PHP Programming

Dr. L.M. Jenila Livingston
VIT Chennai

Presentation Overview

1. PHP-Introduction Background

- **P**ersonal **H**ome **P**age – C kind of scripts written in Perl language, by **Rasmus Lerdorf** in 1995.
- He called the language as PHP/FI – Personal Home Page / Forms Interpreter.
- 1997 Version 2.0 was released
- Then came 3.0, 4.0. They were called PHP simply
- Then came PHP 5.4 PHP 6
- Recent version is PHP 8

PhP

- is a **server-side scripting** language designed primarily for web development
- it now stands for the recursive acronym ***Hypertext Preprocessor***
- PHP code may be embedded into HTML
- or it can be used in combination with various web template systems, web content management systems and web frameworks.

Client Side Scripting Vs Server side Scripting

Client side scripting	Server side scripting
Adds interactivity on the client browser.	Performs interaction from server side
Runs on the client browser	Requires a web server with language parser / interpreter support.
Used mainly for form validation	Used for request/response manipulation and handling
Cannot connect to database or read/write files	Interacts with the database and can do file operations

Client Side Scripting Vs Server side Scripting

Client side scripting	Server side scripting
Can view the source code by clicking on view source on the browser	Code is executed on the the server side. So cannot view the source code by clicking on view source on client browser
Example: Javascript, VB Script	Example: PHP, ASP, JSP

PHP code hidden from user

- Translates the php code into html.
- Then sends the web page to the visitor’s web browser.

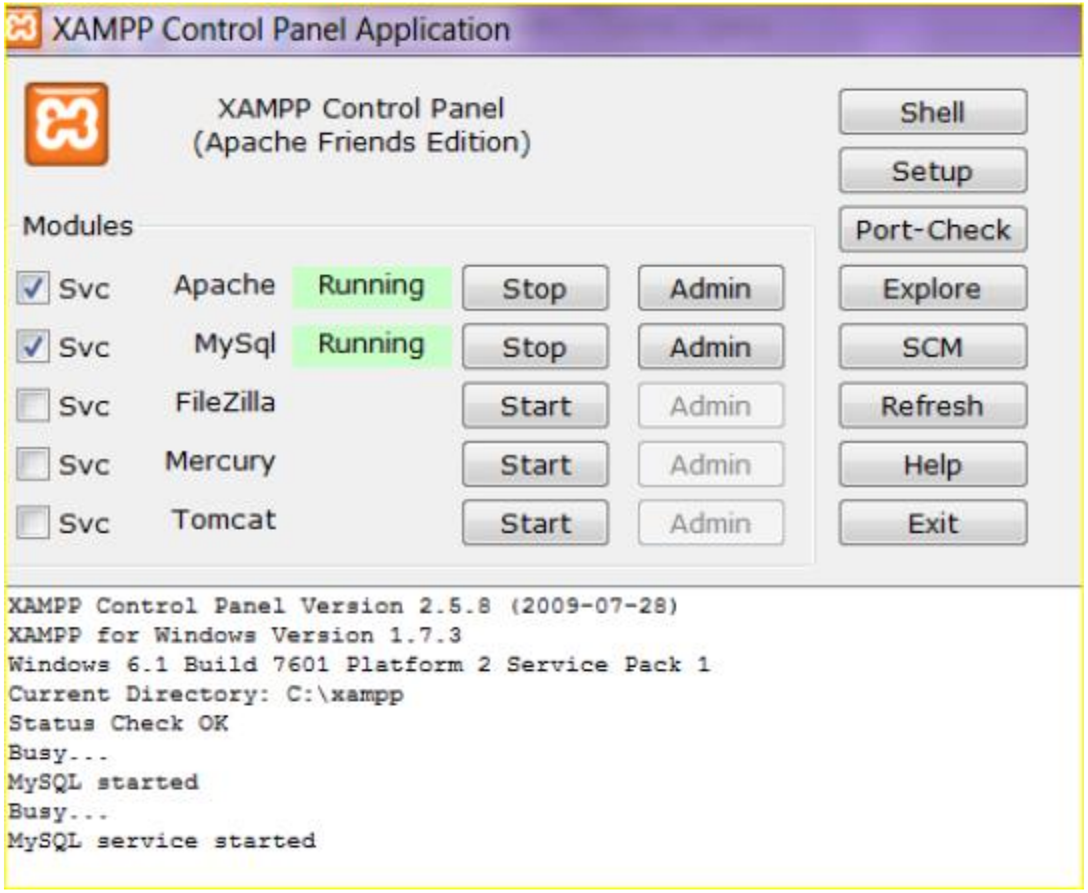
2. Packages

- **WAMP** for Window operating system (**W**indows **A**pache, **M**ySql, **P**HP)
- **LAMP** for Linux operating system (**L**inux **A**pache, **M**ySql, **P**HP)
- **XAMPP** (**X**-os,**A**pache **M**ysql **P**hp **P**erl) X-OS means in runs in all OS

Steps to run your PHP program

- Install XAMPP
- Install Notepad++
- Create a new folder (phpdemo) in hdocs
  - C:\xampp\htdocs\phpdemo
- Open XAMPP Control Panel and run Apache and MYSQL
- Open Notepad++ and type PHP program
- Save it with php file hello.php
- Open any web browser and run your program
  - http://localhost/phpdemo/hello.php

Start the Web Server



```
XAMPP Control Panel Version 2.5.8 (2009-07-28)
XAMPP for Windows Version 1.7.3
Windows 6.1 Build 7601 Platform 2 Service Pack 1
Current Directory: C:\xampp
Status Check OK
Busy...
MySQL started
Busy...
MySQL service started
```

3. PHP Comments

- All php files are saved with extension .php
- The can be written in notepad or any text editor
- Single line comment
  - // this is single line comment
- Multi line comment
  - /*….

    This is a multi line comment */

4. PHP Syntax

- The PHP scripting block

```
//starts with
<?php
php statements;
?> //ends with
```

- Each statement should end with a semicolon.

Sample PHP

```
<?php
echo “hello”;
?>
```

Sample PHP with HTML

```
<html>
<head><title>PHP Program</title> </head>
<body>
<?php
echo “hello”;
?>
</html>
```

JavaScript code as a string in the PHP code

```
<?php
echo '<script type="text/JavaScript">
    prompt(“Hello”);
    </script>' ;
?>
```

PHP Concatenation

> The concatenation operator (.) is used to put two string values together.

> To concatenate two string variables together, use the concatenation operator:

```
<HTML>
<HEAD><u>Concatenation</u><HEAD>
<br><body>
<?php
$a="Hi! Happy";
$b="days";
$c=$a.$b;
echo "$c";
?>
</body>
</html>
```

5. echo and print

- 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.

echo Vs print

```
<?php
$b=0;
$c=17;
$b ? print "true" : print "false";
echo "<br>";
$c ? print "true" : print "false";
//echo throws error
//$b ? echo "true" : echo "false";
?>
```

echo Vs print

Multiple Parameters:

- echo "This ", "string ", "was ", "made ", "with multiple parameters.";
- print "This ", "string ", "was ", "made ", "with multiple parameters.";
- Print statement throws error

Echo example

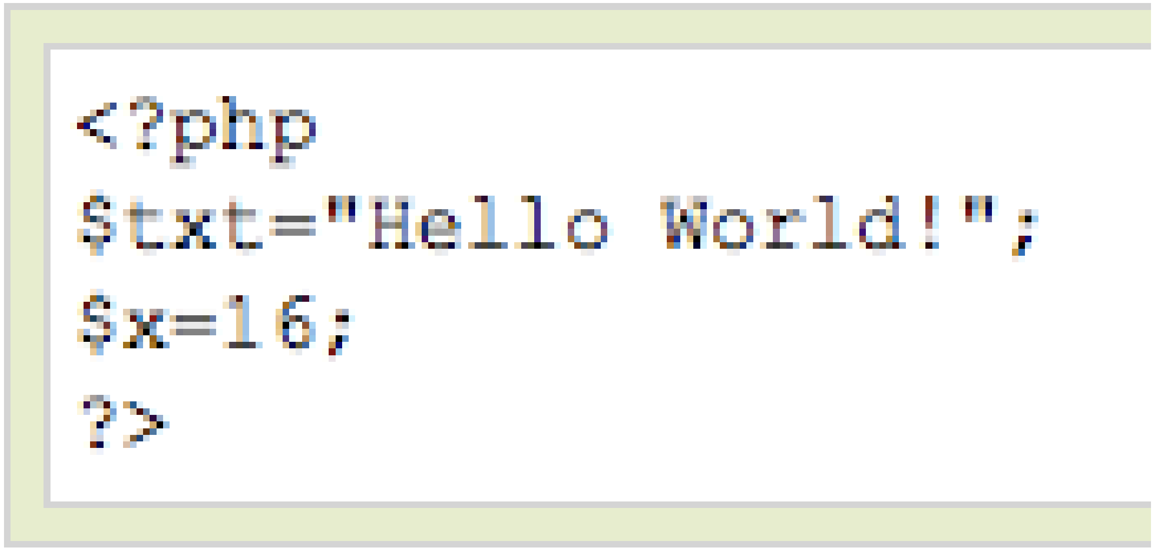
```
<?php
$foo = 25;        // Numerical variable
$bar = "Hello";// String variable

echo $bar;        // Outputs Hello
echo $foo,$bar;// Outputs 25Hello
echo "5x5=",$foo;     // Outputs 5x5=25
echo "5x5=$foo";      // Outputs 5x5=25
echo '5x5=$foo';      // Outputs 5x5=$foo
?>
```

- **Notice** how echo '5x5=$foo' outputs $foo rather than replacing it with 25
- Strings in single quotes (' ') are not interpreted or evaluated by PHP

6. Variable Basics

- Variables start with a **$ symbol**
- Variables in PHP are represented by a **dollar sign followed by the name of the variable.**
- Variable name is **case-sensitive**.
- Variable names follow the same rules as other labels in PHP.
  - A valid variable name
    - starts with a letter or underscore
    - Followed by any number of letters, numbers, or underscores.

PHP Variables



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

Variables

- Example

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var"; // outputs "Bob, Joe"
$4site = ’not yet’; // invalid; starts
  with a number
$_4site = ’not yet’; // valid; starts
  with an underscore
```

Four scalar types

- boolean
- integer
- floating-point number (float)
- string

Scalars

- Very simple:
  - `$foo = true;` (boolean)
  - `$foo= 20;` (integer)
  - `$foo= 3.1415;` (float)

Example

```
<html>
<head> Basic Maths</head>
<body>
<?php
$var1 = 10;
$var2 = 20;
echo "<BR><HR><B> Addition of Two numbers";
$sum=$var1+$var2;
print "<BR>The sum of $var1 + $var2 is $sum";
echo "<BR><HR><B> Sub of Two numbers";
$sub=$var2-$var1;
echo "<BR>The sum of $var2 - $var1 is $sub";
?>
</body>
</html>
```

Type Juggling

- Example

```
<?PHP
$foo = "0";
echo $foo."<BR>";
// $foo is string (ASCII 48)
$foo += 2;
echo $foo."<BR>";
// $foo is now an integer (2)
$foo = $foo + 1.3;
echo $foo;
// $foo is now a float (3.3)
?>
```

Constants

Define() is used to declare a constant

Constant names can contain letters, numbers and the (_) underscore character.

```
<html> <body> Constants <?php
define('STRING_CONSTANT','Student');
define('INTEGER_CONSTANT', 500);
define('FLOAT_CONSTANT',500.50);
echo INTEGER_CONSTANT;
echo FLOAT_CONSTANT;
echo STRING_CONSTANT; ?> </body> </html
```

7. Strings

- A string literal can be specified in three different ways.
  - single quoted
    - Variables not expanded
  - double quoted
    - `$foo = 20;`
    - `echo “The value of foo is $foo”;`

The Length of a String

-
  ```
  <?php
  echo strlen("Hello world!");
  ```

-
  ```
  // outputs 12
  ?>
  ```

Count The Number of Words in a String

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

Reverse a String

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

Search For a Specific Text Within a String

-
  ```
  <?php
  echo strpos("Hello world!", "world");
  ```

-
  ```
  // outputs 6
  ?>
  ```

Replace Text Within a String

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

String Change case

- $a= ‘string function’;
- Strtoupper($a) = STRING FUNCTION
- Strtolower($a) = string function
- Ucfirst($a) = String function
- Ucwords($a) = String Function

String split

- $a = abc@gmail.com
- $temp=**explode**('@', $a);
- explode splits the text string into pieces as specified by the first parameter.
- $temp[0]='abc';
- $temp[1]='gmail.com'

Splitting into chunks

str_split() : function to break the string into fixed-length “chunks”

```
<?php
$str1 = 'Google welcome you';
$chunksize=3;
$chunkarray=str_split($str1,$chunksize);
print_r($chunkarray);
?>
```

```
chunkarray ( [0] => Goo [1] => gle [2] => we [3] => lco [4] => me [5] => you )
```

String Join

- **Implode() / join()**
- #$temp[0]=‘abc’;
- #$temp[1]=‘gmail.com’
- $temp1=**implode**(‘@’, $temp);
- #joins the arrays element with the string passed i.
- $temp1=abc@gmail.comn the first parameter

String repeat

- $stri="Welcome";
- Str_repeat($stri,no.of times);
- Eg. Str_repeat($stri,2);
- Welcome Welcome

ASCII and back

- $char = “A";
- $asc = ord($char);
- echo $asc;
- $asc=65;
- $char = chr($asc);
- echo $char;

COMPARING FOR SIMILARITY

- **Metaphone()**: compare two strings to see if they sound similar.
- echo (metaphone("files")==metaphone("flies"))?"similar"**:"not similar";**
- echo (metaphone("files")==metaphone("miles"))?"similar":"**not similar";**

8. Date Display

```
2022/03/21
```

```
$datedisplay=date(“yy/m/d”);
Print $datedisplay;
```

```
Monday, March 21, 2022
```

```
$datedisplay=date(“l, F j, Y”);
Print $datedisplay;
```

Month, Day & Date Format Symbols

M	Jan
F	January
m	01
n	1

Date of Month	d	01
Date of Month	j	1
Day of Week	l	Monday
Day of Week	D	Mon

Set Date

- Syntax:

date_isodate_set(*object, year, week, day)*

```
<?php
$date=date_create();
// Set the ISO date to the 5th week in 2020
date_isodate_set($date,2020,5);
echo date_format($date,"Y-m-d") . "<br>";

// Set the ISO date to the 3rd day from 5th week in 2020
date_isodate_set($date,2020,5,3);
echo date_format($date,"Y-m-d");
?>
```

Output:

```
2020-01-27

2020-01-29
```

Get Time

- **H** - 24-hour format of an hour (00 to 23)
- **h** - 12-hour format of an hour with leading zeros (01 to 12)
- **i** - Minutes with leading zeros (00 to 59)
- **s** - Seconds with leading zeros (00 to 59)
- **a** - Lowercase Ante meridiem and Post meridiem (am or pm)

Get Time

- <?php
  echo "The time is ".date(“H:i:s") ;

  // Output **The time is 09:30:36**

?>

- <?php
  date_default_timezone_set("Asia/Kolkata ");

  //India time (GMT+5:30)

  echo "The time is " . date("h:i:sa");

//**The time is 09:29:57am**
?>

Set Time

- Syntax:

  date_time_set(*object, hour, minute, second, microseconds)*

```
<?php
$date=date_create("2020-07-01");

date_time_set($date,11,30);
echo date_format($date,"Y-m-d H:i:s") . "<br>";

date_time_set($date,11,30,14);
echo date_format($date,"Y-m-d H:i:s");
?>
```

Set Time

$dt = new DateTime;

// Set Time to zero
$dt->setTime(0, 0);
echo $dt->format('H:i:s');

- // Add one hour

$dt->**add**(new DateInterval('PT1H'));
echo $dt->format('H:i:s');

- // add 2 years, 1 day and 9 seconds

$dt->add(new DateInterval('P2Y1DT9S'));
// P = period, 2Y = 2 years, 1D = 1 day, T = time, 9S = 9 seconds

mktime

- Syntax:

  mktime(*hour, minute, second, month, day, year*)

-
  ```
  <?php
  $d=mktime(10, 14, 52, 7, 11, 2020);
  echo "Created date is " . date("Y-m-d h:i:sa", $d);
  ?>
  ```

9. PHP Arrays

In PHP, there are three kind of arrays:

> **Indexed/ 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

Indexed Arrays

Syntax: $variable=array(values);

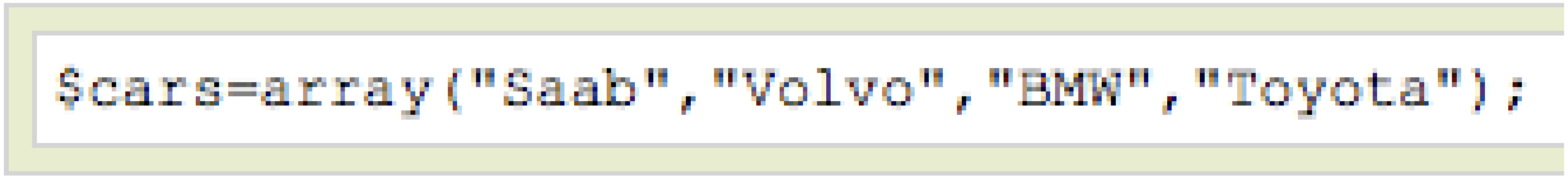
$a = array(10,20,30);

Here, the indices are automatically assigned by PHP as 0,1,2

i.e $a[1] has the value 20

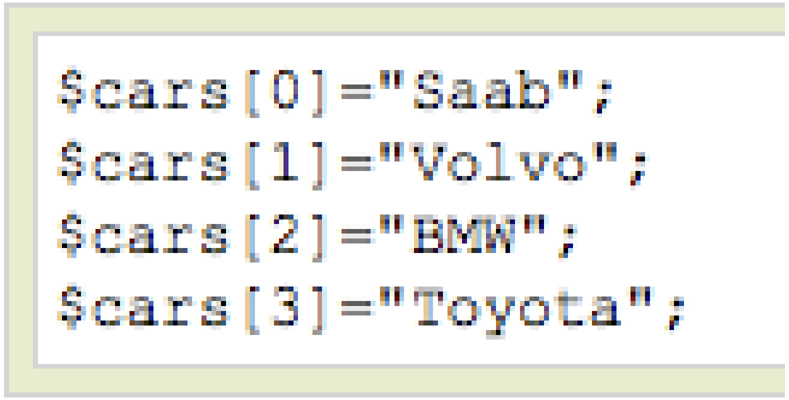
Indexed Arrays

In the following example the index is automatically assigned (the index starts at 0):



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

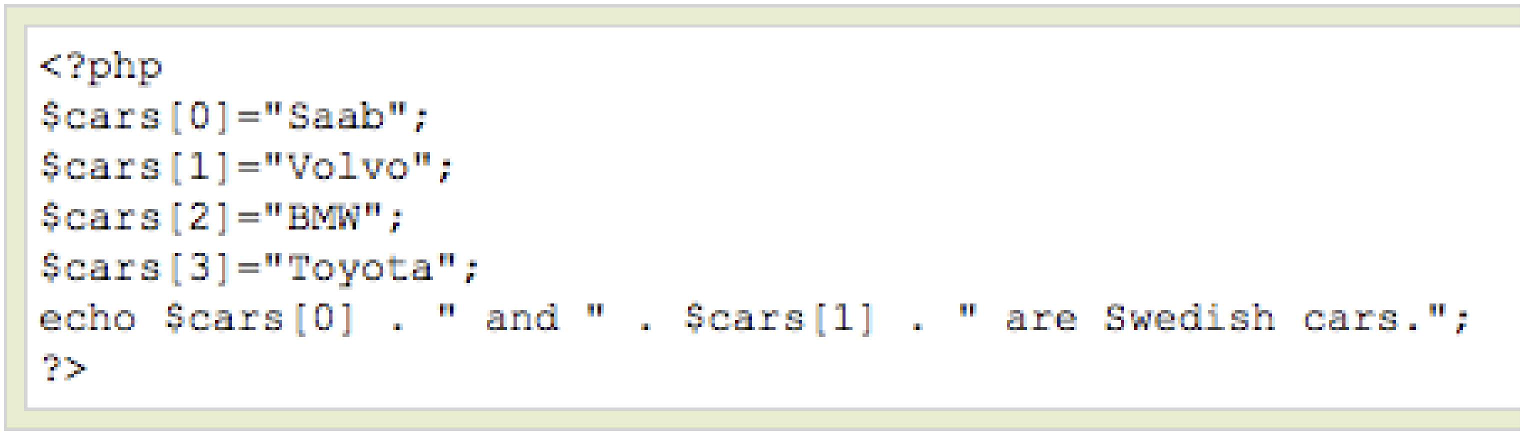
In the following example we **assign the index manually**:



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

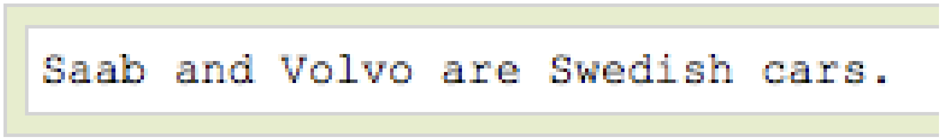
Indexed Arrays

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



```php
<?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.
```

Array using range

to store sequence of numbers in an array.

- **$variable_name=range(from,to,stepsize);**
- $productcode=range(1,10);

#number sequenceelements from 1 to 10

- $productcode=range(1,10,2);

#odd elements

- $alphabets=range('a','z');
- #alphabets from a to z
- $alphabets=range('a','z',2);
- #a,c,e,..

Display the array contents

- <?php
- $letters=range('a','z',2);
- for($i=0;$i<=5;$i++)
- echo $letters[$i]. ' ';
- ?>

Output:

```
a c e g i k
```

Display the array contents: print_r

```
<?php
   $cars=array("Volvo","BMW","Toyota");
    echo $cars[0]."<BR>";
    print_r ($cars);
?>
```

The **print_r** function is used to return an array in a human readable form in the **PHP** programming language.

Output:

```
Volvo
Array ( [0] => Volvo [1] => BMW [2] => Toyota
```

Array Length

- To display the length of the array
- echo sizeof($letters);
- echo count($numbers)

PHP Associative Arrays

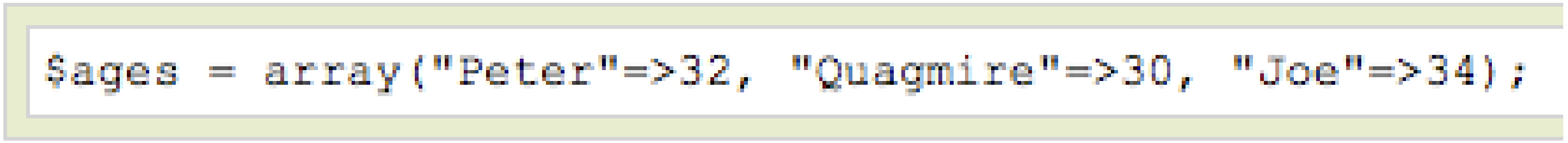
> With an associative array, each ID key is associated with a value.

> 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.

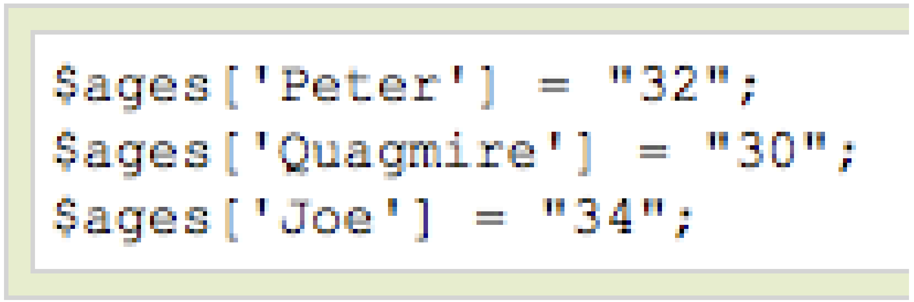
PHP Associative Arrays

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



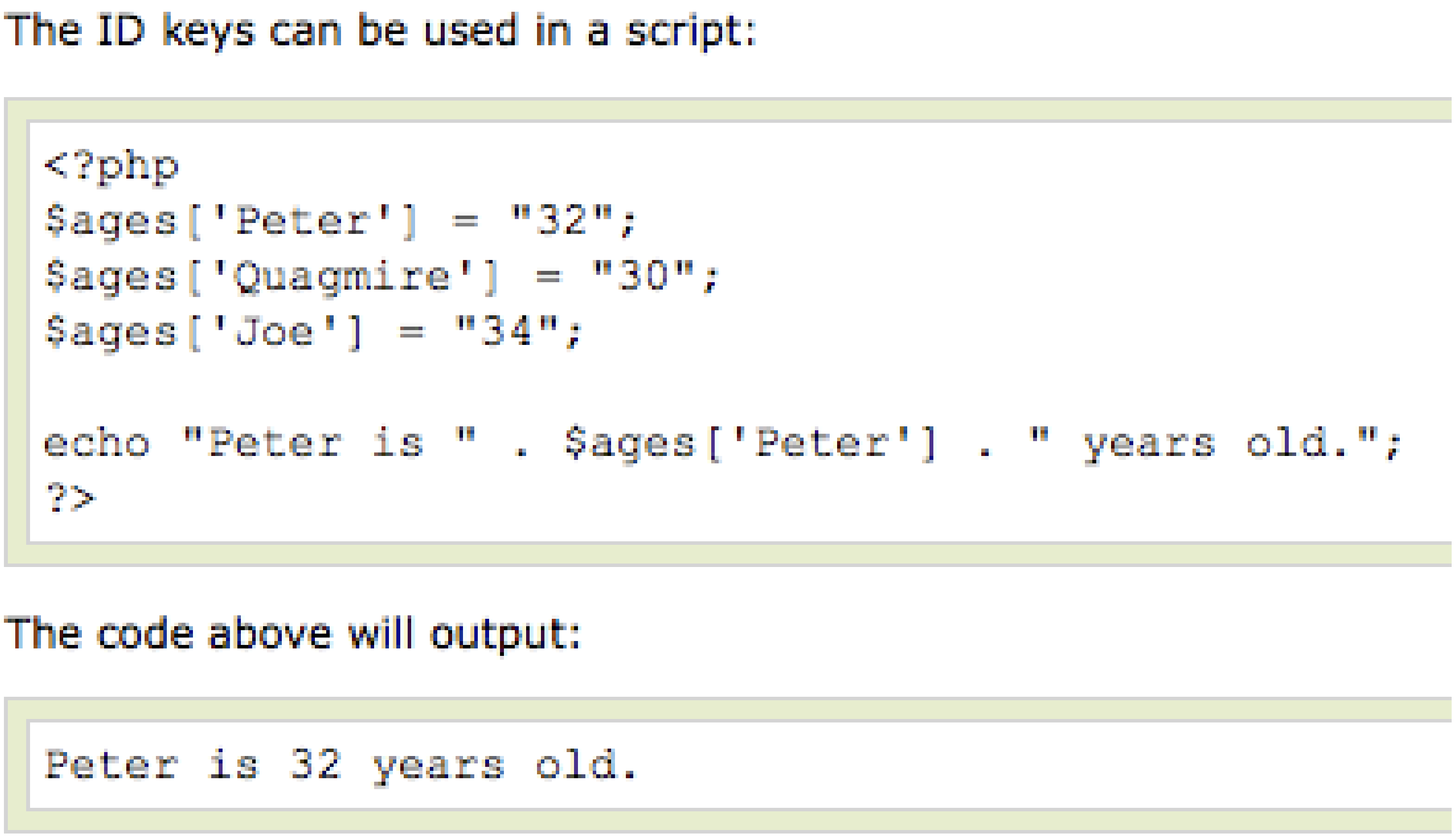
```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

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



```
$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:

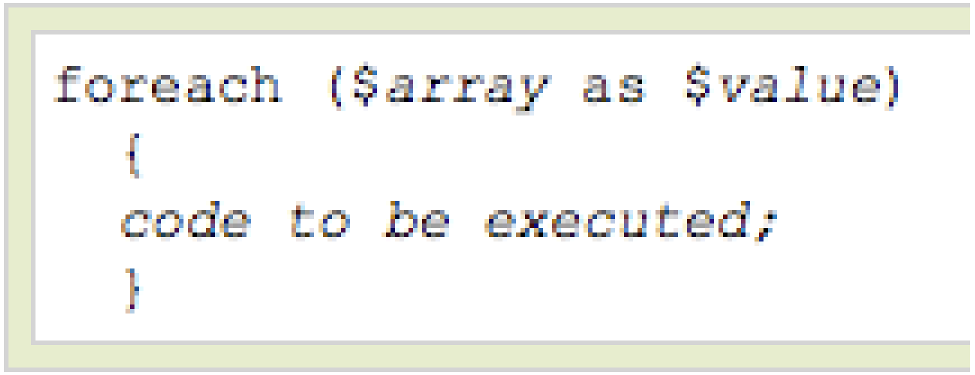
```
Peter is 32 years old.
```

foreach

- The foreach construct provides an easy way to iterate over arrays.
- foreach works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable.

foreach

- For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.



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

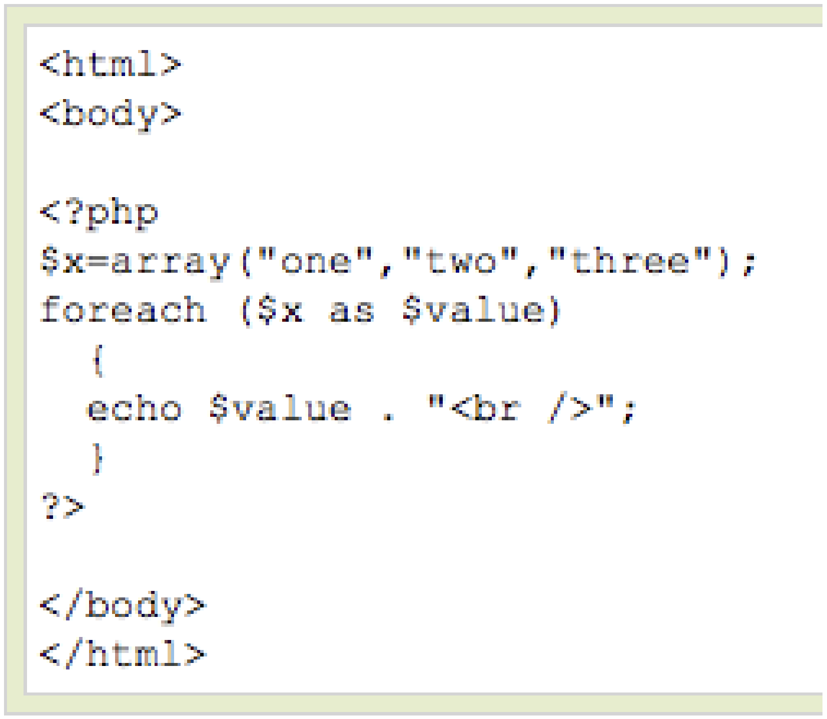
foreach

- There are two syntaxes:

```
foreach ($array_var as $value)
Statements;

foreach ($array_var as $key => $value)
Statements;
```

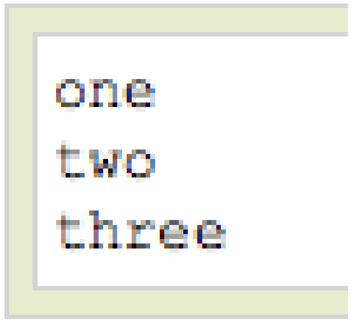
foreach

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>
```

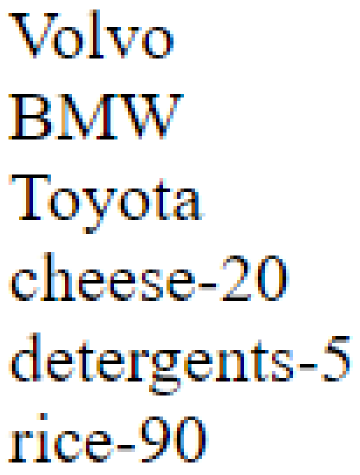
Output:



```
one
two
three
```

foreach

```
$cars=array("Volvo","BMW","Toyota");
foreach ($cars as $c)
echo $c."<BR>";


$prices=array('cheese'=>20,
   'detergents'=>5,'rice'=>90);
foreach ($prices as $k=>$v)
echo $k."-".$v."<BR>";
```



```
Volvo
BMW
Toyota
cheese-20
detergents-5
rice-90
```

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.

Two-dimensional array

CODE	Product Name	Cost
CHE	Cheese	20
DET	Detergents	5
RICE	Rice	90

```
$products=array(
            array('CHE','cheese',20),
            array('DET',detergents',5),
            array('RICE','rice',90));
```

To display the contents

```
for ($row=0;$row<3;$row++)
{
 for ($column=0;$column<3;$column++)
  {
    echo '|'.$products[$row][$column];
  }
echo '<br>';
}
```

Multi Dimensional Array

Food Products		
Code	Product Name	Cost
CHE	Cheese	20
RICE	Rice	90

Domestic Items		
Code	Product Name	Cost
DET	Detergents	5

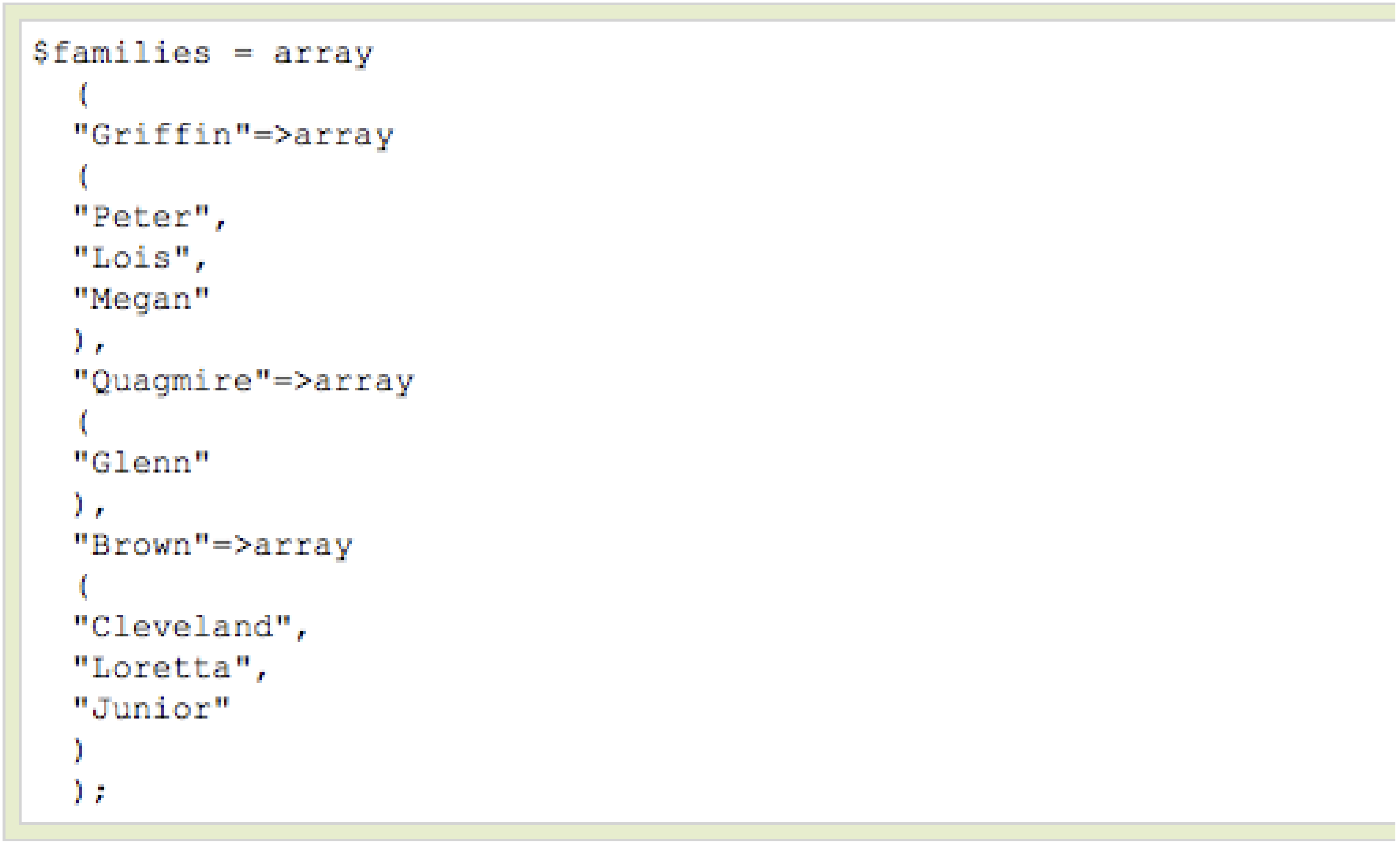
Baby Care		
Code	Product Name	Cost
TEE	Teether	500

Multi Dimensional Array

```
$categories = array(
                    array(
                      array('TEE','Teether',500),
                      array('BOT','Feeding Bottles',350) ),
                   array(
                      array('CHE','Cheese',30),
                      array('RICE''Rice',90) ),
                  array(
                      array('DET','Deterdents',5),
                      array('PHE','Phenol',19) )
              );
```

PHP Multidimensional Arrays

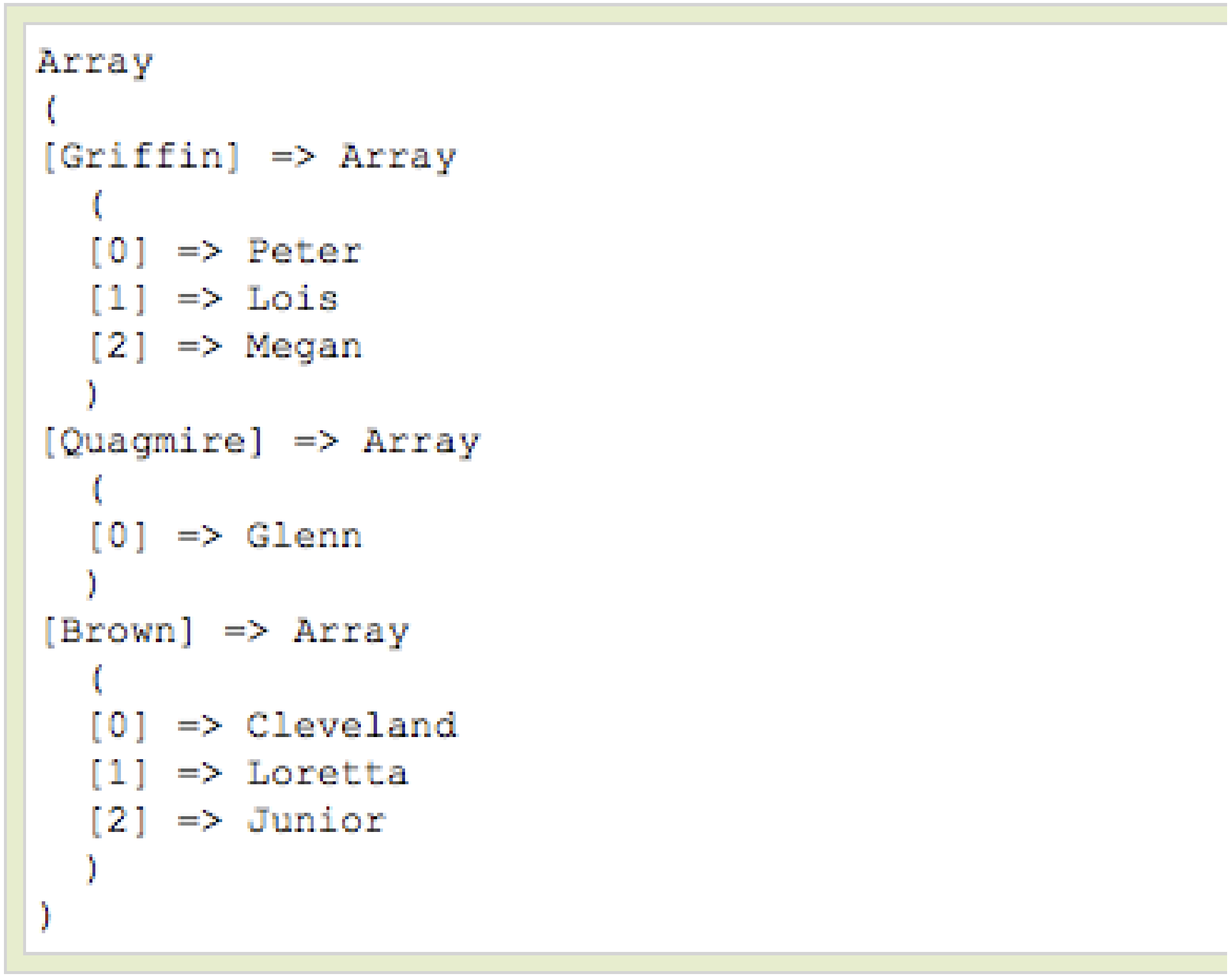
In this example we create a multidimensional array, with automatically assigned ID keys:



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

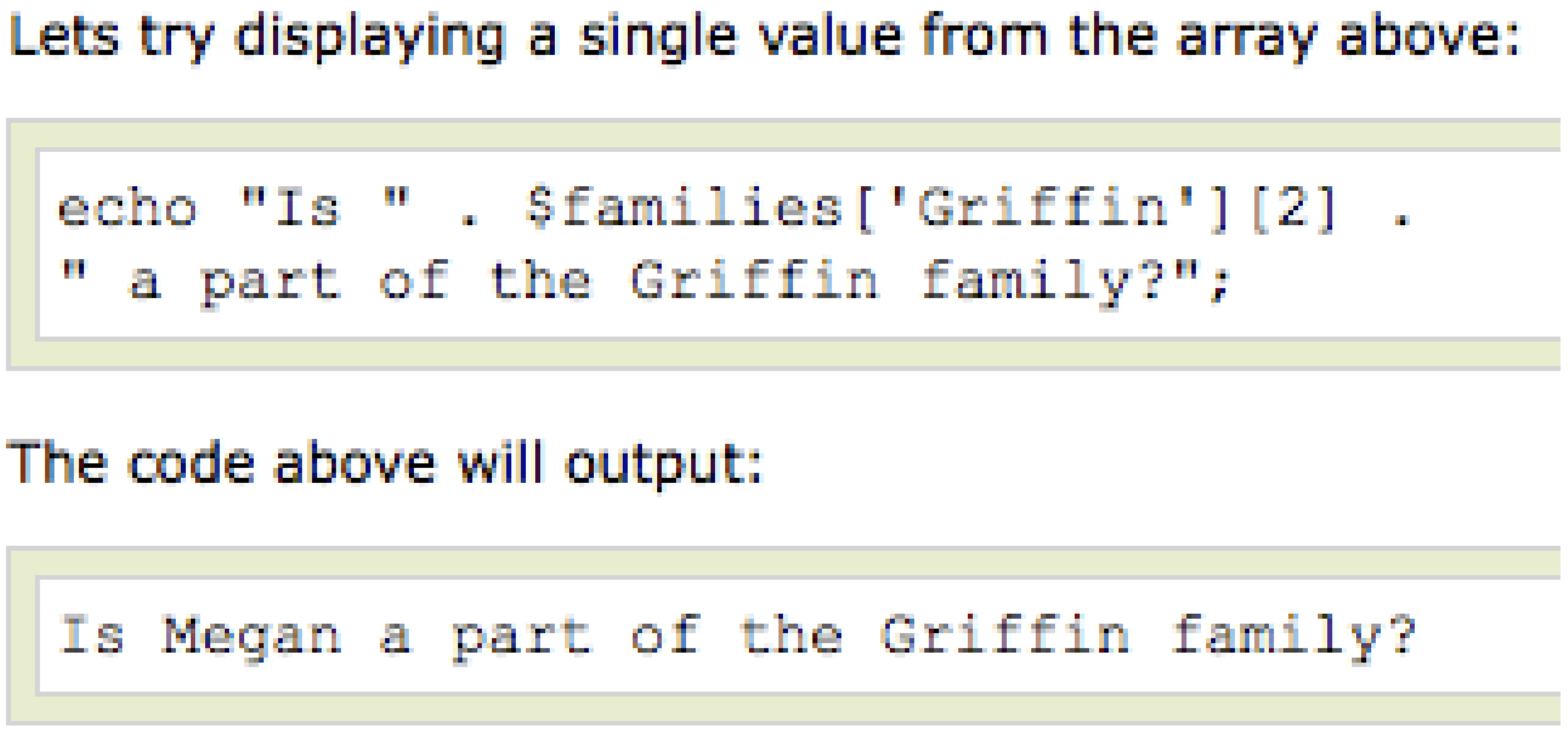
PHP Multidimensional Arrays

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?
```

Sorting of Arrays

- sort()
- Asort()
- Ksort()

$products=array(‘cheese’,’rice’,’detergents’);

- **Sort**($products);

- $prices=array(30,90,5);
- Sort($prices);

Sorting of Arrays

```
$products=array(‘Cheese’=>30,’Rice’=>90,’Deter
  gents’=>5);
asort($products); // sorts by value
print_r($products);
//Array ( [’Detergents’] => 5 [‘Cheese’] => 30 [’Rice’] => 90 )

ksort($products); //sorts by key
print_r($products);
//Array ( [‘Cheese’] => 30 [’Detergents’] => 5 [’Rice’] => 90 )
```

Sorting in Reverse

- Rsort()
- Arsort()
- Krsort()

Arrays - Shuffle

- $pictures =array('chocolate.jpg','burger.jpg','fruits.jpg');
- shuffle($pictures);
- print_r($pictures);

Arrays - Split

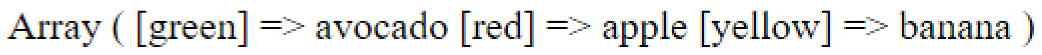
- Array chunk: Chunks an array into arrays withsize elements.
- $input_array = array('a', 'b', 'c', 'd', 'e');
- print_r(**array_chunk**($input_array, 2));

Arrays - Merge

- array_combine: Creates an array by using **one array for keys and another for its values**
- Syntax:

array **array_combine** (array $keys , array $values)

- <?php
- $a = array('green', 'red', 'yellow');
- $b = array('avocado', 'apple', 'banana');
- $c = array_combine($a, $b);
- print_r($c);
- ?>



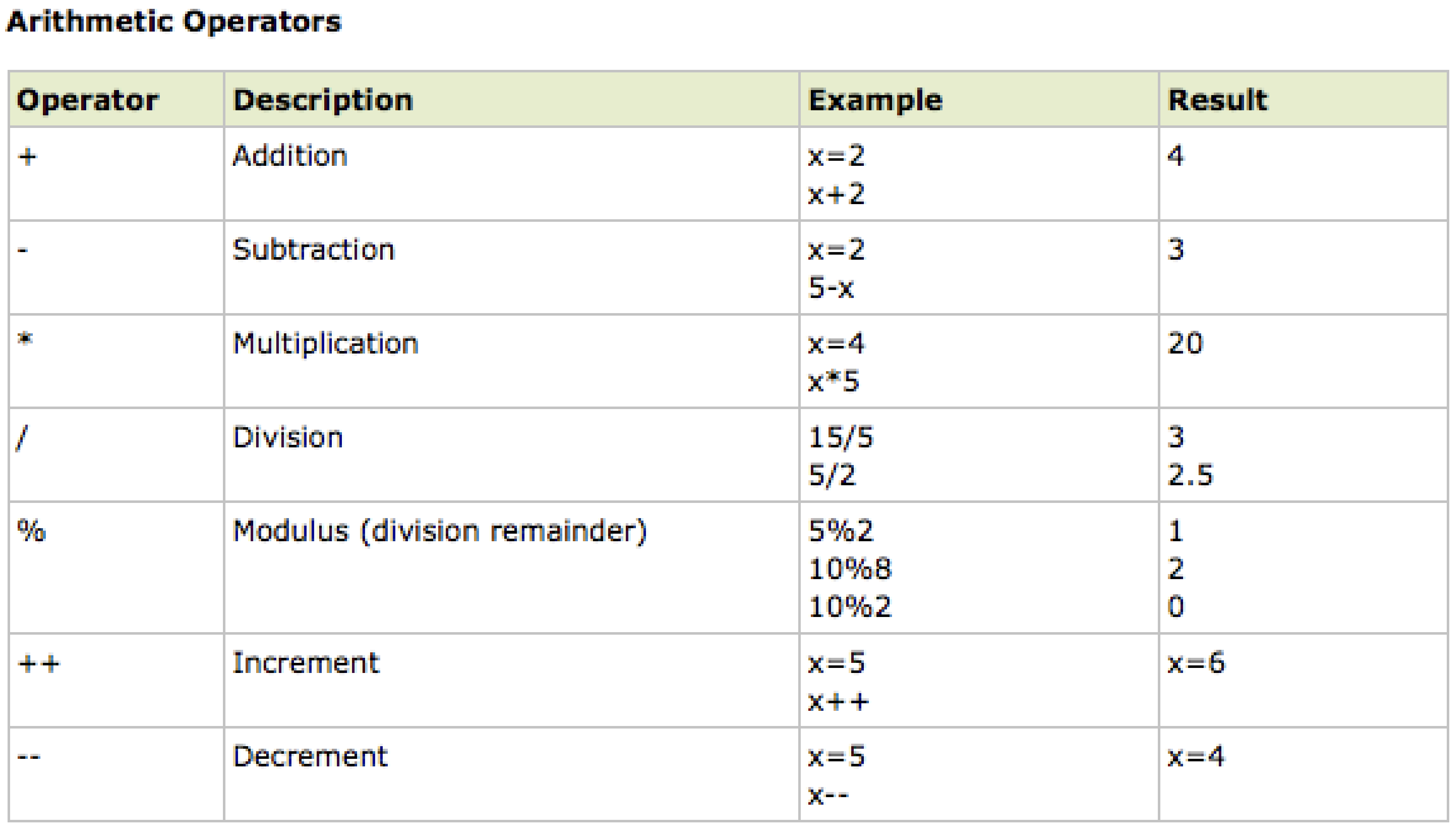
```
Array ( [green] => avocado [red] => apple [yellow] => banana )
```

10. PHP Operators

Operators are used to operate on values. There are four classifications of operators:

- > Arithmetic
- > Assignment
- > Comparison
- > Logical
- > Concatenation

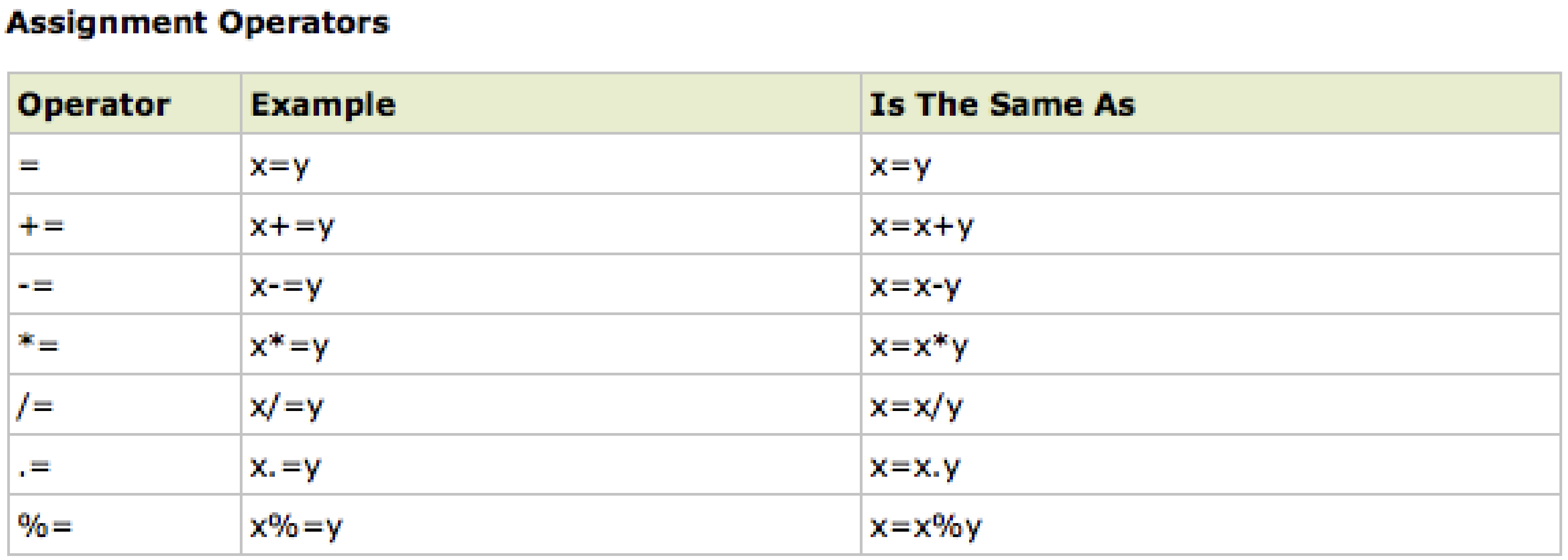
PHP Operators



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
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

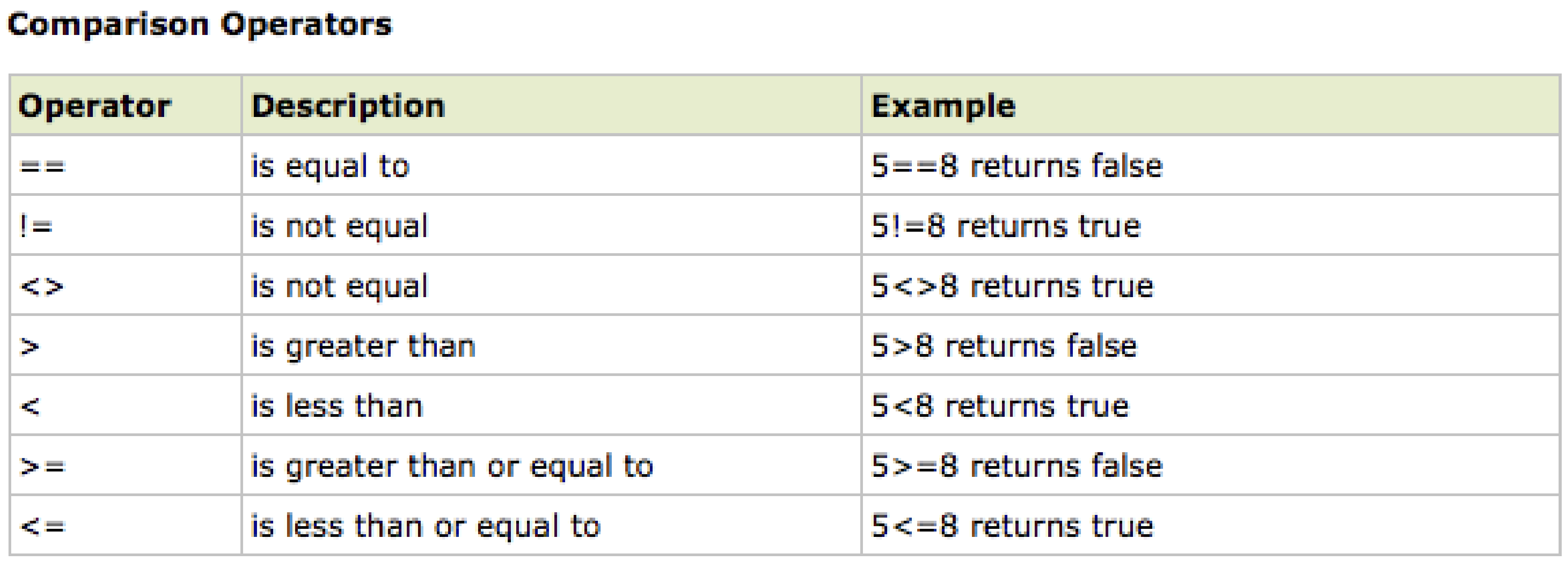
PHP Operators



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

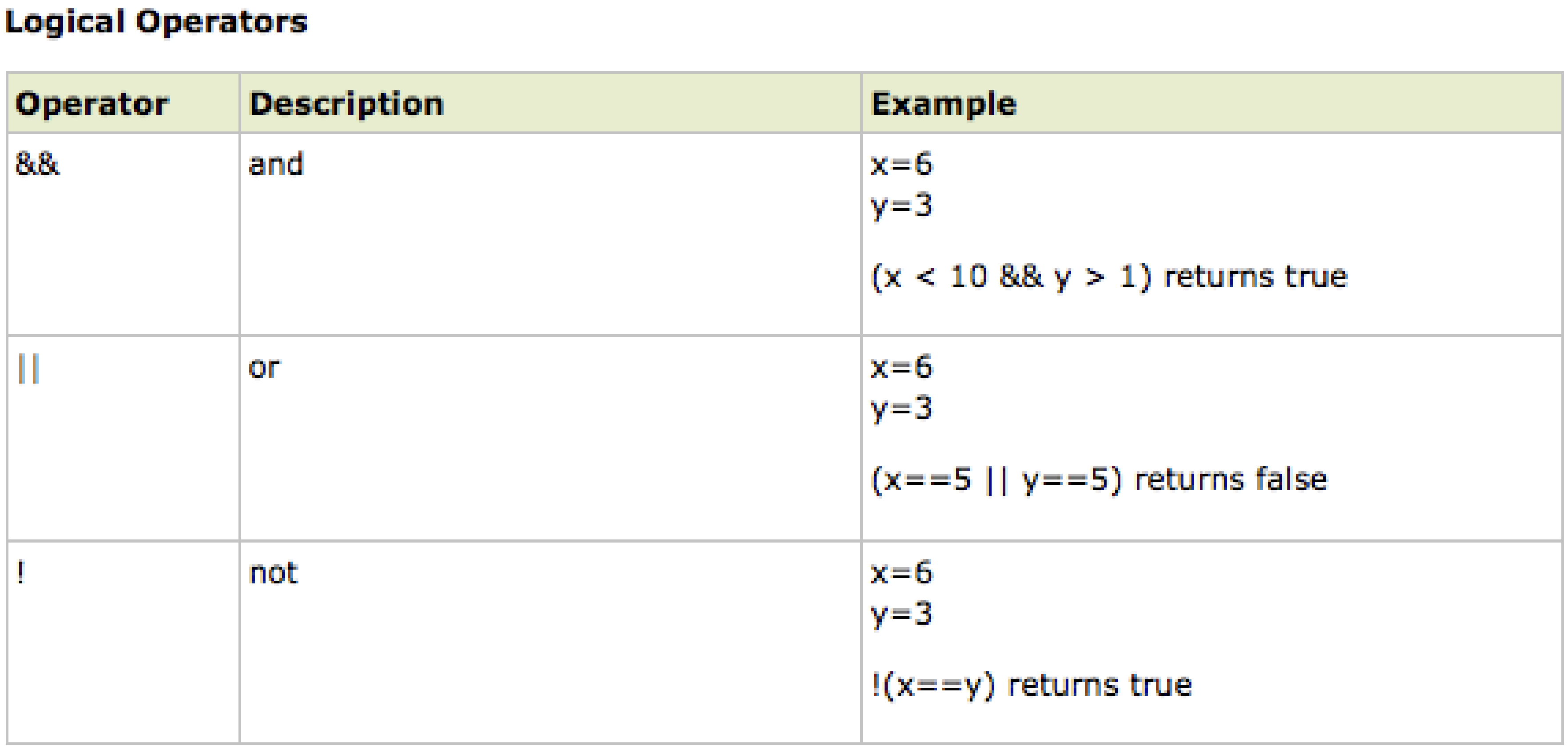
PHP Operators



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



Logical Operators

Operator	Description	Example

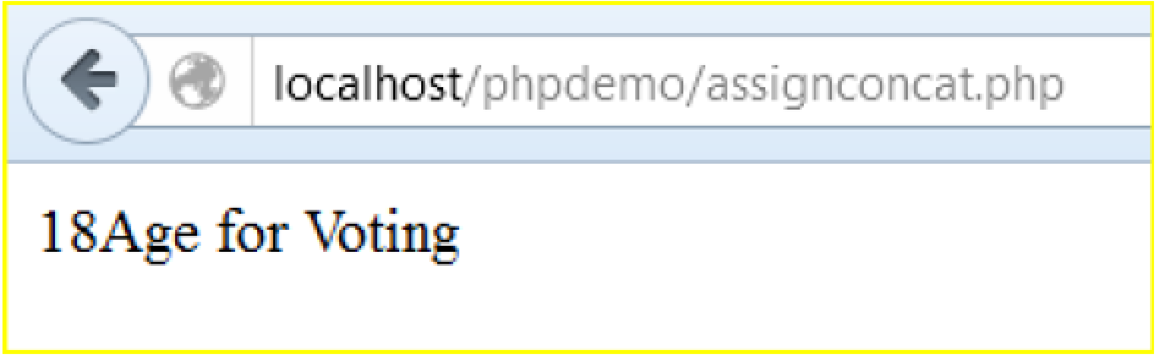
&&	and	x=6
		y=3
		(x < 10 && y > 1) returns true

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

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

Assignment and Concatenation operator

```
<html>
<body>
<?php
$age=18;

echo $age.'Age for Voting';
?>
</body>
</html>
```



localhost/phpdemo/assignconcat.php

```
18Age for Voting
```

Comparison Operator – html part

```
<html><head>
<title>Using Comparison Operators</title></head>
<body>
<form action=comparisonop.php method="post">
<p> Enter the first input as a number <input type="text"
   name="input1" size=10><br>
 Enter the Second input as a number <input type="text"
   name="input2" size=10><br>
 <p>Calculation</p>
<input type ="radio" name="op" value="ADD"> Add<br>
```

Comparison Operator – html part

```
<input type ="radio" name="op" value="SUBTRACT">
   Subtract<br>
<input type ="radio" name="op" value="MULTIPLY">
   Multiply<br>
<input type ="radio" name="op" value="DIVISION">
   Division<br>
<input type ="radio" name="op" value="MOD">
   Mod<br>
<input type ="submit" name="Submit"
   value="Calculate"></p> </form>
</body>
</html>
```

Comparison Operator – php part

```
<html> <head>
<title>Using Comparison Operators</title>
</head>
<body>
<?php
if ($_POST[op]=="ADD")
{
$result=$_POST[input1]+$_POST[input2];
} else if ($_POST[op]=="SUBTRACT")
{
$result=$_POST[input1]-$_POST[input2];
}
```

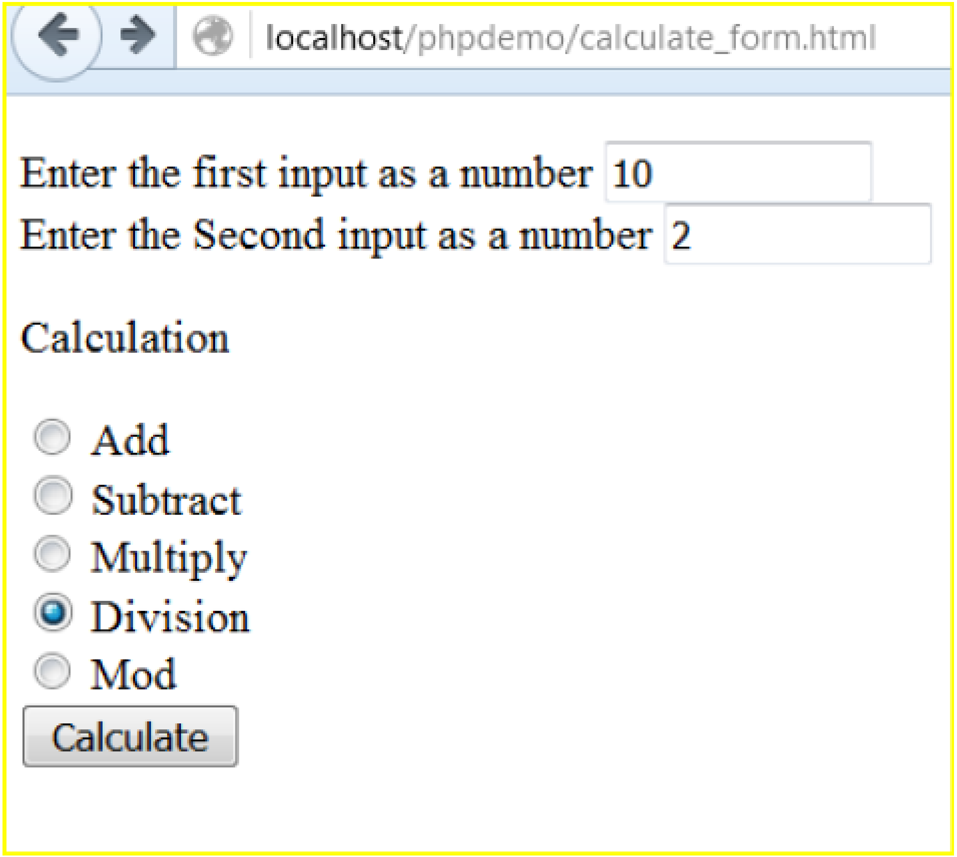
Comparison Operator – php part

```
else if ($_POST[op]=="MULTIPLY")
{
$result=$_POST[input1]*$_POST[input2];
}
else if ($_POST[op]=="DIVISION")
{
$result=$_POST[input1]/$_POST[input2];
}
```

Comparison Operator – php part

```
else if ($_POST[op]=="MOD")
{
$result=$_POST[input1]%$_POST[input2];
}
echo 'The result is'.$result;
?>
</body>
</html>
```

Comparison Operators



localhost/phpdemo/calculate_form.html

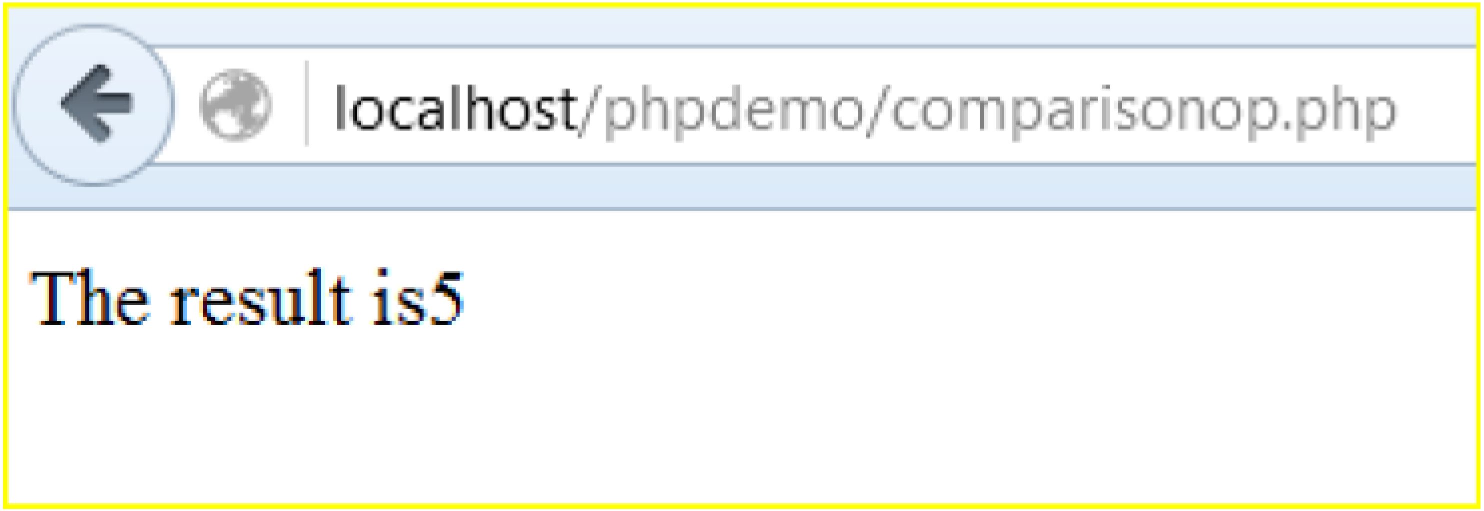
Enter the first input as a number 10

Enter the Second input as a number 2

Calculation

- Add
- Subtract
- Multiply
- [✓ Division]
- Mod

Comparison Operator



```
The result is5
```

11. PHP Conditional Statements

- **If else Statement**

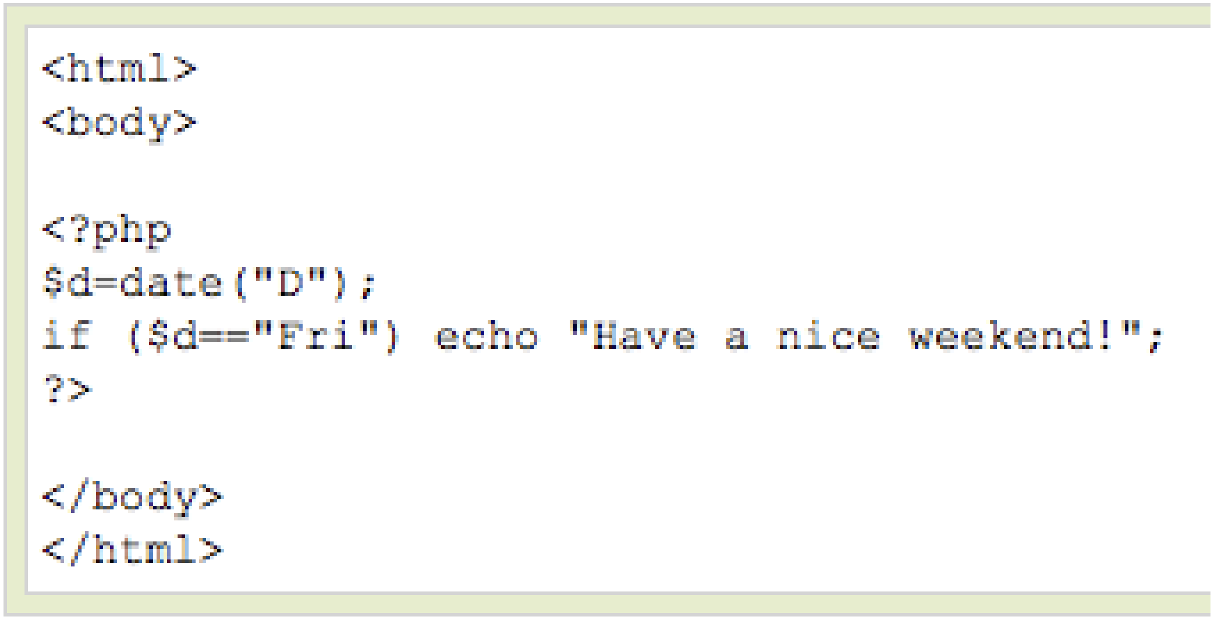
```
if(condn) {….}
elseif(condn) {….}
else { ….}
```

- **Switch case Statement**

```
switch(var)
{
case c1: statements;break
.
Default: statements; break;
}
```

PHP Conditional Statements

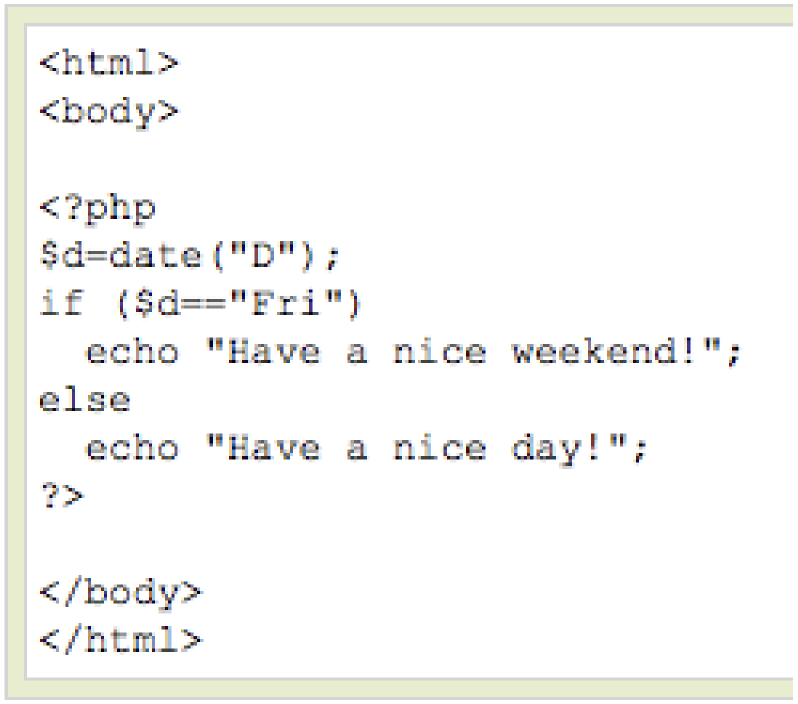
The following example will output "Have a nice weekend!" if the current day is Friday:



```
<html>
<body>

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

</body>
</html>
```

PHP Conditional Statements

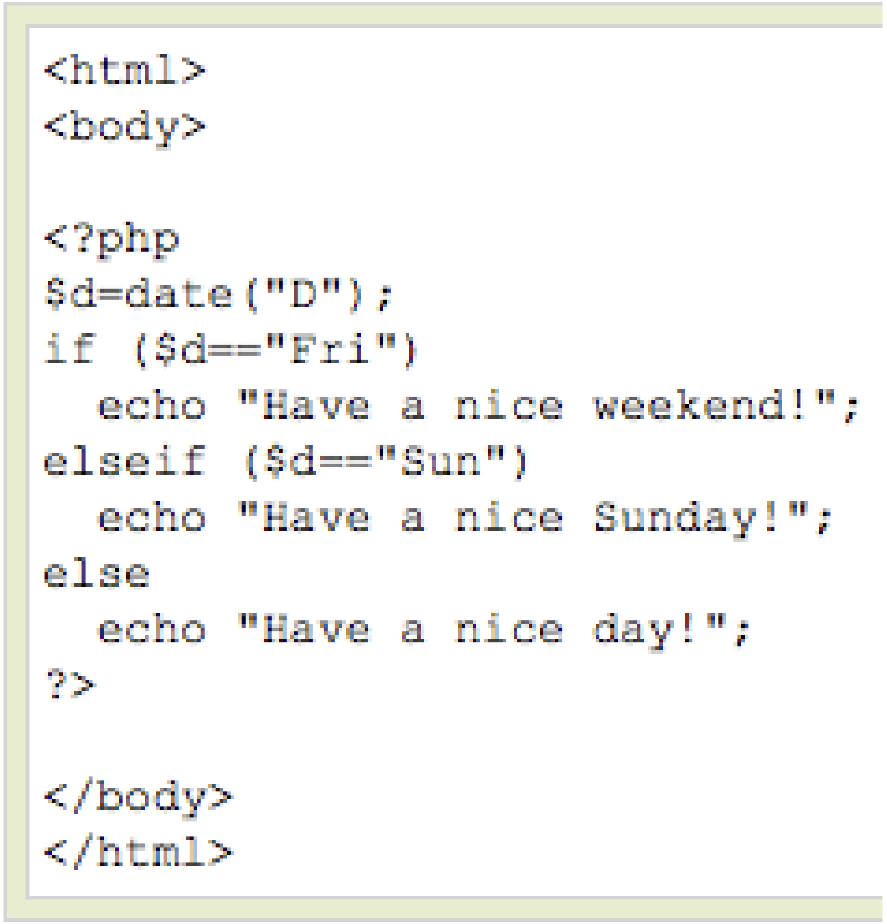
Use the **if....else** statement to execute some code if a condition is true and another code if a condition is false.



```
<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

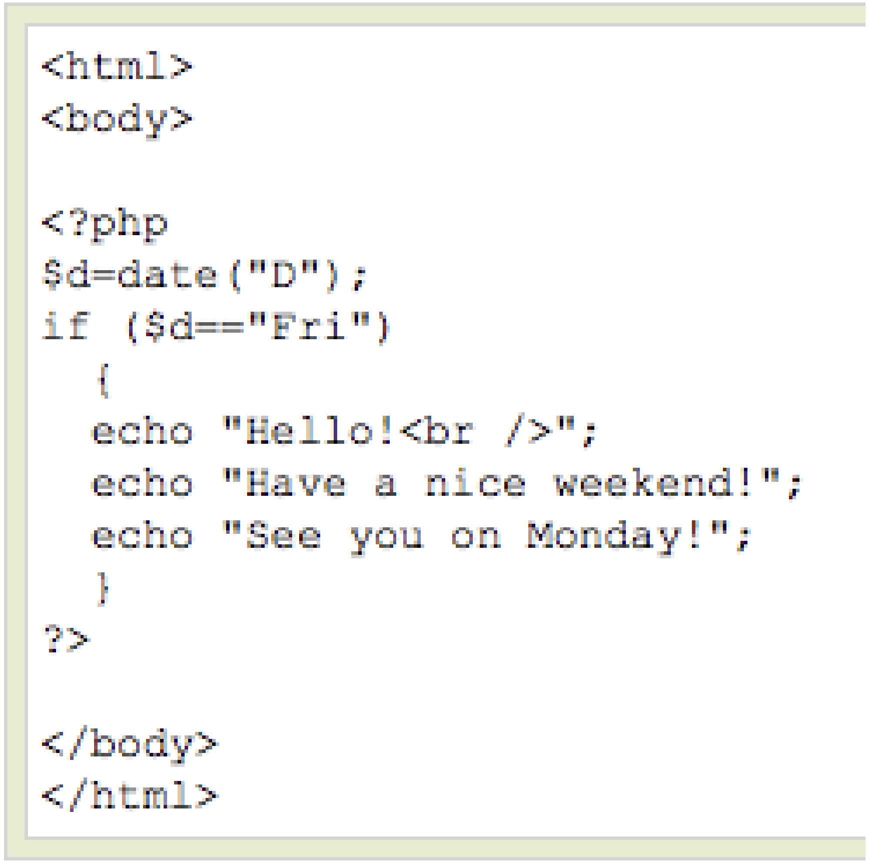
The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":



```
<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

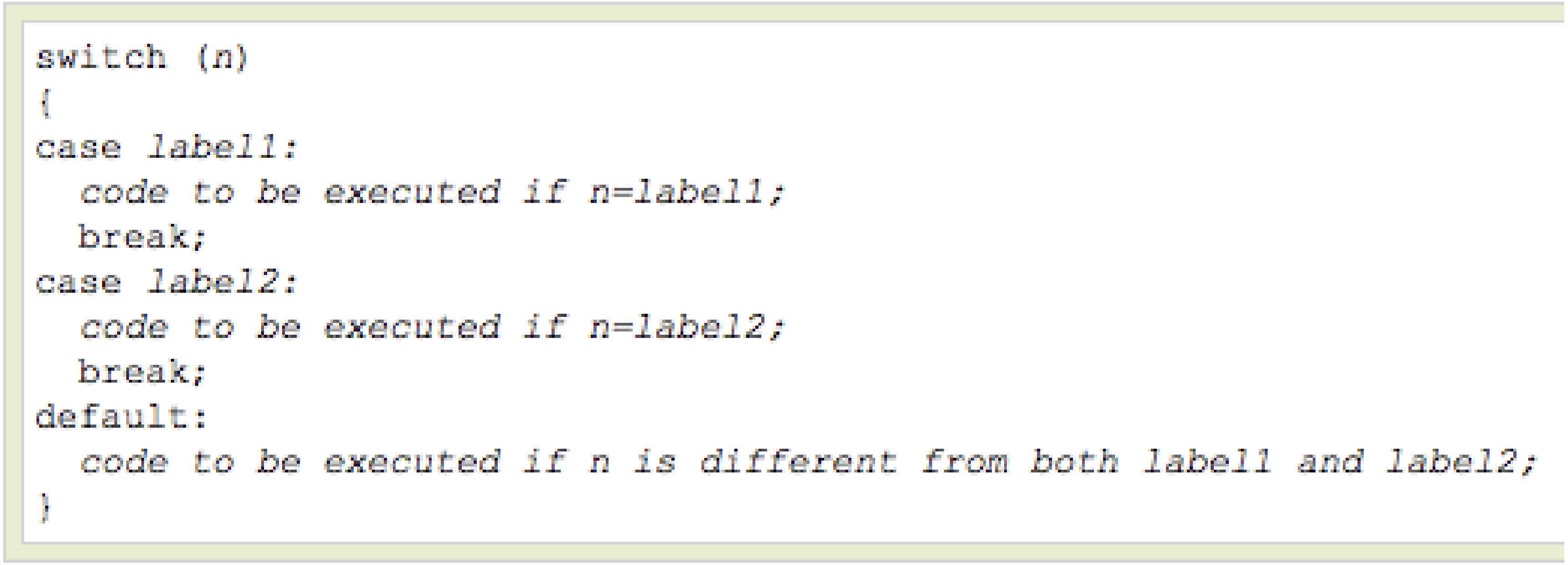
If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces { }



```
<html>
<body>

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

</body>
</html>
```

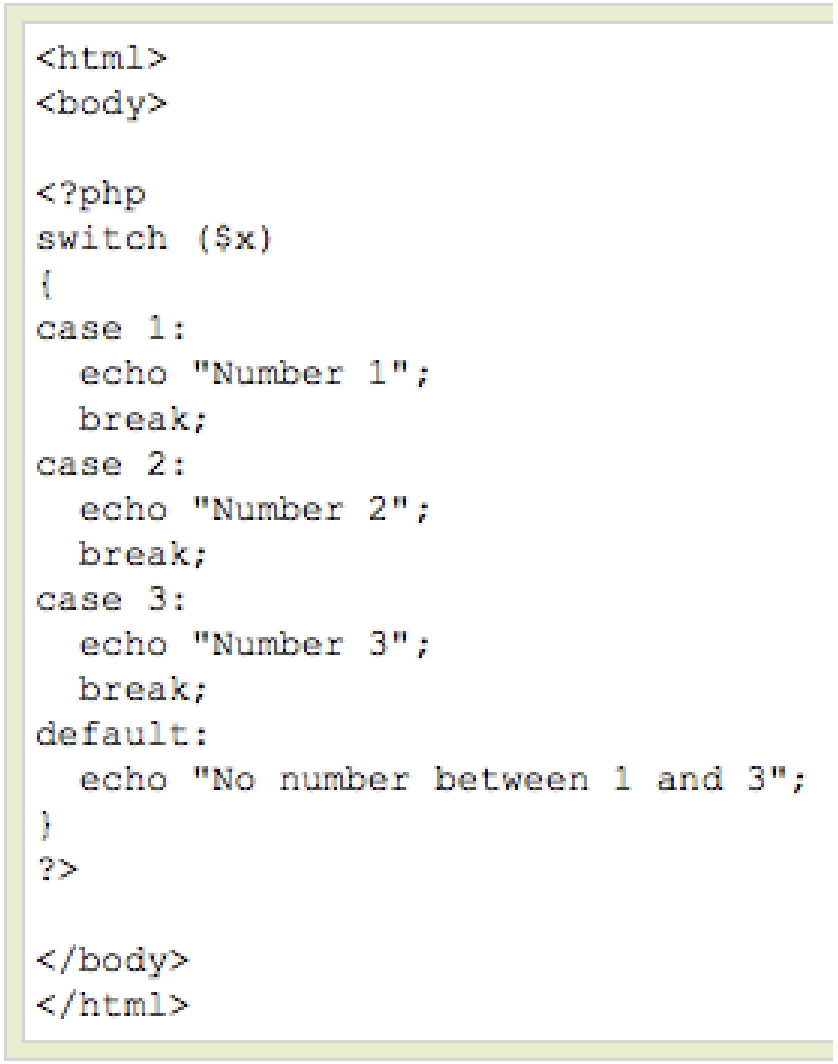
PHP Conditional Statements

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



```
<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>
```

12. Looping Statements

- For

  for(intialisation;condition;increment/decrement)

  { statements }

- While

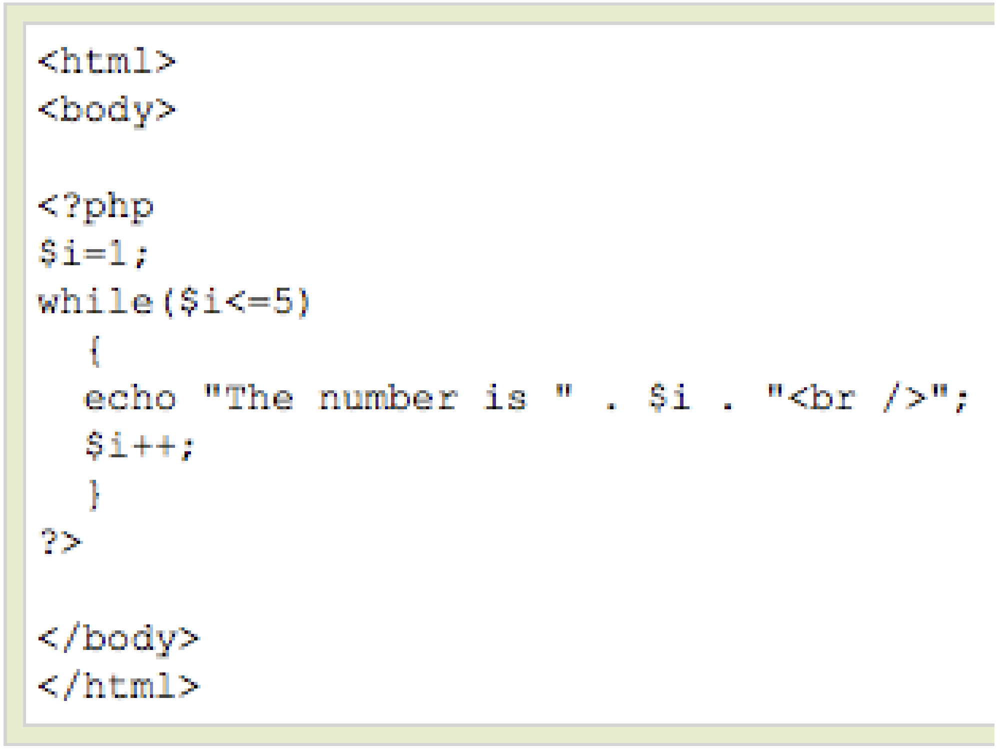
  while(condn) { ….}

- Do While

  do {….} while(condn);

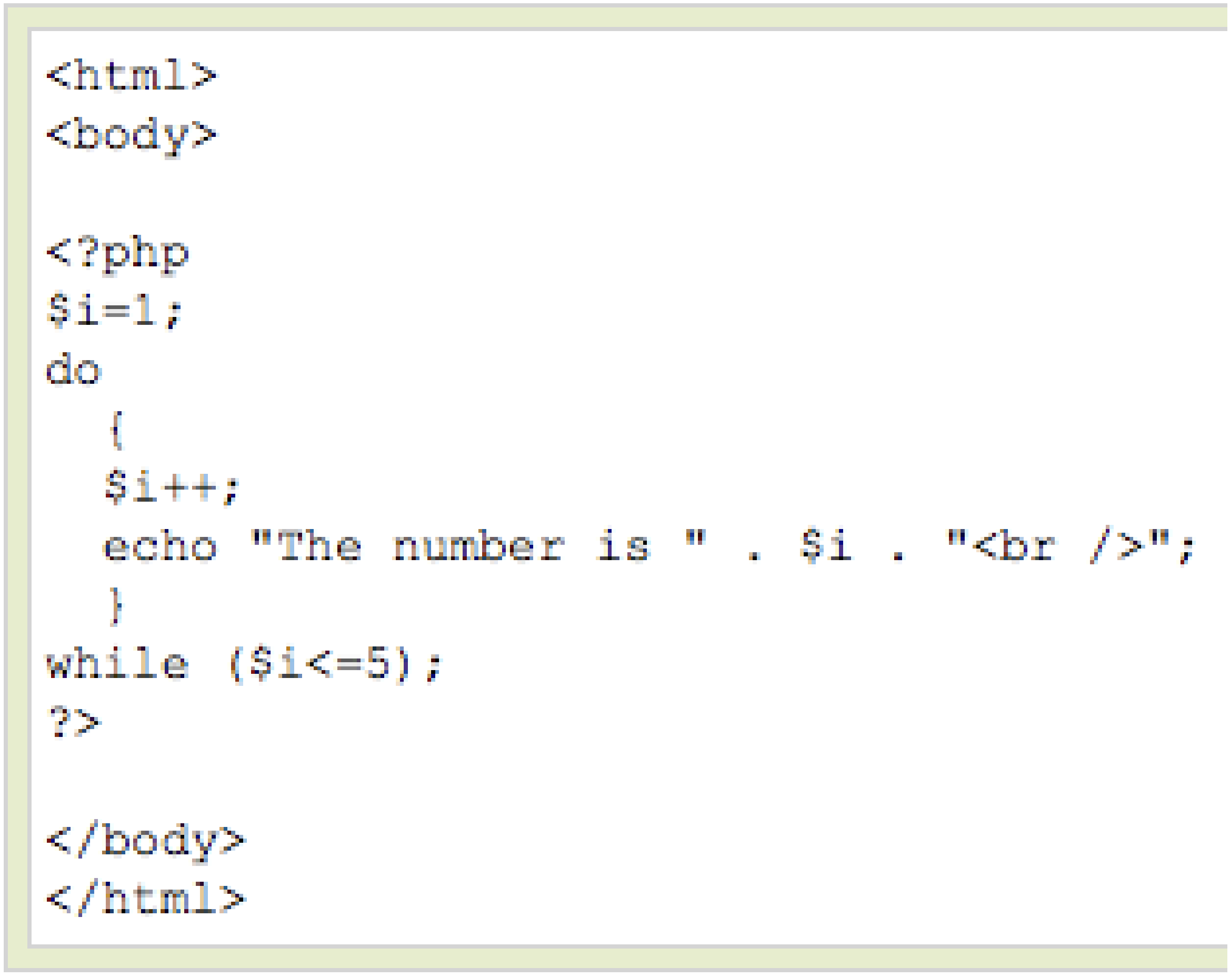
PHP Loops - While

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:



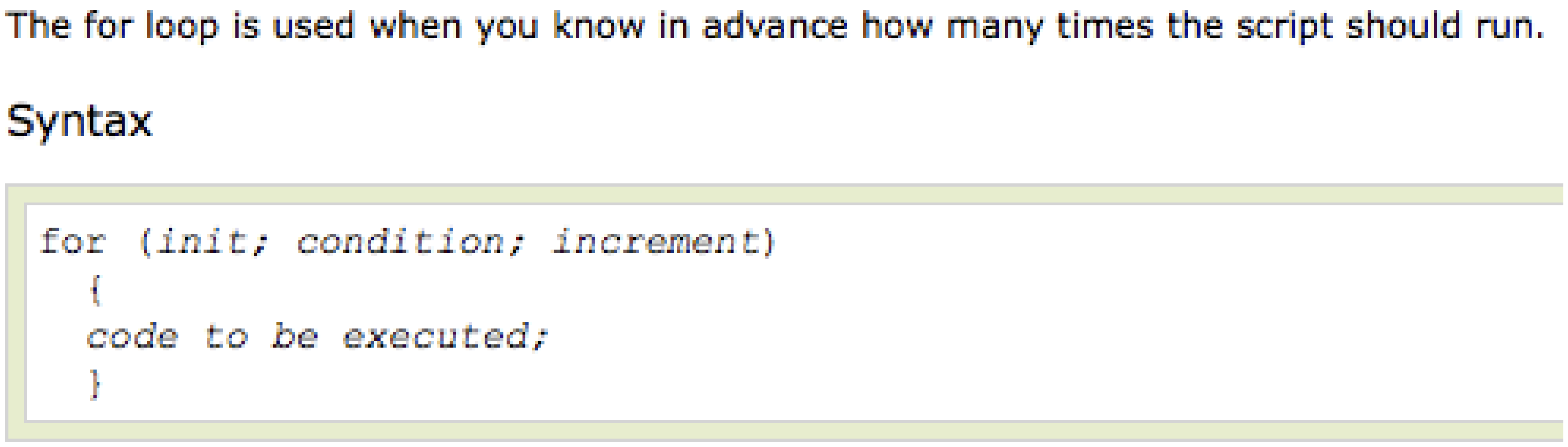
```
<html>
<body>

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

</body>
</html>
```

PHP Loops – Do ... While



```
<html>
<body>

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

</body>
</html>
```

PHP Loops - For



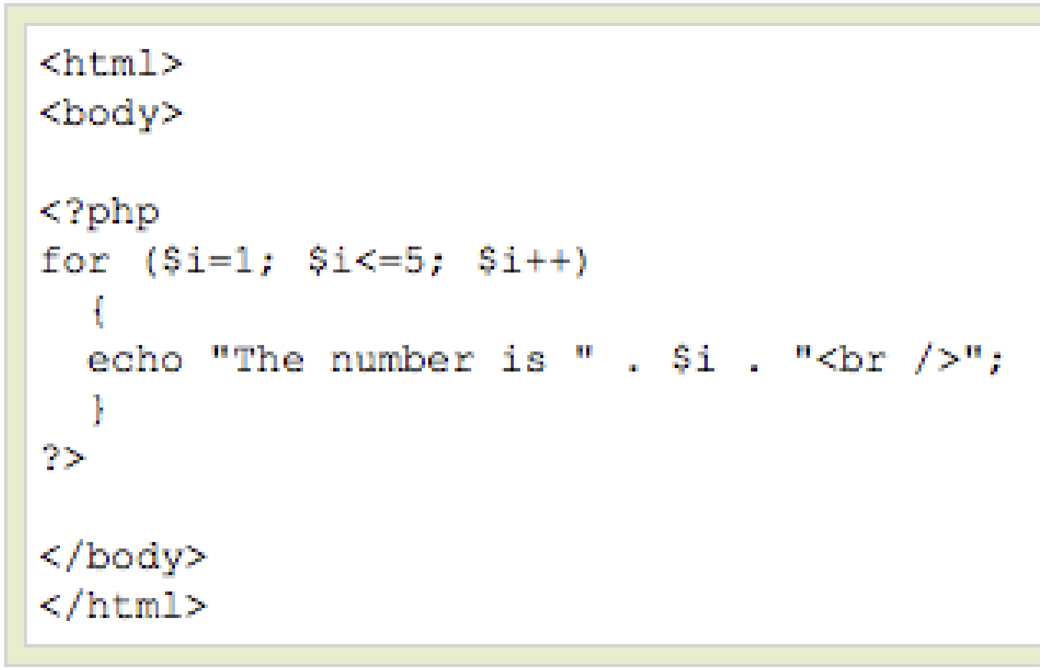
The for loop is used when you know in advance how many times the script should run.

Syntax

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

PHP Loops - For

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>
```

Thank you!