1.5` , `4.231` , `2.0` , etc.

Print   Share