CS100: Software Tools & Technologies Lab I

Introduction to PHP

Vishwesh Jatala

Assistant Professor

Department of EECS

Indian Institute of Technology Bhilai

vishwesh@iitbhilai.ac.in

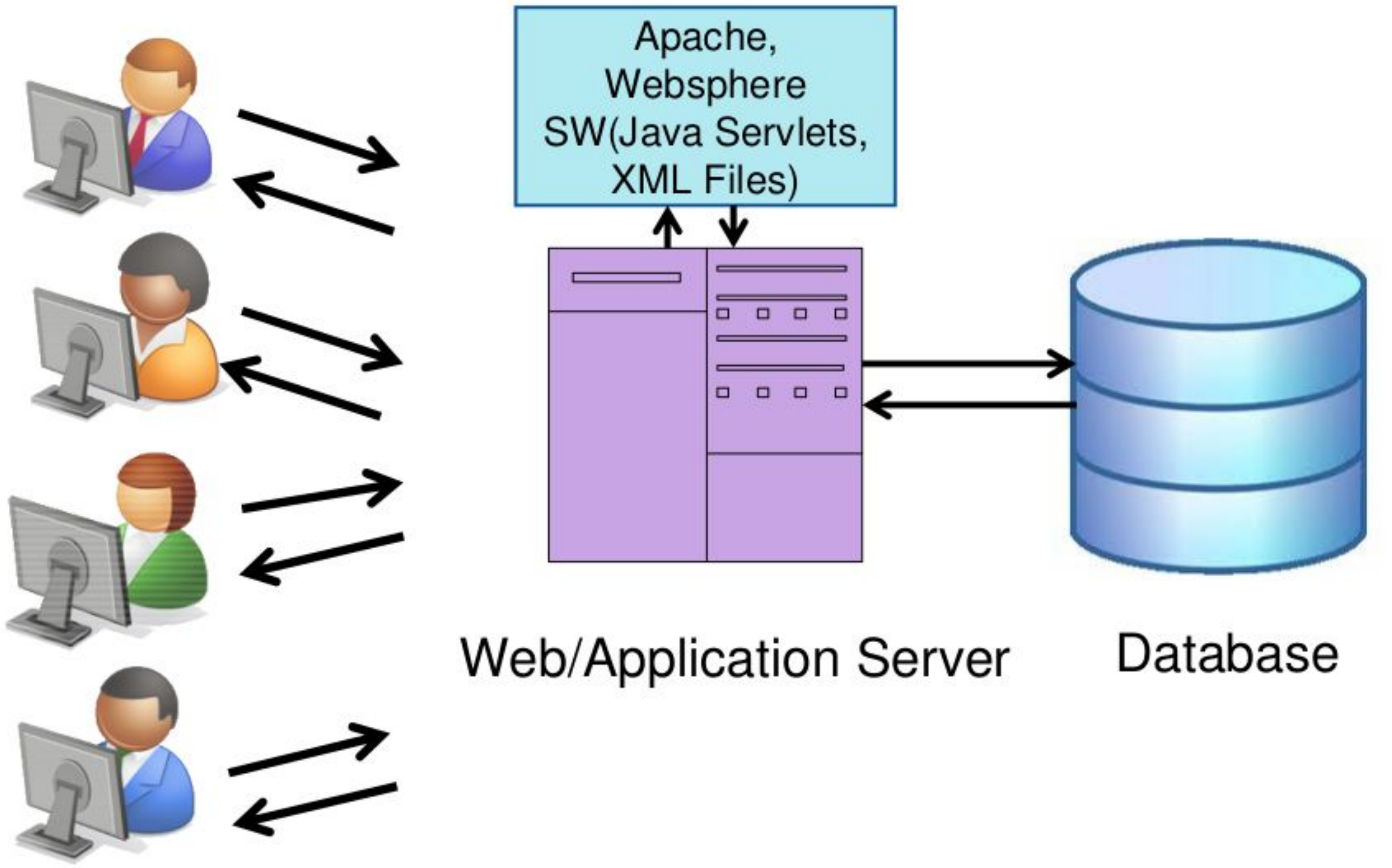
2022-23 W

URLs and web servers

```
http://server/path/file
```

- Usually when you type a URL in your browser:
  - Your computer looks up the server's IP address using DNS
  - Your browser connects to that IP address and requests the given file
  - The web server software (e.g. Apache) grabs that file from the server's local file system
  - The server sends back its contents to you

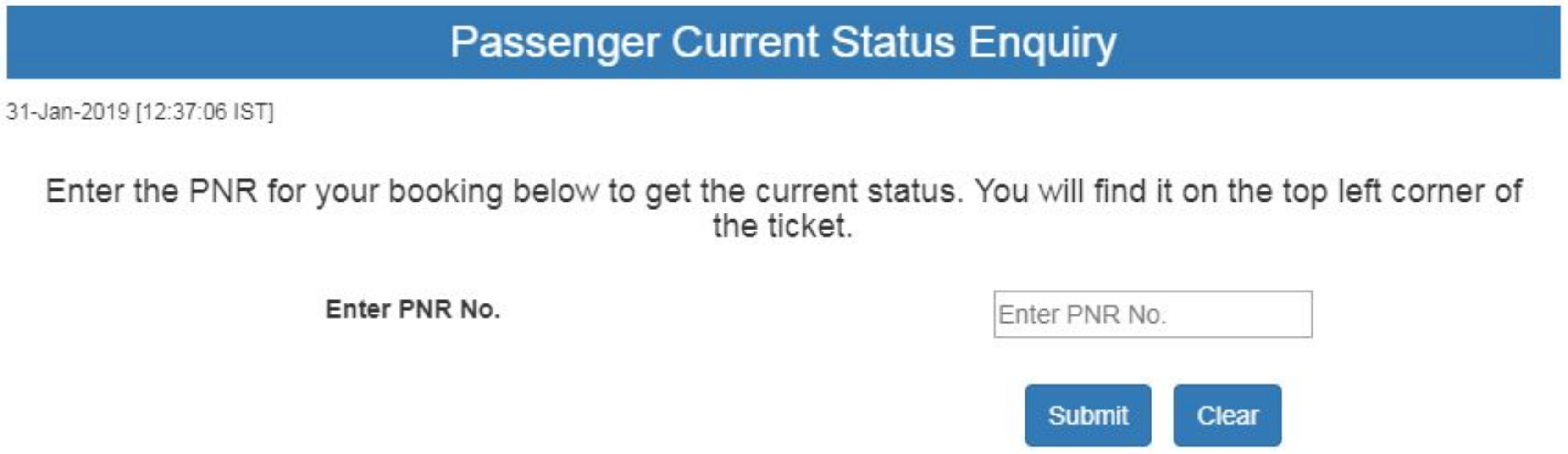
URLs and web servers



URLs and web servers

- http://www.facebook.com/home.php
- Some URLs actually specify programs that the web server should run, and then send their output back to you as the result:
  - The above URL tells the server facebook.com to run the program home.php and send back its output

URLs and web servers

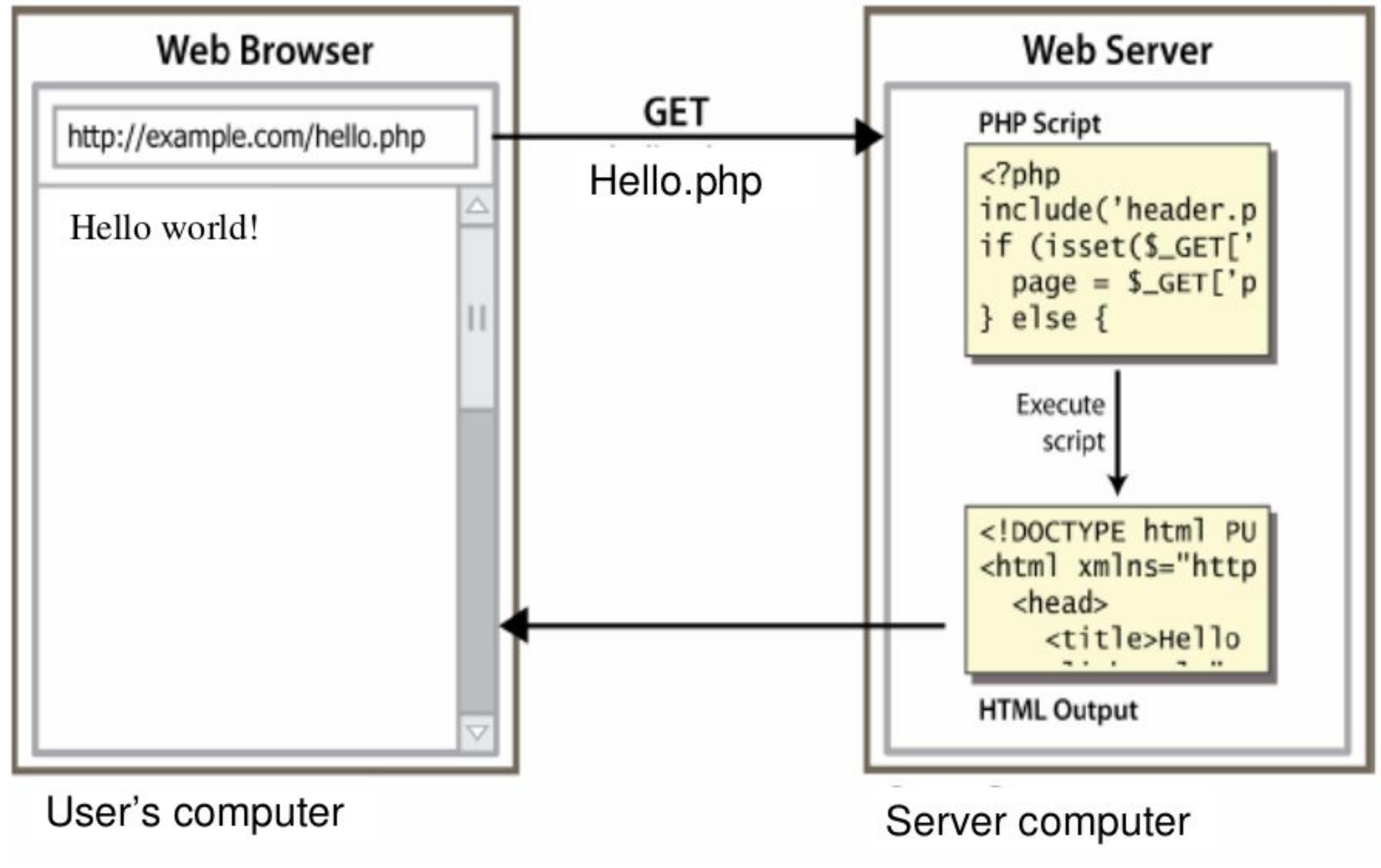

Passenger Current Status Enquiry

31-Jan-2019 [12:37:06 IST]

Enter the PNR for your booking below to get the current status. You will find it on the top left corner of the ticket.

Enter PNR No. Enter PNR No.

URLs and web servers



Server-Side web programming

- Webserver
  - Contains software that allows it to run server side programs
  - Sends back their output as responses to web requests

Server-Side web programming

- Server side scripting:
  - Dynamically edit, change or add any content to a Web page
  - Respond to user queries or data submitted from HTML forms
  - Access any data or databases and return the results to a browser
  - Customize a Web page to make it more useful for individual users

Server-Side web programming

- Server-side pages are programs written using one of many web programming languages/frameworks
  - PHP, Python, Java/JSP, ASP.NET,, Ruby
- Each language/framework has its pros and cons
  - we use PHP

Why PHP?

- Free and open source
- Compatible with many servers
- Simple
- PHP runs on different platforms (Windows, Linux, Unix, etc.)

What is PHP?

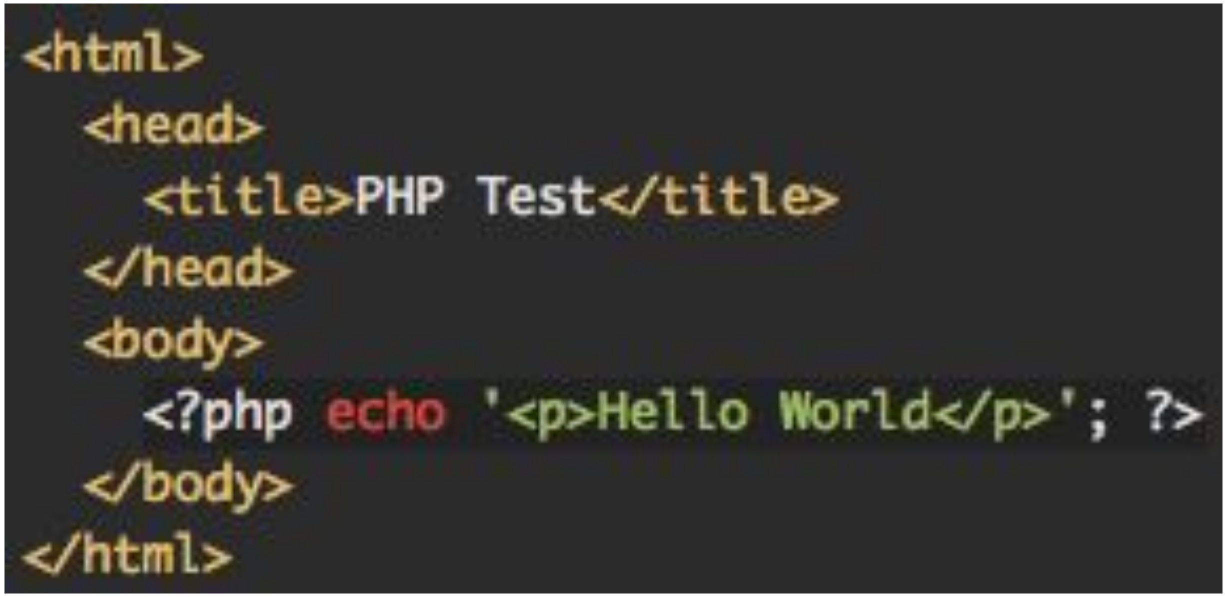
- PHP is a recursive acronym for “PHP: Hypertext Preprocessor”
- Server-side scripting language
- Used to make web pages dynamic:
  - provide different content depending on context
  - interface with other services: database, e-mail, etc.
  - authenticate users process form information
-

PHP Syntax Template

```
HTML content
<?php
PHP code
?>
HTML content
<?php
PHP code
?>
HTML content ...                                          PHP
```

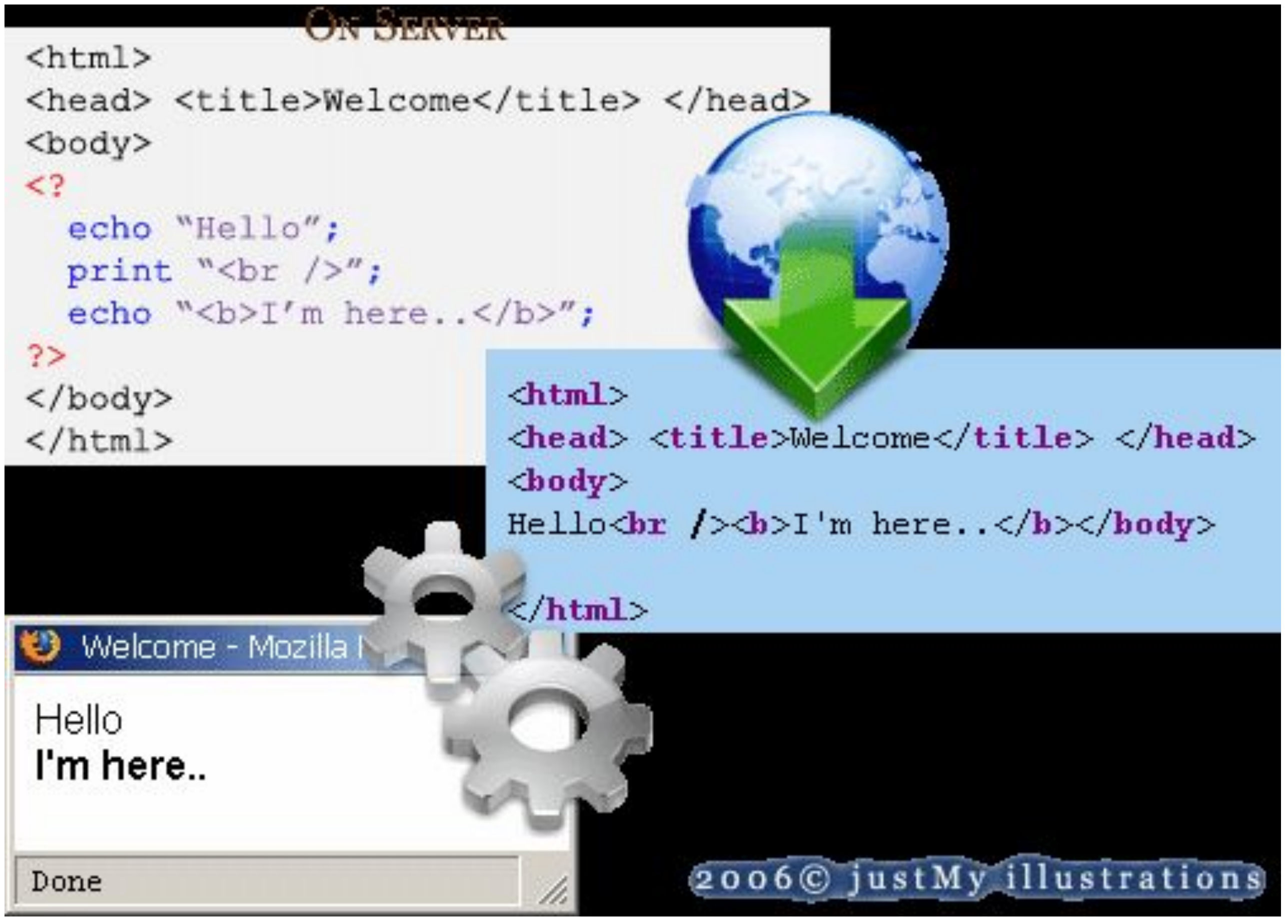
- Contents of a .php file between <?php and ?> are executed as PHP code
- All other contents are output as pure HTML
- We can switch back and forth between HTML and PHP "modes"

Simple Example



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

Simple Example



```
<html>
<head> <title>Welcome</title> </head>
<body>
<?
  echo "Hello";
  print "<br />";
  echo "<b>I'm here..</b>";
?>
</body>
</html>
```

```
<html>
<head> <title>Welcome</title> </head>
<body>
Hello<br /><b>I'm here..</b></body>

</html>
```

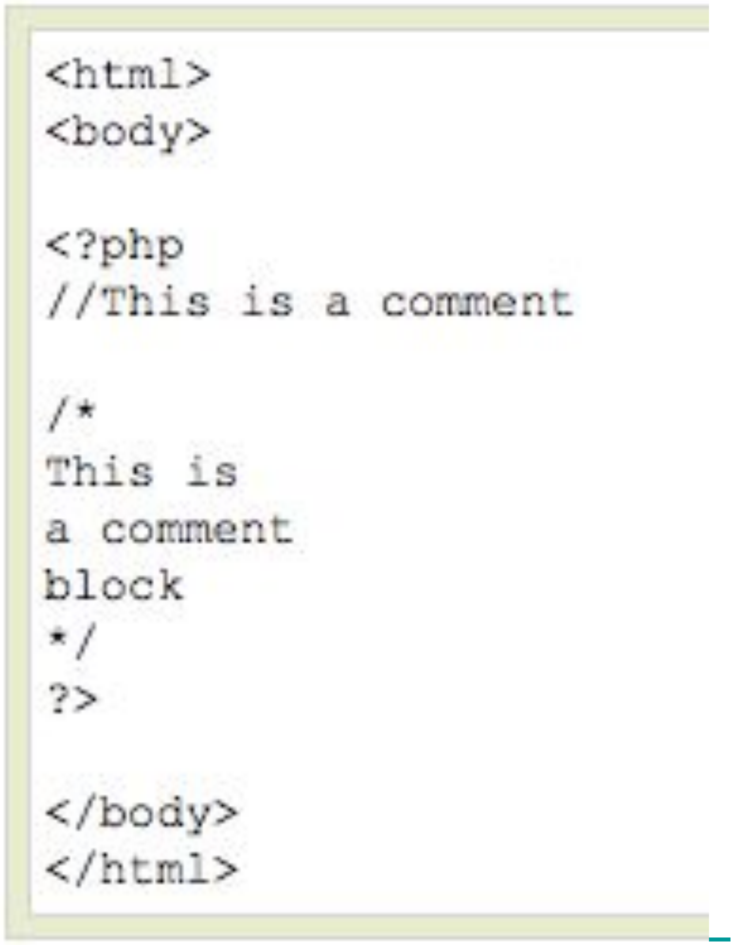
Welcome - Mozilla F

Hello
I'm here..

Done

PHP Comments

In PHP, we use **//** to make a single-line comment or **/*** and ***/** to make a large comment block.



```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

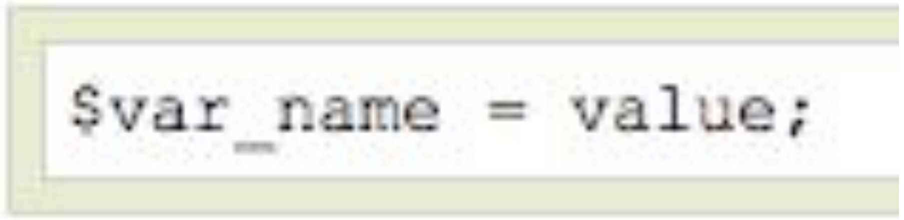
PHP Variables

> 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.

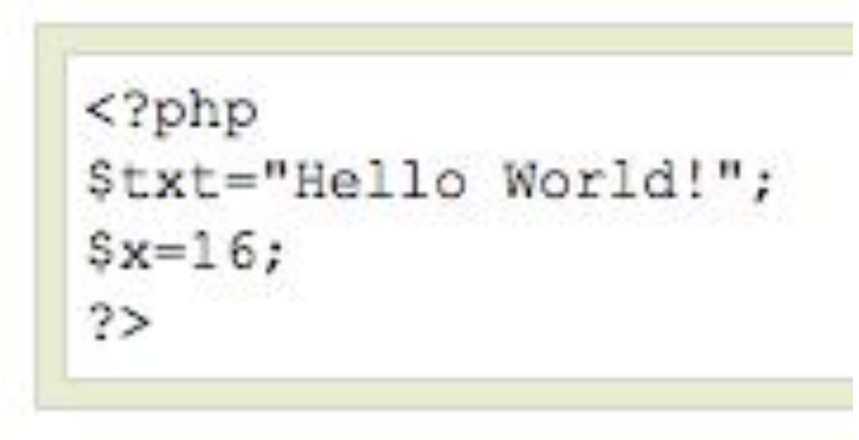
> All variables in PHP start with a $ sign symbol.

> 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.

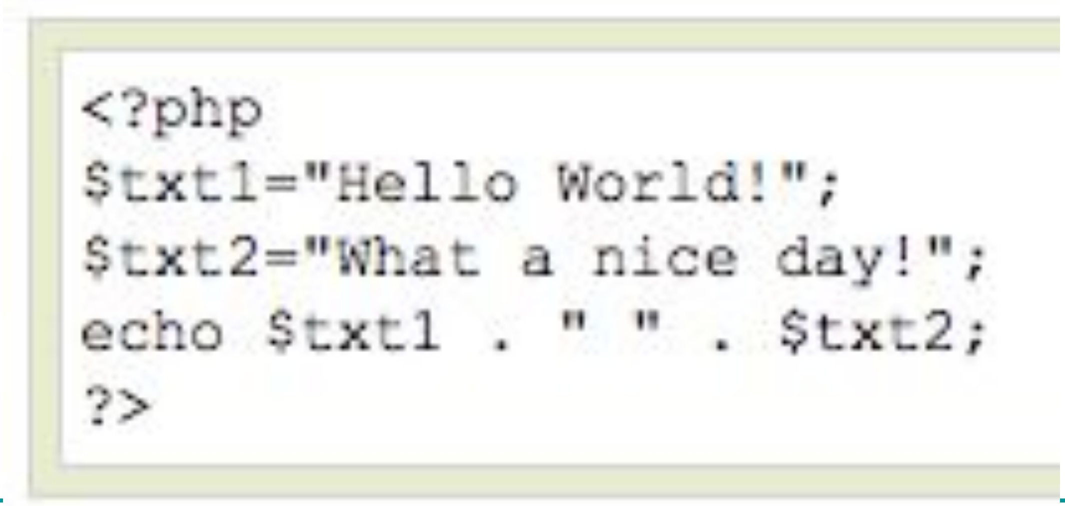
> In the example above, you see that you do not have to tell PHP which data type the variable is.

> PHP automatically converts the variable to the correct data type, depending on its value.

PHP Concatenation

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

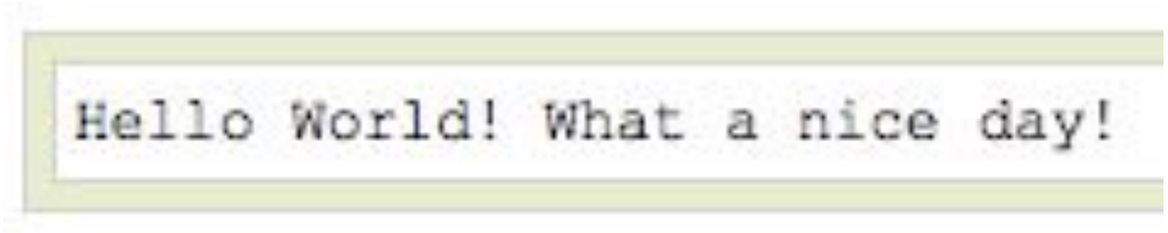
> To concatenate two string variables together, use the concatenation operator:



```
<?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!
```

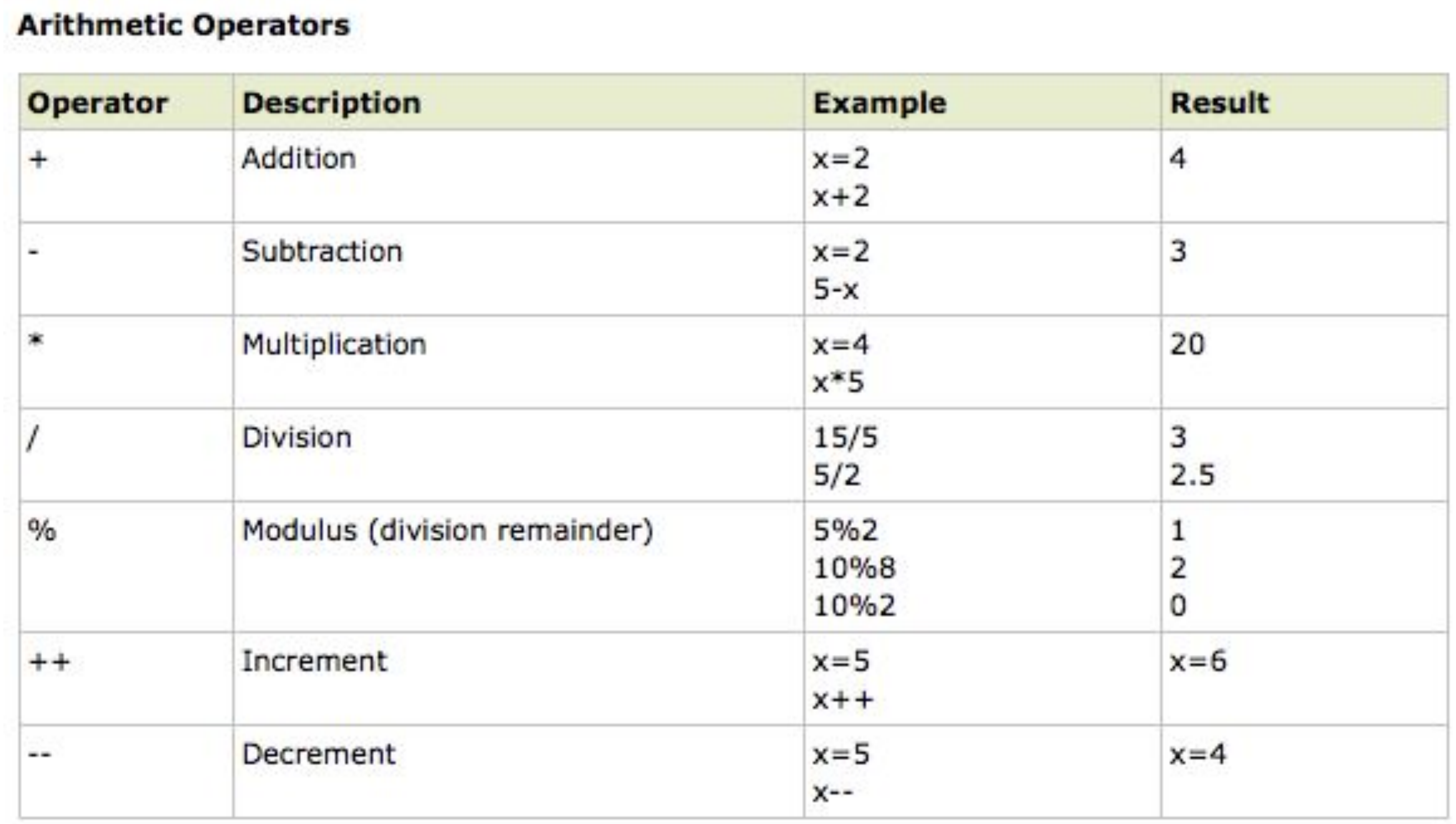
If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

PHP Operators

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

> Arithmetic
> Assignment
> Comparison
> Logical

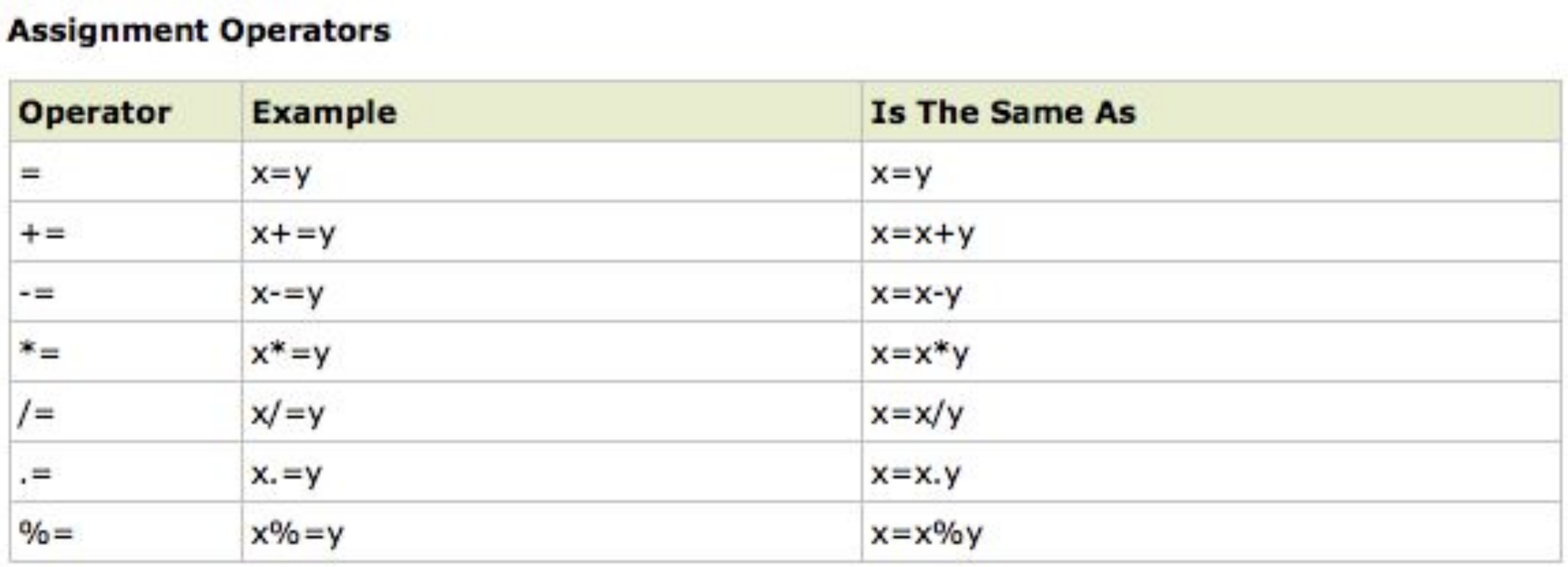
PHP Operators



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

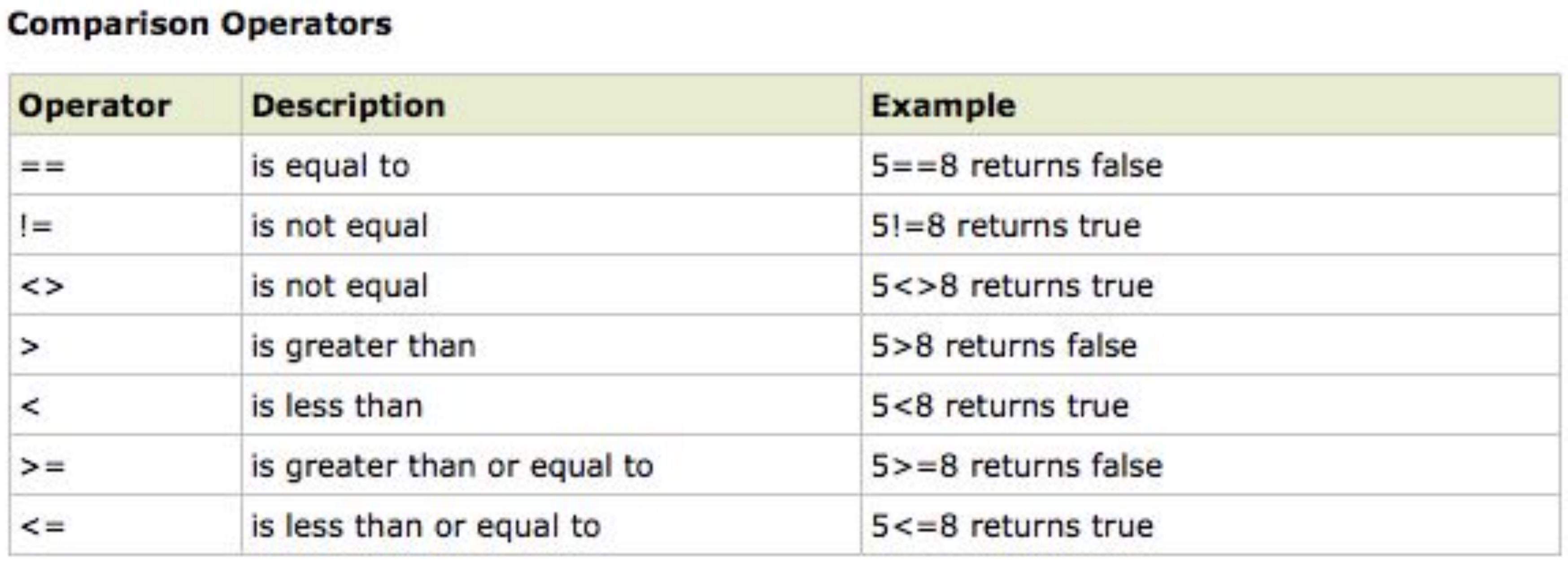
PHP Operators



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

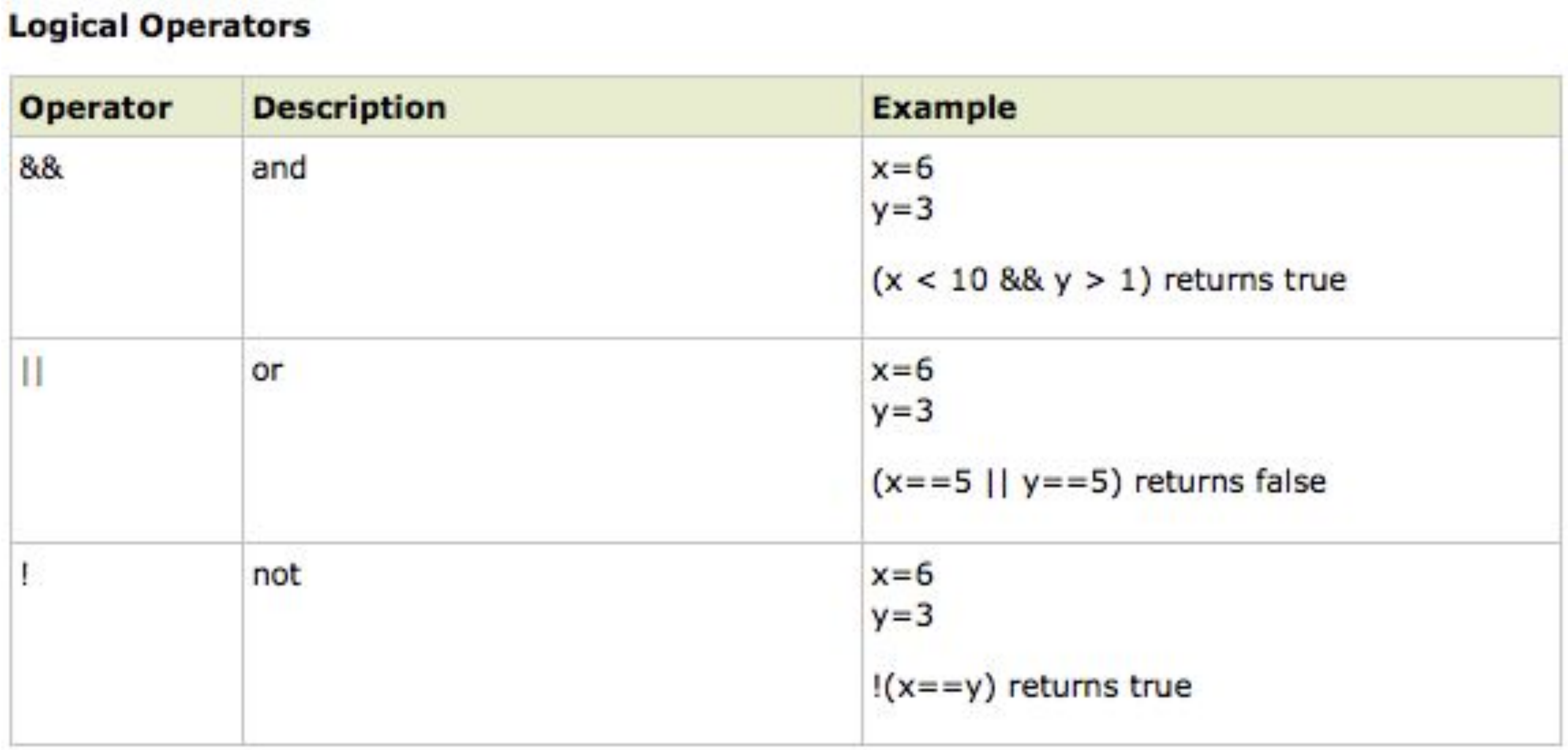
PHP Operators



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

PHP Conditional Statements

> **if** statement - use this statement to execute some code only if a specified condition is true

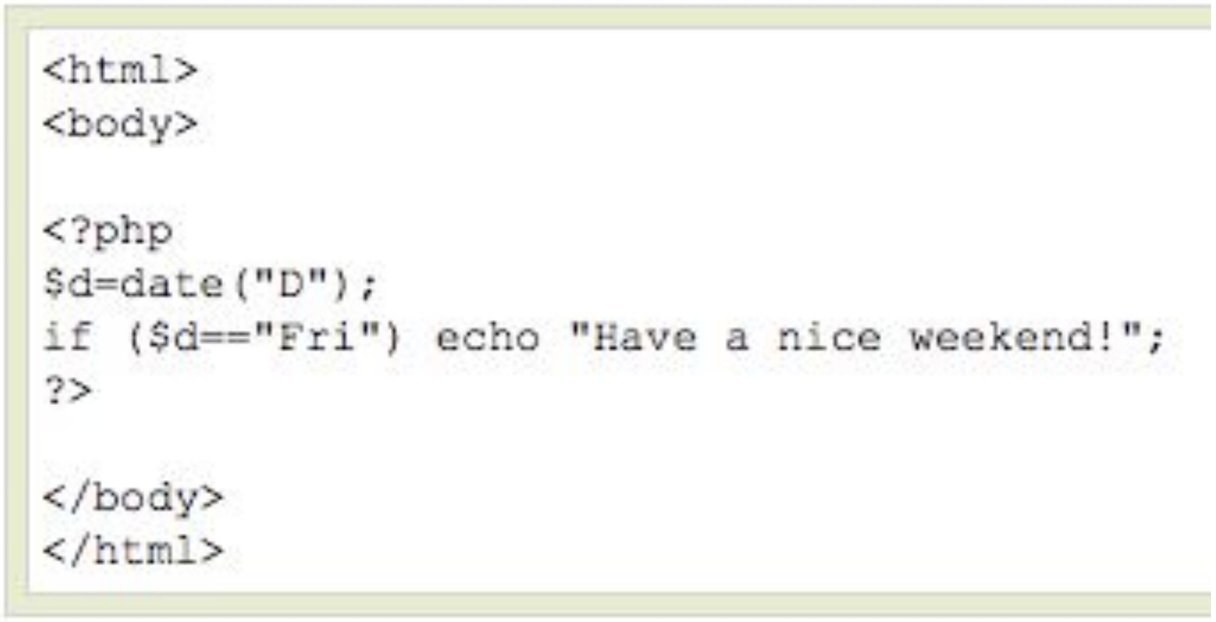
> **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false

> **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

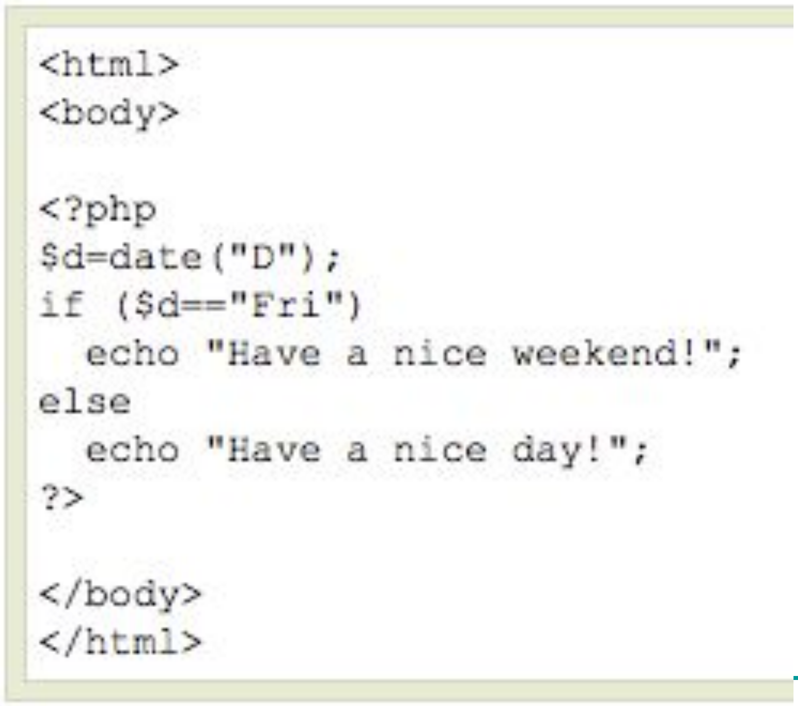
The following example will output "Have a nice weekend!" if the current day is Friday:



```
<html>
<body>

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

</body>
</html>
```

PHP Conditional Statements

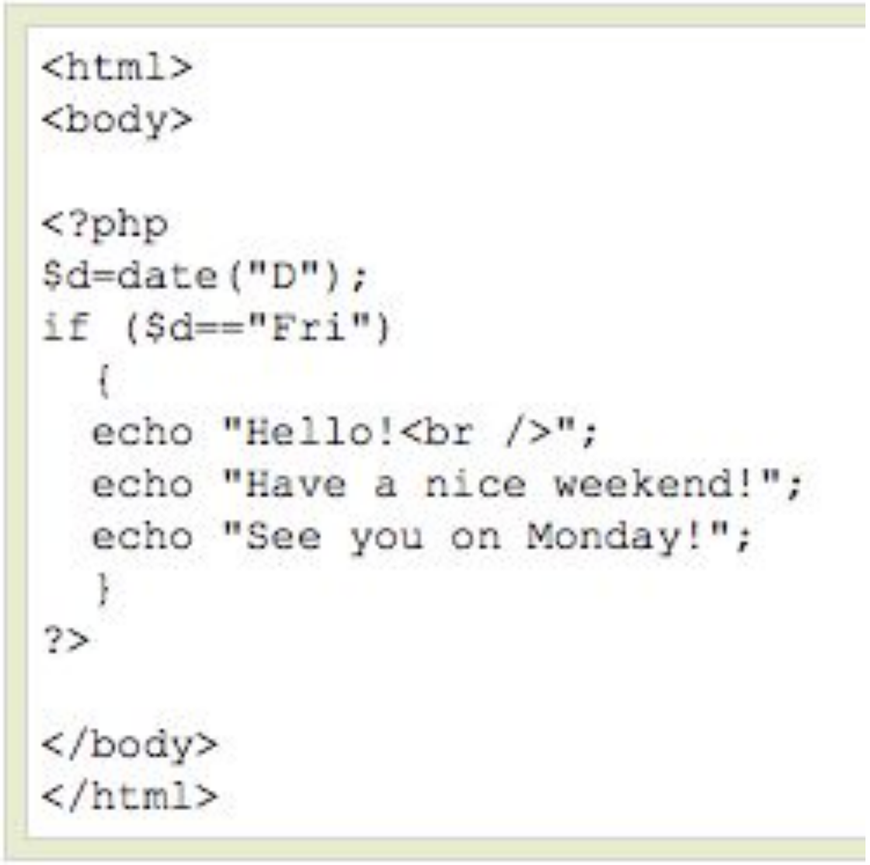
Use the **if....else** statement to execute some code if a condition is true and another code if a condition is false.



```
<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

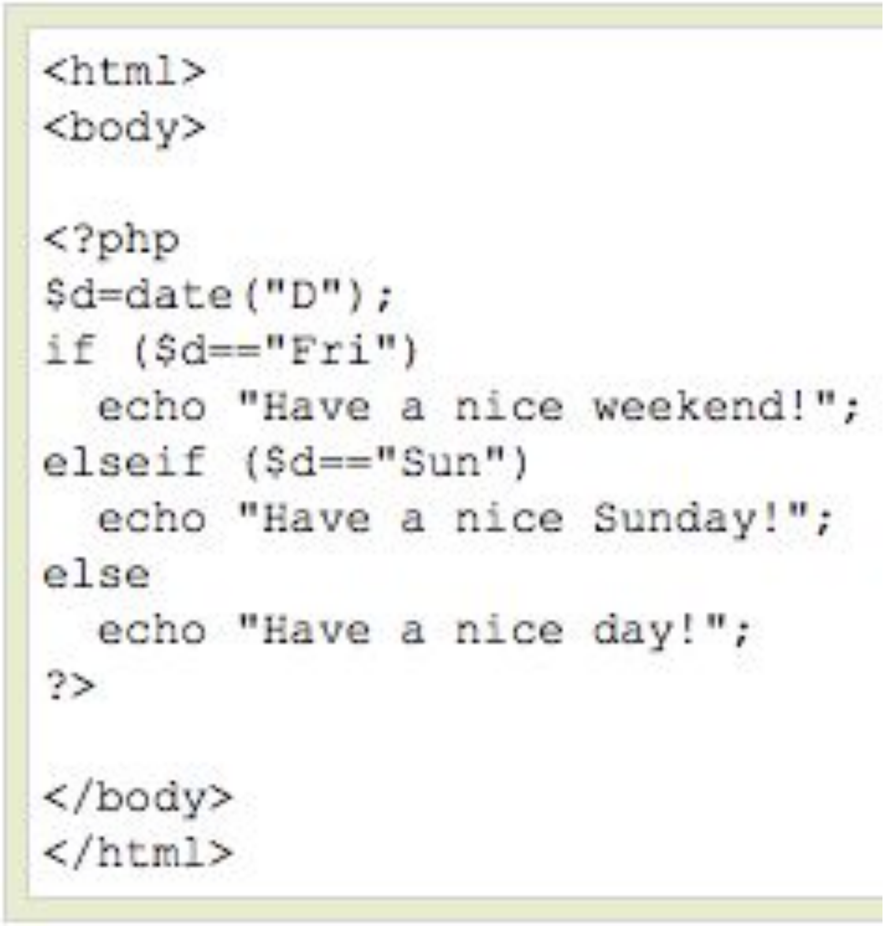
If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces **{ }**



```
<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

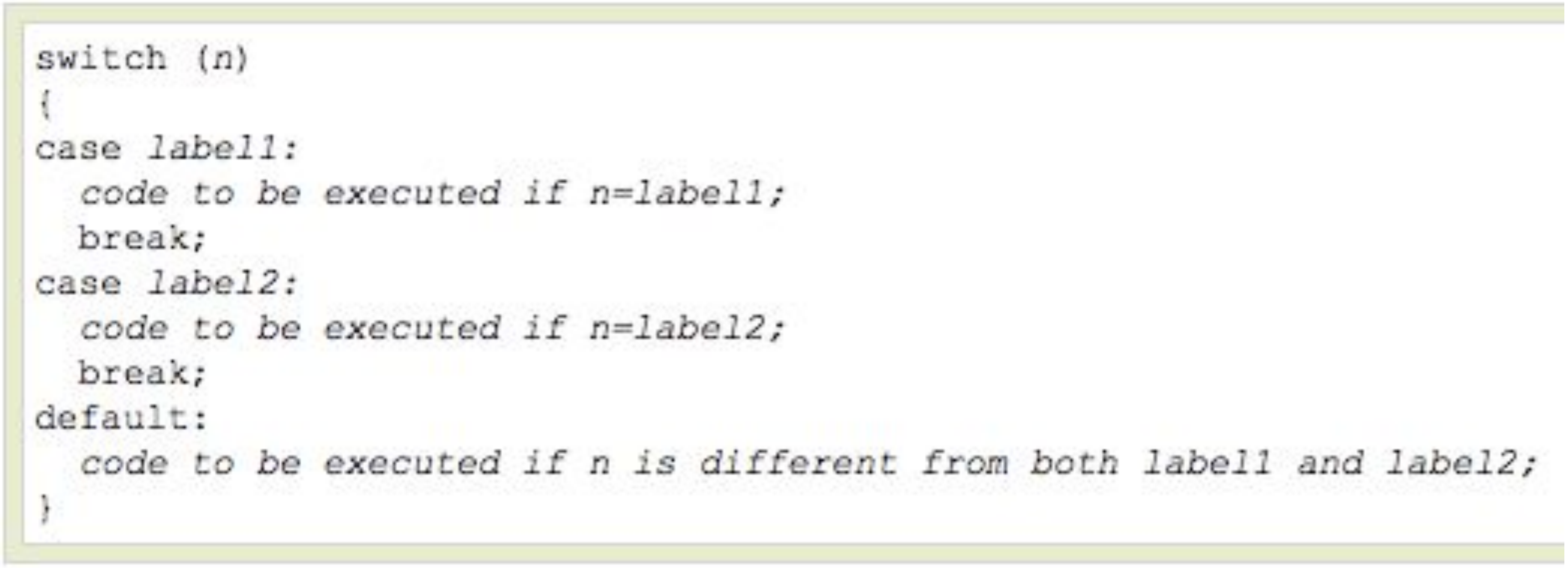
The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":



```
<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>
```

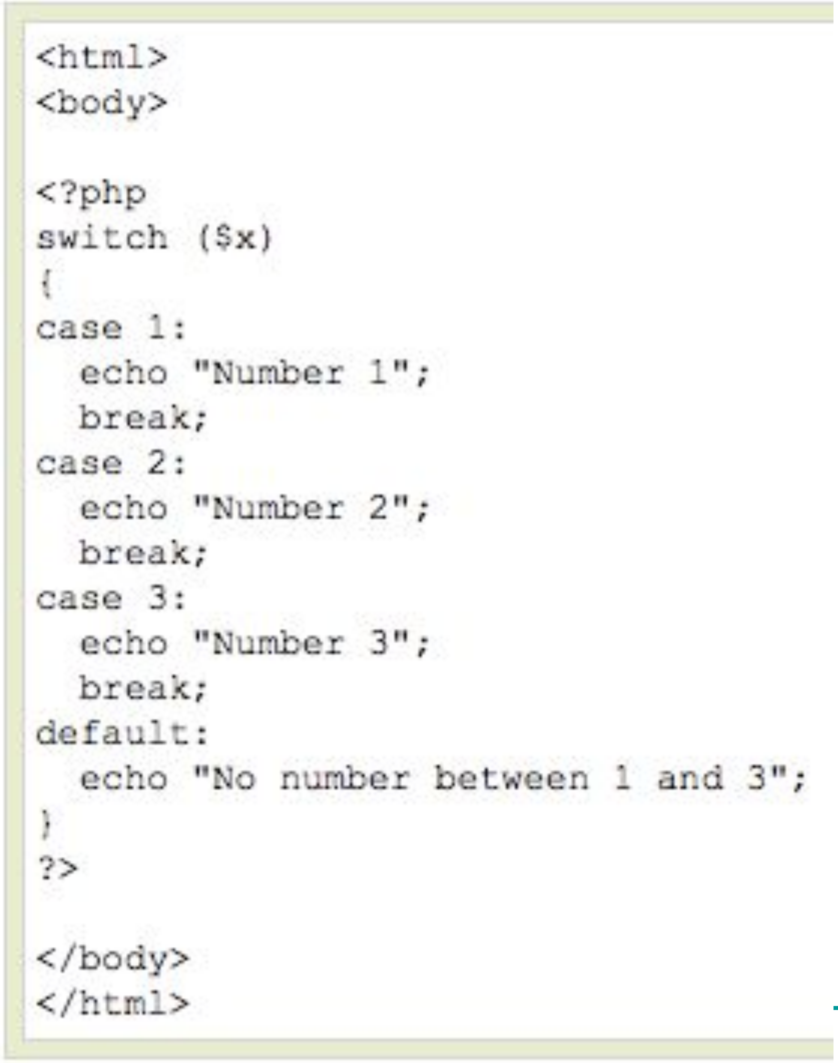
PHP Conditional Statements

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


```
<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 Arrays

> A 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

Types of arrays

> **Numeric array** - An array with a numeric index

> **Associative array** - An array where each ID key is associated with a value

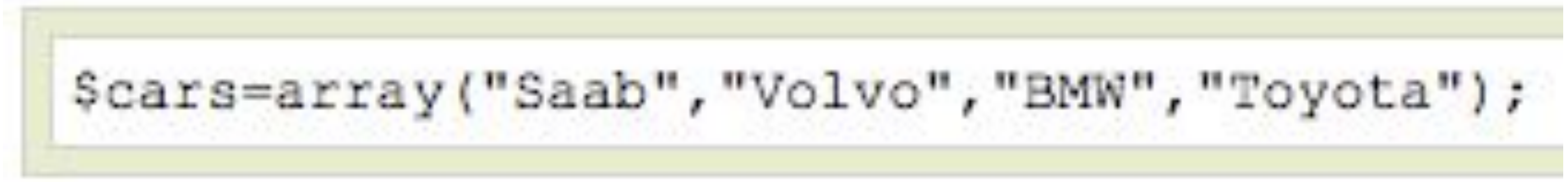
PHP Numeric Arrays

> A numeric array stores each array element with a numeric index.

> There are two methods to create a numeric array.

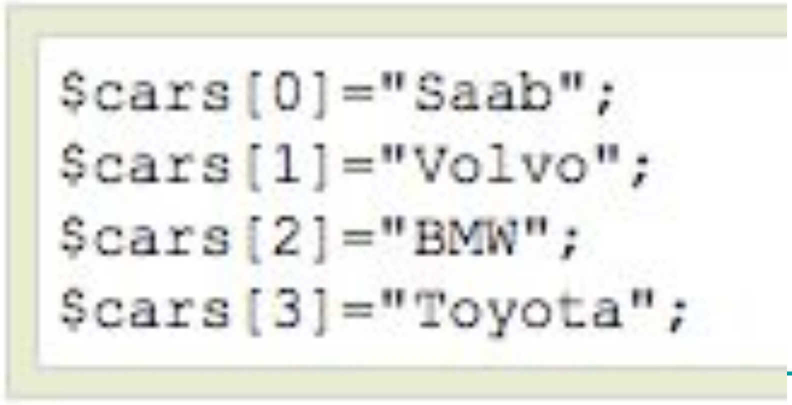
PHP Numeric Arrays

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



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

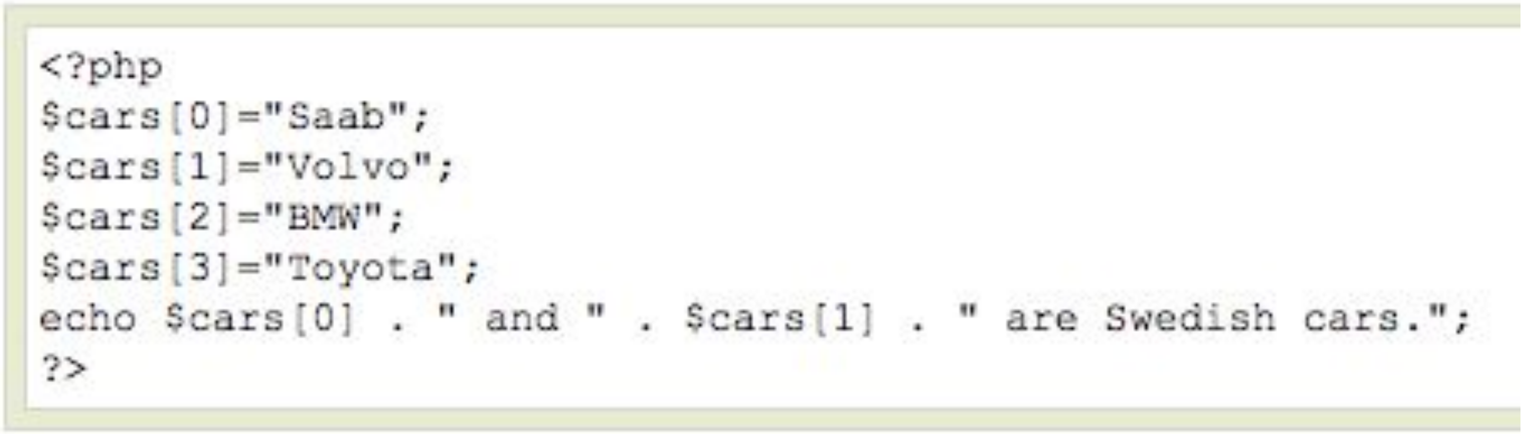
In the following example we assign the index manually:



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

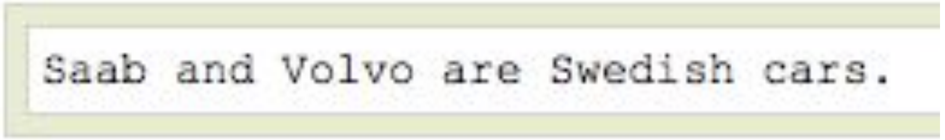
PHP Numeric Arrays

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

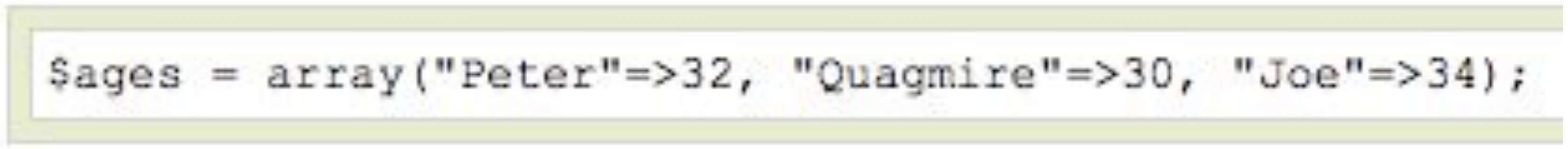
> With an associative array, each ID key is associated with a value.

> 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.

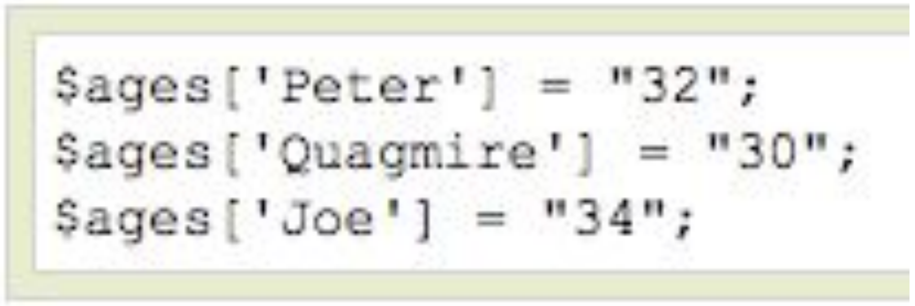
PHP Associative Arrays

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



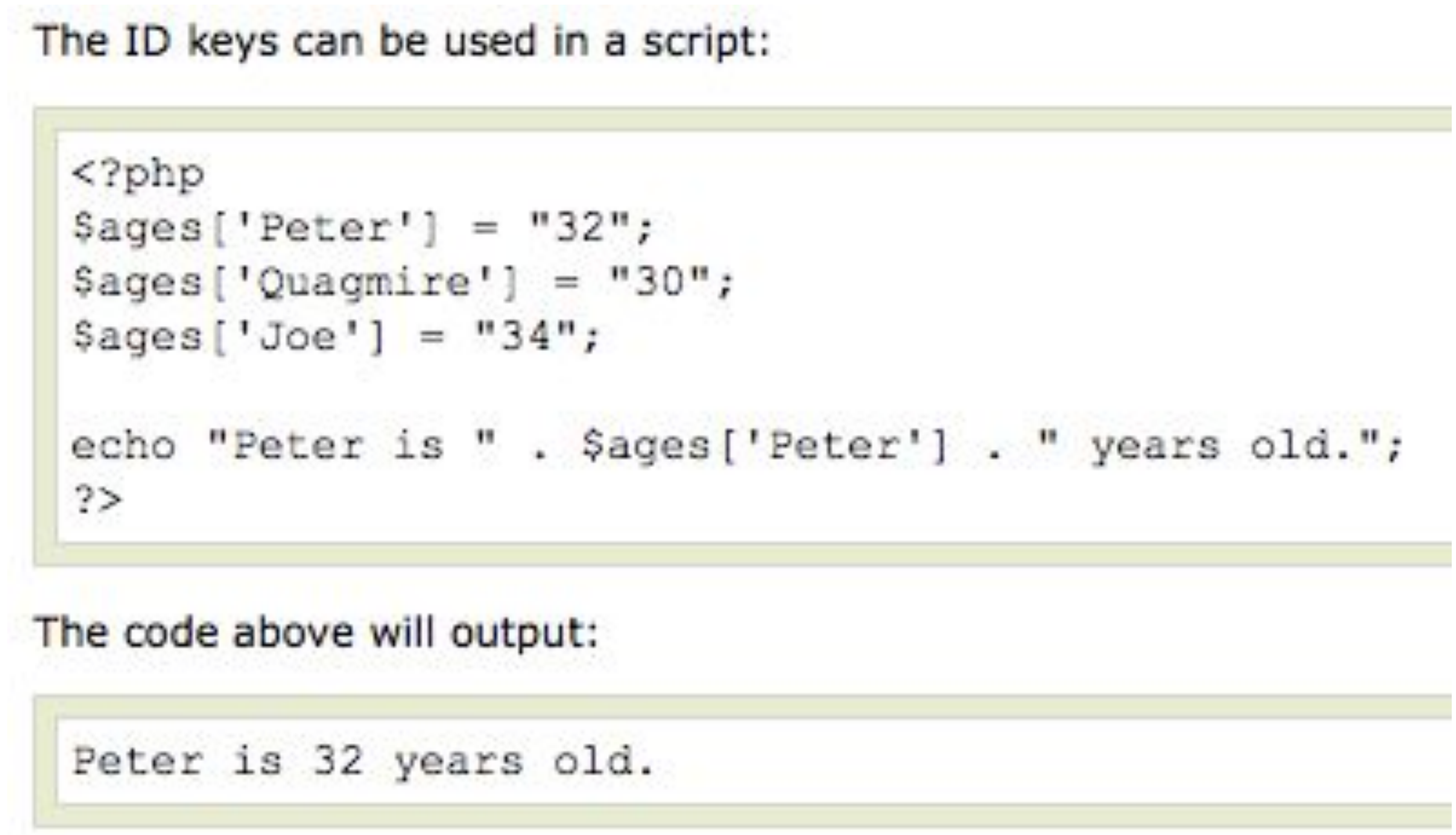
```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

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
<?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 Loops

> **while** - loops through a block of code while a specified condition is true

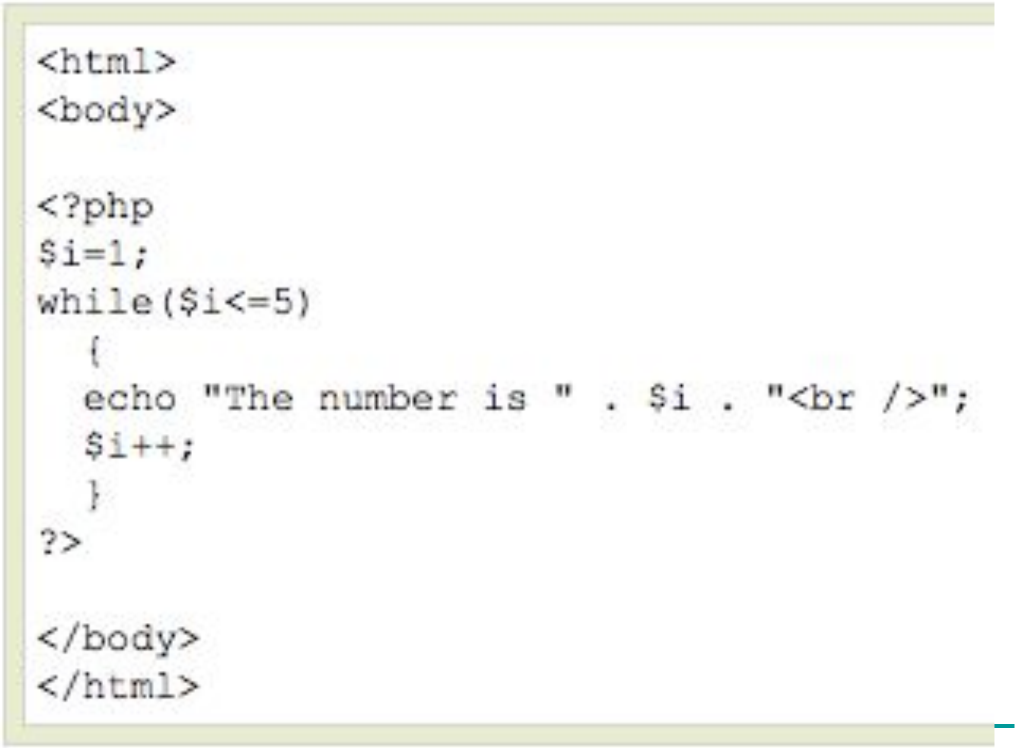
> **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true

> **for** - loops through a block of code a specified number of times

> **foreach** - loops through a block of code for each element in an array

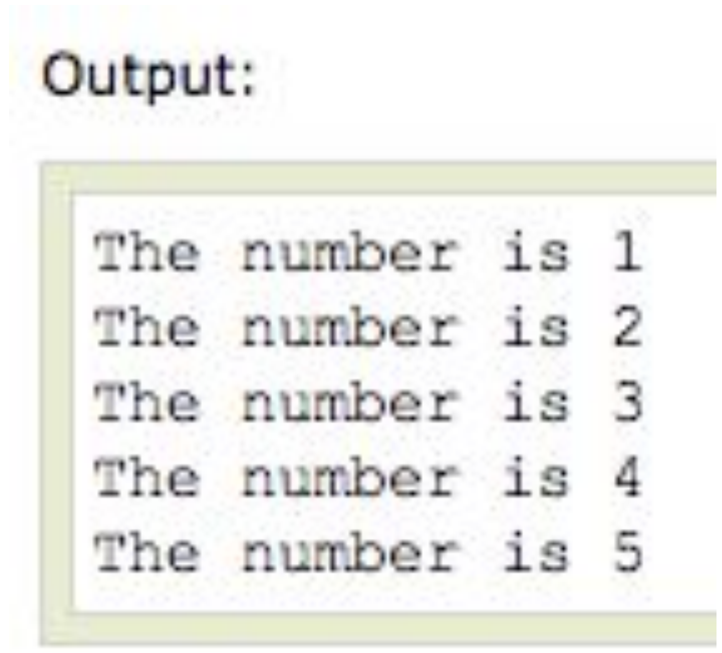
PHP Loops - While

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

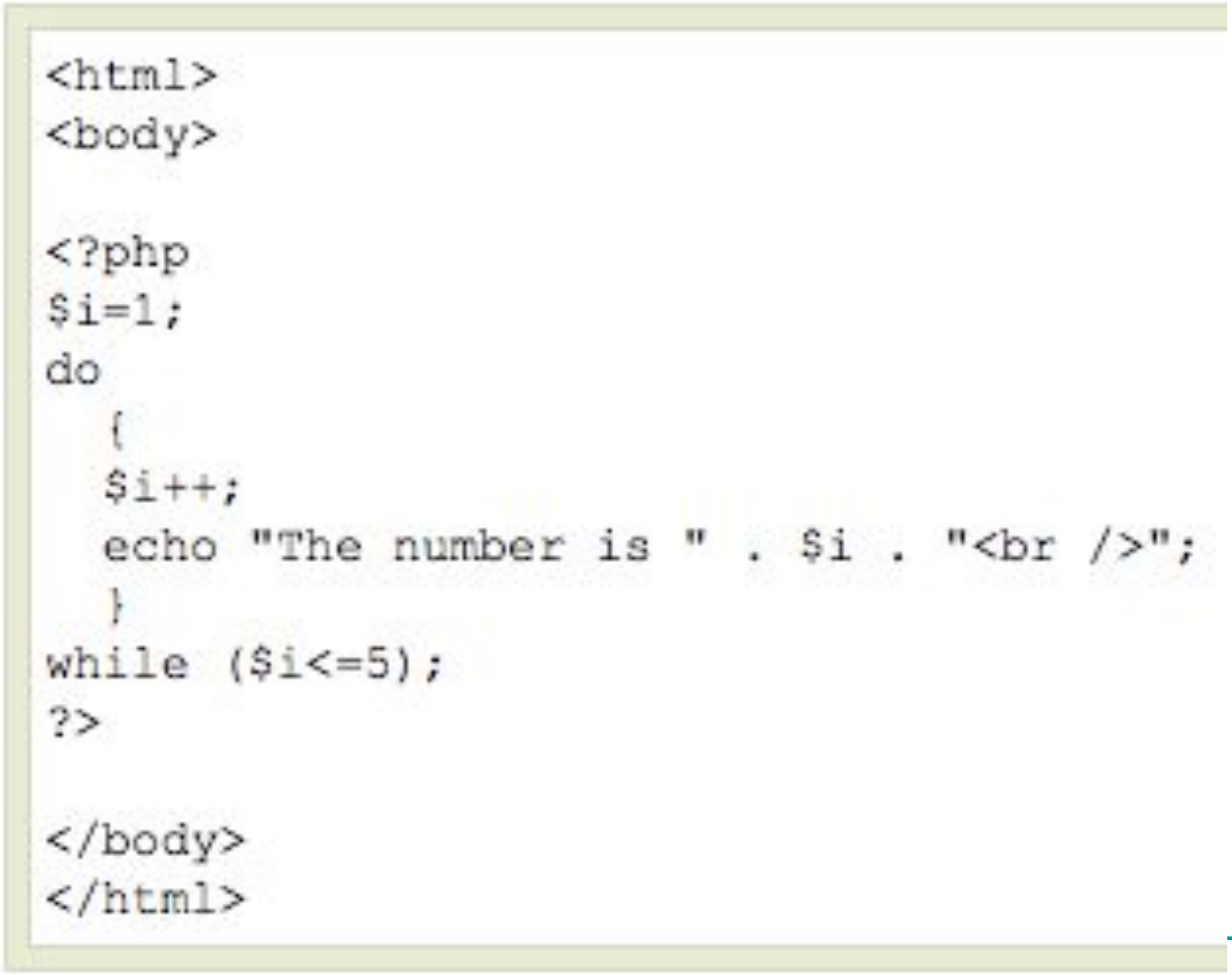

Output:

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

PHP Loops – Do ... While

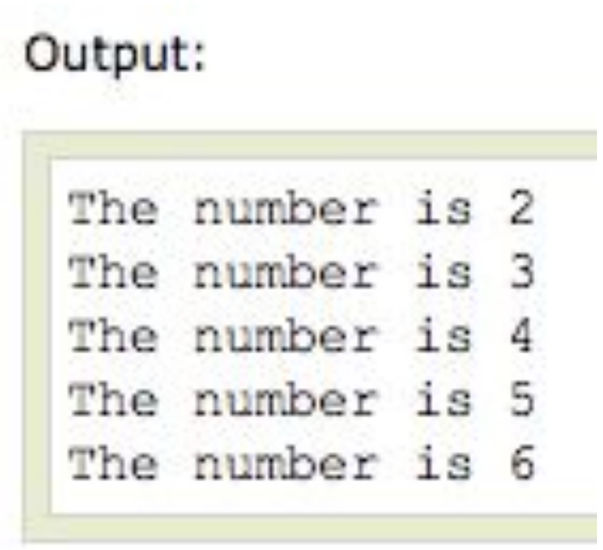
The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

PHP Loops – Do ... While



```
<html>
<body>

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

</body>
</html>
```

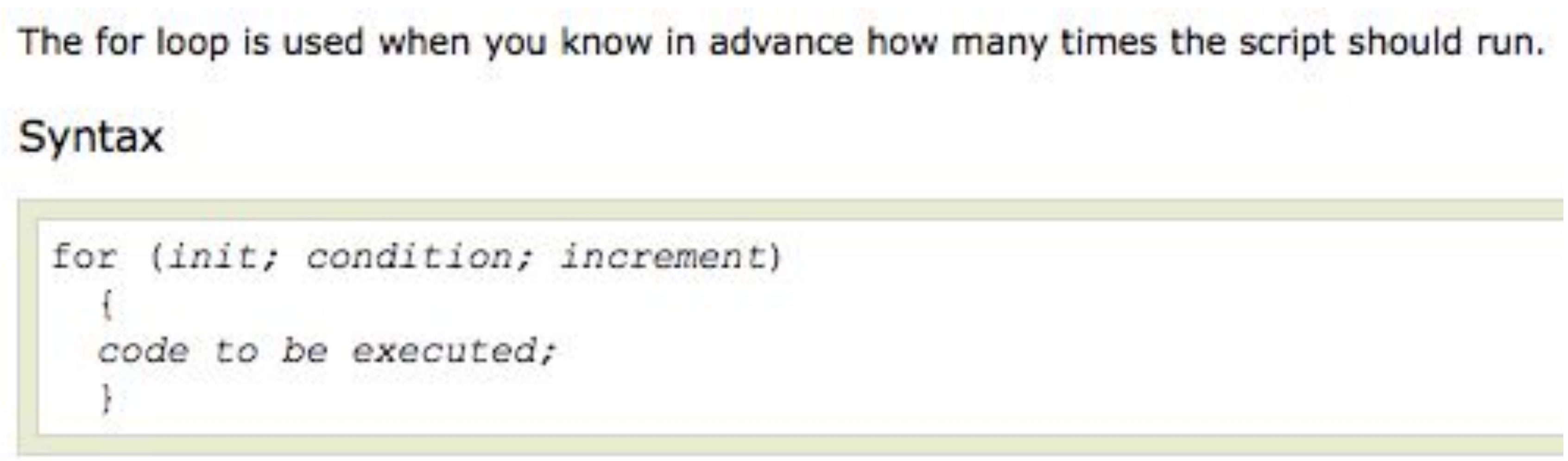
PHP Loops – Do ... While



Output:

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

PHP Loops - For



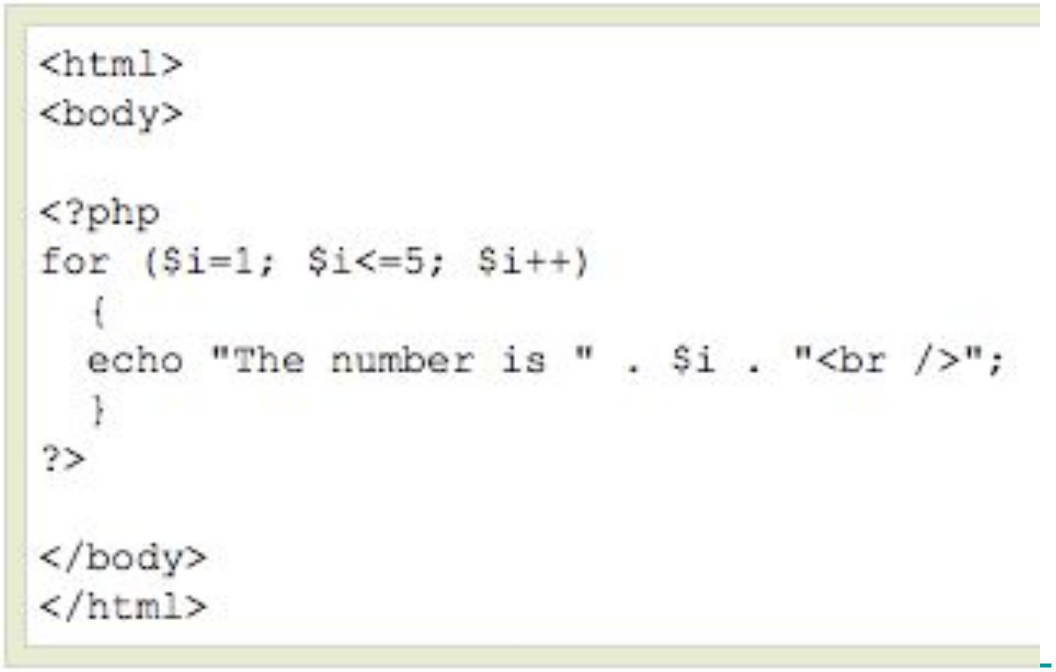
The for loop is used when you know in advance how many times the script should run.

Syntax

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

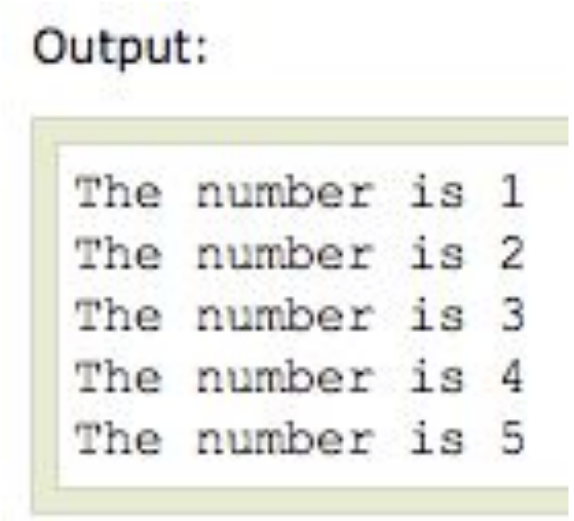
PHP Loops - For

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>
```

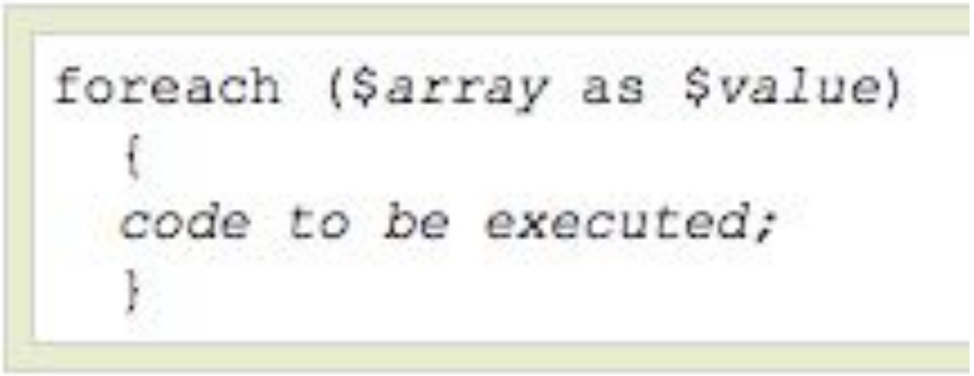
PHP Loops - For



Output:

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

PHP Loops - Foreach

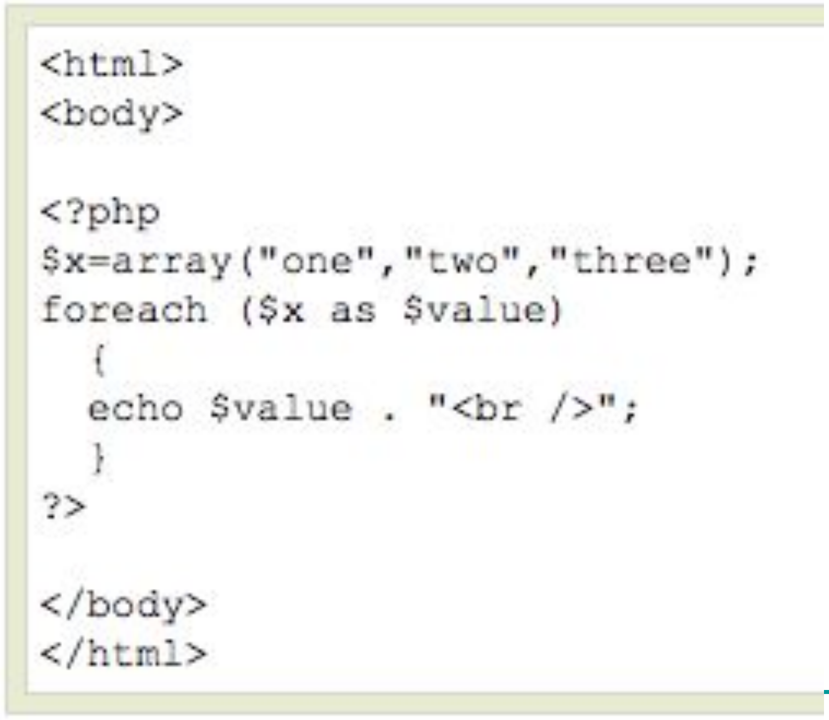


```
foreach ($array as $value)
  {
  code to be executed;
  }
```

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.

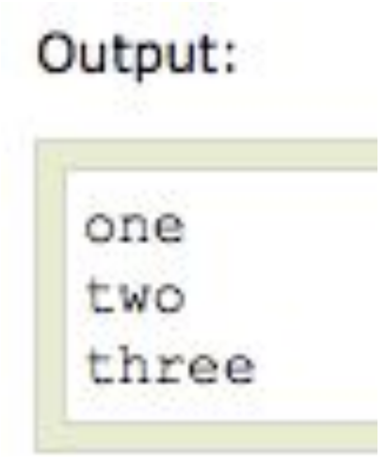
PHP Loops - Foreach

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>
```

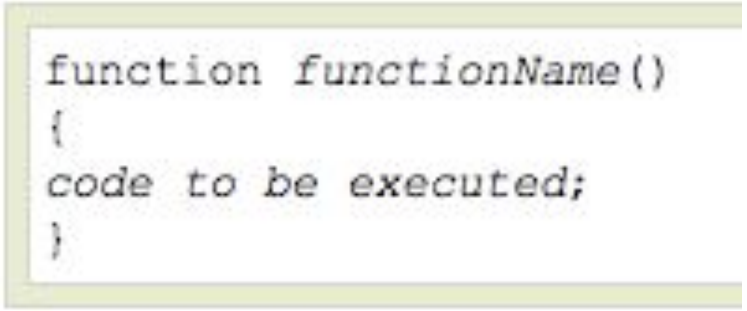
PHP Loops - Foreach


Output:

```
one
two
three
```

PHP Functions

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



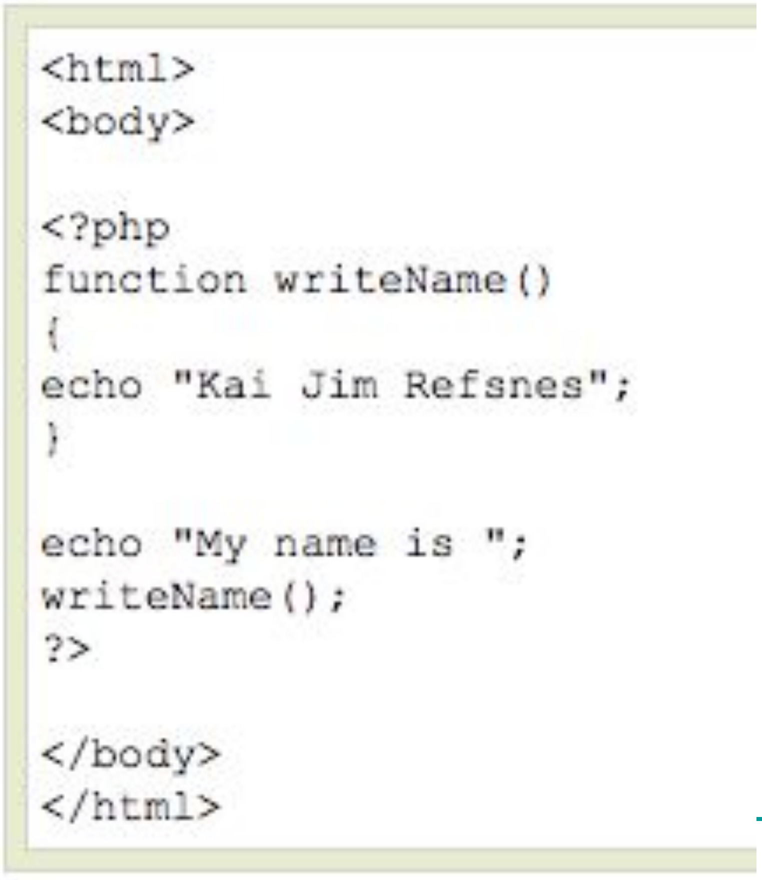
```
function functionName()
{
code to be executed;
}
```

> Give the function a name that reflects what the function does

> The function name can start with a letter or underscore (not a number)

PHP Functions

A simple function that writes a name when it is called:



```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

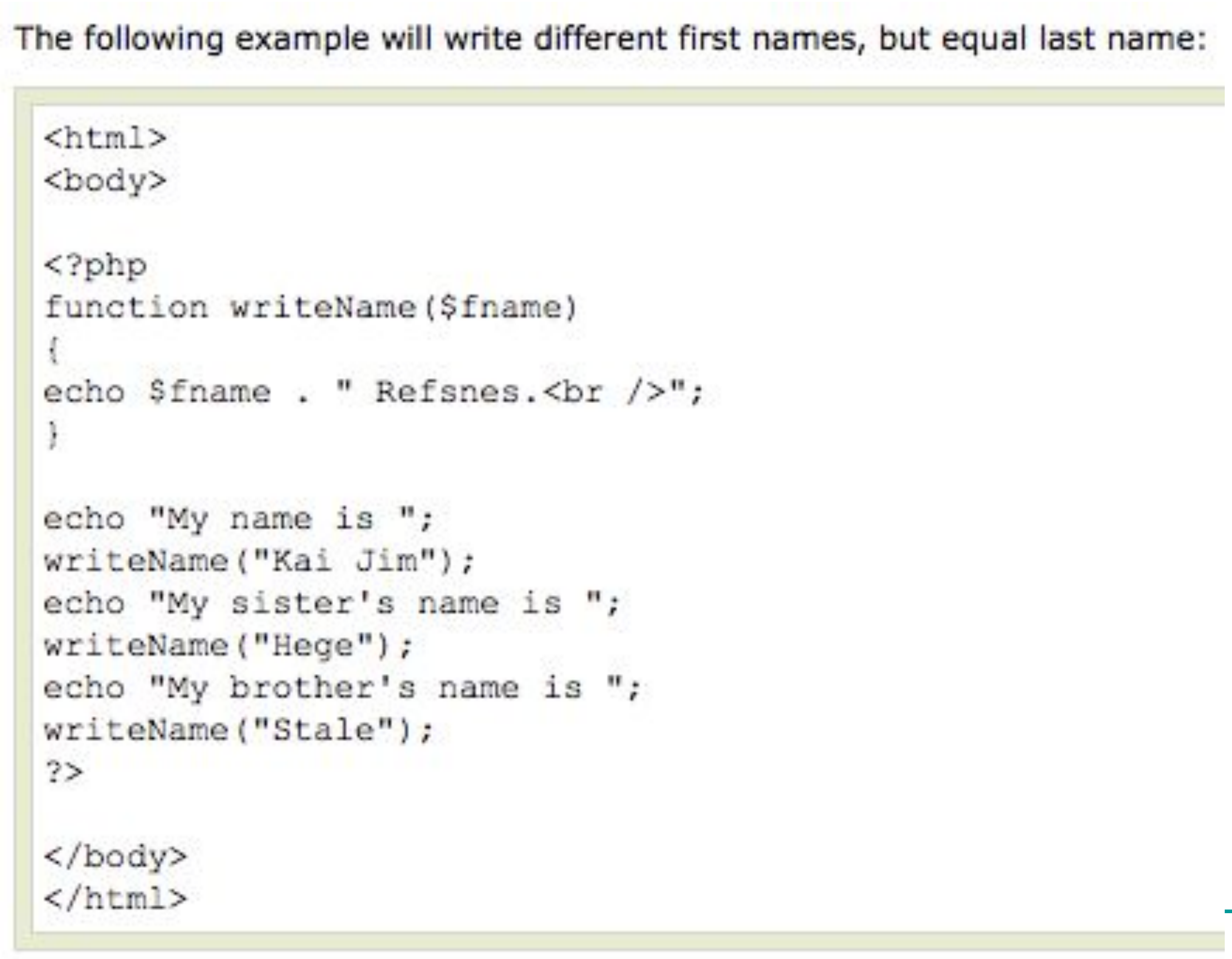
PHP Functions - Parameters

Adding parameters...

> To add more functionality to a function, we can add parameters. A parameter is just like a variable.

> Parameters are specified after the function name, inside the parentheses.

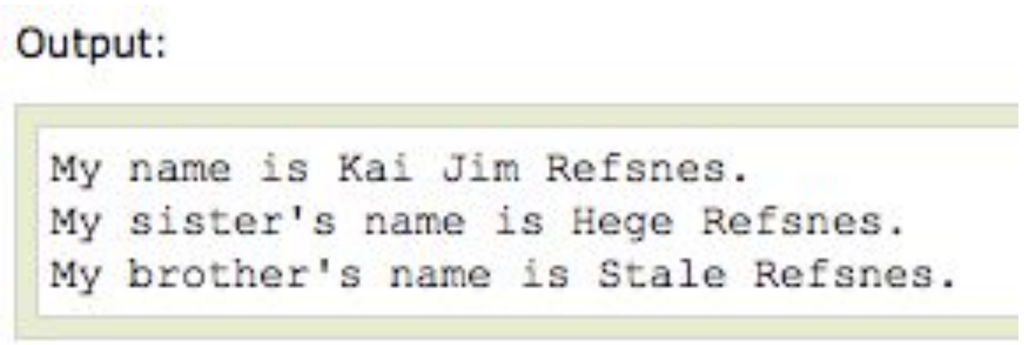
PHP Functions - Parameters



The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

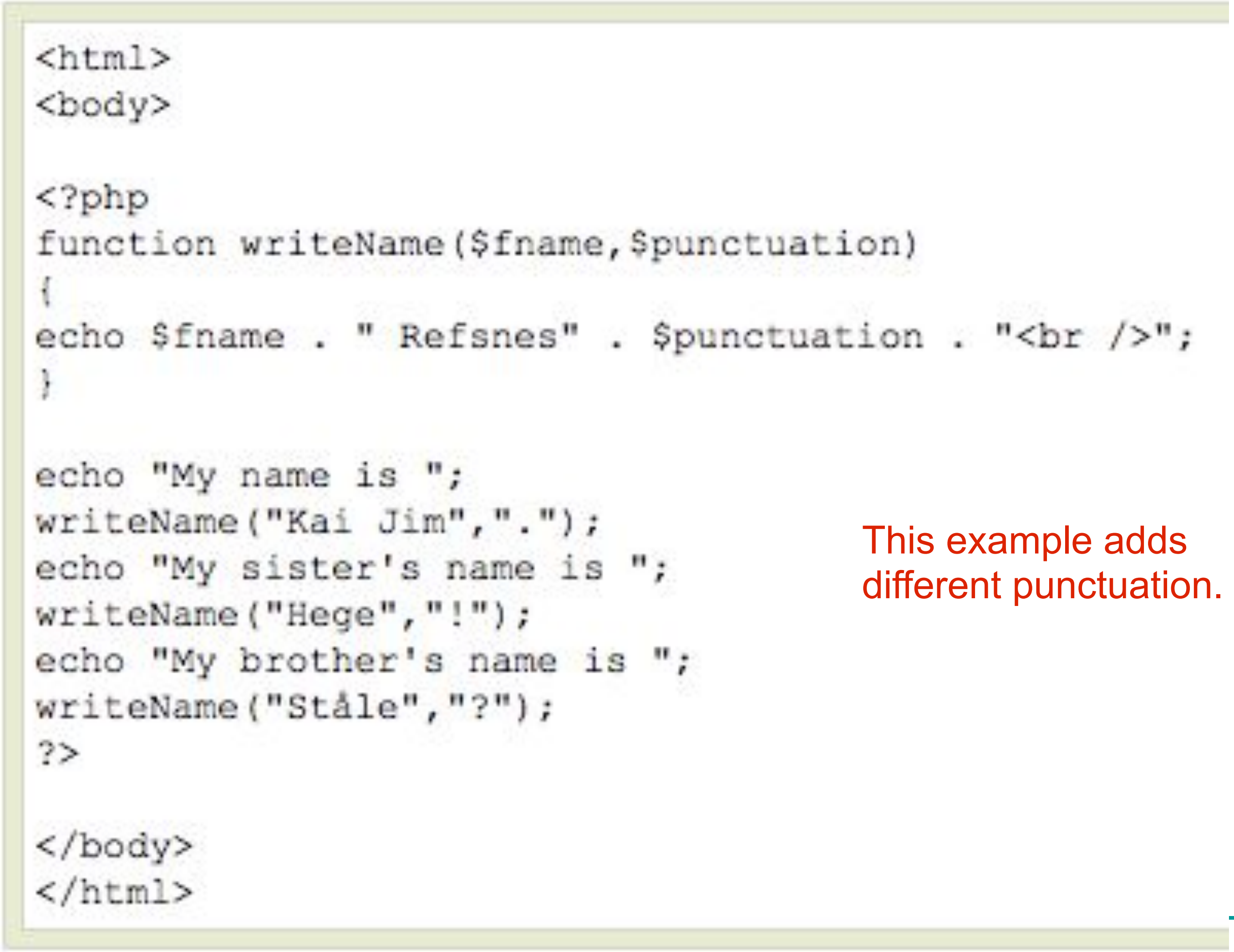
PHP Functions - Parameters



Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

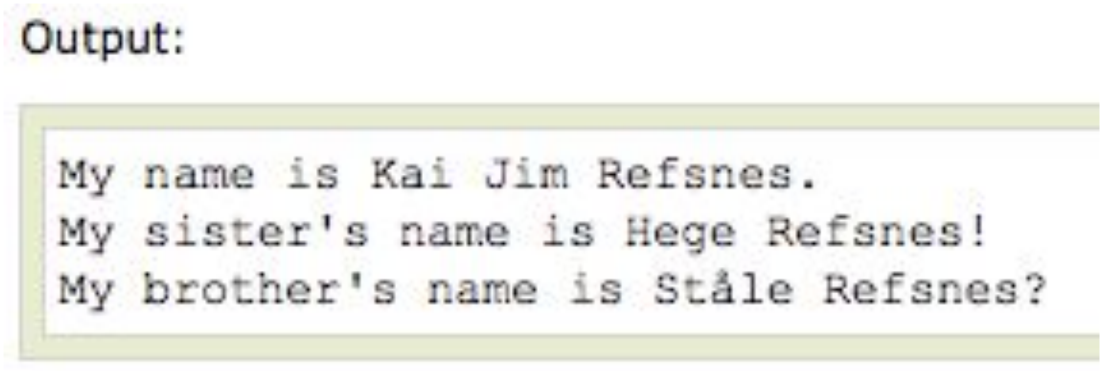
PHP Functions - Parameters



```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

This example adds different punctuation.

PHP Functions - Parameters



Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

References

- Miscellaneous resources from internet
- Lecture notes from https://courses.cs.washington.edu/courses/cse190m/

Thank you!