

# PHP Syntax

- Operators in PHP
- Loop constructs in PHP
- Array in PHP
- Manipulation with array in PHP

- PHP language supports following type of operators.
  - ❖ Arithmetic operators
  - ❖ Assignment operators
  - ❖ Comparison operators
  - ❖ Increment/Decrement operators
  - ❖ Logical operators
  - ❖ String operators

- The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$\$x + \$y$	Sum of $\$x$ and $\$y$
-	Subtraction	$\$x - \$y$	Difference of $\$x$ and $\$y$
*	Multiplication	$\$x * \$y$	Product of $\$x$ and $\$y$
/	Division	$\$x / \$y$	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \$y$	Remainder of $\$x$ divided by $\$y$
**	Exponentiation	$\$x ** \$y$	Result of raising $\$x$ to the $\$y$ 'th power (Introduced in PHP 5.6)

- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
<code>x = y</code>	<code>x = y</code>	The left operand gets set to the value of the expression on the right
<code>x += y</code>	<code>x = x + y</code>	Addition
<code>x -= y</code>	<code>x = x - y</code>	Subtraction
<code>x *= y</code>	<code>x = x * y</code>	Multiplication
<code>x /= y</code>	<code>x = x / y</code>	Division
<code>x %= y</code>	<code>x = x % y</code>	Modulus

- The PHP comparison operators are used to compare two values (number or string)

Operator	Name	Example	Result
==	Equal	<code>\$x == \$y</code>	Returns true if \$x is equal to \$y
===	Identical	<code>\$x === \$y</code>	Returns true if \$x is equal to \$y, and they are of the same type
!=	Not equal	<code>\$x != \$y</code>	Returns true if \$x is not equal to \$y
<>	Not equal	<code>\$x &lt;&gt; \$y</code>	Returns true if \$x is not equal to \$y
!==	Not identical	<code>\$x !== \$y</code>	Returns true if \$x is not equal to \$y, or they are not of the same type
>	Greater than	<code>\$x &gt; \$y</code>	Returns true if \$x is greater than \$y
<	Less than	<code>\$x &lt; \$y</code>	Returns true if \$x is less than \$y
>=	Greater than or equal to	<code>\$x &gt;= \$y</code>	Returns true if \$x is greater than or equal to \$y
<=	Less than or equal to	<code>\$x &lt;= \$y</code>	Returns true if \$x is less than or equal to \$y

# PHP Increment / Decrement Operators

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
<code>++\$x</code>	Pre-increment	Increments <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x++</code>	Post-increment	Returns <code>\$x</code> , then increments <code>\$x</code> by one
<code>--\$x</code>	Pre-decrement	Decrements <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x--</code>	Post-decrement	Returns <code>\$x</code> , then decrements <code>\$x</code> by one

- The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	\$x and \$y	True if both \$x and \$y are true
or	Or	\$x or \$y	True if either \$x or \$y is true
xor	Xor	\$x xor \$y	True if either \$x or \$y is true, but not both
&&	And	\$x && \$y	True if both \$x and \$y are true
	Or	\$x    \$y	True if either \$x or \$y is true
!	Not	!\$x	True if \$x is not true



- PHP has two operators that are specially designed for strings.

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

- Teacher demo about operators for students

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

- The while loop executes a block of code as long as the specified condition is true.
- Syntax

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

- Example

```
<?php  
$x = 1;  
  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

# The PHP do...while Loop

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

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

- Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

# The PHP do...while Loop

- Example

```
<?php  
$x = 1;  
  
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>
```

---

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

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

- Parameters:
  - ❖ *init counter*. Initialize the loop counter value
  - ❖ *test counter*. Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
  - ❖ *increment counter*. Increases the loop counter value

## ■ Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```



# The PHP foreach Loop

- 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.
- Syntax

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

# The PHP foreach Loop

## ■ Example

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

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

- Teacher demo loop constructs for students

- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In PHP, the array() function is used to create an array.
- In PHP, there are three types of arrays:
  - ❖ **Indexed arrays** - Arrays with a numeric index
  - ❖ **Associative arrays** - Arrays with named keys
  - ❖ **Multidimensional arrays** - Arrays containing one or more arrays

- There are two ways to create indexed arrays.
- The index can be assigned automatically (index always starts at 0), like this:

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

- or the index can be assigned manually:

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

## ■ Example

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

- The count() function is used to return the length (the number of elements) of an array

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

- To loop through and print all the values of an indexed array, you could use a for loop

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

- Associative arrays are arrays that use named keys that you assign to them.

- There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

- or  

```
$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";
```

- To loop through and print all the values of an associative array, you could use a foreach loop

```
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
  
foreach($age as $x => $x_value) {  
    echo "Key=" . $x . ", Value=" . $x_value;  
    echo "<br>";  
}  
?>
```



# Multidimensional Arrays

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

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

for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
        echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
}
?>
```

# PHP - Sort Functions For Arrays

- `sort()` - sort arrays in ascending order
- `rsort()` - sort arrays in descending order
- `asort()` - sort associative arrays in ascending order, according to the value
- `ksort()` - sort associative arrays in ascending order, according to the key
- `arsort()` - sort associative arrays in descending order, according to the value
- `krsort()` - sort associative arrays in descending order, according to the key
- Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

- Teacher demo code about arrays for students

- Operators in PHP
- Loop constructs in PHP
- Array in PHP
- Manipulation with array in PHP