GANPAT UNIVERSITY
U. V. PATEL COLLEGE OF ENGINEERING

2CEIT6PE1
WEB TECHNOLOGY

UNIT 3

PHP: Flow control, building blocks, Functions, Array, Objects and Strings

Prepared by: Prof. Megha Patel (Asst. Prof in C.E Dept.)

What is PHP?

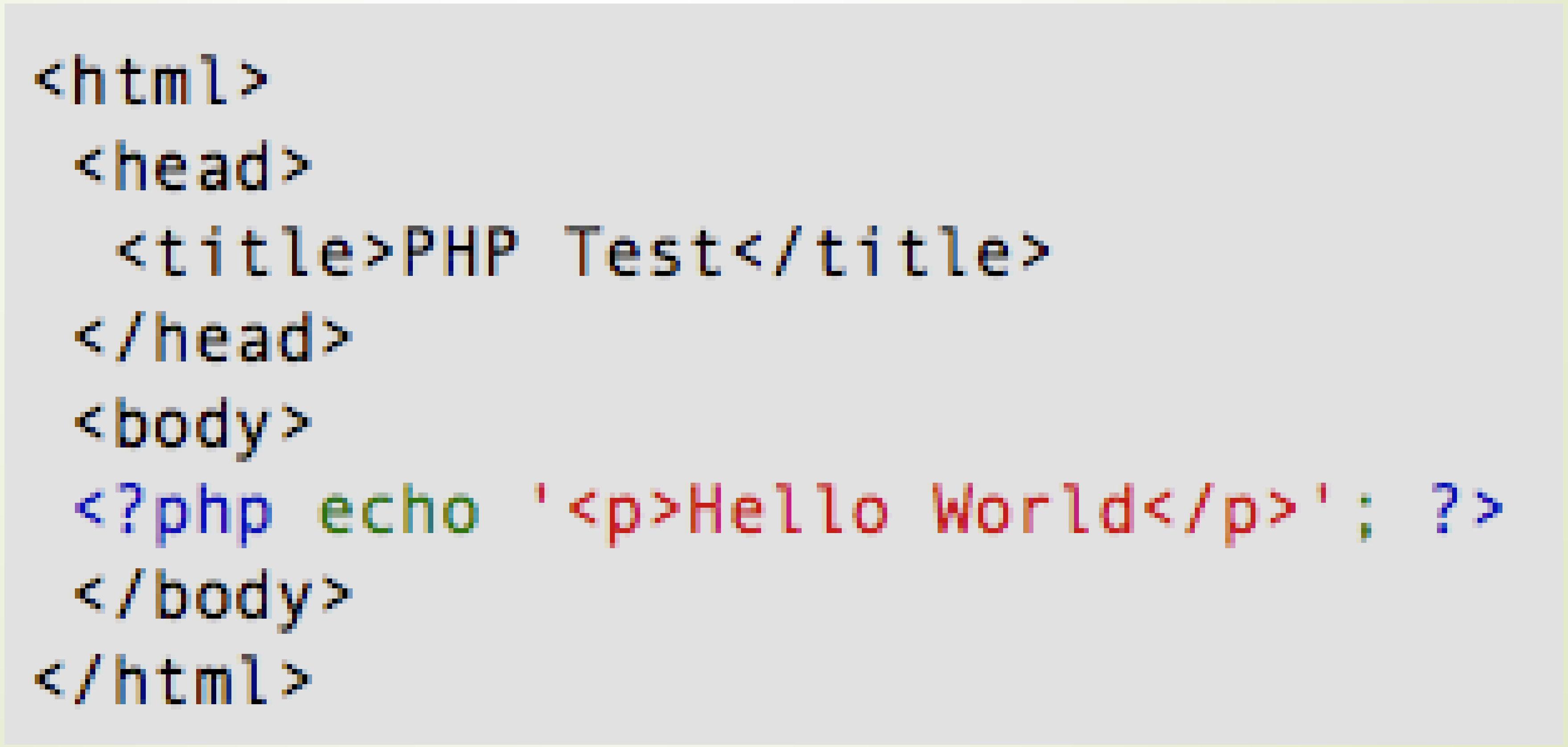
- PHP: Hypertext Preprocessor -- It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.
- PHP files are executed on the web server.
- Therefore we cannot save them anywhere and view them, as with HTML files.
- Must save .php files in subdirectory of web server.

  For Example: htdocs for XAMPP server

  www for WAMP server
- Make call to web server by using local host.

PHP Introduction

- The PHP code is enclosed in <?php and ?> tag.



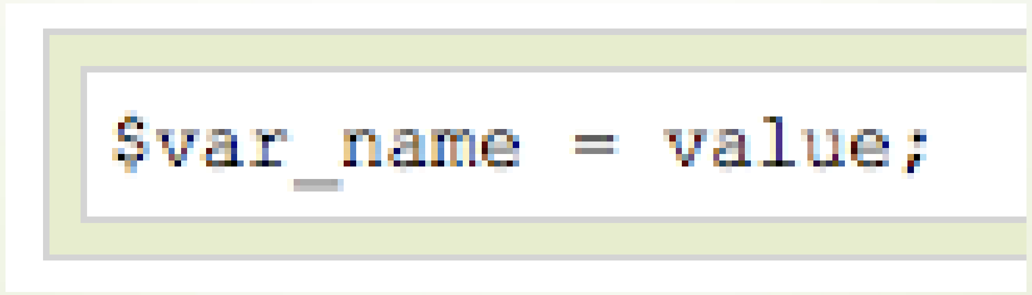
```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

```
<html>
<head>
<title> PHP Introduction </title>
</head>
<body>
This is HTML! <br />
<?php
echo 'This is PHP! <br />'; // prints to screen
/*
Here's a longer
comment
that spans multiple
lines.
*/
?>
</body>
</html>
```

- PHP tags: <?php and ?>
- The echo command
- Single line comment (//)
- Multiple line comment (/* and */)

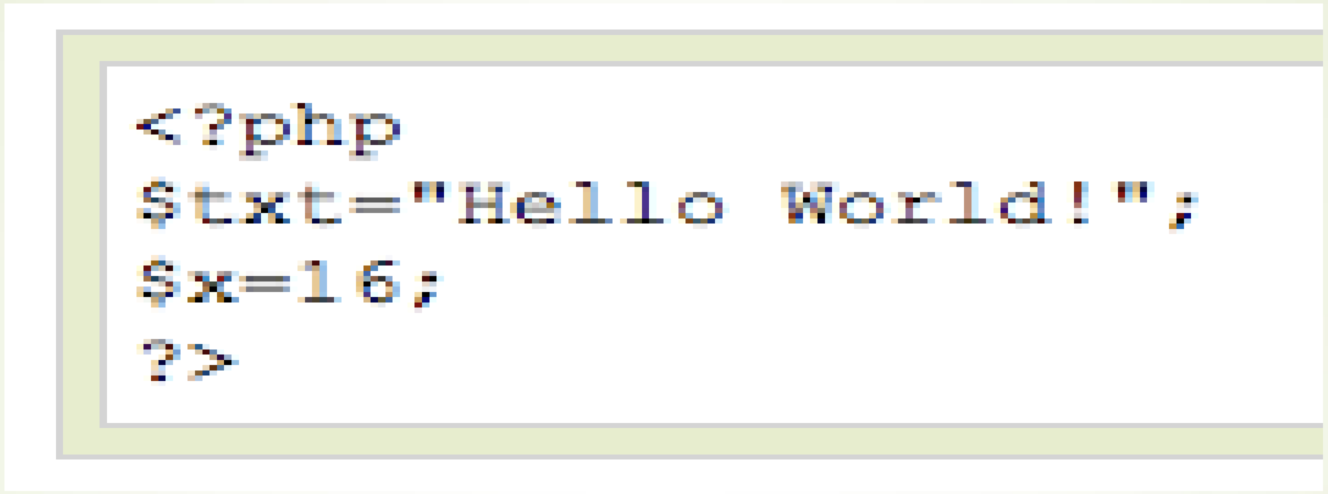
PHP Variables

- Variables are used for storing values, like text strings, numbers or arrays.
- When a variable is declared, it can be used over and over again in your script.
- All variables in PHP start with a $ sign symbol.
- The correct way of declaring a variable in PHP:



```
$var_name = value;
```

PHP Variables



```
<?php
$txt="Hello World!";
$x=16;
?>
```

In PHP, a variable does not need to be declared before adding a value to it.

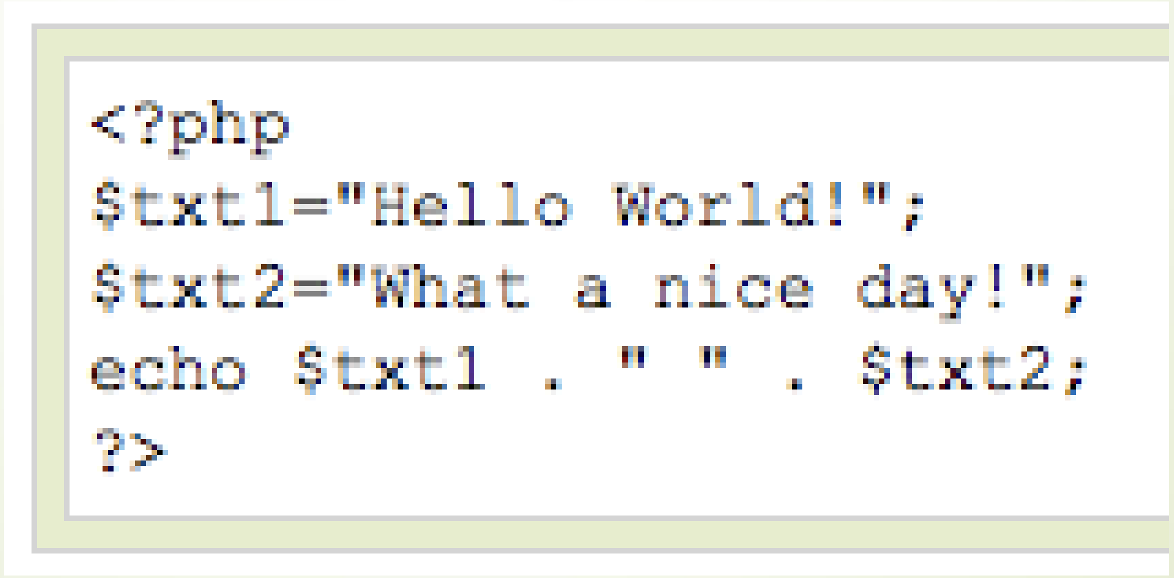
- In the example above, you see that you do not have to tell PHP that the variable is of which datatype.
- PHP automatically converts the variable to the correct data type, depending on its value.

PHP Variables

- A variable name must start with a letter or an underscore "_" -- not a number
- A variable name can only contain alpha-numeric characters, underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString).
- Variable names are case sensitive ($age and $AGE are different variables)

PHP Concatenation

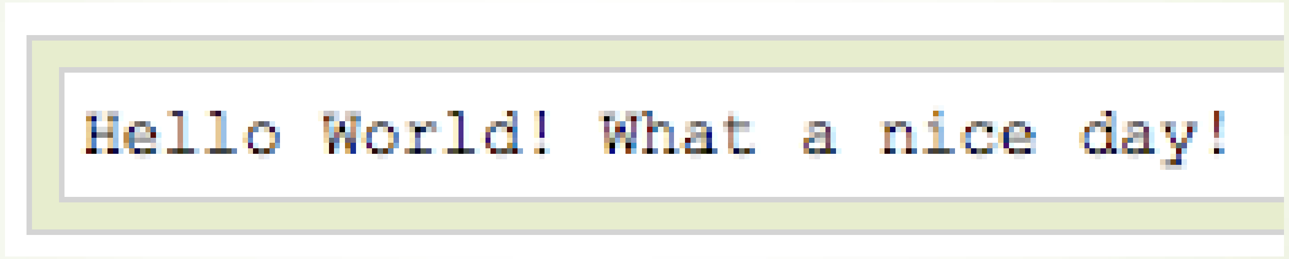
- The concatenation operator (.) is used to put two string values together.
- To concatenate two string variables together, use the concatenation operator:



```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

PHP Concatenation

- The output of the code on the last slide will be:



```
Hello World! What a nice day!
```

- If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

PHP Data Types

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

```
<!DOCTYPE html>
<html>
<body>
<?php
$a = 32;
echo var_dump($a) . "<br>";

$b = "Hello world!";
echo var_dump($b) . "<br>";
```

PHP Data Types

```
$c = 32.5;
echo var_dump($c) . "<br>";

$d = array("red", "green", "blue");
echo var_dump($d) . "<br>";

// Dump two variables
echo var_dump($a, $b) . "<br>";
?>
</body>
</html>
```

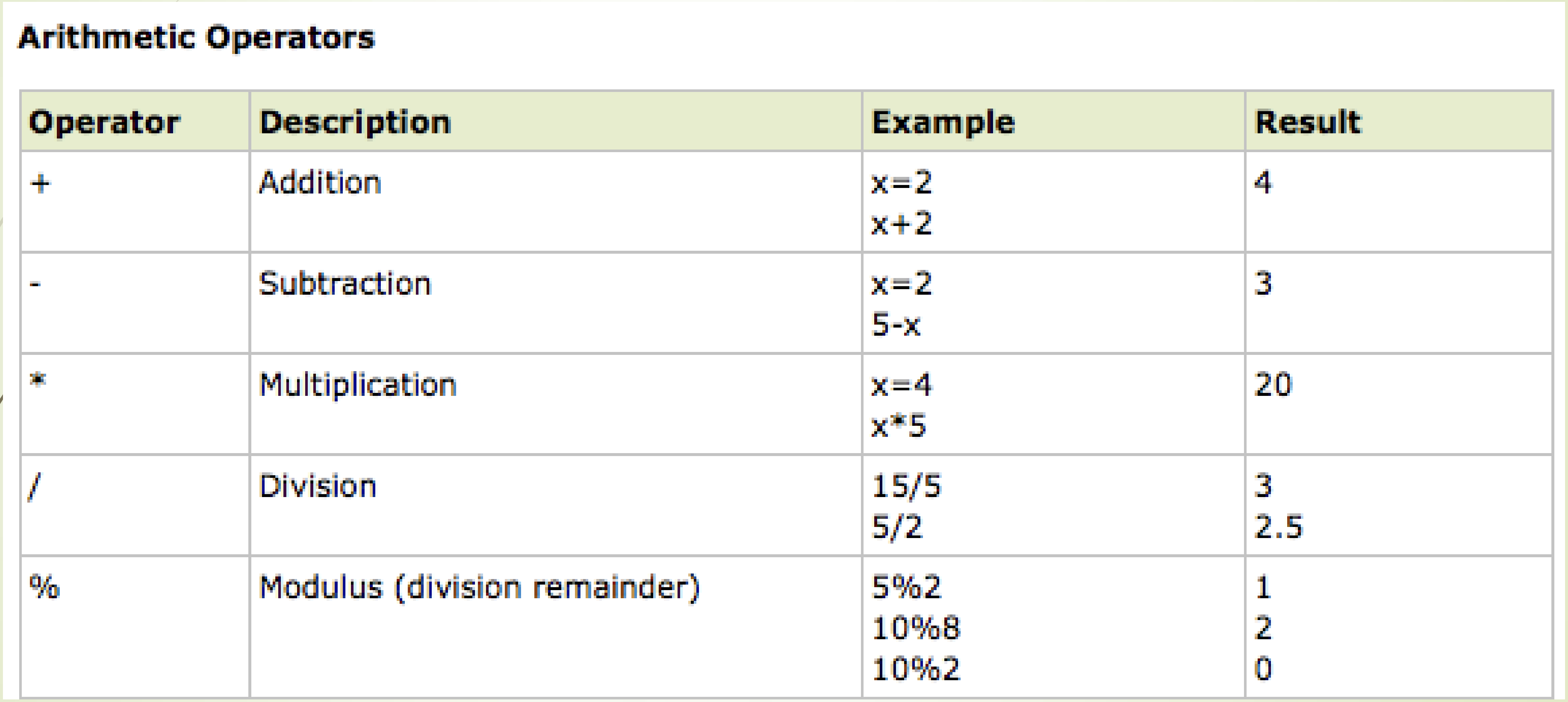
The output of above program:

```
int(32)
string(12) "Hello world!"
float(32.5)

array(3) { [0]=> string(3) "red"
[1]=> string(5) "green" [2]=>
string(4) "blue" }

int(32) string(12) "Hello world!"
```

PHP Operators

- There are five classifications of operators:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators

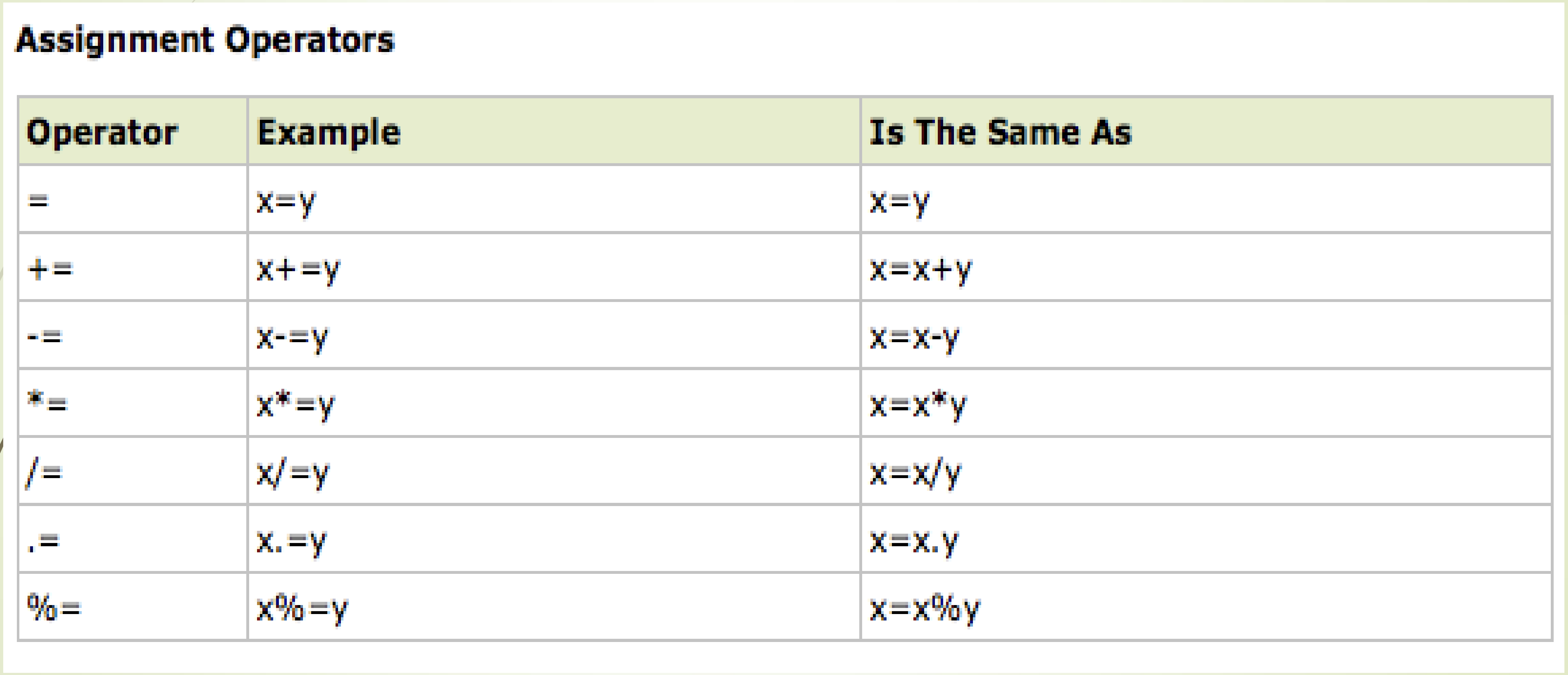
PHP Operators



Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0

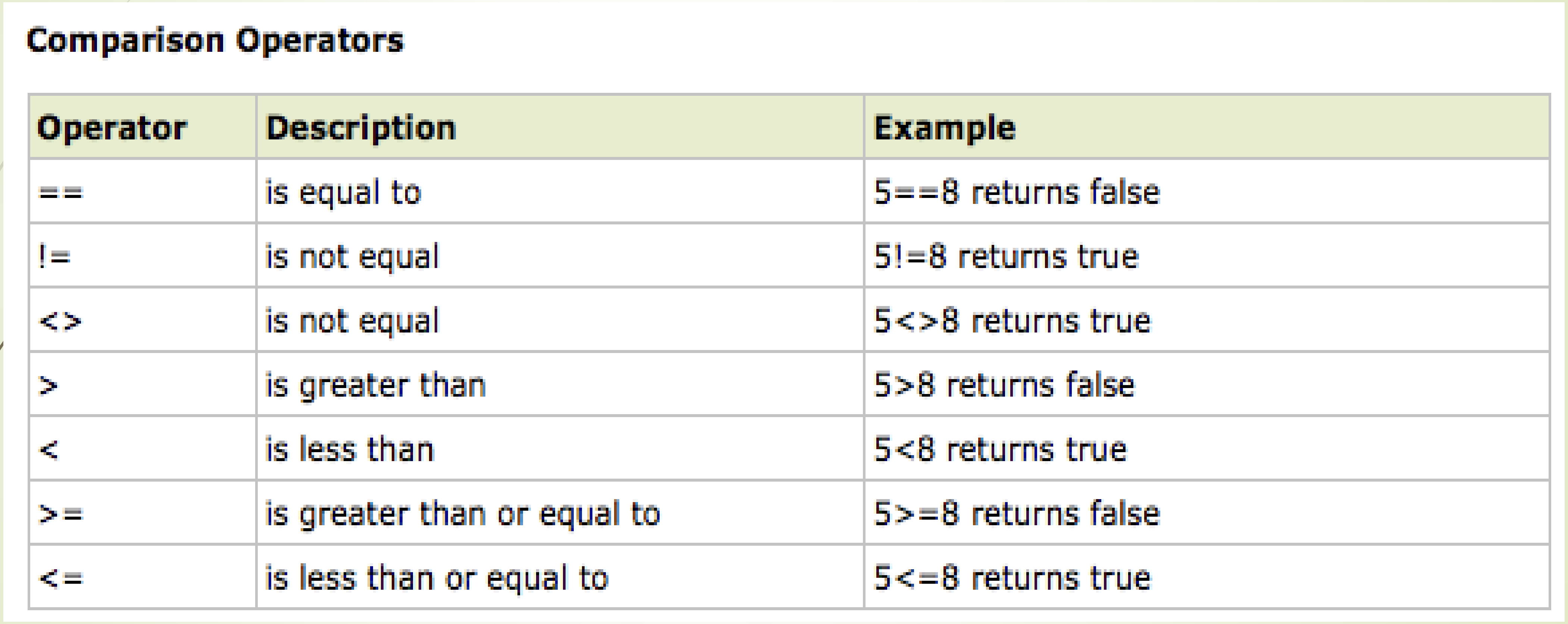
PHP Operators



Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

PHP Operators



Comparison Operators

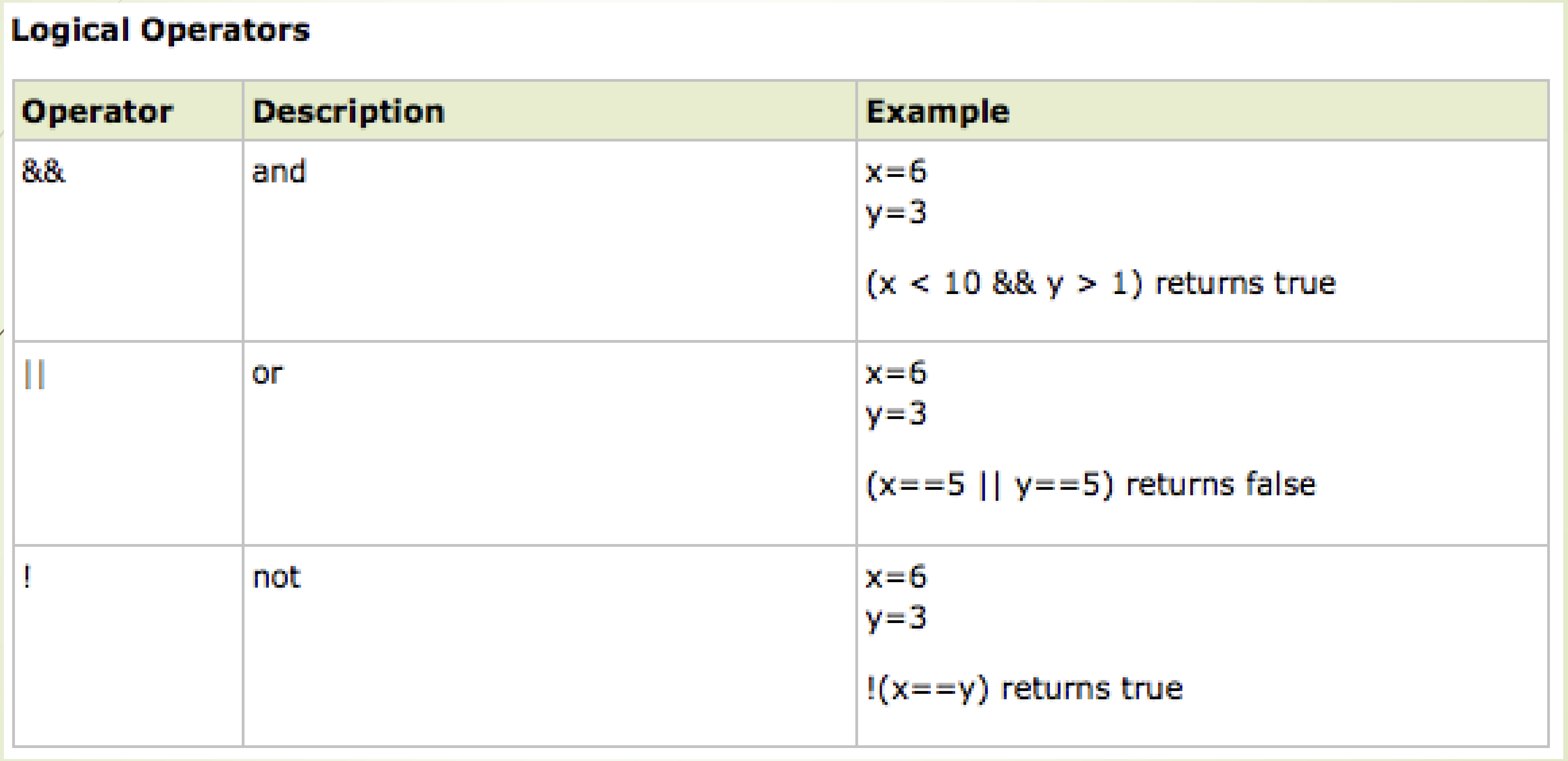
Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

PHP Operators

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

PHP Operators



Logical Operators

Operator	Description	Example

&&	and	x=6
		y=3
		
		(x < 10 && y > 1) returns true

||	or	x=6
		y=3
		
		(x==5 || y==5) returns false

!	not	x=6
		y=3
		
		!(x==y) returns true

PHP Constants

- To create s constant, define() function is used.
- **Syntax:**

  define(name, value, case-insensitive)

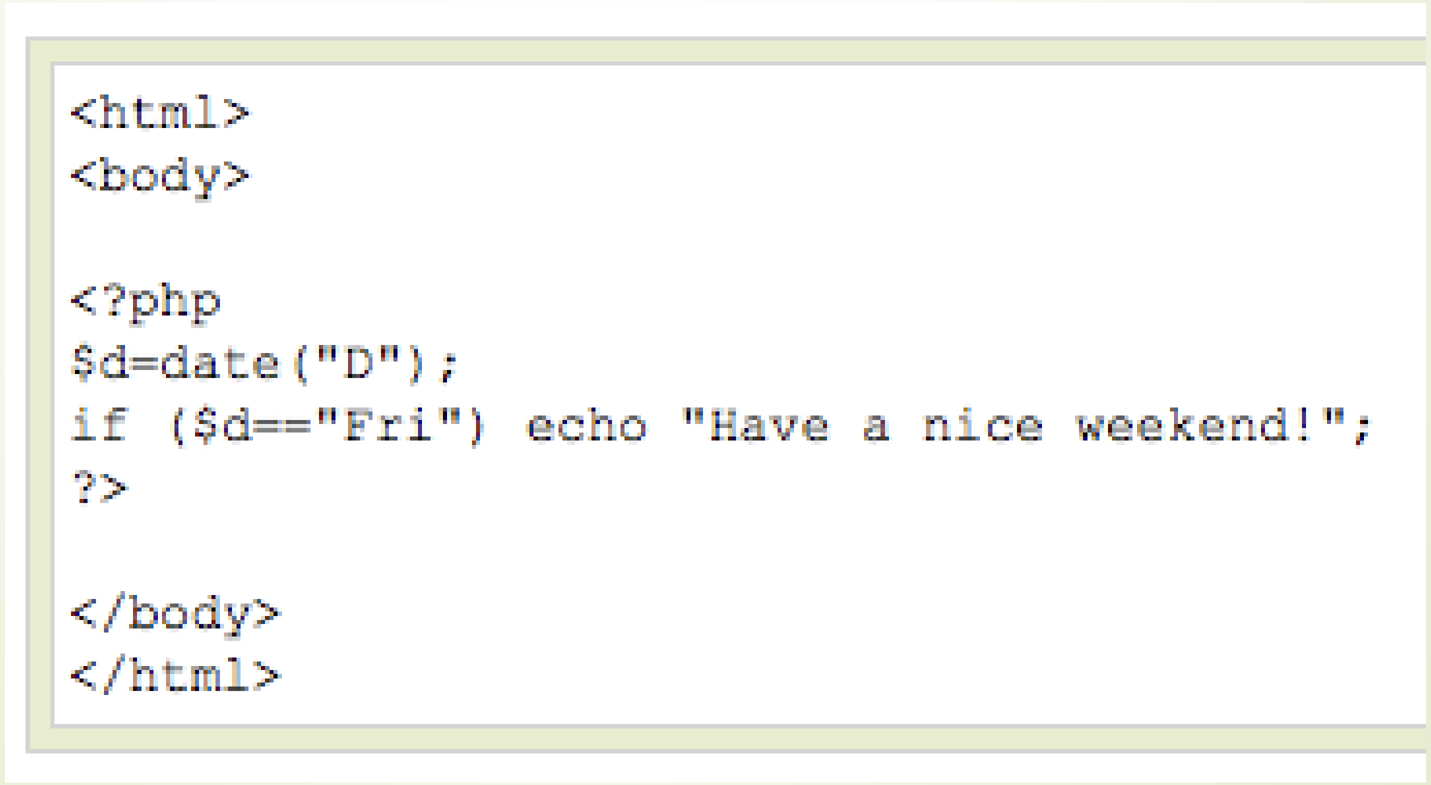
- **Parameters:**
  - name: Specifies the name of the constant
  - value: Specifies the value of the constant
  - case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

PHP Conditional Statements

- if statement - use this statement to execute some code only if a specified condition is true
- if...else statement - use this statement to execute some code if a condition is true and another code if the condition is false
- if...elseif....else statement - use this statement to select one of several blocks of code to be executed
- switch statement - use this statement to select one of many blocks of code to be executed

PHP Conditional Statements

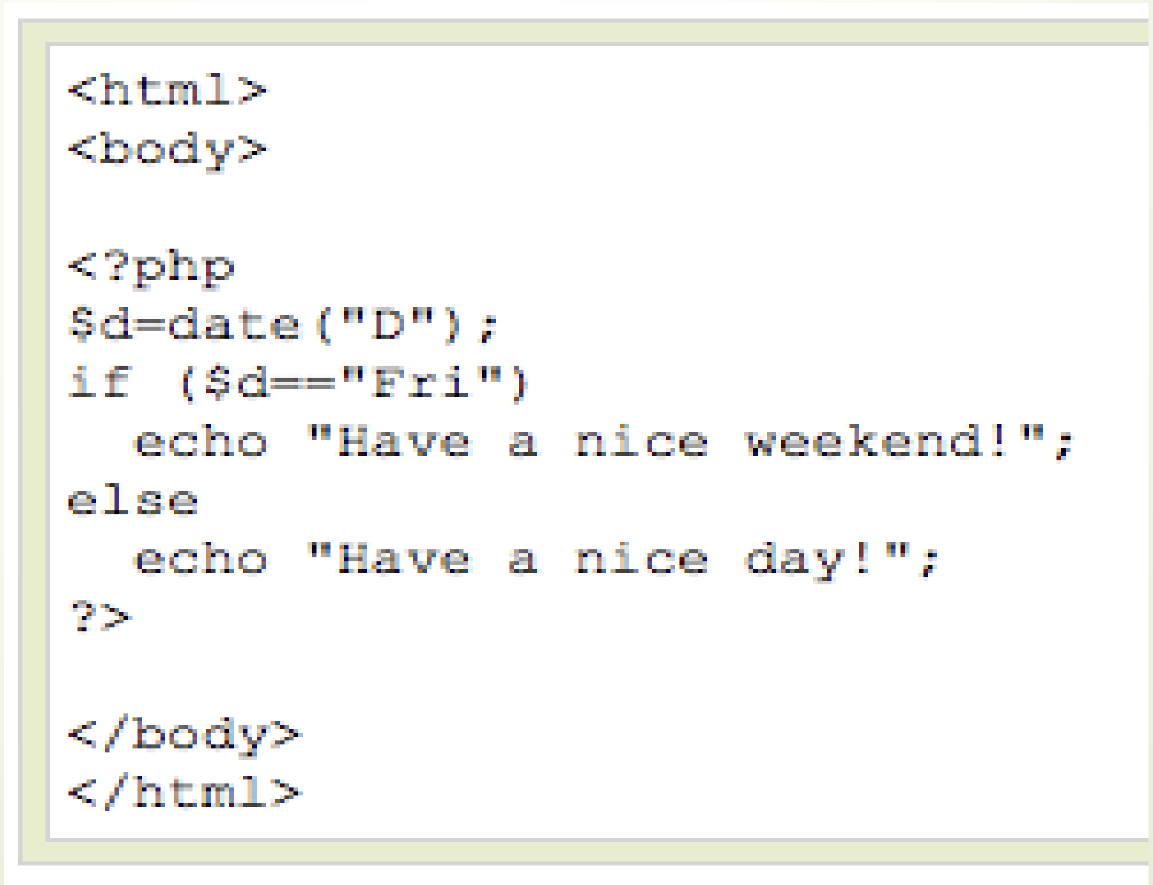
- The following example will output "Have a nice weekend!" if the current day is Friday:



```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

PHP Conditional Statements

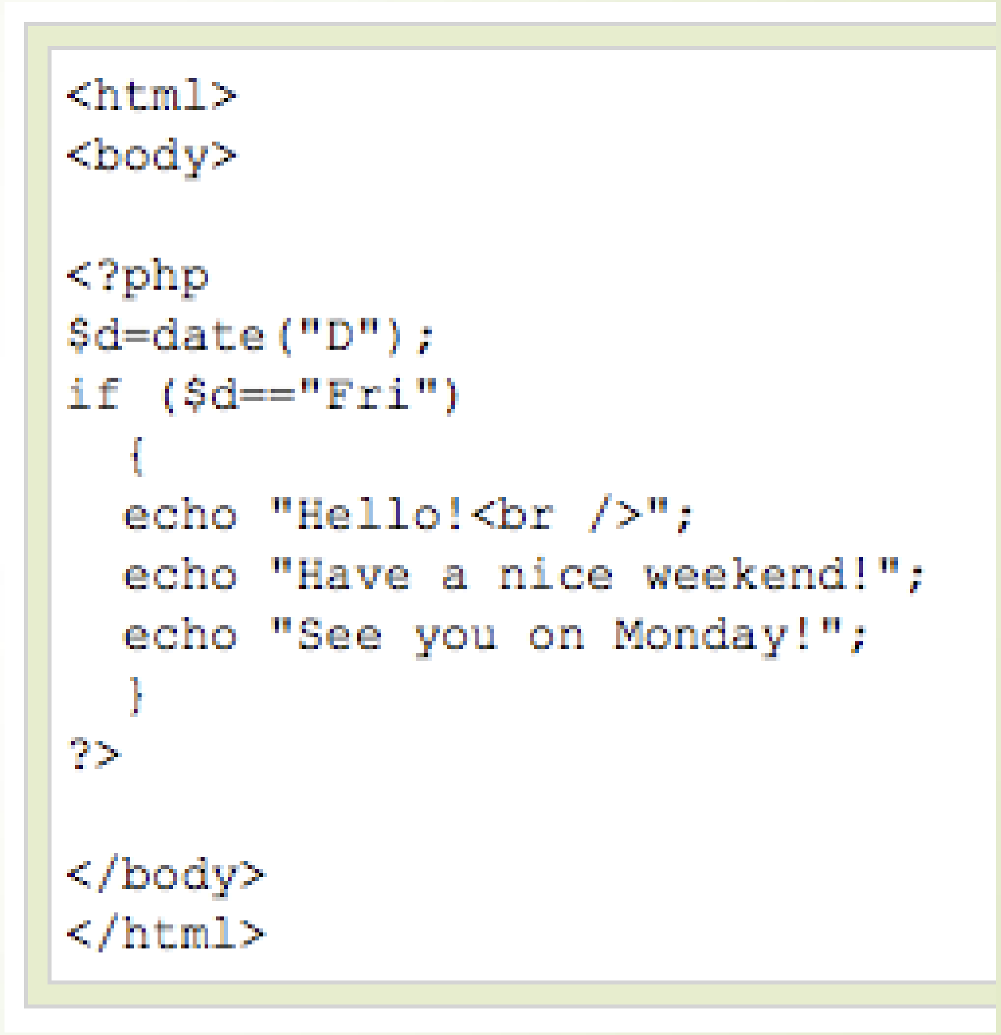
- Use the if....else statement to execute some code if a condition is true and another code if a condition is false.



```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Conditional Statements

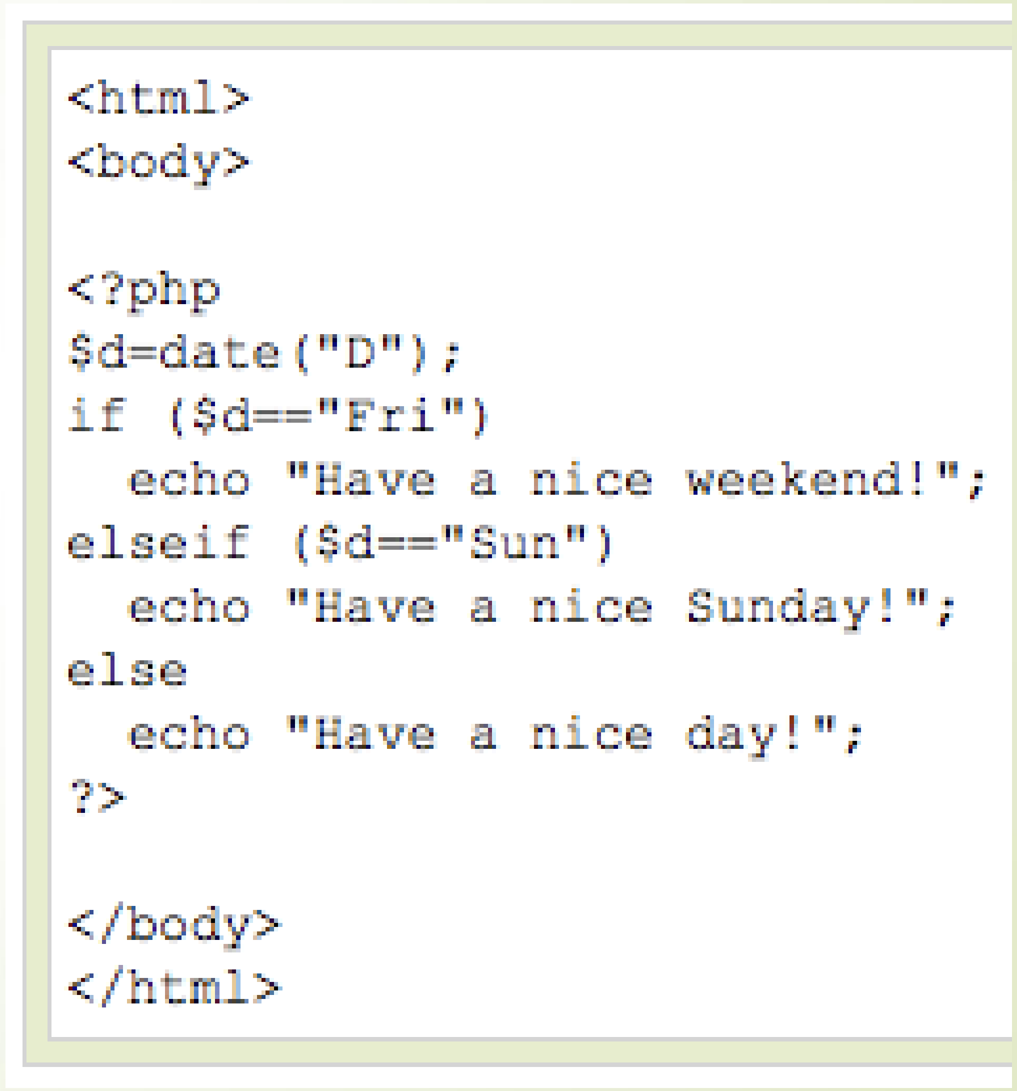
- If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces { }



```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

PHP Conditional Statements

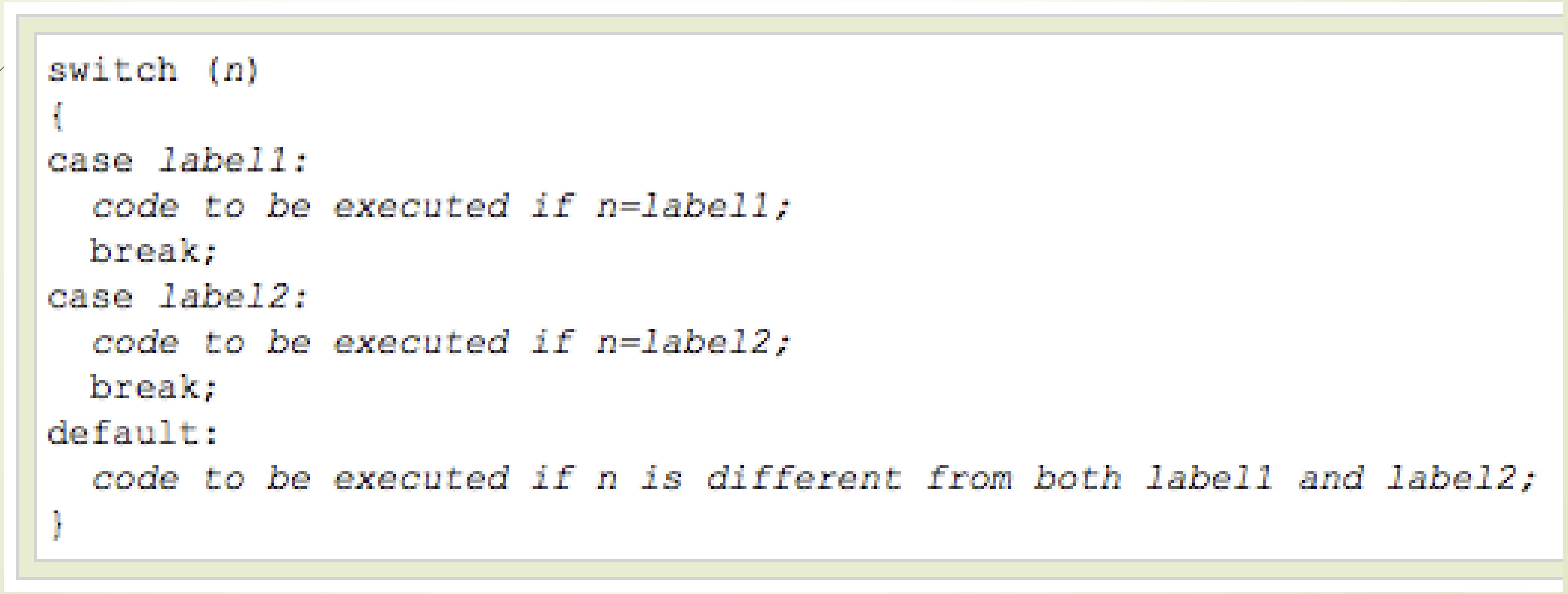
- The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":



```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Conditional Statements

- Use the switch statement to select one of many blocks of code to be executed.

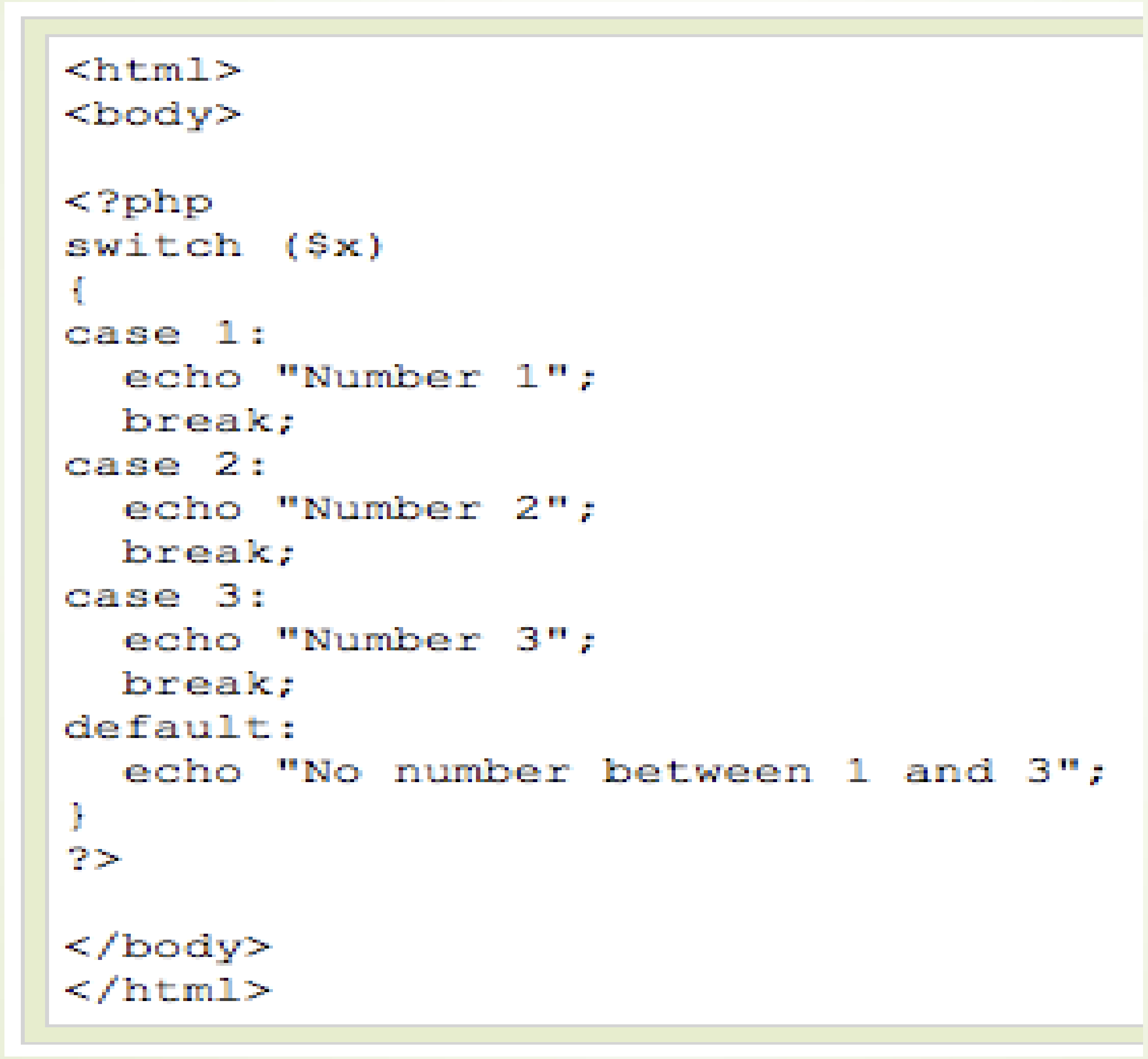


```
switch (n)
{
case label1:
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

PHP Conditional Statements

- For switches, first we have a single expression n (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

PHP Conditional Statements

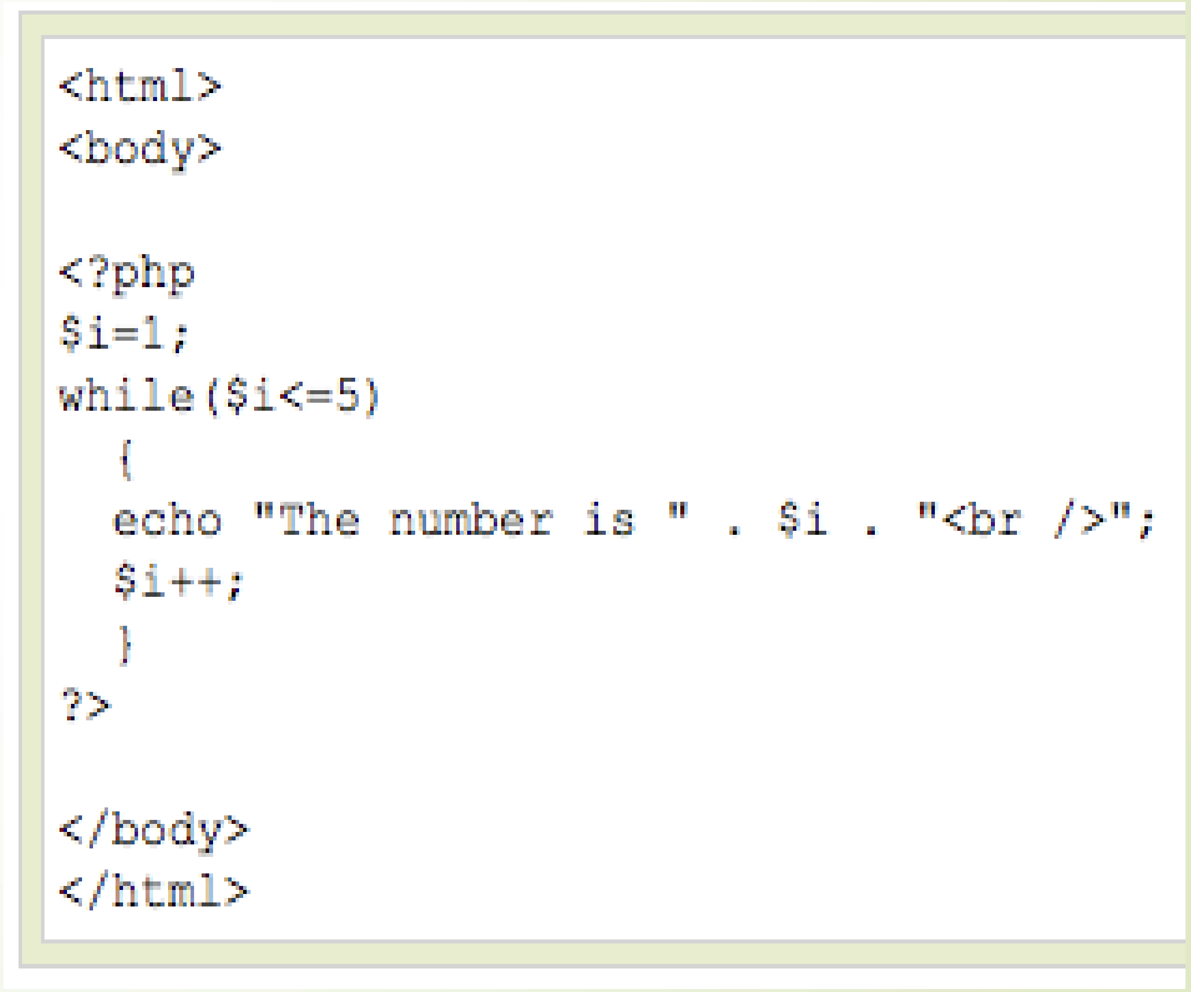

```
<html>
<body>

<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP Loops

- Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.
- In PHP, we have the following looping statements:
- **while** - loops through a block of code while a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- **for** - loops through a block of code a specified number of times

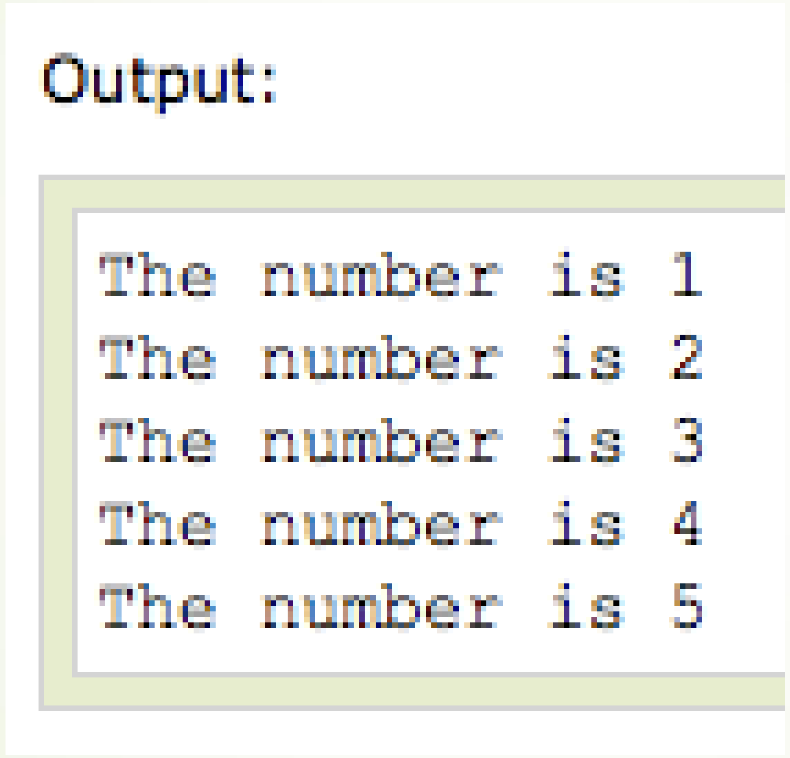
PHP Loops - While

- The while loop executes a block of code while a condition is true. The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:



```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

PHP Loops - While



Output:

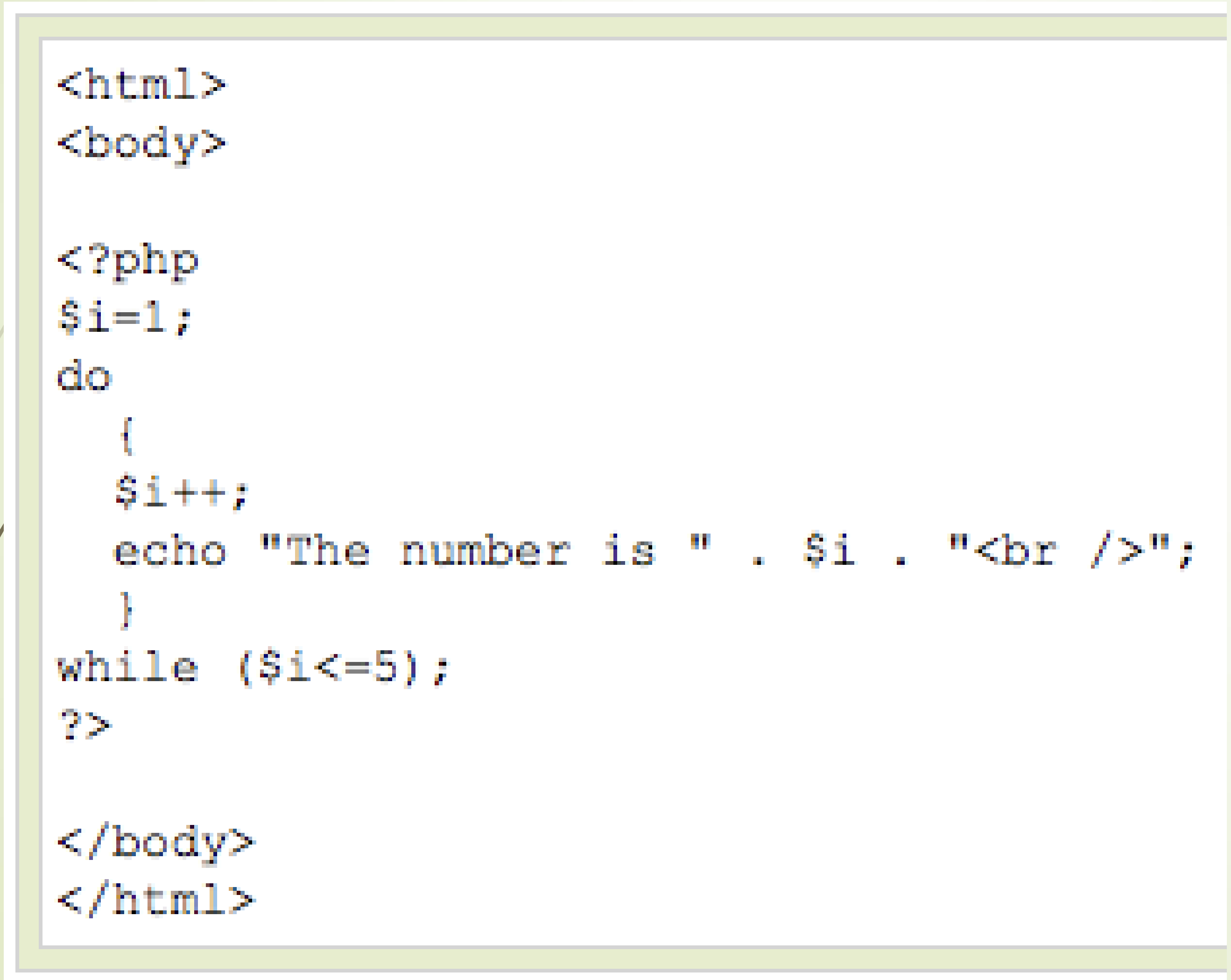
```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops – Do ... While

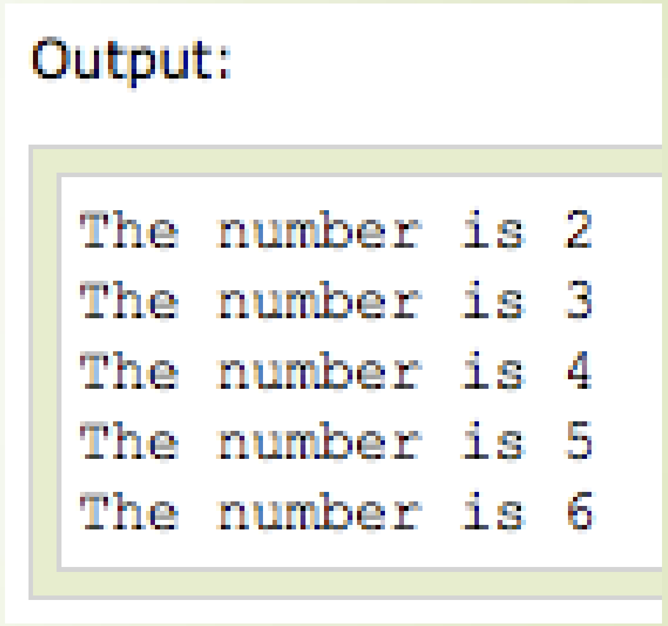
- The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

- The next example defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

PHP Loops – Do … While



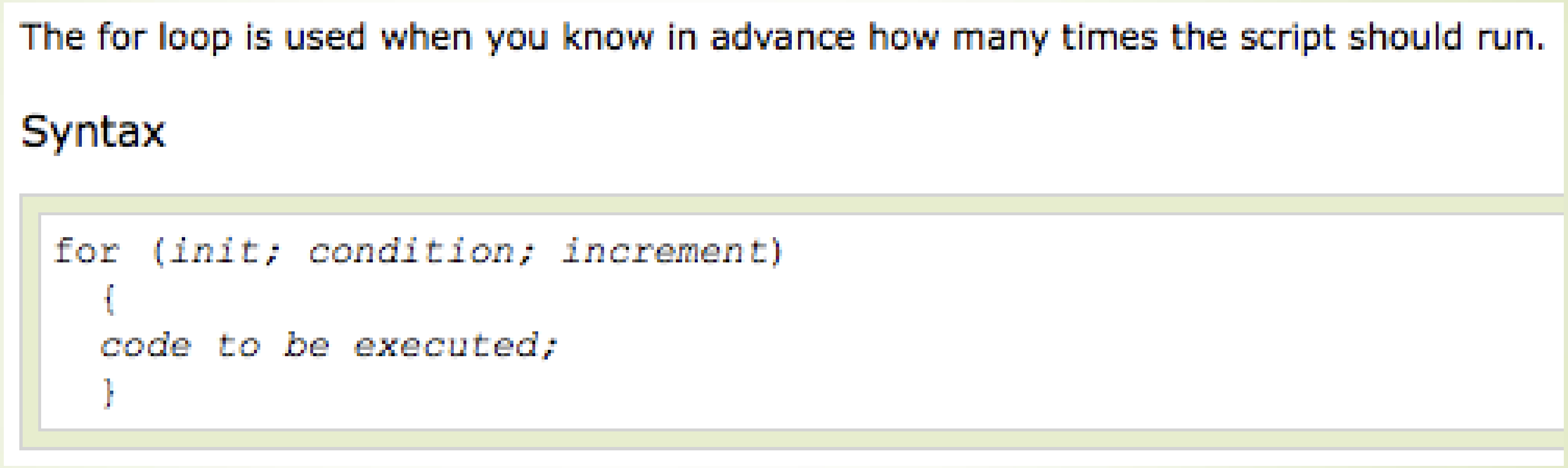
```
<html>
<body>

<?php
$i=1;
do
   {
   $i++;
   echo "The number is " . $i . "<br />";
   }
while ($i<=5);
?>

</body>
</html>
```



Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

PHP Loops - For


The for loop is used when you know in advance how many times the script should run.

Syntax

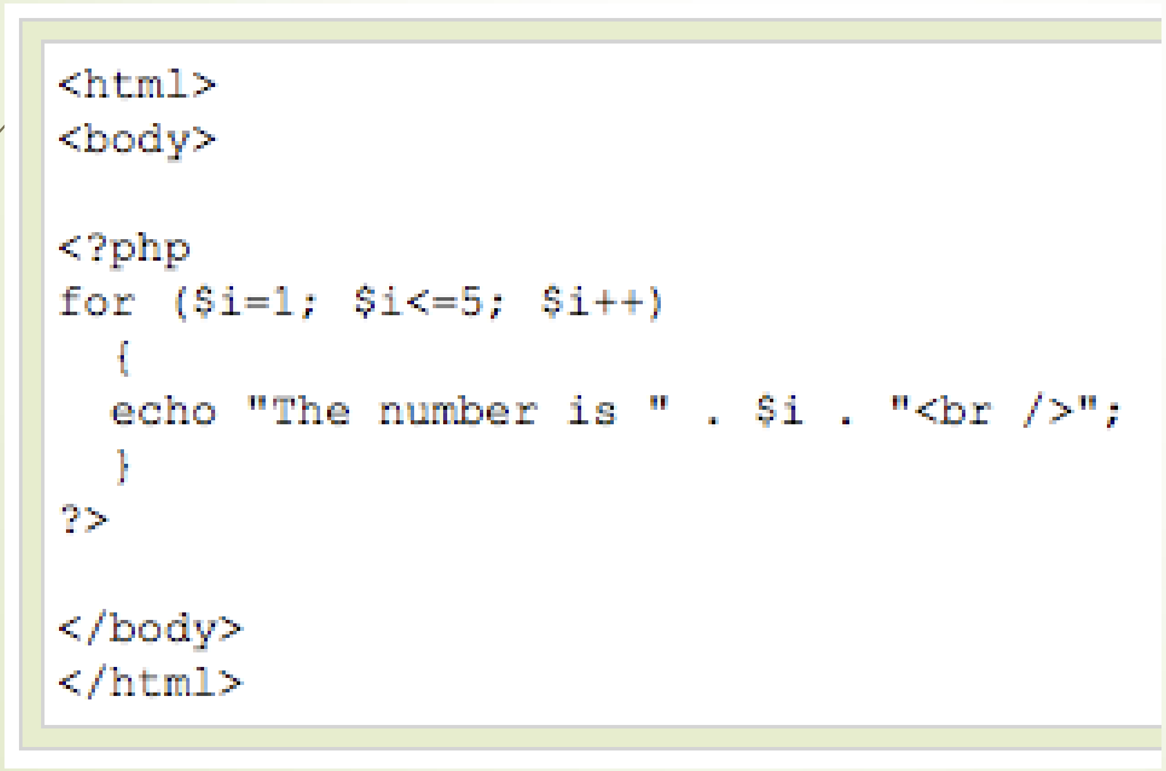
```
for (init; condition; increment)
  {
  code to be executed;
  }
```

PHP Loops - For

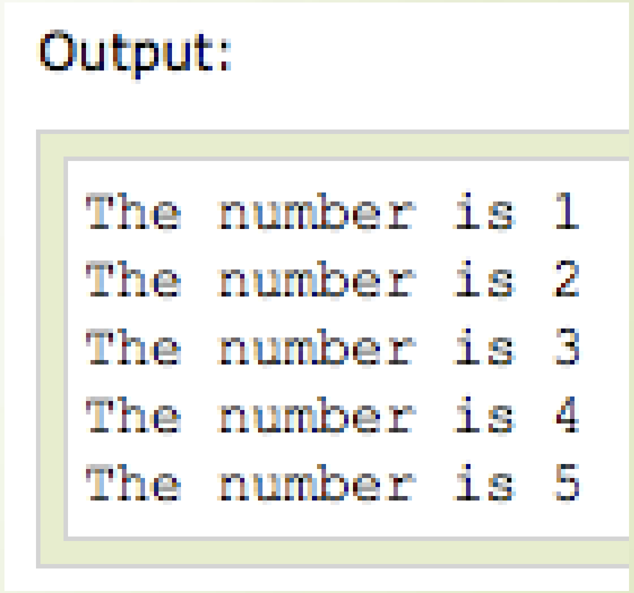
- **Parameters:**
  - **init:** Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
  - **condition:** Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
  - **increment:** Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

PHP Loops - For

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:



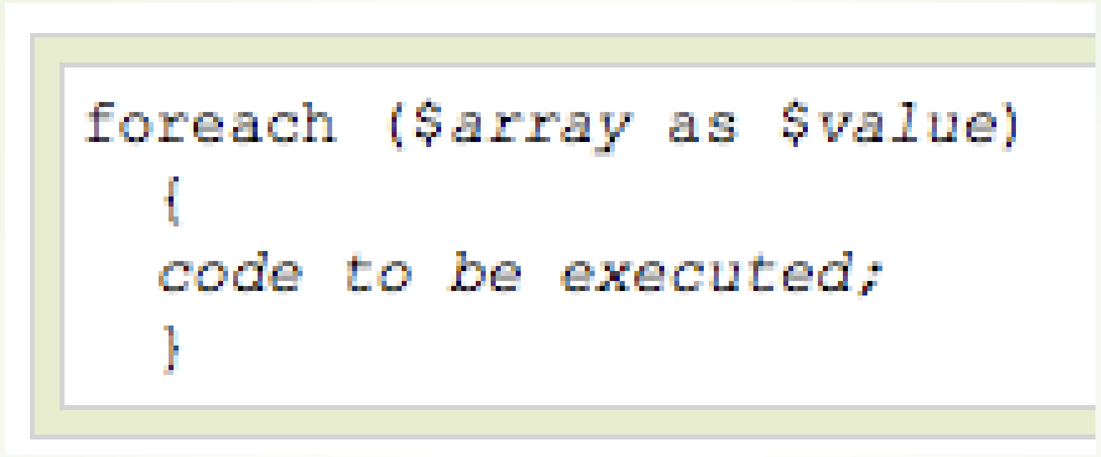
```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```



Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops - Foreach

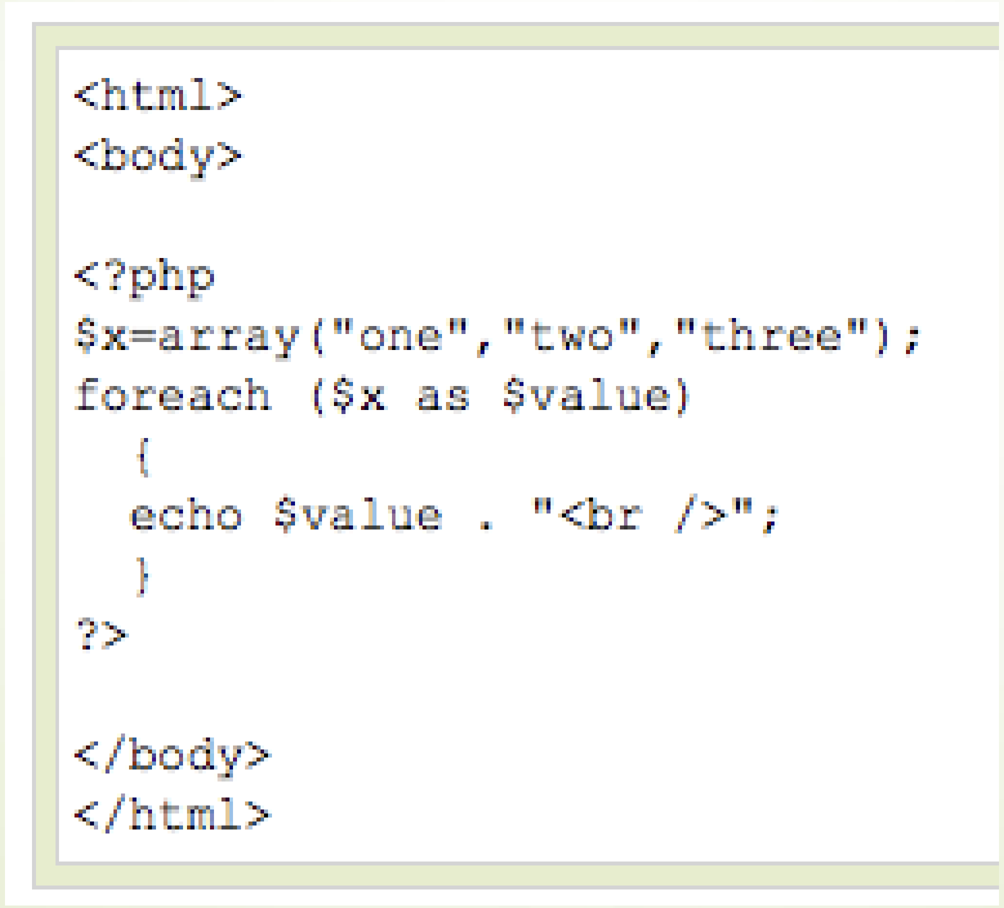


```
foreach ($array as $value)
   {
   code to be executed;
   }
```

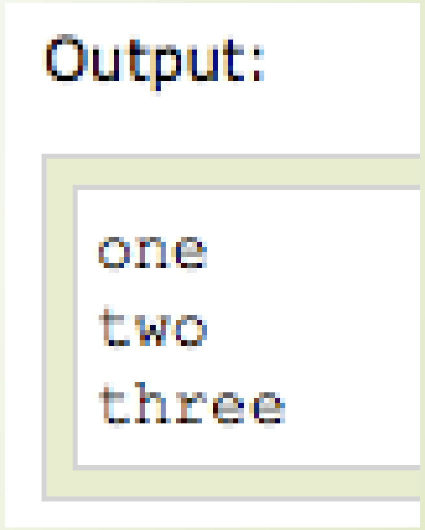
- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

PHP Loops - Foreach

- The following example demonstrates a loop that will print the values of the given array:



```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```



Output:

```
one
two
three
```

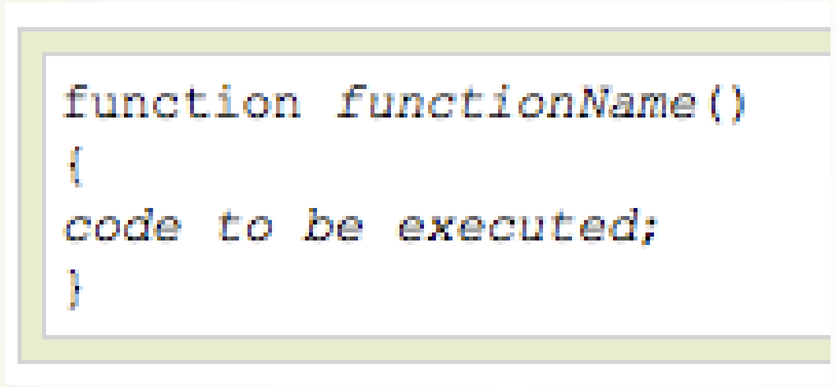
PHP Functions

- PHP has over 1000 **built-in functions** that can be called directly, from within a script, to perform a specific task.

- Besides the built-in PHP functions, it is possible to create your own functions which are called as **User Defined Functions**.
  - A function is a block of statements that can be used repeatedly in a program.
  - A function will not execute automatically when a page loads.
  - A function will be executed by a call to the function.

User Defined Function in PHP

- **Syntax**



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

- Give the function a name that reflects what the function does.
- The function name can start with a letter or underscore (not a number).

User Defined Function in PHP

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

<?php
function writeMsg() {
  echo "Hello world!";
}

writeMsg();
?>

</body>
</html>
```

Output:

```
Hello world!
```

PHP Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.

- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

Global Scope

- A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<!DOCTYPE html>
<html>
<body>
<?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>";
?>
</body>
</html>
```

Output:

```
Variable x inside function is:
Variable x outside function is: 5
```

Local Scope

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

```
<!DOCTYPE html>
<html>
<body>
<?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>";
?>

</body>
</html>
```

Output:

```
Variable x inside function is: 5
Variable x outside function is:
```

PHP 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):

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

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

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

myTest();
echo $y;
?>

</body>
</html>
```

Output: 15

PHP The static Keyword

- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
- To do this, use the static keyword when you first declare the variable:

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

<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
</body>
</html>
```

Output:

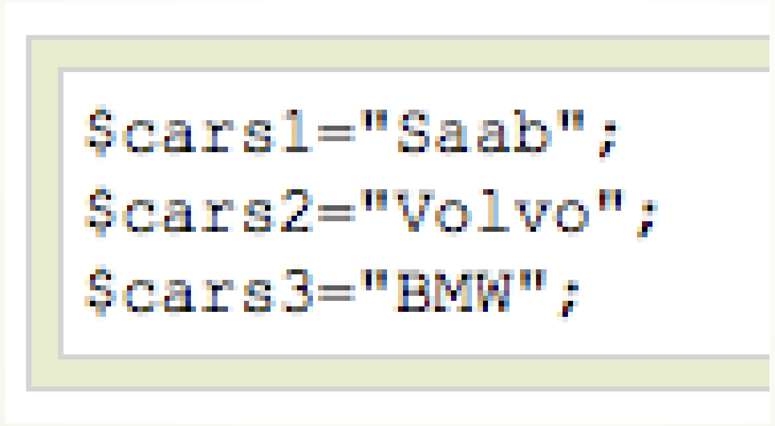
```
0
1
2
```

PHP Arrays

- An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.
- An array is a special variable, which can store multiple values in one single variable.

PHP Arrays

- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:



```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

PHP Arrays

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- The best solution here is to use an array.
- An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
- Each element in the array has its own index so that it can be easily accessed.

PHP Arrays

- In PHP, there are three kind of arrays:
  - **Numeric array** - An array with a numeric index
  - **Associative array** - An array where each ID key is associated with a value
  - **Multidimensional array** - An array containing one or more arrays

PHP Numeric Arrays

- A numeric array stores each array element with a numeric index.
- There are two methods to create a numeric array.
- In the following example the index is automatically assigned (the index starts at 0):

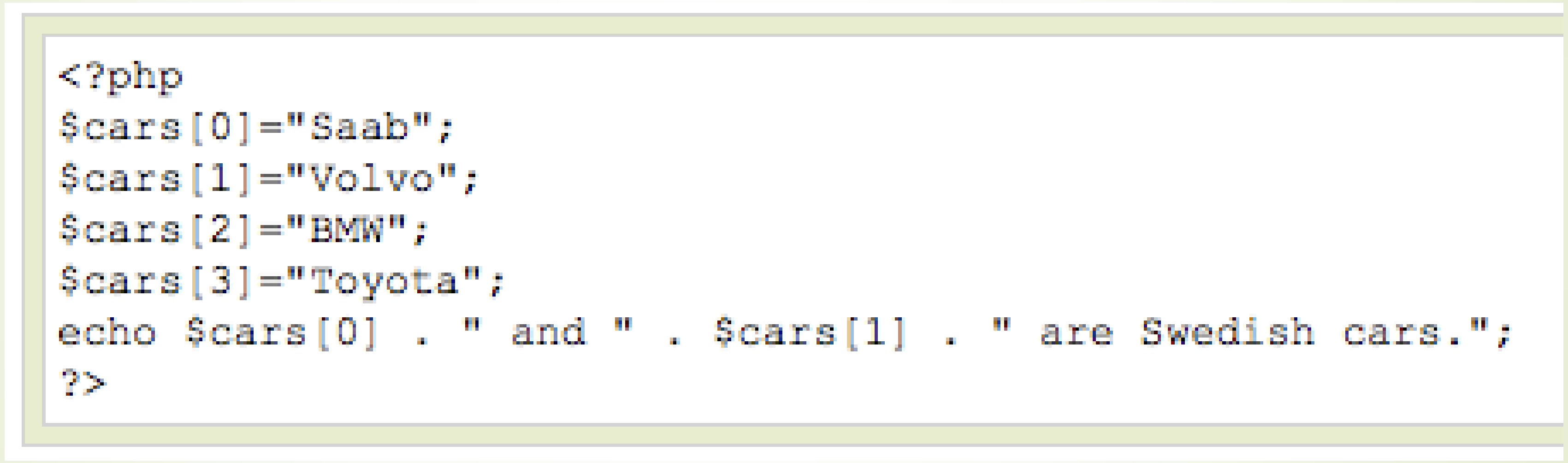
```
$cars=array("Saab","Volvo","BMW","Toyota");
```

In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

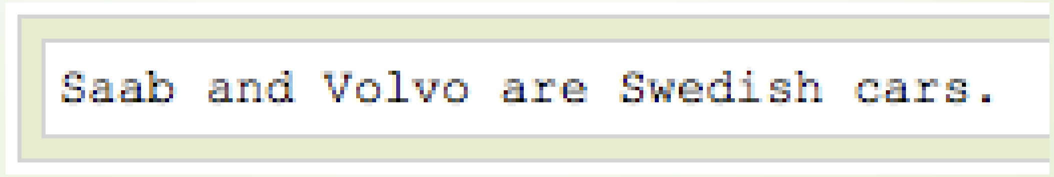
PHP Numeric Arrays

- In the following example you access the variable values by referring to the array name and index:



```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

- The code above will output:



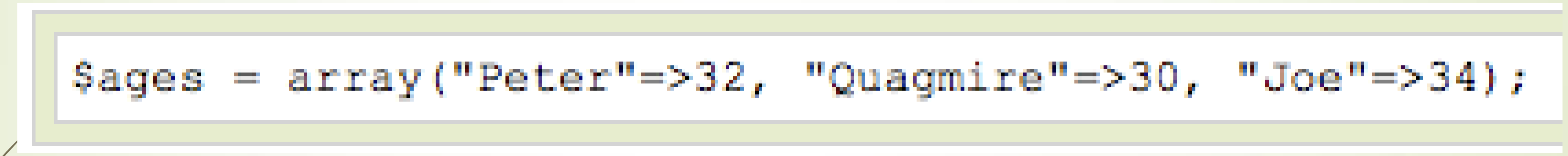
```
Saab and Volvo are Swedish cars.
```

PHP Associative Arrays

- With an associative array, each ID key is associated with a value.
- When storing data about specific named values, a numerical array is not always the best way to do it.
- With associative arrays we can use the values as keys and assign values to them.

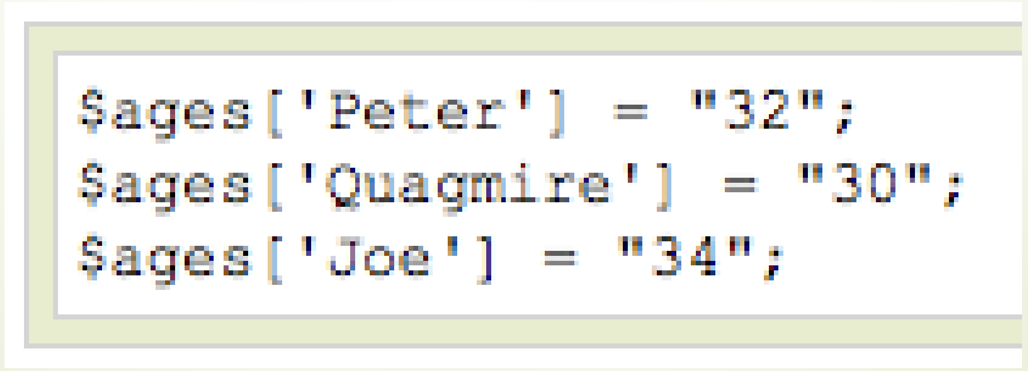
PHP Associative Arrays

- In this example we use an array to assign ages to the different persons:



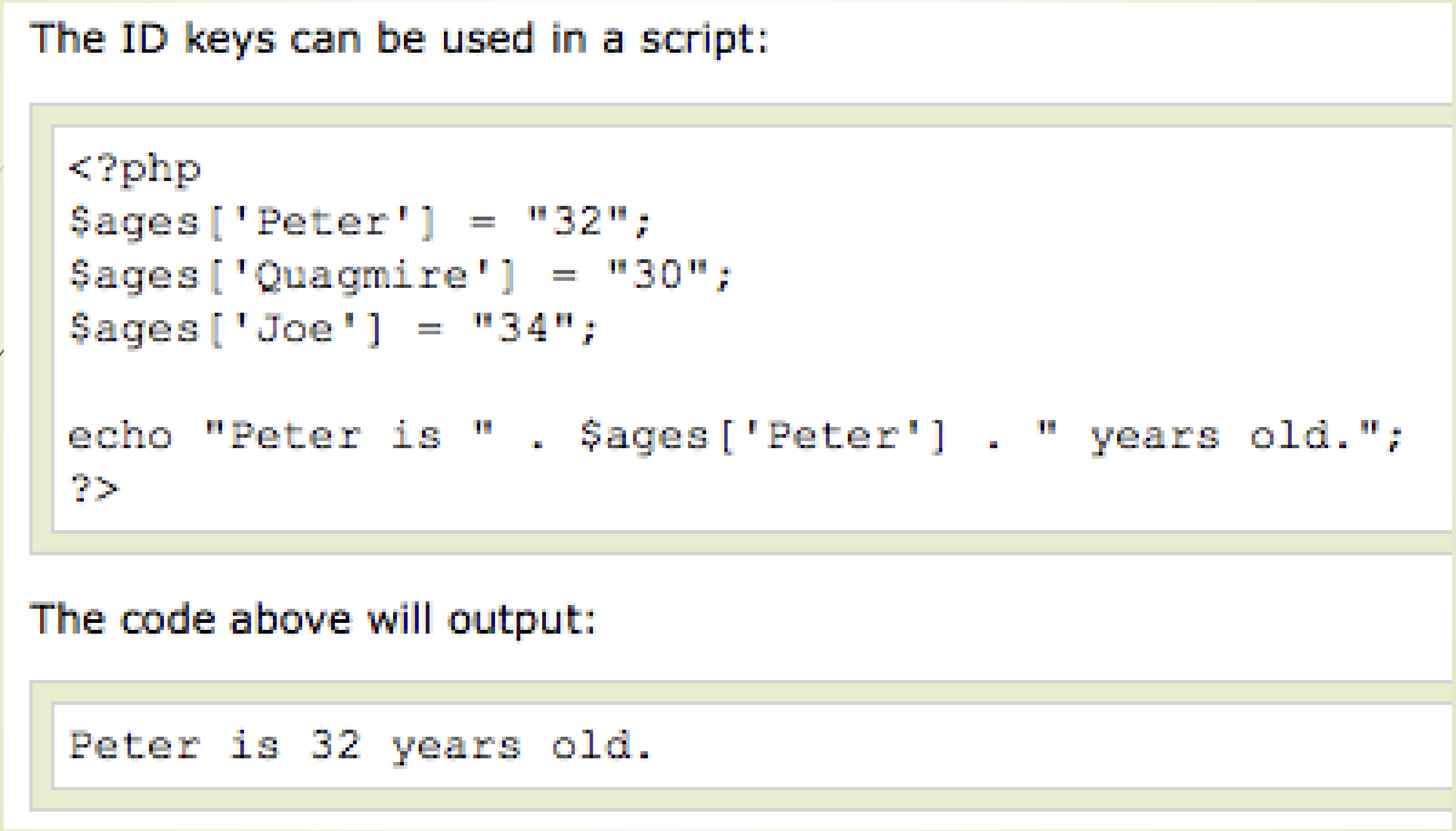
```php
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- This example is the same as the one above, but shows a different way of creating the array:



```php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

PHP Associative Arrays



The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

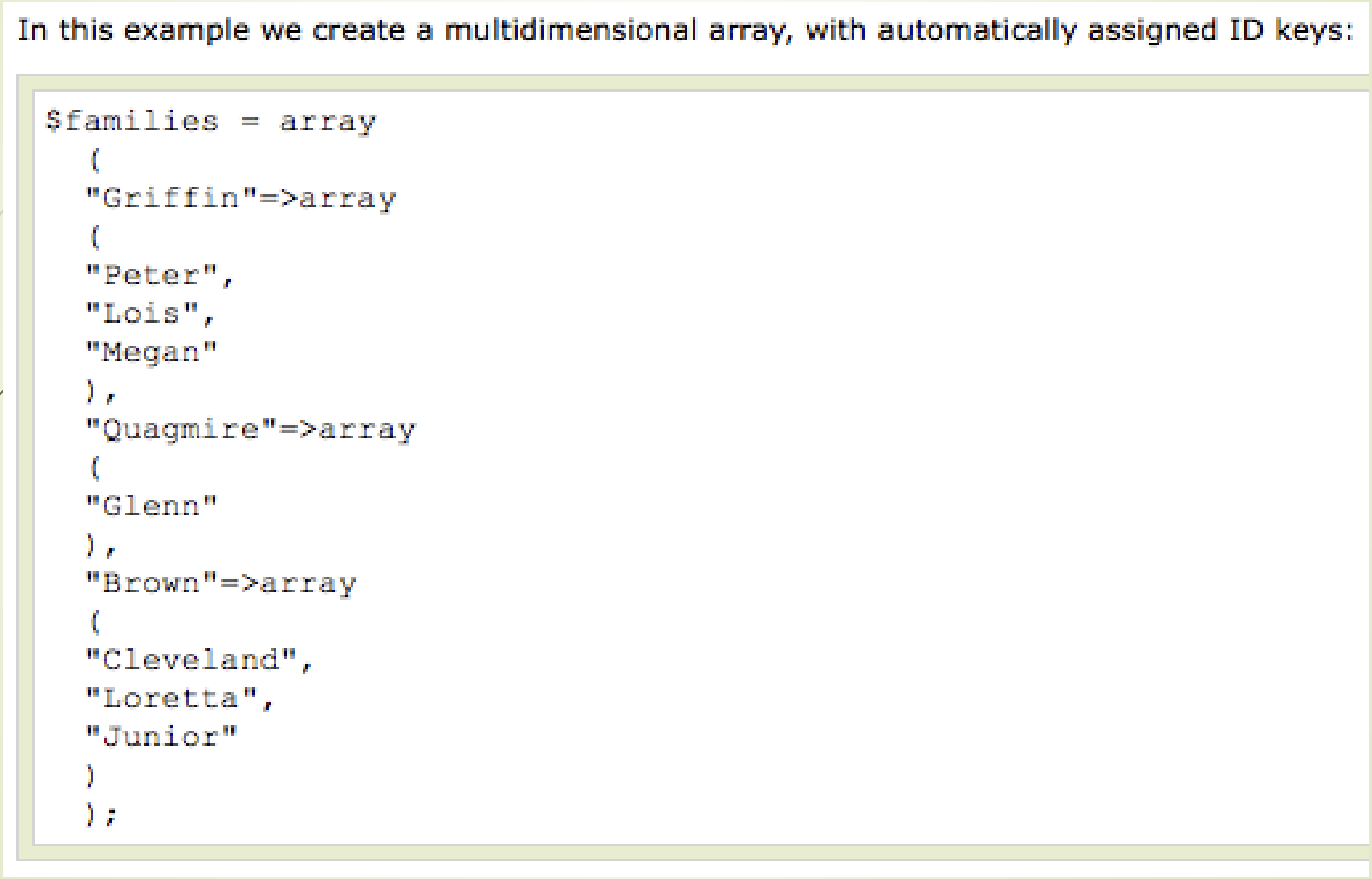
The code above will output:

```
Peter is 32 years old.
```

PHP Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.
- And each element in the sub-array can be an array, and so on.

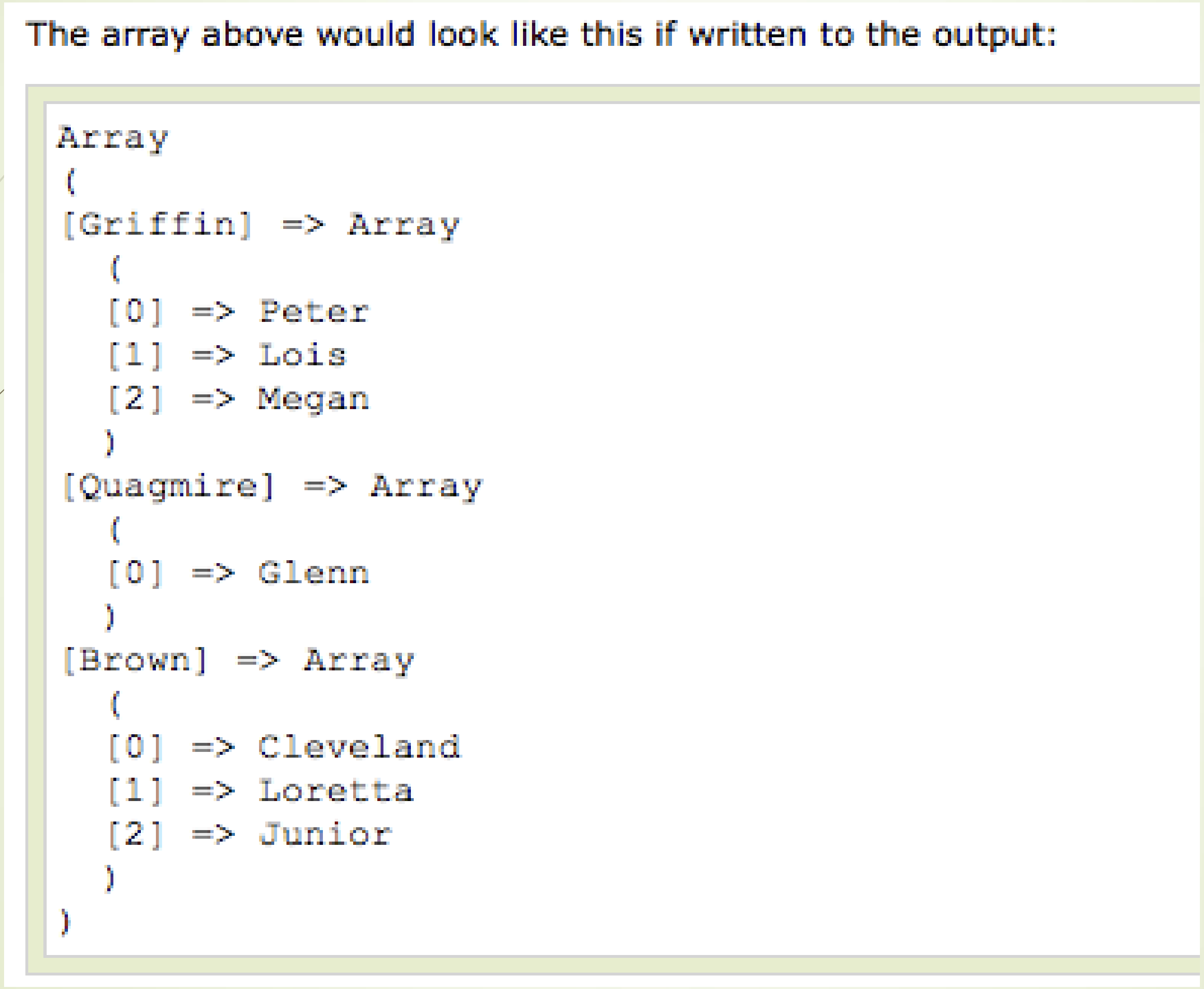
PHP Multidimensional Arrays



In this example we create a multidimensional array, with automatically assigned ID keys:

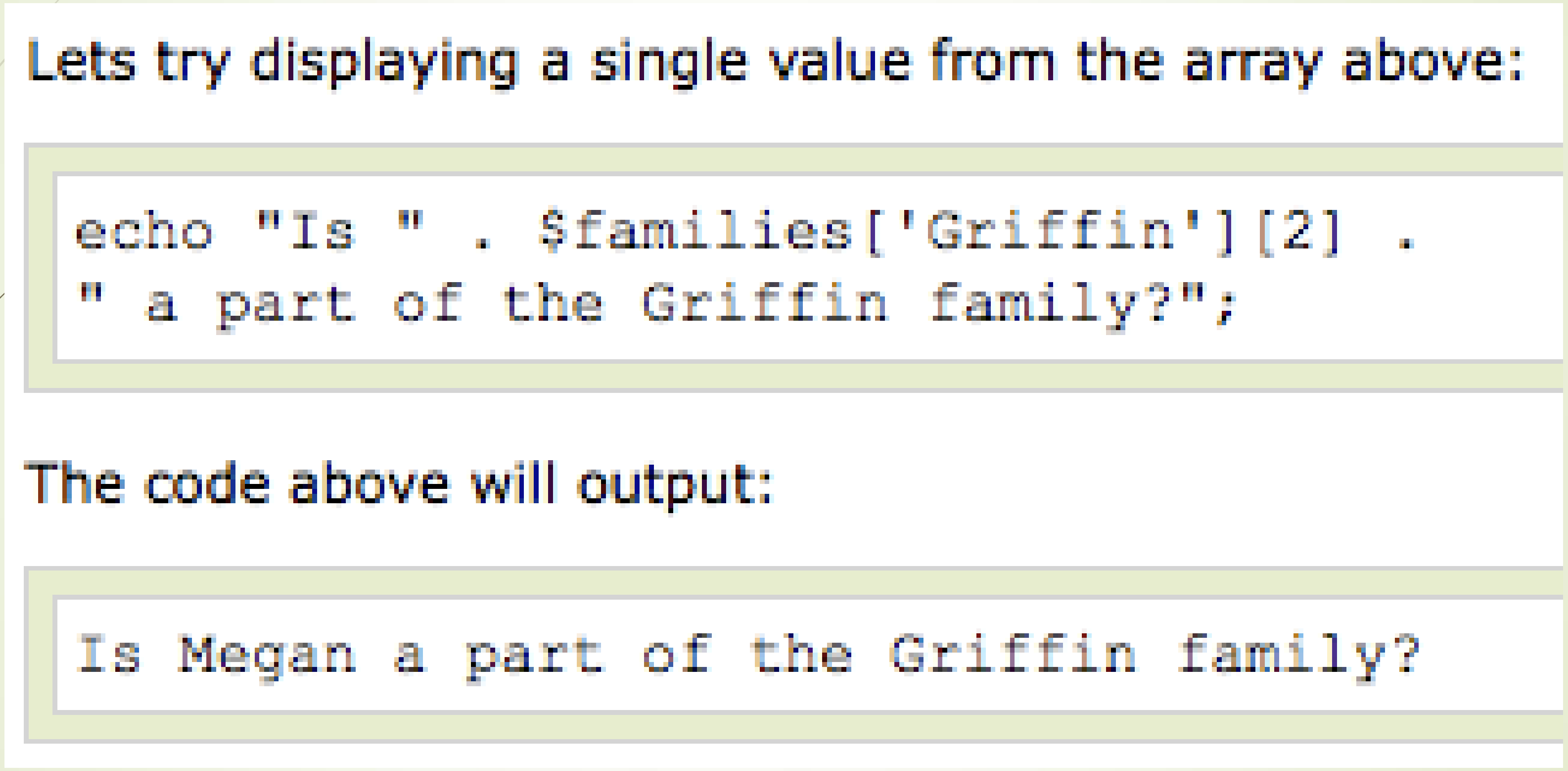
```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

PHP Multidimensional Arrays


The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
  (
  [0] => Peter
  [1] => Lois
  [2] => Megan
  )
[Quagmire] => Array
  (
  [0] => Glenn
  )
[Brown] => Array
  (
  [0] => Cleveland
  [1] => Loretta
  [2] => Junior
  )
)
```

PHP Multidimensional Arrays


Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

PHP Array Functions

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

OUTPUT

```
I like Volvo, BMW and Toyota.
```

String & in-built functions

- Single quote
- Double quote
- heredoc syntax: <<<

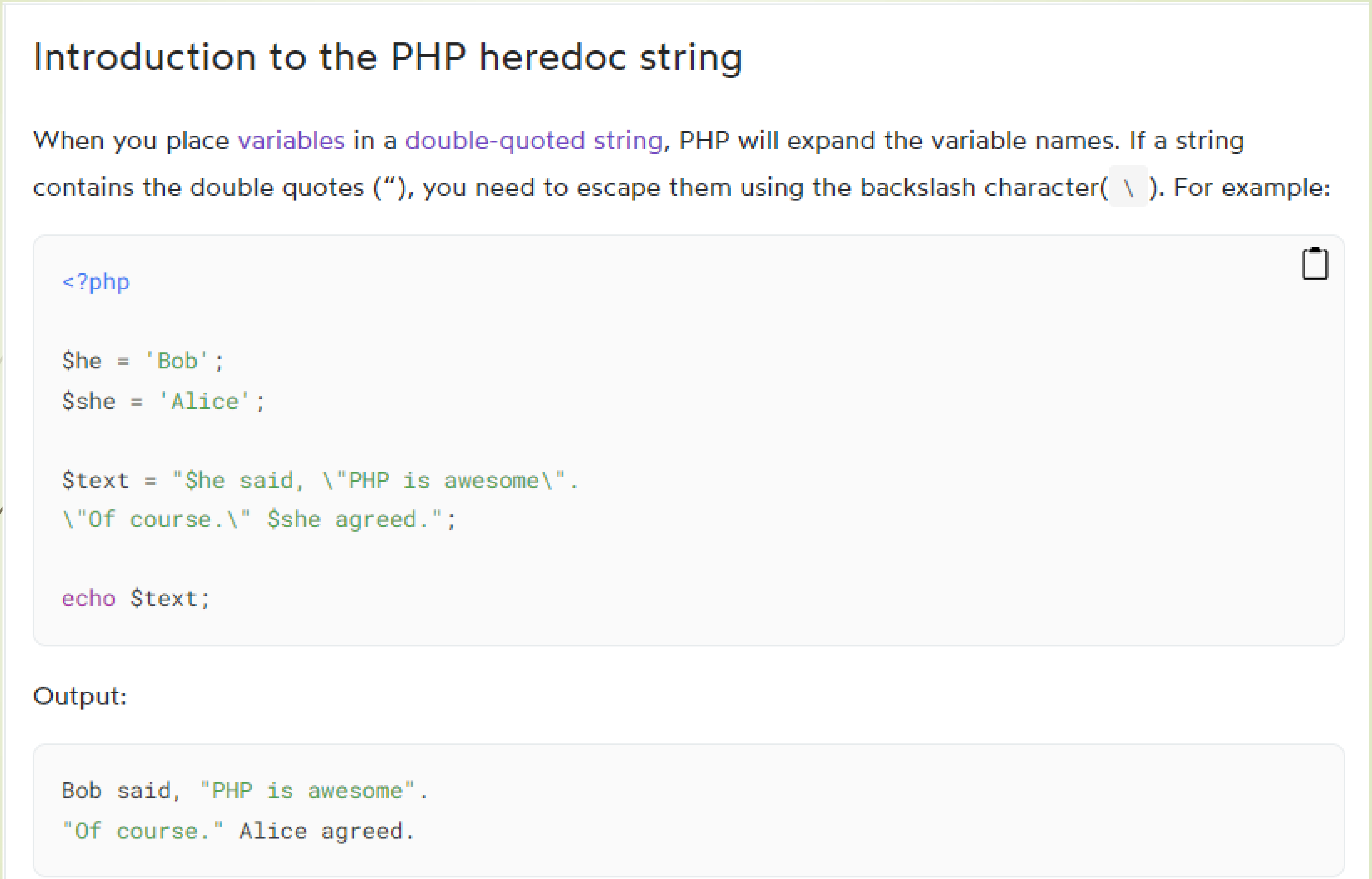
- **Functions:** strlen, strcmp, strrev, strtolower, strtoupper, trim, explode, implode, join, split, crypt, htmlentities, echo, print……

heredoc syntax: <<<

- The heredoc syntax is a way to declare a string variable.
- It use special character **<<<** at the beginning.

Syntax

```
$ nameOfVariable = <<< identifier
// string
// string
// string
identifier;
```



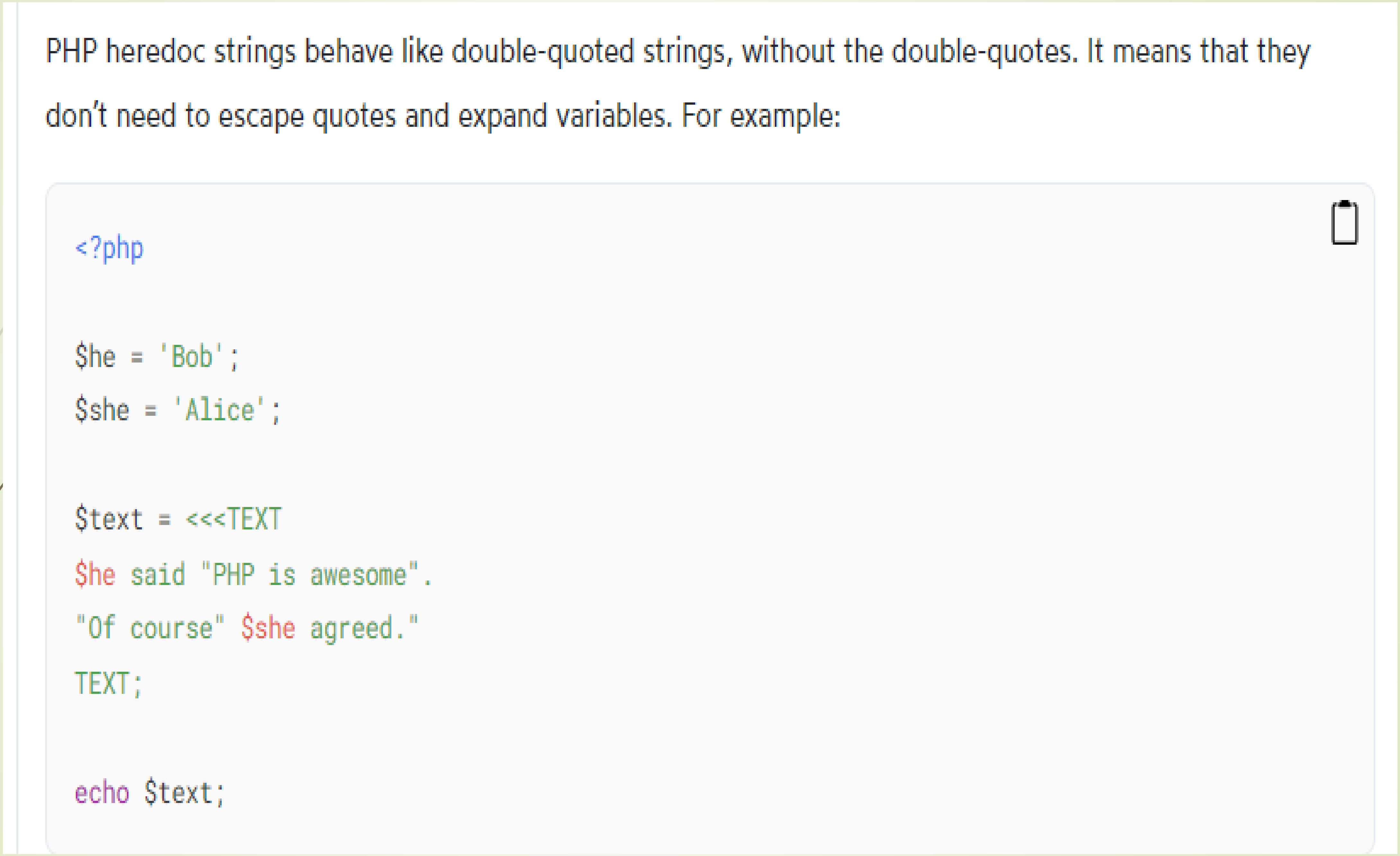
Introduction to the PHP heredoc string

When you place variables in a double-quoted string, PHP will expand the variable names. If a string contains the double quotes ("), you need to escape them using the backslash character(`\`). For example:

```php
<?php

$he = 'Bob';
$she = 'Alice';

$text = "$he said, \"PHP is awesome\".
\"Of course.\" $she agreed.";

echo $text;
```

Output:

```
Bob said, "PHP is awesome".
"Of course." Alice agreed.
```



PHP heredoc strings behave like double-quoted strings, without the double-quotes. It means that they don't need to escape quotes and expand variables. For example:

```php
<?php

$he = 'Bob';
$she = 'Alice';

$text = <<<TEXT
$he said "PHP is awesome".
"Of course" $she agreed."
TEXT;

echo $text;
```

DOs and DON’Ts of heredoc syntax usage

- The variable should start with <<< and identifier in the first line. After the last line, there must be the same identifier again.
- You can have HTML tags as part of the string.
- Don’t attempt to add a function or condition in the strings; this is an erroneous operation and it won’t execute the condition or function.
- Use curly brackets {} to contain any other variable that you want to display the contents of as part of the strings.

User defined function

- Syntax:

```
function functionname()
{
//code to be executed
}
```

- PHP supports Call by Value (default), Call by Reference, Default argument values and Variable-length argument list.

User defined function

- Types of functions:

  - Call by Value (default)
  - Call by Reference
  - Default argument values
  - Variable-length argument list
  - Recursion

PHP Global Variables - Superglobals

Superglobals: Predefined variables in PHP

Always accessible in all scopes.

It can be accessed from any function, class or file.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

PHP Global Variables - Superglobals

PHP $GLOBALS

- $GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
- PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.
- Example:

```
<?php
$x = 10;
$y = 20;
 function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z;
?>
```

PHP Global Variables - Superglobals

PHP $_SERVER

- $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.
- Example:

```
<?php
echo $_SERVER['PHP_SELF'];  //Returns the filename of the currently executing script
echo "<br>";
echo $_SERVER['SERVER_NAME']; //Returns the name of the host server
echo "<br>";
echo $_SERVER['HTTP_HOST']; //Returns the Host header from the current request
echo "<br>";
echo $_SERVER[''REQUEST_METHOD'']; //Returns the request method used to access the page (post)
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT']; //Browser compatibility information
echo "<br>";
echo $_SERVER['SCRIPT_NAME']; //Returns the path of the current script
?>
```

PHP Global Variables - Superglobals

PHP $_GET

- PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".
- $_GET can collect data sent via URL.

Example: Assume we have an HTML page that contains a hyperlink with parameters:

index.html

```
<html>
<body>
 <a href="test.php?subject=WT&technology=PHP">Test $GET</a>
 </body>
</html>
```

Test.php

```
<html>
<body>
 <?php
echo "Study " .
$_GET['subject'] . " at " .
$_GET[‘technology'];
?>
</body>
</html>
```

PHP Global Variables - Superglobals

PHP $_POST

- PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post".
- $_POST is also widely used to pass variables.
- **Example:**

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;  }  }
?>
```

PHP Global Variables - Superglobals

PHP $_REQUEST

- PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.
- The $_REQUEST function is used to get the form information sent with POST method or GET method.

PHP Global Variables - Superglobals

PHP $_ENV:

- $_ENV is another superglobal associative array in PHP.
- It stores environment variables available to current script now been deprecated.
- Environment variables are imported into global namespace. Most of these variables are provided by the shell under which PHP parser is running. Hence, list of environment variables may be different on different platforms.
- This array also includes CGI variables in case whether PHP is running as a server module or CGI processor.
- PHP library has getenv()function to retrieve list of all environment variables or value of a specific environment variable

PHP Global Variables - Superglobals

PHP $_ENV:

Getenv() function: To retrieve list of all environment variables or value of a specific environment variable

Example:

```
<?php
$arr=getenv();
foreach ($arr as $key=>$val)
echo "$key=>$val";
?>
```

PHP String Operators

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

PHP Array Operators

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

Execution Operators

There is one Execution operator defined in PHP. A string inside back-ticks (``) is treated as a DOS command (a shell command in UNIX/Linux) and its output is returned.

Following code executes DIR command and returns result as string.

Example

```
<?php
$list=`dir *.php`;
echo "$list";
?>
```

Error Control Operators

In PHP @ symbol is defined as Error Control Operator. When it is prefixed to any expression, any error encountered by PHP parser while executing it will be suppressed and the expression will be ignored.

Following code tries to open a non-existing file for read operation, but PHP parser reports warning

Example

```
<?php
$fp=fopen("nosuchfile.txt","r");
echo "Hello World
"; ?>
```