❖ Basic PHP Syntax

- ✓ A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.
- ✓ On servers with shorthand support enabled you can start a scripting block with <? and end with ?>. But for maximum compatibility use the standard form (<?php) rather than the shorthand form.
- ✓ A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.
- ✓ Consider an example of a simple PHP script which sends the text "Hello World" to the browser:
- ✓ <html>
  <body>
  <?php echo "Hello World";
  ?>
  </body>
  </html>
- ✓ Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.
- ✓ There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".
- ✓ **Note:** The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.
- ✓ In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

❖ Variables in PHP

- ✓ 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 syntax of declaring a variable in PHP:
  $var_name = value;
  Example-
  <?php
  $txt="PHP is wonderful";
  $num=100;
  ?>
- ✓ PHP is a Loosely Typed Language (in PHP, a variable does not need to be declared before adding a value to it, the variable is declared automatically when you use it).
- ✓ In a strongly typed programming language (like C, C++ etc.), you have to declare (define) the type and name of the variable before using 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.
- ✓ Naming Rules for Variables
- ✓ 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 an underscore ($my_string), or with capitalization ($myString)
- ✓ String Variables in PHP
- ✓ String variables are used for values that contain characters.
- ✓ A string can be used directly in a function or it can be stored in a variable.
- ✓ **The Concatenation Operator**
- ✓ There is only one string operator in PHP.
- ✓ The concatenation operator (.) is used to put two string values together.
- ✓ To concatenate two string variables together, use the concatenation operator:
- ✓ <?php
  $txt1="The first string";
  $txt2="is now concatenated with the second string!";
  echo $txt1 . " " . $txt2;
  ?>
- ✓ Here the concatenation operator is used two times to insert a third string (a space character to separate the two strings).
- ✓ **The strlen() function**
- ✓ The strlen() function is used to return the length of a string.
- ✓ <?php
  echo strlen("I like PHP");// output is10
  ?>
- ✓ The length of a string is often used in loops or other functions, when it is important to know when the string ends.
- ✓ **The strpos() function**
- ✓ The strpos() function is used to search for a character/text within a string.
- ✓ If a match is found, this function will return the character position of the first match. If no match is found, it will return FALSE.
- ✓ <?php
  echo strpos("Hello world!","world");
  ?>
- ✓ The output of the code above will be:6
- ✓ The position of the string "world" in the example above is 6. The reason that it is 6 (and not 7), is that the first character position in the string is 0, and not 1.

- ✓ Conditional Statements
- ✓ Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
- ✓ In PHP we have the following conditional statements:
  1. **if statement** - use this statement to execute some code only if a specified condition is true
  2. **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
  3. **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed

4. **switch statement** - use this statement to select one of many blocks of code to be executed

- ✓ **The if Statement-** Use the if statement to execute some code only if a specified condition is true.
- ✓ Syntax- if (*condition*) *code to be executed if condition is true;*
- ✓ 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>
```
- ✓ Notice that there is no ..else.. in this syntax. The code is executed only if the specified condition is true.
- ✓ **The if...else Statement-** Use the if....else statement to execute some code if a condition is true and another code if a condition is false.
- ✓ **Syntax-** if (*condition*)
  code to be executed if condition is true;
  else
  code to be executed if condition is false;
- ✓ The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":
- ✓
```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```
- ✓ If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces{}
- ✓ **The if...elseif....else Statement-**Use the if....elseif...else statement to select one of several blocks of code to be executed.
- ✓ 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;
- ✓ 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>
```
- ✓ **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;
 default:
 code to be executed if n is different from both label1 and label2;
}
```

- ✓ This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.
- ✓

```
<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>
```

❖ **Array**

✓ 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.

✓ An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

✓ Each element in the array has its own index so that it can be easily accessed.

✓ In PHP, there are three kind of arrays:

1. **Numeric array** - An array with a numeric index
2. **Associative array** - An array where each ID key is associated with a value
3. **Multidimensional array** - An array containing one or more arrays

✓ **Numeric Arrays-**

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

✓ There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

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

2. In the following example we assign the index manually:

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

✓
```
<?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.

✓ **Associative Arrays-**

✓ 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.

✓ Example 1-In this example we use an array to assign ages to the different persons:

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

✓ Example 2-This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

✓ 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.

- ✓ **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.
- ✓ 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"
  )
  );
```

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
   )
)
```

- ✓ Example 2- 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?

- **current()** and **next()** are used to return the value of current index and to move the pointer to point the next index respectively. The following example shows their working details-

```
<?php
 $a=array(1,2,3);
echo current($a);  //outputs 1
next($a);
echo current($a);  //outputs 2
next($a);
echo current($a);  //outputs 3
?>
```

❖ **PHP Loops**

✓ 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 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 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

✓ **The while Loop**

✓ The while loop executes a block of code while a condition is true.

✓ **Syntax-**

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

✓ **Example-** 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>
```

- ✓ **The do...while Statement**
- ✓ 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.

- ✓ **Syntax-**

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

- ✓ Example-The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

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

- ✓ **The for Loop**
- ✓ 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;
 }
```

- Parameters:-*init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop), *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends, *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)
- ✓ **Note:** Each of the parameters above can be empty, or have multiple expressions (separated by commas).

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

```
?>
</body>
</html>
```

- **The foreach Loop**

✓ The foreach loop is used to loop through arrays.

✓ **Syntax-**

```
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.

```
<html>
<body>
<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>
</body>
</html>
```

❖ **PHP Functions**

✓ To keep the script from being executed when the page loads, you can put it into a function.

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

✓ You may call a function from anywhere within a page.

- **Create a PHP Function**

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

✓ **Syntax-**

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

✓ **Note-**Give the function a name that reflects what the function does

✓ **Remember-**The function name can start with a letter or underscore (not a number)

```
<html>
<body>
<?php
function writeName()
{
echo "xyz";
```

```
}

echo "My name is ";
writeName();
?>
</body>
</html>
```

- ✓ 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.
- ✓ Example 1-The following example will write different first names, but equal last name:

```
<html>
<body>
<?php
function writeName($fname,$punctuation)
{
echo $fname . " abc" . $punctuation . "<br />";
}
echo "My name is ";
writeName("xyz ",".");
echo "My sister's name is ";
writeName("def","!");
echo "My brother's name is ";
writeName("rst","?");
?>
</body>
</html>
```

- ✓ To let a function return a value, use the return statement.

```
<html>
<body>
<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}
echo "1 + 16 = " . add(1,16);
?>
</body>
</html>
```

- ❖ **PHP isset() and unset() method-** The isset() method returns true if a variable is active otherwise returns false (which is basically NULL). Whereas the unset() method is used to inactive a variable. Consider the following example-

```
<?php
 $x=1
 if(isset($x))
```

```
echo “variable x is set”
?>
```

❖ **PHP type casting-** Use gettype() method to know the type of a variable and settype() method to change it to a new type. Consider the following example-

```
<?php
   $x=1
echo gettype($x); //output is integer
settype($x,“string”);
echo gettype($x);//output is string
?>
```

- **Note:** A C-like type casting is also permitted in PHP. Consider the following example-

```
<?php
   $x=“This is a string”;
   $y=(int)$x;
    echo gettype($y); //output is integer
?>
```

- To check the type of a variable, consider the following functions- is_array($variable) , is_bool($variable), is_float($variable),is_int($variable), is_null($variable), is_numeric($variable), is_string($variable)
- If true then these functions return 1.

❖ In PHP a constant is defined as follows-
define(“PIE”,3.14); //this is similar to writing #define PIE 3.14 in C language.

❖ PHP global variable-

```
<?php
$x=10;                          //x is declared outside the function, global
function demo()
{
 $x=20;                         //x is declared inside the function, local
 echo "this is local x=>".$x;   //outputs 20
 global $x;                      //use global x
 echo "this is global x=>".$x;  //outputs 10
}
demo();
?>
```

❖ PHP default function argument-

```
<?php
   function cal($x=1,$y="hello")
   {
    echo $y.$x;
    echo "</br>";
   }

   cal();              //outputs hello1
   cal(2,"kello");   //outputs kello2
 ?>
```