Assignment Operators

- 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
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Comparison Operators

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

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

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
++$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

Logical Operators

- 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

String Operators

- 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

Array Operators

- The PHP array operators are used to compare arrays.

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

Conditional Statements

- if statement
- switch statement
- goto statement

If statement

- Simple if

```
if(test-expr)
{
True part
}
statement
```

If...else statement

```
if(test-expr)
{
True part
}
else
{
False part
}
```

Nested if statement

```
if(test-exp 1)
{
   if(test-exp 2)
   {
       true 1 & 2
   }
   else
   {
       false 2
   }
}
else
{
   false part all
}
```

else if ladder :-

```
if(test-exp 1)
   statement -1;
 else if(test-exp 2)
     statement – 2;
  else if(test-exp 3)
              statement –
  3;
     .
      .
       .
       else
         false statement
```

switch statement :-

```
switch(ch)
{
    case value1:
      statement 1;break;
            case value2:
              statement
2;break;
            case value3:
      statement 3; break;
            .
            .
            .
             default :
       statement-false; break;
}
```

- Case value can be of any data type or even any condition.
- We can use break and continue interchangeable.
- Default can be placed anywhere in switch.

The if Statement

- **Syntax**
  - if (*condition*) {
    code to be executed if condition is true;
    }
- The example below will output "Have a good day!" if the current time (HOUR) is less than 20:
- **Example**

```
<?php
  $t = date("H");

  if ($t < "20") {
      echo "Have a good day!";
  }
?>
```

The if...else Statement

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

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

- **Example**

```
<?php
    $t = date("H");
    if ($t < 20)
        echo "Have a good day!";
    else
        echo "Have a good night!";
?>
```

The if...elseif....else Statement

- Use the if....elseif...else statement to specify a new condition to .
- **Syntax**

```
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;
}
```

E.g

```
<?php
$t = date("H");
if ($t < "10")
  echo "Have a good morning!";
elseif ($t < "20")
  echo "Have a good day!";
else
  echo "Have a good night!";
?>
```

The switch Statement

- Use the switch statement to select one of many blocks of code to be executed. Syntax:
- switch (*n*) {
  case *label1:*
  code to be executed if n=label1;
  break;
  case *label2:*
  code to be executed if n=label2;
  break;
  case *label3:*
  code to be executed if n=label3;
  break;
  ...
  default:
  code to be executed if n is different from all labels;
  }

Example

```
<?php
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, or green!";
    }
?>
```

Switchtest.php

Loops

- 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

Loops

while loop :-

```
while (expr)
  statement;
     ….
}
```

do..while loop :-

```
do{

statements;
}while(expr);
```

for loop :-

```
for(initialization; condition; incr/decr)
{
   statement;
}
```

do..while loop is executed at least once whereas while and for loop executes only when condition is satisfied.

The while Loop

- 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 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);
```

- **Example**

```
<?php
 $x = 1;

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

Cont...

- Example

```
<?php
  $x = 6;

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

The for Loop

- 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

Cont...

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

The Foreach Loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- **Syntax**
  - foreach ($*array* as $*value*) {
    code to be executed;
    }
- **Example**

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

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

Arrays

- An array stores multiple values in one single variable.
- **Example**

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

- **What is an Array?**

- An array is a special variable, which can hold more than one value at a time.
- 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 = "Volvo";
  $cars2 = "BMW";
  $cars3 = "Toyota";

Create an Array

- In PHP, the array() function is used to create an array:
- 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

Indexed 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";
- E.g.

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

Get The Length of an Array - The count() Function

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

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

- **Loop Through an Indexed Array**

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    $arrlength = count($cars);
    for($x = 0; $x < $arrlength; $x++) {
        echo $cars[$x];
        echo "<br>";
    }
?>
```

Associative Arrays

- 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";
- The named keys can then be used in a script:
- **Example**

```
<?php
 $age = array("Peter"=>"35",
 "Ben"=>"37", "Joe"=>"43");
 echo "Peter is " . $age['Peter'] . "
 years old.";
?>
```

Loop Through an Associative Array

- To loop through and print all the values of an associative array, you could use a foreach loop, like this:
- **Example**

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

 foreach($age as $x => $x_value) {
     echo "Key=" . $x . ", Value="
 . $x_value;
     echo "<br>";
 }
?>
```

Adding elements with [ ]

- The empty brackets add an element to the array.
- The element has a numeric key that's one more than the biggest numeric key already in the array.
- If the array doesn't exist yet, the empty brackets add an element with a key of 0.

Adding elements with [ ]

Example

```
// Create $lunch array with two elements
// This sets $lunch[0]
$lunch[] = 'Chana masala';

// This sets $lunch[1]
$lunch[] = 'Chole bhature';

// Create $dinner with three elements
$dinner=array('Dum    Aloo‘,'Gajar    ka
  Halwa’, ’Butter Roti');

// Add an element to the end of $dinner
// This sets $dinner[3]
$dinner[] = 'Palak Paneer';
```

Arrays

- **Finding the size of an array**

```
echo count($cars);
```

Assigning a Range of Values

- range( ) : function creates an array of consecutive integer or character values between the two values you pass to it as arguments.

```
array range ($start , $limit [, number $step = 1 ] )
```

- Parameters
  - Start - First value of the sequence.
  - Limit - The sequence is ended upon reaching the limit value.

- step value is given, it will be used as the increment between elements in the sequence.

- step should be given as a positive number. If not specified, step will default **to 1.**

Padding an Array

- To create an array initialized to the same value, use array_pad( ).
  array array_pad (array $input, int $pad_size mixed $pad_value)
  input - Initial array of values to pad.
  pad_size - New size of the array.
  pad_value-Value to pad if *input* is less than pad_size.
  Example

```
$scores = array(3, 10);
$padded = array_pad($scores, 5, 0);
  // $padded is now array(3, 10, 0, 0, 0)
```

- If you want the new values added to the start of the array, use a negative second argument:

```
$padded = array_pad($scores, -5, 0);
```

Searching a Value from array

array_search()

used to search and locate a specific value in the given array. If it successfully finds the specific value, it returns its corresponding key value. If the element is found twice or more, then the first occurrence of the matching value's key will be returned.

array_search($value, $array, strict_parameter)

1. **value (required):** This parameter represents the value that the user wishes to search in the given array.
2. **$array (required):** This parameter represents the original array, in which the user wants to search the element.
3. **strict_parameter (optional)**: It is an optional parameter that can be set either to TRUE or FALSE. It represents the strictness of the array search. By default, this parameter is set to Boolean FALSE.
    1. If strict_parameter is set to TRUE, then the function looks for similar values in the array, i.e., a string 200 will not be considered same as integer 200. Hence, both the values are different.
    2. If strict_parameter is set to Boolean FALSE, strictness is not retained, i.e., a string 200 will be considered same as integer 200.

Cont.

Searchdemo.php

```
<?php

$array_name = array(“red ”, “blue", “green”, “white", "purple");
$search_value = “white";
 $x=  array_search($search_value, $array_name)
  print_r(“$search_value is at position $x);
?>
```

in_array() : check whether value is in array or not

syntax : in_array($value,$array)

Value is found in array then returns true o.w false.

Count values in array

array_count_values($array) : return the count of each values in the arra

```
<?php
    $a=array("A","Cat","Dog","A","Dog");
    print_r(array_count_values($a));
 ?>
```

Output :

```
 Array ( [A] => 2 [Cat] => 1 [Dog] => 2 )
```

```
<?php
$a=array("A","Cat","Dog","A","Dog");
  $cnt = array_count_values($a);
 echo  $cnt['A'];
 ?>
```

Out put : 2

Array Functions

array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_assoc()	Compare arrays, and returns the differences (compare keys and values)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_diff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using a user-defined key comparison function)

Array Functions

array_diff_ukey()	Compare arrays, and returns the differences (compare keys only, using a user-defined key comparison function)
array_fill()	Fills an array with values
array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)
array_intersect_assoc()	Compare arrays and returns the matches (compare keys and values)
array_intersect_key()	Compare arrays, and returns the matches (compare keys only)
array_intersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function)
array_intersect_ukey()	Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function)

Array Functions

array_key_exists()	Checks if the specified key exists in the array
array_keys()	Returns all the keys of an array
array_map()	Sends each value of an array to a user-made function, which returns new values
array_merge()	Merges one or more arrays into one array
array_merge_recursive()	Merges one or more arrays into one array recursively
array_multisort()	Sorts multiple or multi-dimensional arrays
array_pad()	Inserts a specified number of items, with a specified value, to an array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array

Array Functions

array_rand()	Returns one or more random keys from an array
array_reduce()	Returns an array as a string, using a user-defined function
array_replace()	Replaces the values of the first array with the values from following arrays
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array

Array Functions

array_udiff()	Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function)
array_udiff_assoc()	Compare arrays, and returns the differences (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_udiff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using two user-defined key comparison functions)
array_uintersect()	Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function)
array_uintersect_assoc()	Compare arrays, and returns the matches (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_uintersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using two user-defined key comparison functions)
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
array_walk()	Applies a user function to every member of an array

Array Functions

array_walk_recursive()	Applies a user function recursively to every member of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values
count()	Returns the number of elements in an array
current()	Returns the current element in an array
each()	Deprecated from PHP 7.2. Returns the current key and value pair from an array
end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a specified value exists in an array

Array Functions

key()	Fetches a key from an array
krsort()	Sorts an associative array in descending order, according to the key
ksort()	Sorts an associative array in ascending order, according to the key
list()	Assigns variables as if they were an array
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
natsort()	Sorts an array using a "natural order" algorithm
next()	Advance the internal array pointer of an array
pos()	Alias of current()
prev()	Rewinds the internal array pointer
range()	Creates an array containing a range of elements

Array Functions

reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an indexed array in descending order
shuffle()	Shuffles an array
sizeof()	Alias of count()
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function and maintains the index association
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array by values using a user-defined comparison function

Multidimensional Arrays

- A multidimensional array is an array containing one or more arrays.
- PHP understands multidimensional arrays that are two, three, four, five, or more levels deep.
- However, arrays more than three levels deep are hard to manage for most people.

Two-dimensional Arrays

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

- We can store the data from the table above in a two-dimensional array, like this:
- $cars = array (array("Volvo",22,18), array("BMW",15,13), array("Saab",5,2), array("Land Rover",17,15));

Cont...

- Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.
- To get access to the elements of the $cars array we must point to the two indices (row and column):
- **Example**

```
<?php
$cars = array (array("Volvo",22,18),
  array("BMW",15,13), array("Saab",5,2),
  array("Land Rover",17,15));
echo $cars[0][0].": In stock: ".$cars[0][1].",
  sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].",
  sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].",
  sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].",
  sold: ".$cars[3][2].".<br>";
?>
```

Cont...

- Accessing two dimensional array using for loop :
- **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>";
}
?>
```

Multidimensional Array

- The values in an array can themselves be arrays. This lets you easily create multidimensional arrays:

```
$row_0 = array(1, 2, 3);
$row_1 = array(4, 5, 6);
$row_2 = array(7, 8, 9);
$multi = array($row_0, $row_1, $row_2);
```

- You can refer to elements of multidimensional arrays by appending more []s:

```
$value = $multi[2][0];
// row 2, column 0. $value = 7
```

Creating variables from associative array

- The extract( ) function automatically creates local variables from an array. The indexes of the array elements are the variable names

```
$person = array('name' => 'Fred', 'age' => 35, 'wife' => 'Betty');
extract($person); // $name, $age, and $wife are now set
print $name;
```

- If a variable created by the extraction has the same name as an existing one, the extracted variable overwrites the existing variable.

```
$shape = "round";
$array = array("cover" => "bird", "shape" => "rectangular");
extract($array, EXTR_PREFIX_SAME, "book");
echo "Cover: $cover, Book Shape: $book_shape,
```

Searching for elements in array

- The in_array( ) function returns true or false, depending on whether the first argument is an element in the array given as the second argument:

in_array(*to_find*, *array*)

Printing arrays

- print_r()
  - **print_r()** displays information about a variable in a way that's readable by humans.

var_dump()

- This function displays structured information about one or more expressions that includes its type and value.
- Example :-

```
<?php
$a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y',
 'z'));
print_r ($a);
?>
Array ( [a] => apple [b] => banana [c] => Array ( [0] => x [1]
=> y [2] => z ) )
```

Cont...

- Example :-

```
<?php
  $a = array(1, 2, array("a", "b", "c"));
  var_dump($a);
?>
```

```
array(3)
{ [0]=> int(1)
  [1]=> int(2)
  [2]=> array(3)
            { [0]=> string(1) "a"
              [1]=> string(1) "b"
              [2]=> string(1) "c" }
 }
```

Sorting one array

Effect	Ascending	Descending
Sort array by values, then re-assign indexes starting with 0	sort( ) (sort.php)	rsort( )
Sort array by values	asort( ) (asort.php)	arsort( )
Sort array by keys	ksort( ) (ksort.php)	krsort( )