Introduction to PHP – Part 2

Primary source: https://www.w3schools.com/php/

Internet Programming 2, Lesson 2

April 2024

PHP Data Types
PHP Strings
PHP Numbers
PHP Constants
PHP Operators

Recall: Part 1 About PHP:

Basic PHP Syntax: <?php…?>; echo

Comments in PHP: // # /*…*/

PHP Case Sensitivity: Keywords are case-insensitive; all variable names are case-sensitive.

PHP Variables & Scope: $name; PHP is loosely typed; global, local and static variables

PHP Constants: use define() function to created constants, used for values that remain the same for a period – e.g. vatRate; Globally available

echo and print – technical differences: speed; return value, number of parameters

WAMPSERVER – development environment

PHP Data Types

Variables can store data of different types, and different data types can do different things.
PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

A string is a sequence of characters, like "Hello world!".
A string can be any text inside quotes. You can use single or double quotes:

```php
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

Output

```
Hello world!
Hello world!
```

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

Output

```
int(5985)
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```php
<?php
$x = 10.365;
var_dump($x);
?>
```

Output

float(10.365)

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```php
$x = true;
$y = false;
```

PHP Array

An array stores multiple values in one single variable.
In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```php
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

Output

array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

- A class is a template for objects, and an object is an instance of a class.
- When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
- Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.
- When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
- If you create a __construct() function, PHP will automatically call this function when you create an object from a class.
- Example on next slide…

```php
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . "
" . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

Output

My car is a black Volvo!
My car is a red Toyota!

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

```php
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

Output

```
NULL
```

PHP 5 echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small:
echo has no return value while print has a return value of 1 so it can be used in expressions.
echo can take multiple parameters (although such usage is rare) while print can take one argument.
echo is marginally faster than print.

We can use single or double quotation marks for string values. However, there is a different when used in the output statements. Try out the PHP below:

```
<?php
$x = 10;
echo "echo double: the value of x is $x <br />";
echo 'echo single: the value of x is $x <br />';
print "print double: the value of x is $x <br />";
print 'print single: the value of x is $x <br />';
?>
```

Output

```
echo double: the value of x is 10
echo single: the value of x is $x
print double: the value of x is 10
print single: the value of x is $x
```

PHP Strings

Recall that the period (full-stop / dot) is the string concatenation symbol in PHP.

```
<?php
$txt1 = "Hello";
$txt2 = " world!";
echo $txt1 . $txt2;
?>
```

Output

Hello world!

Get The Length of a String

The PHP **strlen()** function returns the length of a string.

Example:

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

The output of the code above will be: 12.

Count The Number of Words in a String

The PHP **str_word_count()** function counts the number of words in a string:

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

The output of the code above will be: 2.

Reverse a String

The PHP **strrev()** function reverses a string:

Example

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

The output of the code above will be: !dlrow olleH.

Search For a Specific Text Within a String

The PHP **strpos()** function searches for a specific text within a string.
If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
The example below searches for the text "world" in the string "Hello world!":

Example

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

The output of the code above will be: 6.

NOTE: The first character position in a string is 0 (not 1).

Return part of a string:

The **substr()** function returns a part of a string.

```
<?php
// Positive numbers:
echo substr("Hello world",0,10)."<br>";
echo substr("Hello world",1,8)."<br>";
echo substr("Hello world",0,5)."<br>";
echo substr("Hello world",6,6)."<br>";
echo "<br>";

// Negative numbers:
echo substr("Hello world",0,-1)."<br>";
echo substr("Hello world",-10,-2)."<br>";
echo substr("Hello world",0,-6)."<br>";
?>
```

```
Hello worl
ello wor
Hello
world

Hello worl
ello wor
Hello
```

Note: If the start parameter is a negative number and length is less than or equal to start, length becomes 0.

Syntax

substr(string,start,length)

Parameter	Description
string	Required. Specifies the string to return a part of
start	Required. Specifies where to start in the string
	A positive number - Start at a specified position in the string
	A negative number - Start at a specified position from the end of the string
	0 - Start at the first character in string
length	Optional. Specifies the length of the returned string. Default is to the end of the string.
	A positive number - The length to be returned from the start parameter
	Negative number - The length to be returned from the end of the string

Replace Text Within a String

The PHP **str_replace()** function replaces some characters with some other characters in a string.

The example below replaces the text "world" with "Dolly":

Example

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

The output of the code above will be: Hello Dolly!

The **strtolower()** function converts a string to lowercase.

```
<?php
echo strtolower("Hello WORLD.");
?>
```

hello world.

Related functions:

- strtoupper() - converts a string to uppercase
- lcfirst() - converts the first character of a string to lowercase
- ucfirst() - converts the first character of a string to uppercase
- ucwords() - converts the first character of each word in a string to uppercase

The **strtoupper()** function converts a string to uppercase.

```
<?php
echo strtoupper("Hello WORLD!");
?>
```

HELLO WORLD!

Related functions:

- strtolower() - converts a string to lowercase
- lcfirst() - converts the first character of a string to lowercase
- ucfirst() - converts the first character of a string to uppercase
- ucwords() - converts the first character of each word in a string to uppercase

Return the ASCII value of a character:

The ord() function returns the ASCII value of the first character of a string.

```
<?php
echo ord("h")."<br>";
echo ord("hello")."<br>";
?>
```

104
104

Syntax ord(string)

Parameter	Description
string	Required. The string to get an ASCII value from

Return characters from different ASCII values:

The chr() function returns a character from the specified ASCII value.

```
<?php
echo chr(52) . "<br>"; // Decimal value
echo chr(052) . "<br>"; // Octal value
echo chr(0x52) . "<br>"; // Hex value
?>
```

4
*
R

The ASCII value can be specified in decimal, octal, or hex values. Octal values are defined by a leading 0, while hex values are defined by a leading 0x.

Syntax chr(ascii)

Parameter	Description
ascii	Required. An ASCII value
Return Value:	Returns the specified character

Check out the complete PHP String reference at: https://www.w3schools.com/php/php_ref_string.asp

PHP Numbers

One thing to notice about PHP is that it provides automatic data type conversion.
So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.
This automatic conversion can sometimes break your code.

PHP Integers

2, 256, -256, 10358, -179567 are all integers.
An integer is a number without any decimal part.
An integer data type is a non-decimal number between -2147483648 and 2147483647 in 32 bit systems, and between -9223372036854775808 and 9223372036854775807 in 64 bit systems.
A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.
Note: Another important thing to know is that even if 4 * 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

```php
<?php
$x = 5985;
var_dump(is_int($x));

$x = 59.85;
var_dump(is_int($x));
?>
```

Output

```
bool(true)
bool(false)
```

Here are some rules for integers: [Aaaah but you know this…]

- An integer must have at least one digit
- An integer must NOT have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

PHP has the following predefined constants for integers:

- PHP_INT_MAX - The largest integer supported
- PHP_INT_MIN - The smallest integer supported
- PHP_INT_SIZE - The size of an integer in bytes

PHP has the following functions to check if the type of a variable is integer:

- is_int()
- is_integer() - alias of is_int()
- is_long() - alias of is_int()

PHP Floats

A float is a number with a decimal point or a number in exponential form.
2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.
The float data type can commonly store a value up to 1.797693134862 3E+308 (platform dependent), and have a maximum precision of 14 digits.

PHP has the following predefined constants for floats (from PHP 7.2):

- PHP_FLOAT_MAX - The largest representable floating point number
- PHP_FLOAT_MIN - The smallest representable positive floating point number
- - PHP_FLOAT_MAX - The smallest representable negative floating point number
- PHP_FLOAT_DIG - The number of decimal digits that can be rounded into a float and back without precision loss
- PHP_FLOAT_EPSILON - The smallest representable positive number x, so that x + 1.0 != 1.0

PHP has the following functions to check if the type of a variable is float:

- is_float()
- is_double() - alias of is_float()

```php
<?php
$x = 10.365;
var_dump(is_float($x));
?>
```

Output

```
bool(true)
```

PHP Infinity

A numeric value that is larger than PHP_FLOAT_MAX is considered infinite.
PHP has the following functions to check if a numeric value is finite or infinite:

- is_finite()
- is_infinite()

However, the PHP var_dump() function returns the data type and value:

```
<?php
$x = 1.9e411;
var_dump($x);
?>
```

Output

float(INF)

PHP NaN

NaN stands for Not a Number.
NaN is used for impossible mathematical operations.
PHP has the following functions to check if a value is not a number:

- is_nan()

However, the PHP var_dump() function returns the data type and value:

```
<?php
$x = acos(8);
var_dump($x);
?>
```

Output

float(NAN)

PHP Numerical Strings

The PHP is_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.

```php
<?php
$x = 5985;
var_dump(is_numeric($x));

$x = "5985";
var_dump(is_numeric($x));

$x = "59.85" + 100;
var_dump(is_numeric($x));

$x = "Hello";
var_dump(is_numeric($x));
?>
```

Output

```
bool(true)
bool(true)
bool(true)
bool(false)
```

Note: From PHP 7.0: The is_numeric() function will return FALSE for numeric strings in hexadecimal form (e.g. 0xf4c3b00c), as they are no longer considered as numeric strings.

PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The (int), (integer), or intval() function are often used to convert a value to an integer.

```php
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;

echo "<br>";

// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

Output

```
23465
23465
```

PHP Constants

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (*no $ sign before* the constant name).
- **Note:** Unlike variables, constants are *automatically global across the entire script*.

Create a PHP Constant

To create a constant, use the define() function.

Syntax

```
define(name, value, case-insensitive)
```

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Example;: Create a constant with a **case-sensitive** name:

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

Output

Welcome to W3Schools.com!

Example: Create a constant with a **case-insensitive** name:

```php
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

Output

Welcome to W3Schools.com!

PHP Constant Arrays

In PHP7, you can create an Array constant using the define() function.

Example: Create an Array constant:

```php
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
```

Output

Alfa Romeo

Constants are Global

Constants are automatically global and can be used/accessed across the entire script.

Example: This example uses a constant inside a function, even if it is defined outside the function:

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
  echo GREETING;
}

myTest();
?>
```

Output

Welcome to W3Schools.com!

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.
The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.
The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

```php
<!DOCTYPE html><html>
<body>
<?php
// if empty($user) = TRUE, set $status = "anonymous"
echo $status = (empty($user)) ? "anonymous" : "logged in";
echo("<br>");

$user = "John Doe";
// if empty($user) = FALSE, set $status = "logged in"
echo $status = (empty($user)) ? "anonymous" : "logged in";
?>
</body>
</html>
```

Output

anonymous
logged in

```
<!DOCTYPE html>
<html>
<body>

<?php
  // variable $user is the value of $_GET['user']
  // and 'anonymous' if it does not exist
  echo $user = $_GET["user"] ?? "anonymous";
  echo("<br>");

  // variable $color is "red" if $color does not exist or is null
  echo $color = $color ?? "red";
?>

</body>
</html>
```

Output

anonymous
red

Exercise 1, continued: *Note that for now, we will not obtain any input from the user, but will hard-code the potential input into the source code, merely for learning purposes – when we cover forms – which will be mechanism for transmitting user input from the client browser to the server, we will return to these examples and modify them to work in an environment where the input is provided by the user.*

1-3. Write the following PHP Script – save your code in a file called Ex1-3.php
Create the following variables in php and assign the values indicated:

- A variable to hold the length of a rectangle, assign the value 10 m to the variable.
- A variable to hold the width of a rectangle and assign the value 20m to such a php variable.
- A variable to hold the radius of a circle and assign this the value of 2m.
- Now write the code to determine and display the area of the rectangle and the area of the circle.

1-4. Write a php Script to split an email address into its constituent parts. Cater for the following formats: userID@mut.ac.za; userID@mut.com; firstName.surname@mut.ac.za; [assign these to a variable and the perform the splits]

1-5. Write a PHP Script to split a website URL into its constituent parts. Cater for the following formats: http://mut.ac.za; https://www.mut.ac.za; https://moodle.mut.ac.za; http://www.Ramdeyal.com; https://www.pkramdeyal.co.za;

1-6. Write a PHP Script to determine the number of each coin denomination in an amount that a cashier/teller will have to give a customer as change. Assign the test amount [in the range 0 to R9.99c) to a variable and then determine the number of coins of each denomination:1c; 2c; 5c; 10c; 20c; 50c, R1, R2, R5…The main objective is to give the customer the LEAST number of coins possible. Assume that the teller has an unlimited number of coins for each denomination.

For example, if the amount of change due to the customer is 88c, then your output should be as follows:

50c	1
20c	1
10c	1
5c	1
2c	1
1c	1

Total number of coins is: 6

Note that the change can also be given as:

50c	1
20c	1
10c	1
5c	1
1c	3

But does not consider the issue of using the minimum number of coins possible – here it is 7. Hence the first solution is correct!