PHP

What is PHP

- **PHP is a powerful server-side scripting language for creating dynamic and interactive websites.**
- **PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.**
- **PHP is perfectly suited for Web development and can be embedded directly into the HTML code.**
- **The PHP syntax is very similar to Perl and C.**
- **PHP is often used together with Apache (web server) on various operating systems. It also supports ISAPI and can be used with Microsoft's IIS on Windows.**

What You Should Already Know

- HTML
- Some scripting knowledge

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- PHP combined with MySQL are cross-platform

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Get Started

- Download PHP
- Download MySQL Database
- Download Apache Server
- WAMP Server

Basic PHP Syntax

- A PHP scripting block always starts with **<?php** and ends with **?>**. Eg.

```
<?php
//Code goes here…….
?>
```

First Example

```
<html>
<body>
 <?php
echo "Hello World";
//This is a comment
/* This is a comment block */
 ?>
</body>
 </html>
```

Variables in PHP

- Variables are used for storing values, like text strings, numbers or arrays.
- When a variable is set it can be used over and over again.
- All variables in PHP start with a $ sign symbol. Syntax :

  ```
  $var_name = value;
  ```

  Eg.:

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

Variable Naming Rules

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and 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 underscore ($my_string), or with capitalization ($myString)
- Variables are case sensitive.

PHP’s Supported Datatypes

- Scalar Datatypes
- Compound Datatypes

Scalar Datatypes

- Capable of containing a single item of information.
- Boolean:

```
$alive = false; // $alive is false.
$alive = 1; // $alive is true.
$alive = -1; // $alive is true.
$alive = 5; // $alive is true.
$alive = 0; // $alive is false.
```

- Integer
- Float
- String

Compound Datatypes

- *Compound datatypes* allow for multiple items of the same type to be aggregated under a single representative entity.
- Array
- Object

Converting Between Datatypes Using Type Casting

- Converting values from one datatype to another is known as *type casting*.
- Eg:
- $score = (double) 13;

  // $score = 13.0
- $score = (int) 14.8;

  // $score = 14
- $sentence ="This is a sentence";

  echo (int) $sentence;

  // returns 0

Cast Operators	**Conversion**
(array)	Array
(bool) or (boolean)	Boolean
(int) or (integer)	Integer
(object)	Object
(real) or (double) or (float)	Float
(string)	String

Implicit Type casting

-
  ```
  <?php
  $total = 5; // an integer
  $count = "15"; // a string
  $total += $count; // $total = 20 (an integer)
  ?>
  ```

-
  ```
  <?php
     $total = " 45 aaaaa";
  $incoming = 10;
  $total = $incoming + $total; // $total = 55
  ?>
  ```

-
  ```
  <?php
  $total = "1.0";
  if ($total) echo "We're in positive territory!";
  ?>
  ```

Type Identifier Functions

- is_array()
- is_bool(),
- is_float(),
- is_integer(),
- is_null(),
- is_numeric(),
- is_object(),
- is_scalar(),
- is_string()

```
• <?php
$item = 43;
printf("The variable \$item is of type array: %d <br />",
   is_array($item));

printf("The variable \$item is of type integer: %d <br
   />",is_integer($item));

printf("The variable \$item is numeric: %d <br />",
   is_numeric($item));
?>
```

Variable Scope types

- Local variables
- Function parameters
- Global variables

  GLOBAL $somevar;
- Static variables

STATIC $somevar;

Constants

- A *constant* is a value that cannot be modified throughout the execution of a program.
- The define() function defines a constant by assigning a value to a name.

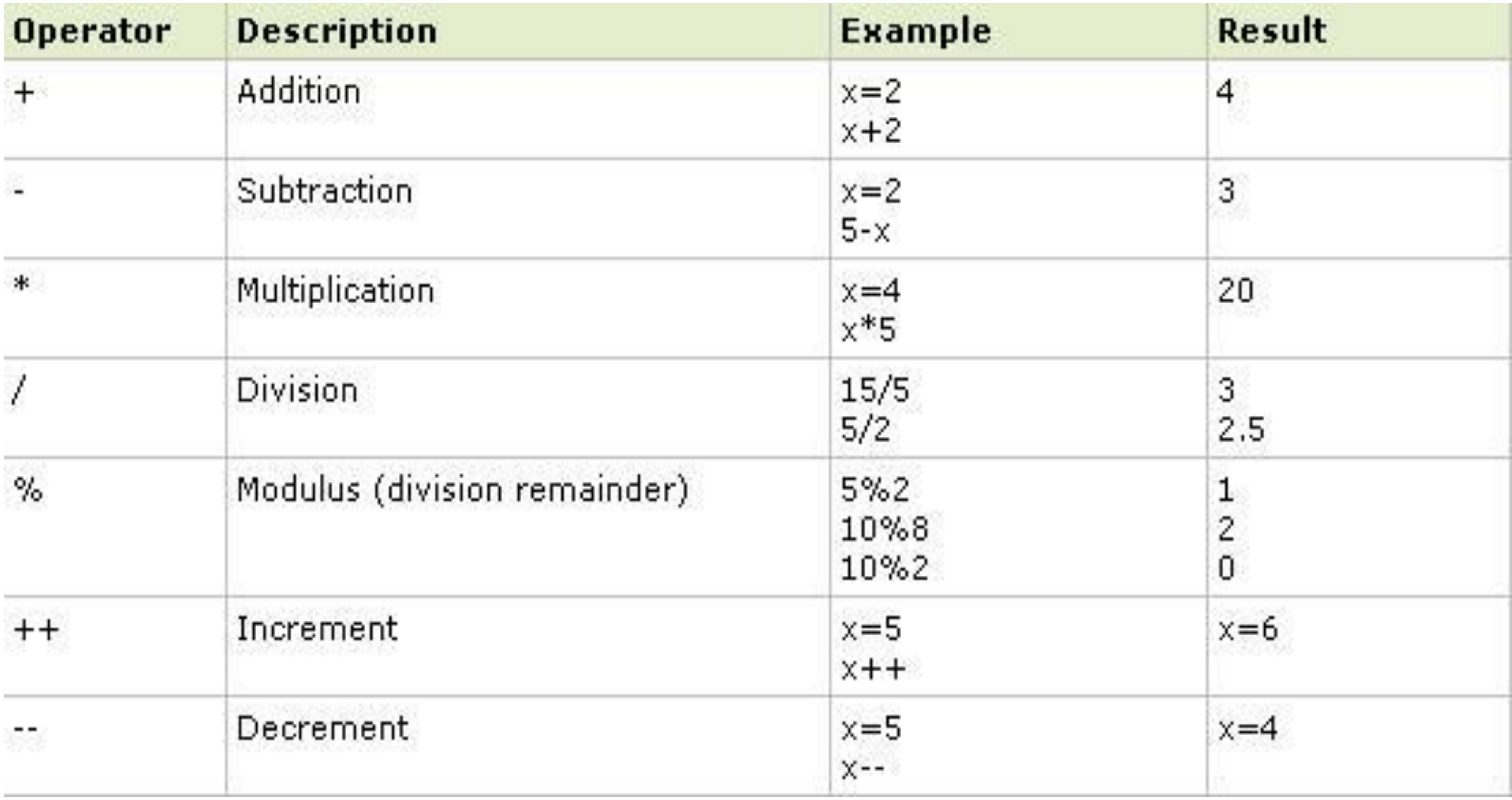
  bool define(string name, mixed value [, bool case_insensitive])

-
  ```
  <?php
  define("CONSTANT", "Hello world.");
  echo CONSTANT; // outputs "Hello world."
  echo Constant; // outputs "Constant" and issues a notice.

  define("GREETING", "Hello you.", true);
  echo GREETING; // outputs "Hello you."
  echo Greeting; // outputs "Hello you."

  ?>
  ```

PHP Operators

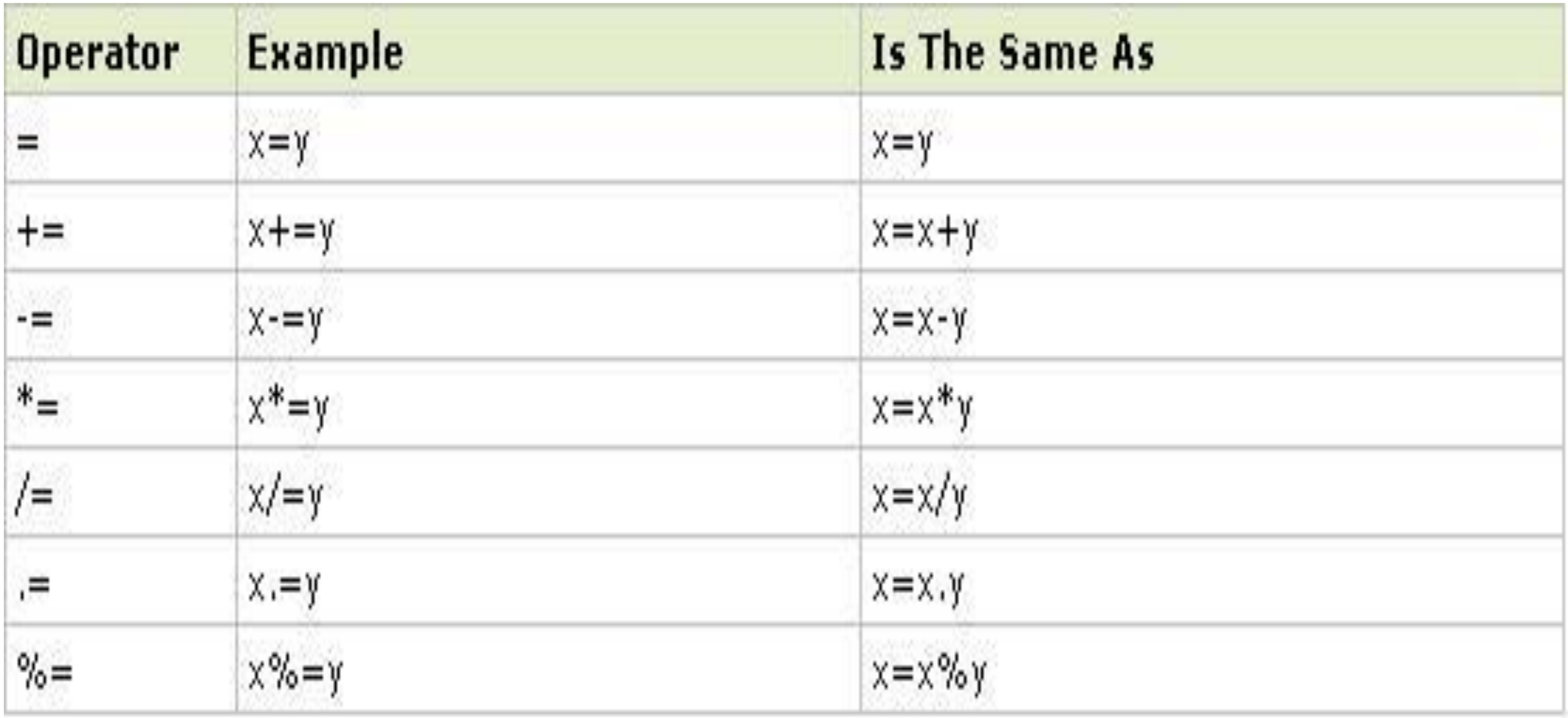
- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators

Arithmetic Operators



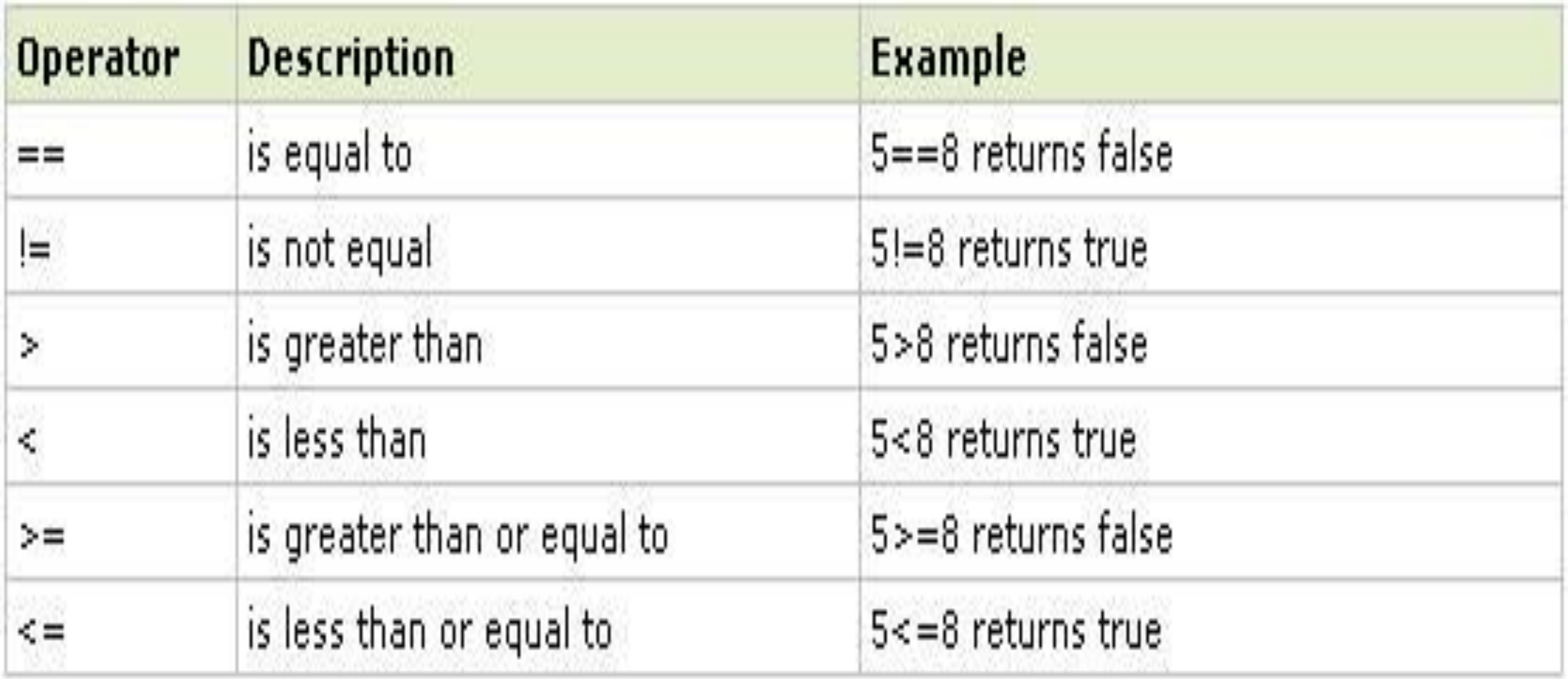
Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators



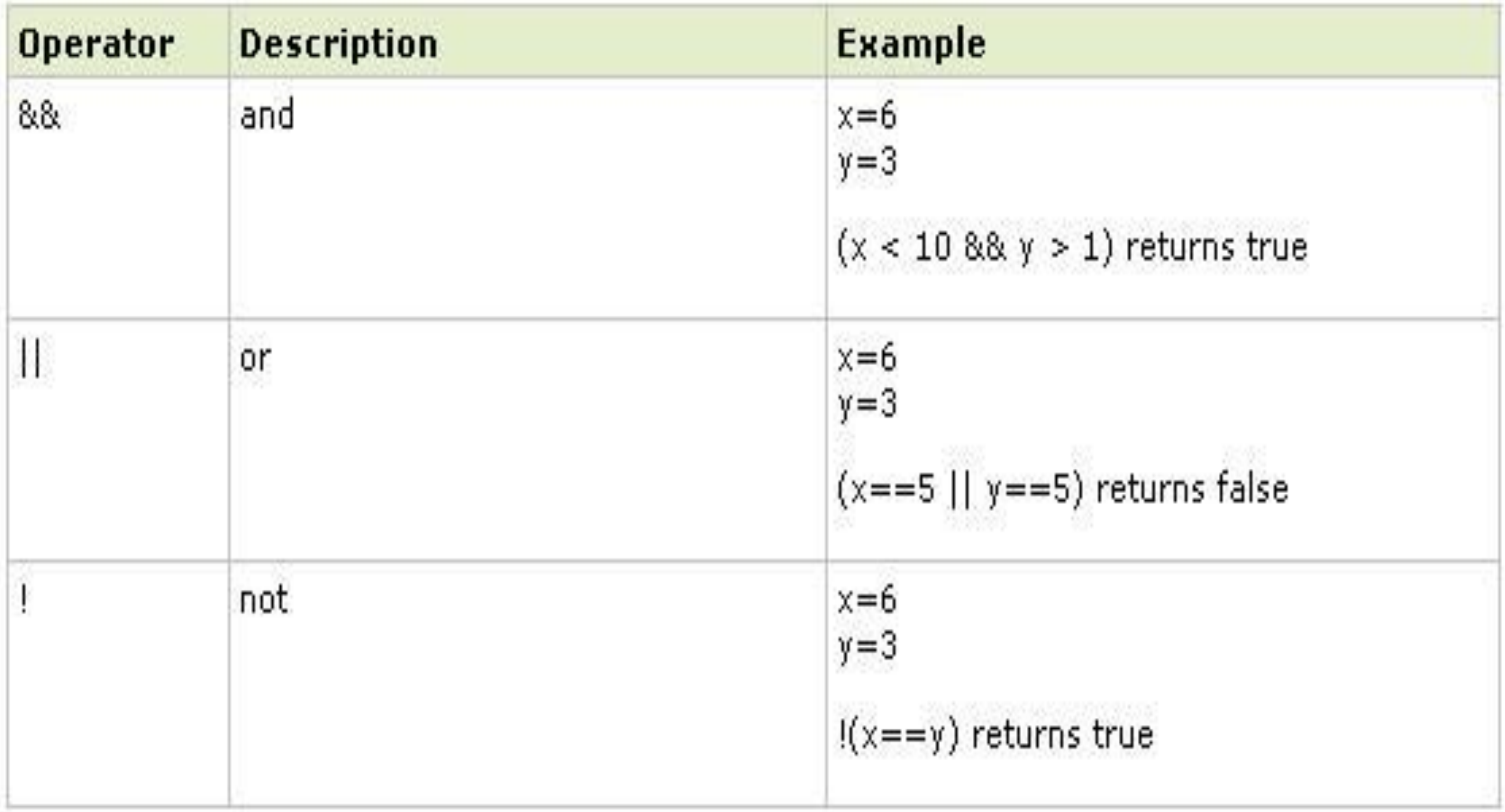
Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators



Operator	Description	Example
==	is equal to	5==8 returns false
!=	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

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

Control Structures

- Conditional Statements
- Looping Statements

Conditional Statements

- **if...else statement** - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** - is used with the if...else statement to execute a set of code if **one** of several condition are true
- **switch statement-** If you want to select one of many blocks of code to be executed, use the Switch statement.

if-else

- if (*condition*)

  {

//code to be executed if condition is true;

  else

//code to be executed if condition is false;

  }

- if (*condition*)

//code to be executed if condition is true;

  elseif (*condition*)

//code to be executed if condition is true;

  else

//code to be executed if condition is false;

switch

- switch (*expression*)

```
{
    case label1:
        //code to be executed if expression = label1;
        break;
    case label2:
        //code to be executed if expression = label2;
        break;
    default:
        //code to be executed if expression is different from both label1
          and label2;
 }
```

Looping Statements

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

- while (*condition*)

  //code to be executed;

- do

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

for,foreach Statement

- for (*init*; *cond*; *incr*)

  {

  code to be executed;

  }

- foreach (*array* as *value*)

  {

  code to be executed;

  }

Arrays

- An *array* is traditionally defined as a group of items that share certain characteristics, such as similarity.
- Each element in the array has its own ID so that it can be easily accessed.
- There are three different kind of arrays:
- **Numeric array** - An array with a numeric ID key
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

Numeric Arrays

- A numeric array stores each element with a numeric ID key.
- Eg:

1. $names = array(“XYZ",“ABC",“LMN");

2. $names[0] = “XYZ";

   $names[1] = “ABC";

   $names[2] = “LMN";

Associative Arrays

- An associative array, each ID key is associated with a value.
- $ages = array(“XYZ"=>32, “ABC"=>30, “LMN"=>34);
- $ages[‘XYZ'] = "32";

  $ages[‘ABC'] = "30";

  $ages[‘LMN'] = "34";

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.
- $families = array

```
(
    "Griffin"=>array ( "Peter", "Lois", "Megan" ),
  “yellow"=>array ( "Glenn" ),
  "Brown"=>array ( "Cleveland", "Loretta", "Junior" )
);
```

- asort()
- arsort()
- krsort()
- ksort()
- rsort()
- shuffle()
- sort()
- array_multisort()

PHP Functions

- A function is a block of code that can be executed whenever we need it.
- All functions start with the word "function()"
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{" - The function code starts after the opening curly brace
- Insert the function code
- Add a "}" - The function is finished by a closing curly brace

```
<html>
    <body>
   <?php
   function writeMyName()
    {
                         echo “XYZ name";
   }
   writeMyName();
    ?>
   </body>
   </html>
```

Adding parameters

-
  ```
  <html>
        <body>
        <?php
        function writeMyName($fname)
         {
                          echo $fname . " Technology Pvt Ltd <br>";
        }
        echo "My Company name is ";
        writeMyName(“Bitwise");
           echo "My Company name is ";
        writeMyName(“Zensar");
            echo "My Company name is ";
         writeMyName(“Cogni");
        ?>
        </body>
         </html>
  ```

Default Argument Values

- must appear at the end of the parameter list.
- must be constant expressions.
- function calcSalesTax($price, $tax=.0675)

```
{
$total = $price + ($price * $tax);
echo "Total cost: $total";
}
```

Returning Values from a Function

- The return statement returns any ensuing value back to the function caller.
- function calcSalesTax($price, $tax=.0675)

```
{
$total = $price + ($price * $tax);
return $total;
}
```

Returning Multiple Values

- The list() construct offers a convenient means for retrieving values from an array.
- Eg:

```
<?php
$colors = array("red","blue","green");
list($red, $blue, $green) = $colors;
?>
```

-
  ```
  <?php
  function retrieveUserProfile()
  {
  $user[] = “YourName";
  $user[] = “Youremailid@compname.in";
  $user[] = "English";
  return $user;
  }
  list($name, $email, $language) = retrieveUserProfile();
  echo "Name: $name, email: $email, language:
     $language";
  ?>
  ```

Function Libraries

- Reusability
- PHP libraries are created via the simple aggregation of function definitions in a single file.
- Save this Library and include in the target PHP page using Server Side Includes.

Server Side Includes

- Inserting the content of a file into a PHP file before the server executes it.
- Used to create functions, headers, footers, or elements that can be reused on multiple pages.
- include(filename)
- include_once (filename)
- require (filename)
- require_once (filename)

The include() Function

- Eg

```
<html>
<body>
<?php include("header.php");
?>
<h1>
Welcome to my home page
</h1>
 <p>
Some text
</p>
</body>
</html>
```

The require() Function

- Similar like include with following differences:
  - The file will be included in the script in which the require() construct appears, regardless of where require() is located.
  - Script execution will stop if a require() fails, whereas it may continue in the case of an include().

- include_once (filename)
  - Ensures a File Is Included Only Once
- require_once (filename)
  - Ensures a File Is Required Only Once

What we have learnt so far……

- What is Server side scripting
- PHP Basics
- Variables, Arrays,Functions