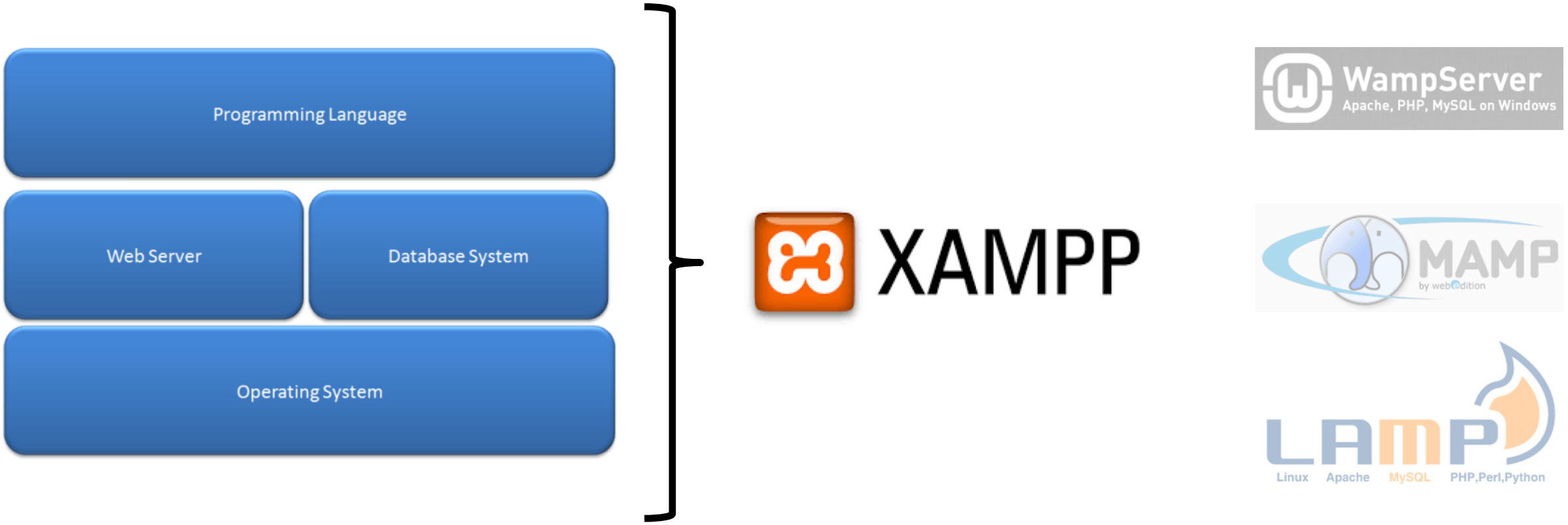
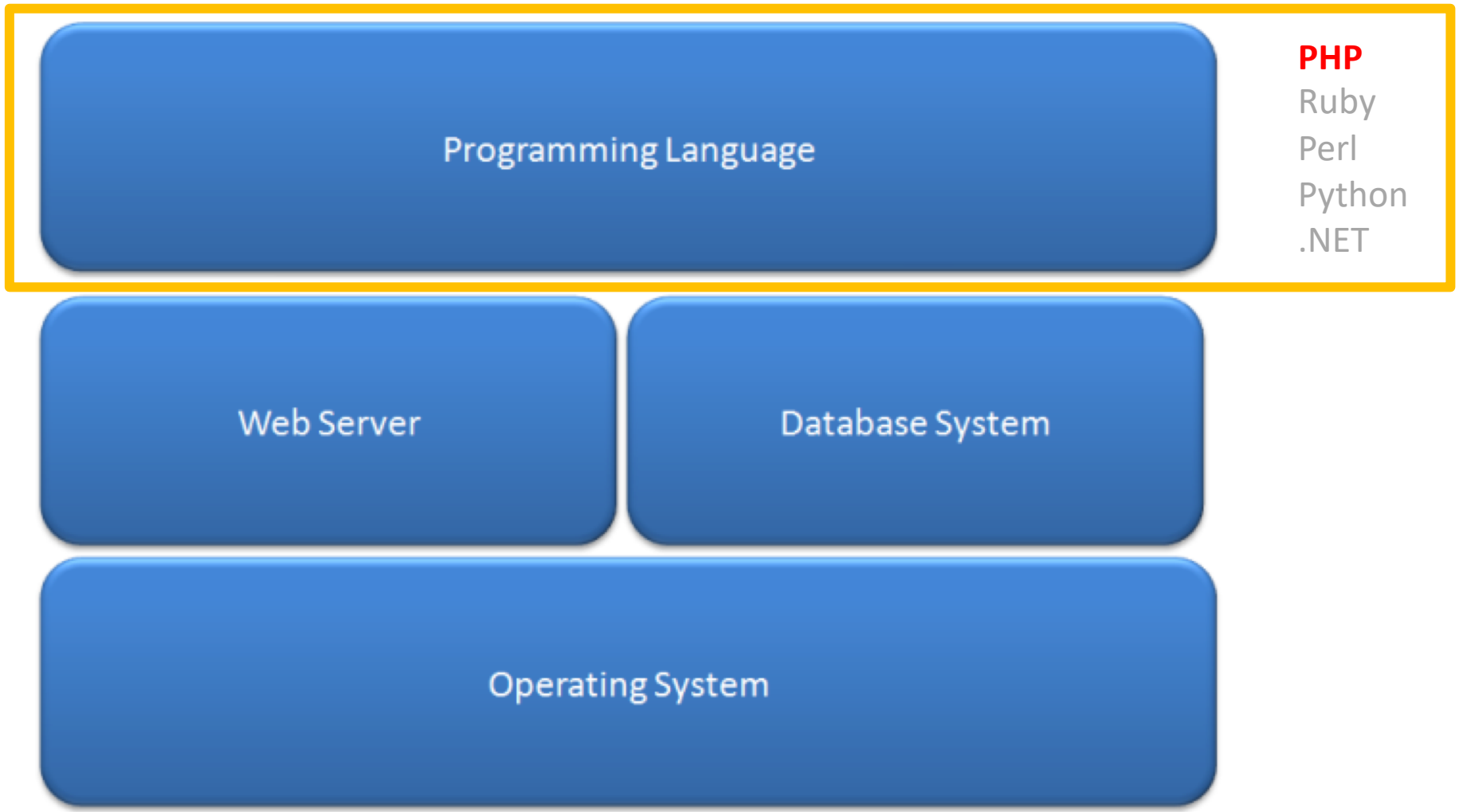
COMP30680
Web Application Development

PHP part 1

David Coyle
d.coyle@ucd.ie

A word on our development stack




Programming Language
PHP
Ruby
Perl
Python
.NET
Web Server
Database System
Operating System

PHP

PHP is an acronym for "PHP: Hypertext Preprocessor"
PHP files have extension ".php"
PHP files can contain text, HTML, CSS, JavaScript, and PHP code
PHP code are executed on the server, and the result is returned to the browser as plain HTML

Why PHP?
Runs on various platforms.
Is compatible with almost all servers used today.
Supports a wide range of databases.
Is free and open source.

Note

The newest version of PHP is PHP 7, but PHP 5 is still widely used.

The core topics we discuss are equally applicable to both PHP 5 and PHP 7.

What about to PHP 6?

echo and print

In PHP there are two basic ways to get output: echo and print. Both 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
- print has a return value of 1 so it can be used in expressions.

- echo can take multiple parameters
- print can take one argument

- echo is marginally faster than print.

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

In this example PHP is used to echo HTML markup

See **echo_example.php**

Basic PHP Syntax

A PHP script can be placed anywhere in the document.
A PHP script starts with **<?php** and ends with **?>**:

```
<?php
// PHP code goes here
?>
```

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

A PHP file normally contains HTML tags, and some PHP scripting code.

Note: PHP statements end with a semicolon (;).

Comments in PHP

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

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>

</body>
</html>
```

Variables

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Note: When you assign a text value to a variable, put quotes around the value.

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

PHP is a Loosely Typed Language

You do not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value.

Case Sensitivity

In PHP keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

```php
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
```

All are legal and equal.

However, all variable names are case-sensitive.

```php
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Only this statement will display the values of $color.

Conditional Statements

In PHP you have the following conditional statements:

- **if statement** - executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif....else statement** - executes different codes for more than two conditions
- **switch statement** - selects one of many blocks of code to be executed

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is diff
}
```

Loops

You have the following looping statements:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

```
while (condition is true) {
    code to be executed;
}
```

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

```
do {
    code to be executed;
} while (condition is true);
```

```
foreach ($array as $value) {
    code to be executed;
}
```

PHP Operators

Most should be familiar. Some are slightly different.

Arithmetic Operators

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)

Assignment Operators

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

Comparison Operators

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

Increment / Decrement Operators

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

String Operators

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

Logical Operators

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

Functions

A user defined function declaration starts with the word "function":

```
function functionName() {
    code to be executed;
}
```

A function will be executed by a call to the function.

Information can be passed to functions through arguments. Return values are optional.

PHP also allows for a Default Argument Value in a function declaration:

```
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
```

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

Scope

Variables can be declared anywhere in the script.

PHP has three different variable scopes: **local, global and static.**

Scope: local

Variables can be declared anywhere in the script.

PHP has three different variable scopes: **local, global and static.**

- **local** - A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

Scope: global

Variables can be declared anywhere in the script.

PHP has three different variable scopes: local, global and static

- **global** - A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function.

```php
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

The global keyword

The global keyword is used to access a global variable from within a function.
To do this, use the global keyword before the variables (inside the function):

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The static keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes you want a local variable NOT to be deleted.

To do this, use the **static** keyword when you first declare the variable

```php
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

Each time the function is called $x will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

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.

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

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

- ***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

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

PHP Global Variables - Superglobals

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:
$GLOBALS
$_SERVER
$_REQUEST
$_POST
$_GET
$_FILES
$_ENV
$_COOKIE
$_SESSION

We will talk more about superglobals in the next class.

Data types

PHP supports the following data types:

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

The PHP var_dump() function returns the data type and value

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

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

Data types

PHP supports the following data types:

- **String – See the reference for functions on strings** http://www.w3schools.com/php/php_ref_string.asp
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Data types

PHP supports the following data types:

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

Data types

PHP supports the following data types:

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

Data types

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- **NULL -** a variable of data type NULL is a variable that has no value assigned to it
- Resource

Data types

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- **Resource –** used to store of a reference to functions and resources external to PHP. We will speak about this when we get to database calls.

PHP objects

An object is a data type which stores data and information on how to process that data.

In PHP, an object must be explicitly declared.
First you declare a class of object, using the class keyword. A class is a structure that can contain properties and methods

```php
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

Note the syntax for accessing object variables.

PHP arrays

There are three types of arrays in PHP:

- **Indexed arrays** - Arrays with a numeric index:

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

- **Associative arrays** - Arrays with named keys:

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

- **Multidimensional arrays** - Arrays containing one or more arrays

Multidimensional arrays

A multidimensional array is an array containing one or more arrays.

PHP understands multidimensional arrays that are two, three, four, five, or more levels deep.

The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

Multidimensional arrays

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

You can store the data from this table in a two-dimensional array:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

To get access to the elements of the $cars array we must point to the two indices (row and column):

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

php.ini

The filesystem functions are part of the PHP core.

There is no installation needed to use these functions.

The behavior of the filesystem functions is affected by settings in **php.ini**. This file is used to configure your PHP installation. To find it search for php.ini within your installation of XAMP.

Where necessary these notes point out settings you may need to check in php.ini, particularly if you have problems running any of the PHP examples.

Questions, Suggestions?

Next:

PHP: Connecting to a data base.