

## UNIT-V

### IMPLEMENT SERVER SIDE SCRIPTING USING PHP

#### **INTRODUCTION TO PHP:**

- PHP stands for **PHP: Hypertext Preprocessor**
- PHP is a widely-used, open source scripting language
- PHP is free to download from PHP resource: [www.php.net](http://www.php.net)
- PHP files can contain text, HTML, JavaScript code, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have a default file extension of ".php"
- PHP can create, open, read, write, and close files on the server
- PHP can restrict users to access some pages on your website
- PHP can encrypt data
- PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)

#### **CHARACTERISTICS OF PHP:**

- simplicity
- Efficiency
- Security
- Flexibility
- Familiarity.

#### **5.1 Understand the installation of PHP**

To start using PHP, you need

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host with PHP Support, If your server has activated support for PHP you do not need to do anything. Just create some .php files, place them in your web directory, and the server will automatically parse them for you. You do not need to compile anything or install any extra tools. Because PHP is free, most web hosts offer PHP support. Fortunately from online we have XAMPP software to run php files.

**XAMPP** stands for Cross-Platform (X), Apache (A), MySQL (M), PHP (P) and Perl (P). It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing purposes. Everything you need to set up a web server – server application (Apache), database (MySQL), and scripting language (PHP) – is included in a simple extractable file.

XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server is extremely easy as well. Web development using XAMPP is especially beginner friendly, as this popular PHP and MySQL for beginners.

#### **How to Install XAMPP for Windows**

##### **1.Download XAMPP**

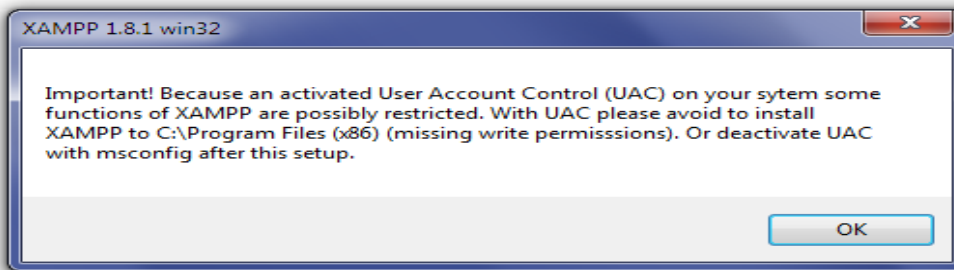
Download the installer file for the latest version of XAMPP, and save the file to your computer.

XAMPP for Windows 1.8.1, 30.9.2012		
Version	Size	Content
<b>XAMPP Windows 1.8.1</b>		Apache 2.4.2, MySQL 5.5.27, PHP 5.4.7, OpenSSL 1.0.1c, phpMyAdmin 3.5.2-2, XAMPP Control Panel 3.1.0, Webalizer 2.23-04, Mercury Mail Transport System v4.62, FileZilla FTP Server 0.9.41, Tomcat 7.0.30 (with mod_proxy_ajp as connector), Strawberry Perl 5.16.0.1 Portable For Windows 2000, XP, Vista, 7.
☑ Installer	99 MB	Installer MD5 checksum: 2c067c31725fda3c71c6d43483b4df4c
☑ ZIP	184 MB	ZIP archive MD5 checksum: 924e9cdc0fc49984e0c4916aa8f31c18
☑ 7zip	84 MB	7zip archive MD5 checksum: 462f6bc3c9e96a8c9228927ff8e0d217

## 2. Installing XAMPP

Next, you need to open the folder where you saved the file, and double-click the installer file.

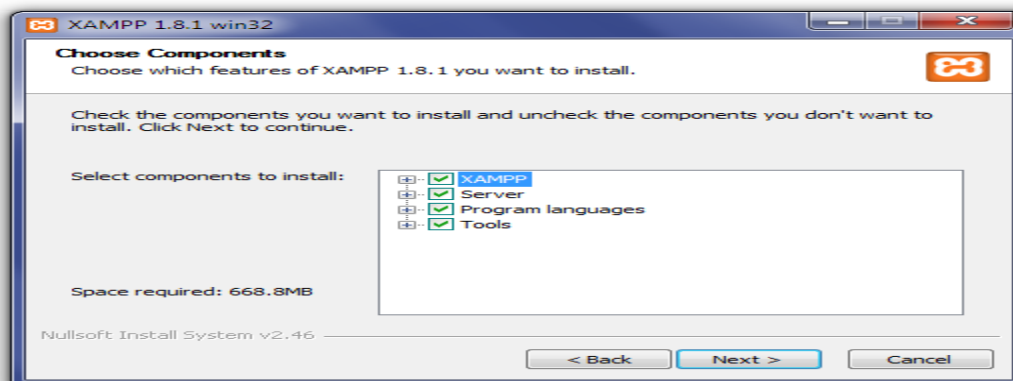
For Windows 7 users, you will see a window pop up, warning you about User Account Control (UAC) being active on your system. **Click OK** to continue the installation.



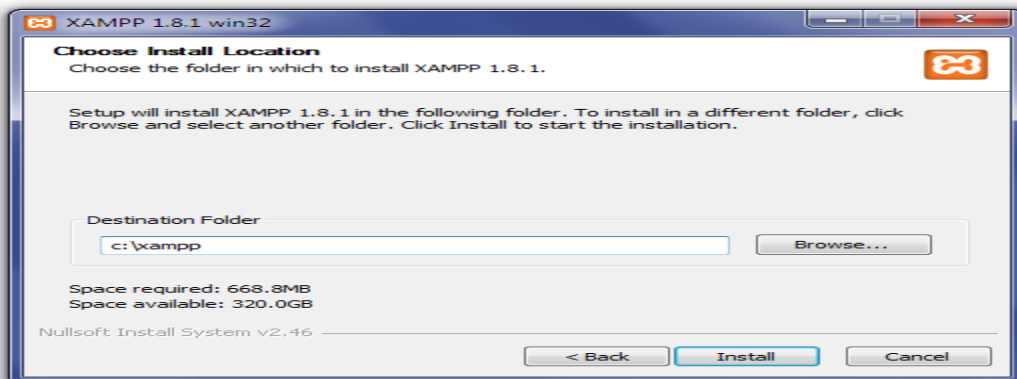
Next you will see the Welcome To The XAMPP Setup Wizard screen. **Click Next** to continue the installation.



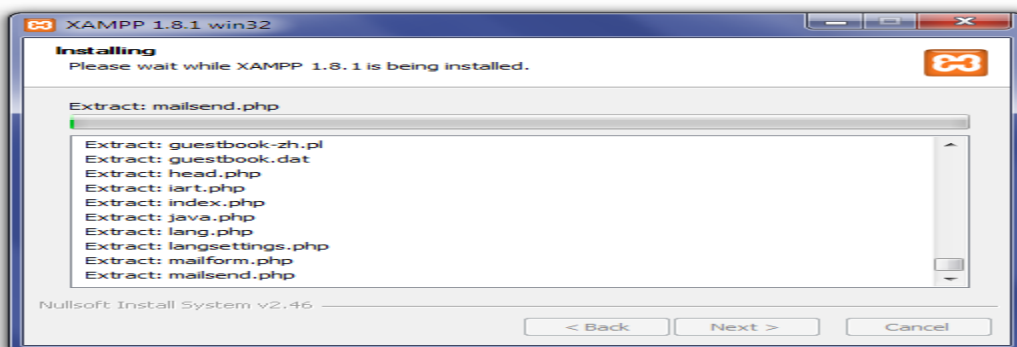
The Choose Components screen will appear next. This screen will allow you to choose which components you would like to install. To run XAMPP properly, all components checked need to be installed. **Click Next** to continue.



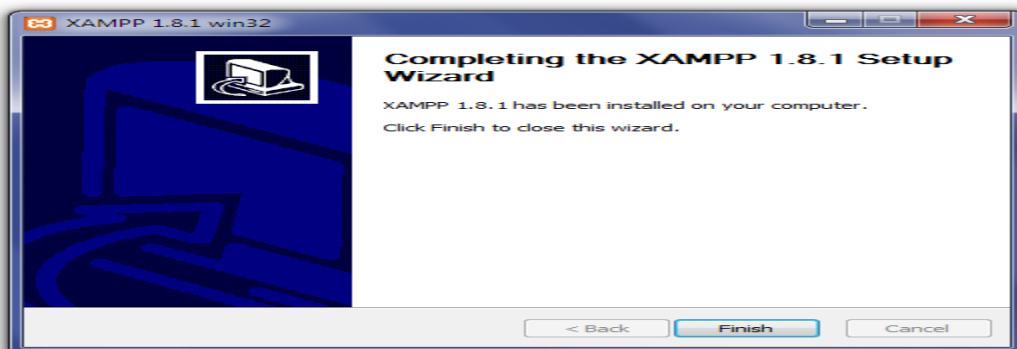
Next you will see the Choose Install Location screen. Unless you would like to install XAMPP on another drive, you should not need to change anything. **Click Install** to continue.



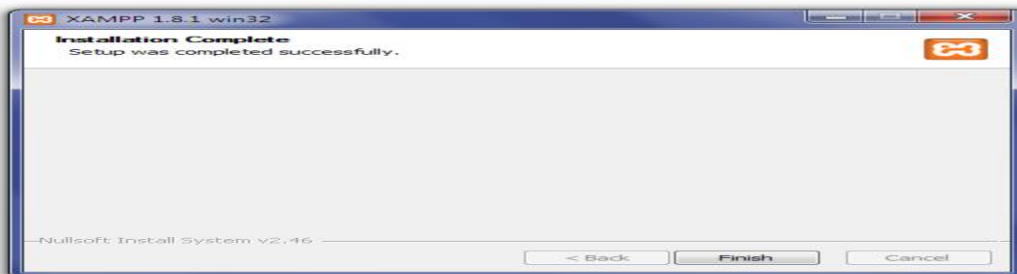
XAMPP will begin extracting files to the location you selected in the previous step.



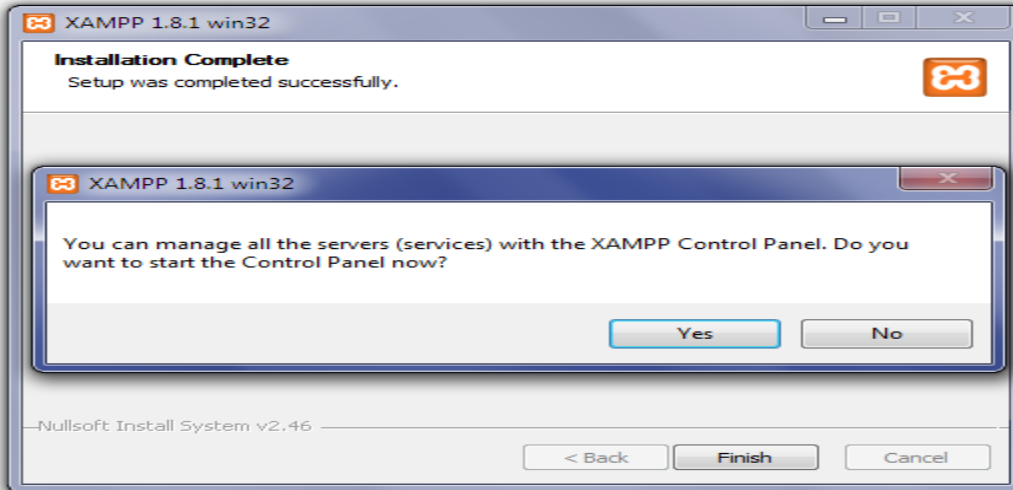
Once all of the files have been extracted, the Completing The XAMPP Setup Wizard screen will appear. **Click Finish** to complete the installation.



The Installation Complete screen will now appear. **Click Finish** to begin using XAMPP.

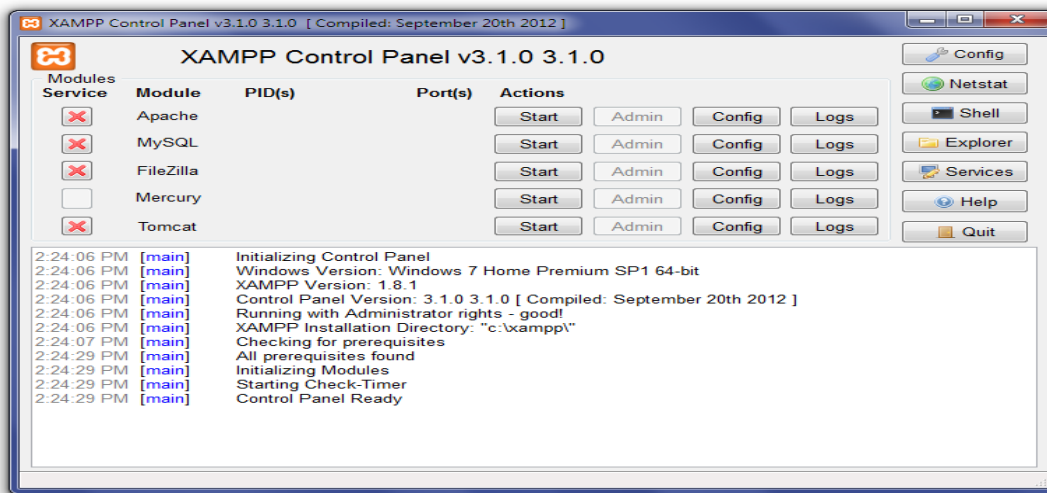


After clicking Finish in the previous screen, you will be asked if you want to open the XAMPP Control Panel. **Click Yes.**

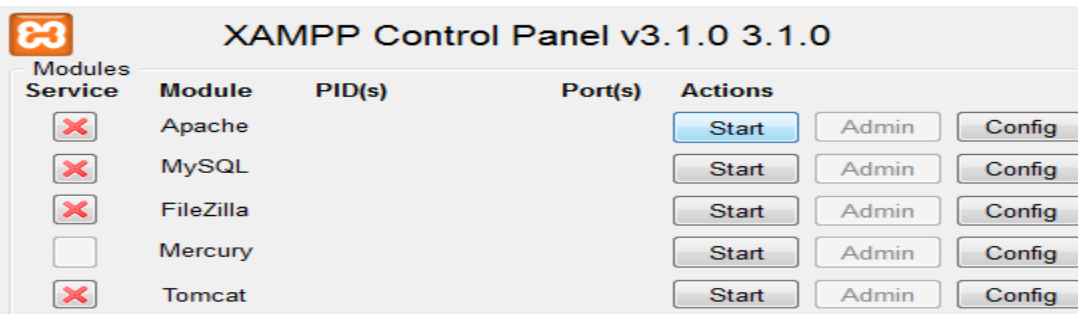


### 3. Starting XAMPP

The XAMPP Control Panel allows you to manually start and stop Apache and MySQL, or install them as services.



To start Apache or MySQL manually, **click the Start button** under **Actions** next to that module.



## 5.2 EXPLAIN THE FUNDAMENTALS OF PHP

### Basic PHP Syntax:

PHP allows user to create dynamic web pages by embedding PHP scripts into a HTML page by using `<?php` and `?>` tags.

A file can be saved with extensions `.php`, `.php3`, `.phtml` e.t.c

Each PHP statement must end with a semicolon(`;`).

**Echo** is used to display text on the browser.

A PHP script can be placed anywhere in the document.

A PHP script starts with `<?php` and ends with `?>`:

### SAMPLE PROGRAM:

```
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. With PHP, there are two basic statements to output text in the browser: **echo** and **print**.

### 5.2.1 Combine PHP Code and HTML Code

PHP and HTML can be combined in two ways.

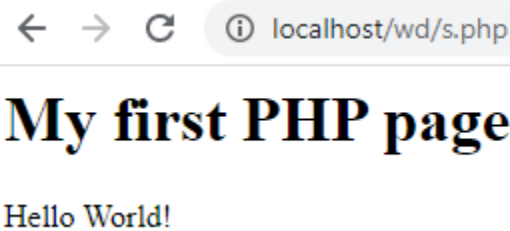
- 1) Embedding PHP code into an HTML file.
- 2) Embedding HTML code into PHP

### Example for embedding PHP code into HTML file

```
<html>
<head>
<title>my php program</title>
</head>
<body>
<?php
    print "welcome to php";
    echo "hi";
?>
</body>
</html>
```

### Embedding HTML code into PHP

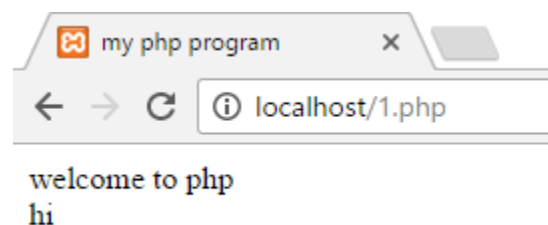
```
<?php
echo "<html>";
echo "<body>";
echo "<title>My PHP</title>";
echo "<body>";
echo "<h1>welcome to php</h1>";
echo "</body>";
echo "</html>";
?>
```



← → ↻ ⓘ localhost/wd/s.php

# My first PHP page

Hello World!



my php program ×

← → ↻ ⓘ localhost/1.php

welcome to php  
hi



My PHP × +

← → ↻ | localhost/ht.php

welcome to php

## 5.2.2 List and Explain Various Data Types

### PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean

**String:** A string is a sequence of characters, like "Hello world!". PHP does not have a character data type, like other languages. therefore, a single character is also considered as a string of length 1. The maximum length of a string is unlimited,.

**Integer:** An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

**Float:** A float (floating point number) is a number with a decimal point or a number in exponential form.

**Boolean:** A Boolean represents two possible states: TRUE or FALSE.

### 5.2.3 Declare Variables and Constants:

**VARIABLES:** Variables are "containers" for storing information.

**Syntax:** \$variable\_name=value;

**Rules for PHP variables:**

- A variable starts with the \$ sign, followed by the name of the variable
- A variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z,0-9, and \_ )
- A variable name should not contain spaces
  
- Variable names are case sensitive (\$y and \$Y are two different variables)
- PHP automatically converts the variable to the correct data type

```
$txt="Hello world!";
```

```
$x=5;
```

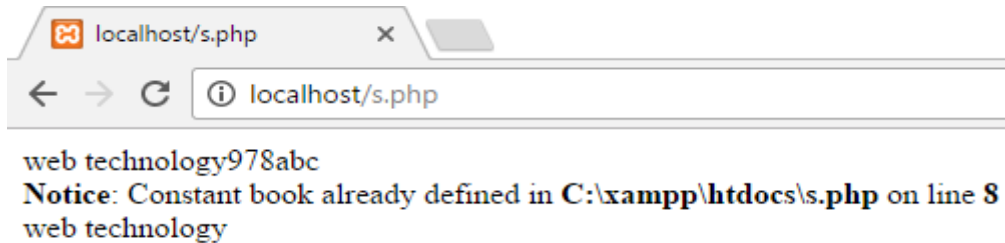
**CONSTANTS:**

- The PHP constants are similar to PHP variables
- They can be created by using the combination of upper case letters,lower case letters,numbers and underscores.
- The scope of PHP constants is global hence they can be accessed anywhere in a PHP script.
- The PHP constants can be created by using a built-in function called define().
- The function define() has two arguments.
- The first argument defines the name of the constant whereas the second argument defines the value of the constant.
- Once the value of a constant is defined, it cannot be changed throughout the program by usingthe define() function.

**Syntax:** define(string name, mixed value)

**Example:**

```
<?php
define('book',"web technology");
define('wd',"978");
define('AUTHOUR',"abc");
print book."<br>";
print wd."<br>";
print AUTHOUR."<br>";
define('book',"concepts of Programming language");
print book;
?>
```

**Output:****5.2.4 Use various Operators:**

**Operator:** Operator is a symbol that specifies and performs certain action on the variables and value.

**PHP Arithmetic Operators:** Used to perform various mathematical operations.

Operator	Name	Description	Example	Result
$x + y$	Addition	Sum of x and y	$2 + 2$	4
$x - y$	Subtraction	Difference of x and y	$5 - 2$	3
$x * y$	Multiplication	Product of x and y	$5 * 2$	10
$x / y$	Division	Quotient of x and y	$15 / 5$	3
$x \% y$	Modulus	Remainder of x divided by y	$5 \% 2$	1
			$10 \% 8$	2
			$10 \% 2$	0
$- x$	Negation	Opposite of x	$- 2$	
$a . b$	Concatenation	Concatenate two strings	"Hi" . "Ha"	HiHa

**PHP Assignment Operators:** Used to assign a value to variable.

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
<code>a .= b</code>	<code>a = a . b</code>	Concatenate two strings

**PHP Incrementing/Decrementing Operators:** Used to performing incrementing and Decrementing values

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

**PHP Comparison Operators:** Used to compare numerical values

Operator	Name	Description	Example
<code>x == y</code>	Equal	True if x is equal to y	<code>5==8</code> returns false
<code>x === y</code>	Identical	True if x is equal to y, and they are of same type	<code>5=== "5"</code> returns false
<code>x != y</code>	Not equal	True if x is not equal to y	<code>5!=8</code> returns true
<code>x &lt;&gt; y</code>	Not equal	True if x is not equal to y	<code>5&lt;&gt;8</code> returns true
<code>x !== y</code>	Not identical	True if x is not equal to y, or they are not of same type	<code>5!== "5"</code> returns true
<code>x &gt; y</code>	Greater than	True if x is greater than y	<code>5&gt;8</code> returns false
<code>x &lt; y</code>	Less than	True if x is less than y	<code>5&lt;8</code> returns true
<code>x &gt;= y</code>	Greater than or equal to	True if x is greater than or equal to y	<code>5&gt;=8</code> returns false
<code>x &lt;= y</code>	Less than or equal to	True if x is less than or equal to y	<code>5&lt;=8</code> returns true



**PHP Logical Operators:** Used to perform logical operations

Operator	Name	Description	Example
x and y	And	True if both x and y are true	x=6 y=3 (x < 10 and y > 1) returns true
x or y	Or	True if either or both x and y are true	x=6 y=3 (x==6 or y==5) returns true
x xor y	Xor	True if either x or y is true, but not both	x=6 y=3 (x==6 xor y==3) returns false
x && y	And	True if both x and y are true	x=6 y=3 (x < 10 && y > 1) returns true
x    y	Or	True if either or both x and y are true	x=6 y=3 (x==5    y==5) returns false
! x	Not	True if x is not true	x=6 y=3 !(x==y) returns true

**Concatenation Operator:**

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

The example below shows how to concatenate two string variables together

```
<?php
$txt1="Hello world!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

**5.3 IMPLEMENT VARIOUS LOOP STATEMENTS WITH EXAMPLES:** A loop statement allows us to

execute a statement or group of statements multiple times

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

**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 first sets a variable *i* to 1 (\$i=1;).

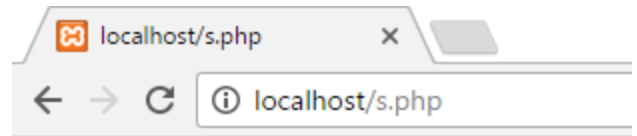
Then, the while loop will continue to run as long as *i* is less than, or equal to 5. *i* will increase by 1 each time the loop runs:

```

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

```

### Output:



```

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

```

### 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

```

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

```

### Output:



```

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

```

### For Loop:

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

### Syntax :

```

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

```

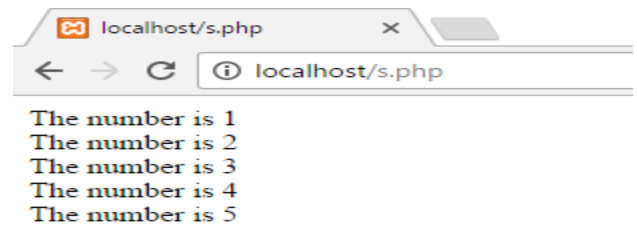
### Parameters:

- **initialization:** 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 iteration)

### Example

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



### foreach Loop:

The foreach loop is used to loop through arrays.

### Syntax

**foreach (\$array as \$value)**

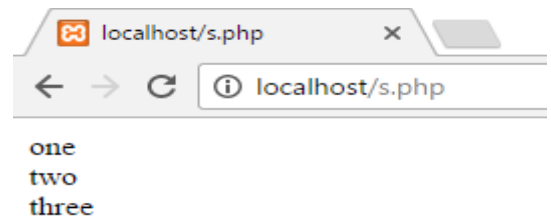
```
{
    code to be executed;
}
```

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.

### Example

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>
```



## 5.4 IMPLEMENT VARIOUS CONDITIONAL STATEMENTS WITH EXAMPLES

Conditional statements are used to execute a statement or a group of statement based on certain conditions.

In PHP we have the following conditional statements:

- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes code if a condition is true and another code if the condition is false
- **if...else if...else statement** - selects one of several blocks of code to be executed
- **switch statement** - selects one of many blocks of code to be executed

### if Statement:

The if statement is used to execute some code **only if a specified condition is true.**

## Syntax

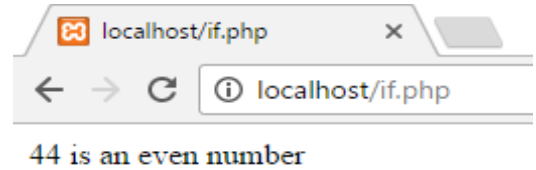
**if** (*condition*)

```
{  
  code to be executed if condition is true;  
}
```

The example below will output "Have a good day!" if the current time is less than 20:

### Example

```
<?php  
$t=3;  
if ($t<20)  
{  
  echo "Have a good day!";  
}  
?>
```



### 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

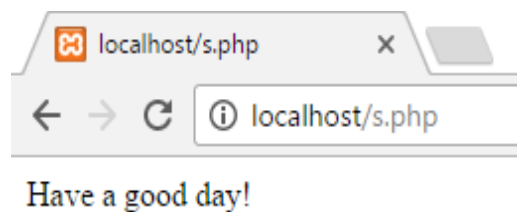
**if** (*condition*)

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

The example below will give output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

### Example:

```
<?php  
$number=44;  
if(($number%2)!=0)  
  echo "$number is an odd number";  
else  
  echo "$number is an even number";  
?>
```



### if...else if...else Statement:

Use the if...else if...else statement to **select one of several blocks of code to be executed.**

## Syntax

**if** (*condition*)

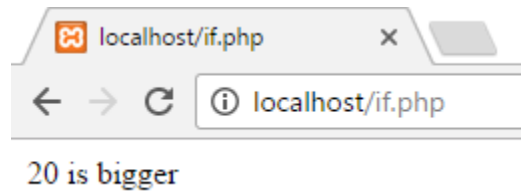
```
{  
  code to be executed if condition is true;  
}
```

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

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

### **Example**

```
<?php
$a=10;$b=15;$c=20;
if (($a>$b)&&($a>$c))
    echo "$a is bigger";
else if ($b>$c)
    echo "$b is bigger ";
else
    echo "$c is bigger ";
?>
```



### **Switch statements:**

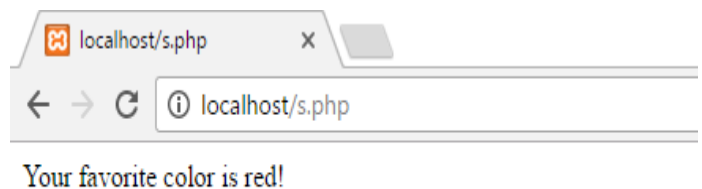
The switch statement is used to perform different actions based on different conditions.

### **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;
}
```

### **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!";}
```

```
?>
```

## **5.5 UNDERSTAND STRING MANIPULATION USING STRING FUNCTIONS**

### **String functions:**

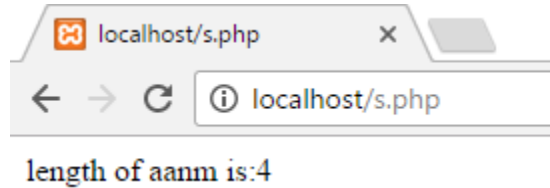
PHP has several functions for string manipulation. Some of the most commonly used functions are discussed below.

**a)strlen():**The strlen() function returns the number of characters in a string

**Syntax:** int strlen(string \$str)

#### **sample program:**

```
<?php
$org="aanm";
print"length of $org is:";//prints 4
print strlen($org);
?>
```



**b)strcmp():** The strcmp() function compares two strings based on their ASCII values. The comparison is case sensitive.

It returns one of the following values:

0 if both strings are equal

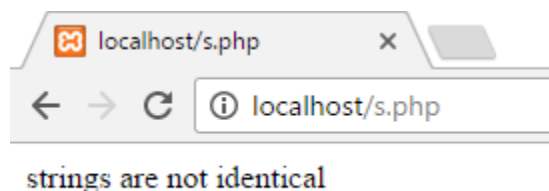
-1 if str1 less than str2

1 if str1 is greater than str2

**Syntax:**int strcmp(string \$str1,string \$str2)

#### **Example**

```
<?php
$str1="jan";
$str2='january';
if(strcmp($str1,$str2)!=0)
print "strings are not identical";
?>
```

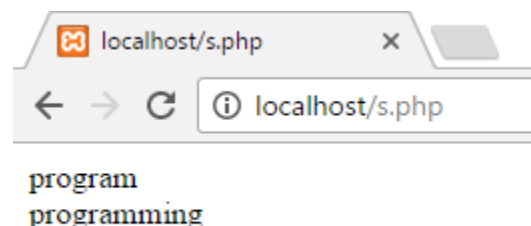


**c)substr():** The substr() function, returns the substring starting from the start position,till the end of the string,if length is not specified then the substring consists of the characters between the start and start+length.

**Syntax:** string substr(string \$str,int \$start[,int \$length])

#### **Example**

```
<?php
$sub="unix programming";
print substr($sub,5,7);//prints program
print"<br/>";
print substr($sub,5);//prints programming
?>
```

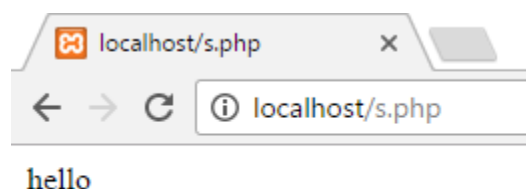


**d)strtolower():** The strtolower() function converts all characters in the given string to lowercase letters and returns the modified string.

**Syntax:**String strtolower(string \$str)

#### **Example**

```
<?php
$var='HELLO';
print strtolower($var);//prints hello
?>
```

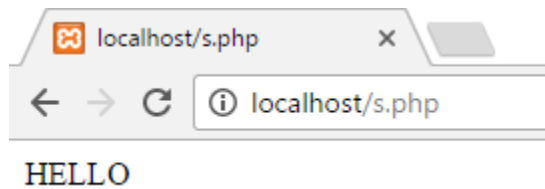


**e)strtoupper()**The strtoupper()function,converts all the characters in the given string to uppercase letters and returns the modified string

**Syntax:**string strtoupper(string \$str)

**Example**

```
<?php
$var="hello";
print strtoupper($var);//prints HELLO
?>
```



**f)trim():**The trim() function removes whitespace and other predefined characters from both sides of a string.Related functions

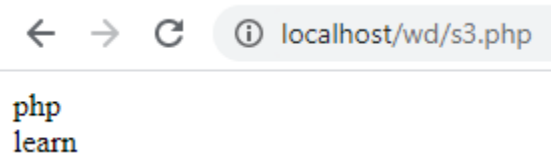
**ltrim()** - Removes whitespace or other predefined characters from the left side of a string

**rtrim()** - Removes whitespace or other predefined characters from the right side of a string

**Syntax:**string trim(string \$str[,string list-of-chars])

**sample program:**

```
<?php
$str="learn php";
echo ltrim($str,"learn")."<br>";
echo rtrim($str,"php")."<br>";
?>
```



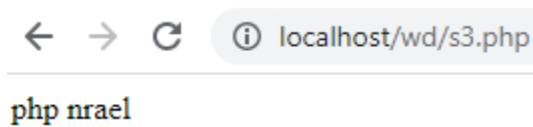
**g)str\_word\_count():**-This function counts the number of words in a string.

**Example:**

```
<?php
$str="learn php";//o/p:2
echo str_word_count($str);
?>
```

**h)strrev():** This function reverses a string.

```
<?php
$str="learn php";
echo strrev($str);
?>
```



**i)str\_replace() :-** This function replaces some characters with some other characters in a string.

**Example**

```
<?php
$str="learn php";
echo str_replace($str,'php','PHP');
?>
```

**j)ucfirst():**-The ucfirst() function converts the first character of a string to uppercase.

```
<?php
$str="learn php";
echo ucfirst($str);
?>
```

**k)ucwords() :**The ucwords() function converts the first character of each word in a string to uppercase.

```
<?php
$str="learn php";
echo ucwords($str);
?>
```

**5.7 IMPLEMENT ARRAYS**

Arrays in PHP are flexible and built-in data structures that are used to store multiple data elements.

**Types of arrays:**

In PHP, there are three types of arrays

- **Indexed arrays** - Arrays with numeric index

**Index array contains two types**

1. Single dimensional array
2. multidimensional arrays

- **Associative arrays** - Arrays with named keys

**Associative array contains two types**

1. Single dimensional array
2. multidimensional arrays

- **Multidimensional arrays** - Arrays containing one or more arrays

### **5.7.1, 5.7.2 Single and multidimensional arrays, declaration of arrays:**

**Index array:** Arrays with numeric index

**Single dimensional array:** A list of variables can be represented using single subscript is called single dimensional array.

There are two ways to create indexed single dimensional arrays:

**First method :** The index can be assigned automatically (index always starts at 0):

**Creation of Single dimensional indexed array:** `$arrayname=array(value1,value2,.....,valuen);`

**Ex:** `$fruits=array("mango","banana","apple","orange");`

**Accessing an array element:**

The brackets with subscript of key is used to access an individual array element.

Eg:-`$fruits[0]=mango;`

**second method :**

the index can be assigned manually:

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

The following example creates an indexed array named \$cars, assigns three elements to it, and then prints a text containing the array values:

**Example:**

```
<?php  
$fruits=array("mango","orange","apple","banana");  
echo"<h1>by using foreach loop</h1></br>";  
foreach($fruits as $i)  
{  
echo"$i</br>";  
}  
?>
```

← → ↻ ⓘ localhost/wd/s3.php

**by using foreach loop**

```
mango  
orange  
apple  
banana
```

**multidimensional arrays:** An array contain more than one array is called multidimensional array.

**Creation of Indexed multi dimensional array:**

`$arrayname=array(array(value1,value2,.....,valuen), array(value1,value2,.....,valuen),.....);`

**Example:** `$arr=array(array(1,2,3,4), array(5,6,7,8),array(9,10,11,12));`

**Accessing an array element:**

Eg:-`$arr[0][0]=1;`

`$arr[2][1]=10;`

**Sample program:**

```
<?php  
$arr=array(array(1,2,3),array("abc","xyz","def"),array(18,19,20));
```



```

echo"<h1>by using foreach loop</h1></br>";
foreach($arr as $i)
{
foreach($i as $j)
{
echo"$j </br>";
}
}
?>

```

← → ↻ ⓘ localhost/wd/s3.php

## by using foreach loop

```

1
2
3
abc
xyz
def
18
19
20

```

### PHP Associative Arrays:

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

**Single dimensional array:** A list of variables can be represented using single subscript is called single dimensional array.

There are two ways to create an associative single dimensional array:

#### Creation of Single dimensional Associative array:

**\$arrayname=array(key=>value1,key=>value2,.....,key=>valuen);**

\$marks=array("Peter"=>90,"Ben"=>100,"Joe"=>95);

or:

```
$marks['Peter']=66;
```

```
$marks['Ben']=95;
```

```
$marks['Joe']=56;
```

The named keys can then be used in a script:

#### Accessing an array element:

```
$marks['peter']=90;
```

```
$marks['ben']=100;
```

```
$marks['joe']=95;
```

#### Example

```
<?php
```

```
$marks=array("mani"=>90,"sai"=>80,"deep"=>70);
```

```
echo"<h1>by using foreach loop</h1></br>";
```

```
foreach($marks as $i=>$j)
```

```
{
```

```
echo"$i=>$j</br>";
```

```
}
```

```
?>
```

← → ↻ ⓘ localhost/wd/s3.php

## by using foreach loop

```

mani=>90
sai=>80
deep=>70

```

### Multidimensional Arrays

A multidimensional array is an array containing one or more 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.

**Creation of multi dimensional Associative array:** \$arrayname=array(key=>

array(key=>value1,key=>value2,.....,key=>valuen),.....);

**Example:-** `$marks=array(("Peter"=>array("os"=>90,"ds"=>80), "Ben"=> array("os"=>80,"ds"=>80), "Joe"=> array("os"=>60,"ds"=>80));`

### Accessing an array element:

```
$marks['peter']['os']=90;
```

```
$marks['ben']['ds']=80;
```

```
$marks['joe']['os']=60;
```

### **Sample program:**

```
<?php
```

```
$marks=array("mani"=>array("wd"=>90,"mp"=>70),"sai"=>array("mp"=>100,"se"=>80));
```

```
echo"<h1>by using foreach loop</h1></br>";
```

```
foreach($marks as $i=>$x)
```

```
{
echo"$i=>$x</br>";
```

```
foreach($x as $j=>$l)
```

```
{
echo"$j=>$l</br>";
```

```
}
```

```
}
```

```
?>
```

← → ↻ ⓘ localhost/wd/s3.php

## by using foreach loop

```
mani=>Array
wd=>90
mp=>70
sai=>Array
mp=>100
se=>80
```

### 5.7.3 Manipulation of array:

PHP provides various functions for dealing with arrays. They are:

**a)unset():**Used to delete an individual element of an array as well as the whole array.

**Eg:** `$rank=array(5,12,10,20);`

```
unset($rank[1]); //This statement deletes 12 from the array
```

```
unset($rank); //This statement deletes the entire array.
```

**b)array\_key\_exists():**This function defines whether an element with specified key exists in the array and returns a Boolean value. It returns TRUE, if the key exists in the array; otherwise it returns FALSE.

**Ex:**`$tmp=array("hyd"=>35,"delhi"=>40);`

```
If(array_key_exists("hyd",$tmp))
```

```
{
```

```
Print "the key hyd exists";
```

```
}
```

**c)in\_array():** This function determines, whether the specified value exists in the array. It takes two parameters an expression and an array. If the expression evaluates to a value which is present in the array, then it returns TRUE; otherwise it returns FALSE.

**Eg:**`$grades=array(50,100,73.9,84);`

```
If(in_array("100",$grades))
```

```
{
```

```
Print "grade 100 is found in the array";
```

```
}
```

### d)is\_array():

This function checks whether the variable is in the array. It returns true, if the variable is in the array, otherwise it returns FALSE.

**Eg:** \$city=array("hyderabad","banglore","delhi","Mumbai");  
If(is\_array(\$city))  
{  
    Print "\$city is an array";  
}

#### **e)array keys():**

This function returns an array, whose elements are the keys of given array. The returned array has numeric keys 0,1,2, and so on.

**Eg:** \$employees=array("ash"=>100,"shaz"=>420);  
\$names=array\_keys(\$employees);  
Now,the elements of array \$names are ash,shaz.

#### **f)array values():**

This function returns an array whose elements are the values of the given array. The returned array has numeric keys 0,1,2 and so on.

**Eg:** \$codes=array\_values(\$employees);  
Now,the elements of array \$codes are 100,420

#### **g)implode():**

This function takes two parameters a separator character and an array. It concatenates the elements of given array, and each separated by a separator character and returns the result to the string.

**Eg:** \$words=array("have","a","good\_","day");  
\$message=implode("",\$words);  
Now, the variable \$message has the value. "have a good-day"

**f)explode():**This function is inverse of implode. It takes two parameters a substring delimiter and a string. It tokenizes the given string into sub string using substring delimiter and the returns substrings in an array.

**Eg:**  
\$message="have a nice day";  
\$words=explode("",\$message);  
Now, the variable \$word contains("have","a","good-","day")

#### **g)sizeof()**

This function,determines the number of elements in an array

**Eg:** \$os=array(1,2,3);  
echo sizeof(\$os);

### **5.8 Implement functions:**

- PHP functions are similar to other programming languages.
- A function is a block of statements that can be repeatedly in a program.
- PHP have two types of functions :
  - 1) User-defined Function.
  - 2) Built-in Function.
- **User-defined Function** : Function created by user .
- **Built-in Function** : Function created by PHP

#### **5.8.1 Define user defined functions:**

- User-defined function is just a name we give to a block of code that can be executed whenever we need it
- A function will be executed by a call to the function.

User defined function will have following elements:-

**1)Keyword “function”:-** User defined function requires this keyword for function declaration. It is placed prior to function name. It is placed prior to function name.

**2)Function name:** It is name of function defined by user.It is written after keyword ‘function’ and followed by braces. A function can start with lowercase letter/underscore with any number of characters, underscores or numbers.

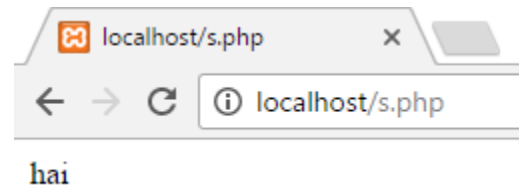
**3)Argument list:** Parameters required for the function.Afunction may have zero or more arguments. These arguments become local variables for the function.

**4)Function code:**Code present with in the function body.Function starts and ends with a curly brace.

**5)Opening and closing braces:**These braces enclose the function code representing the opening and closing of function code.

**Syntax:**

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



**Sample program:**

```
<?php  
function myfunction()  
{  
    echo "hai";  
}  
myfunction();  
?>
```

**5.8.2State The Importance Of User Defined Functions:**

- ✓ It provides modularity to the program.
- ✓ Easy code Reuseability. You just have to call the function by its name to use it.
- ✓ In case of large programs with thousands of code lines, debugging and editing becomes easier if you use functions.

**5.8.3 Describe the process of passing arguments:**

- The parameter that are passed in the call to a function are called actual parameters
- Parameters that are listed in the function definition are called formal parameters.
- Actual parameters can be any expression or variable, but formal parameters must be variables.  
There are two parameter-passing mechanisms ‘pass-by-value’ and ‘pass-by-reference’.

**Pass-by-value:**

- 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
- Copies the actual parameters into formal parameters.

**Syntax:**

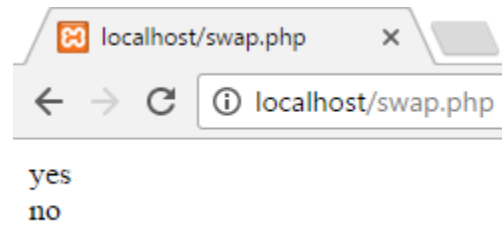
```
<?php  
function myfunction($par1, $par2 ,.....)  
{  
    echo "This is the first function with parameters to me ";  
}  
?>
```

**Example:**

```

<?php
function swap($x,$y)
{
    $z=$x;
    $x=$y;
    $y=$z;
}
$x="no";
$y="yes";
swap($x,$y);
print "$x";
print "<br>";
print "$y";
?>

```



### **Pass-by-reference:**

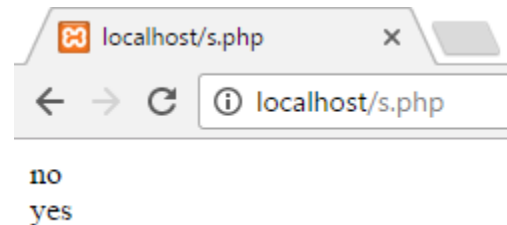
Pass-by-reference, provides two-way communication by passing the address to actual parameters rather than values, to the function.

### **Example:**

```

<?php
function swap(&$x,&$y)
{
    $z=$x;
    $x=$y;
    $y=$z;
}
$x="yes";
$y="no";
swap($x,$y);
print "$x";
print "<br>";
print "$y";
?

```



### **5.8.4.Explain the scope and lifetime of a variable:**

#### **Scope of a variable:**

The visibility of a variable within the script is called scope of that variable.

- PHP has three variable scopes:
  - ✓ local
  - ✓ global
  - ✓ static

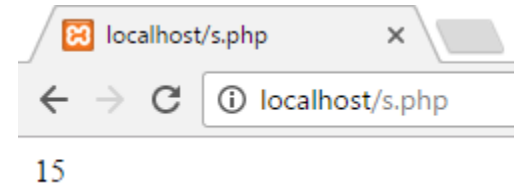
**Local:** A variable defined in the function is called local variable. The scope of variable is local to the function in which it is defined. It cannot be used outside the function.

#### **Global:**

A variable defined outside the function is called global variable. This variable can be accessed from anywhere in the program. Global variables can be accessed from any part of the script, EXCEPT from within a function. To access a global variable from within a function, use the **global** keyword.

### **Example:**

```
<?php
$x=5; // global scope
$y=10; // global scope
function myTest()
{
    global $x,$y;
    $y=$x+$y;
}
myTest();
echo $y; // outputs 15
?>
```



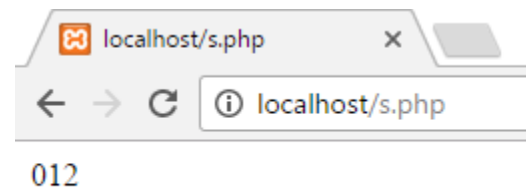
### **Lifetime of a variable:**

The lifetime of a variable in a PHP function begins, when the variable is first used and ends when the function execution terminates. That is, after the termination of function execution, the value of local variable is lost. In order to retain this value throughout different calls to function, we must declare local variable as static.

The lifetime of a local variable in a function begins when the variable is first used in function and ends when the execution ends.

### **Example:**

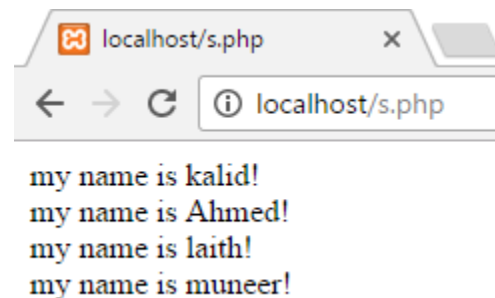
```
<?php
function myTest()
{
    static $x=0;
    echo $x;
    $x++;
}
myTest();
myTest();
myTest();
?>
```



## **5.8.5 small programs using functions:**

### **Sample program on parameter passing:**

```
<?php
function myname($firstName)
{
    echo "my name is ". $firstName . "!<br />";
}
myname("kalid");
myname("Ahmed");
myname("laith");
myname("muneer");
?>
```



### **Function Returning Values :**

In addition to being able to pass functions information, you can also return a value.

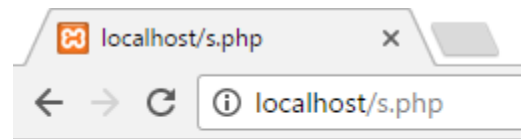
To capture this value you can set a variable equal to the function.

**Syntax:**

```
$myVar = somefunction();
```

**Sample program:**

```
<?php
function mySum($numX, $numY)
{
    return ($numX + $numY);
}
$myNumber = 0;
echo "Before call function, myNumber = " . $myNumber . "<br />";
$myNumber = mySum(3, 4);
echo "After call function, myNumber = " . $myNumber . "<br />";
?>
```



Before call function, myNumber = 0  
After call function, myNumber = 7

**5.9 Implement the concept of accessing databases:**

**5.9.1 Understand basic database concepts:**

**Database:**

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. Database consists of tables. The tables contains records of data in the form of rows and columns.

**Query:**

A query is a request for information from a database.

Examples: select, insert, delete, update.

**5.9.2 steps for connecting to a database:**

In order to connect to a Mysql database, first a link to it must be opened. This can be done with the help of `mysqli_connect()`. After completion of the work the link to the database is closed.

**Syntax:**

```
mysqli_connect(server,user,password);
```

Server	<b>Optional - The host name running database server.</b>
User	<b>Optional - The username accessing the database. If not specified, then default is the name of the user that owns the server process.</b>
Password	<b>Optional - The password of the user accessing the database. If not specified, then default is an empty password.</b>

**Closing connection:**

A connection will be closed automatically when the script ends. To close the connection we use `mysqli_close()`.

**Sample connection:**

```
<?php
$con=mysqli_connect('localhost','root','');
if($con)
```

```

{
echo "connection established";
}
else
{
echo " not connected";
}
?>

```

← → ↻ ⓘ localhost/wd/s3.php

connection established

### **Creating a database:**

A database is created by using CREATE DATABASE statement of MYSQL.

### **Syntax:**

**CREATE DATABASE databasename;**

The above statement is used in function of PHP called mysqli\_query().It will send query to MYSQL connection.

### **Sample program:**

```

<?php
$con=mysqli_connect('localhost','root','');
if(!$con)
{
echo "not connected";
}
echo " connection established</br>";
$s="create database aa";
if(mysqli_query($con,$s))
{
echo"database created";
}
else
{
echo" not created";
}
?>

```

← → ↻ ⓘ localhost/wd/s3.php

connection established  
database created

### **Creation of table:**

A table is created by using CREATE TABLE of MYSQL.

### **Syntax:**

**Create table table\_name(column1 datatype(size),column2 datatype(size),.....);**

This statement must be emebded in the function of PHP called mysqli\_query().

### **Sample program:**

```

<?php
$con=mysqli_connect('localhost','root','');
if(!$con)
{
echo "not connected";
}

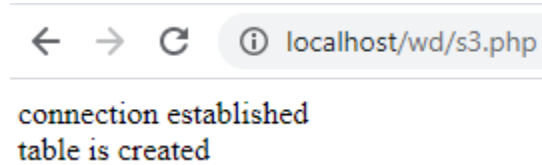
```



```

echo " connection established</br>";
mysqli_select_db($con,'aa');
$q="create table student(sid int,sname varchar(20),sage int)";
if(mysqli_query($con,$q)
{
echo"table is created";
}
else
{
echo"table is not created";
}
?>

```



### **5.9.3 List and explain the steps:**

#### **5.9.3.1 Retrieving data from a table:**

Data can be retrieved by using select statement of mysql.

#### **Syntax:**

**select \* from table name;**

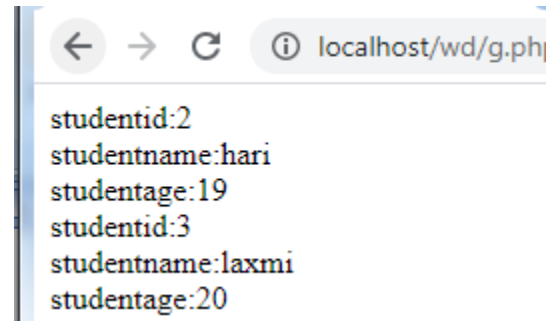
The above statement must be mentioned in mysqli\_query() function.

#### **Sample program:**

```

<?php
$con=mysqli_connect("localhost","root","");
if(!$con)
{
    die("unable to connect");
}
mysqli_select_db($con,'aa');
$sql="select *from student";
$retval=mysqli_query($con,$sql);
while($row=mysqli_fetch_array($retval,MYSQLI_ASSOC))
{
    echo
"studentid:{$row['sid']}<br>". "studentname:{$row['sname']}<br>". "studentage:{$row['sage']}<br>";
}
mysqli_close($con);
?>

```



#### **5.9.3.2 Inserting data into a table:**

Data can be inserted into table by using insert command.

#### **Syntax:**

**insert into tablename values("value1","value2");**

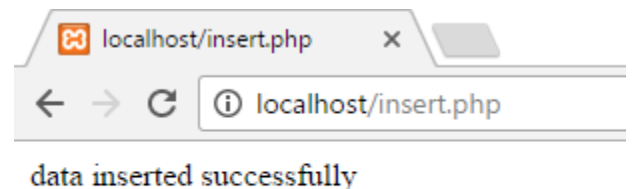
The above syntax must be embedded in mysqli\_query() function of PHP

#### **sample program:**

```

<?php
$con=mysqli_connect('localhost','root','');

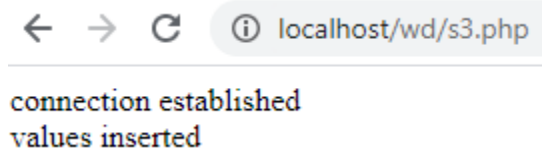
```



```

if(!$con)
{
echo "not connected";
}
echo " connection established</br>";
mysqli_select_db($con,'aa');
$q1="insert into student(sid ,sname,sage)values(1,'sai',18),(2,'hari',19),(3,'laxmi',20)";
if(mysqli_query($con,$q1))
{
echo"values inserted";
}
else
{
echo"values not inserted";
}
?>

```



### **5.9.3.3 Updating data in a table:**

Data in the table can be modified by using update statement

#### **Syntax:**

**Update table\_name set column\_name=new\_value where column\_name=value**

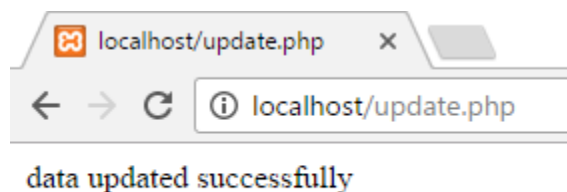
The above statement must be embedded with mysql\_query() function of PHP

#### **Sample program:**

```

<?php
$con=mysqli_connect('localhost','root','');
if(!$con)
{
echo "not connected";
}
echo " connection established</br>";
mysqli_select_db($con,'aa');
$q4="update student set sage=18 where sid=3";
if(mysqli_query($con,$q4))
echo"data updated sucessfully</br>";
else
echo"data not updated</br>";
?>

```



### **5.9.3.4 Deleting data from table:**

Data from table can be deleted by using delete from MYSQL.

#### **Syntax:**

**delete from table\_name where column\_name=value**

#### **Sample program:**

```

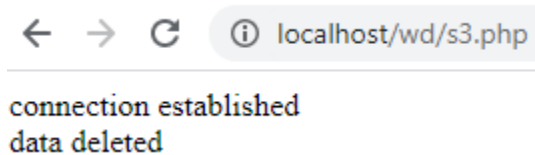
<?php
$con=mysqli_connect('localhost','root','');

```

```

if(!$con)
{
echo "not connected";
}
echo " connection established</br>";
mysqli_select_db($con,'aa');
$q4="delete from student where sid=1";
if(mysqli_query($con,$q4))
echo"data deleted</br>";
else
echo"data not deleted</br>";
}
?>

```



### **5.10.Cookies and sessions:**

- PHP cookie is a small piece of information which is stored at client browser.
- It is used to recognize the user .Cookie is created at server side and saved to client browser.
- Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.
- Each cookie on the user’s computer is connected to a particular domain.
- Each cookie be used to store up to 4kB of data.
- A maximum of 20 cookies can be stored on a user’s PC per domain

#### **Session:**

It is used to store the user information on the server for temporary purpose. The information is gathered from the time user logs in upto the time user logout. This is called as session information. The time period is called session.

It will be deleted whenever the user closes the website.

### **5.10.3 Create and delete a cookie:**

#### **Creating a cookie:**

A cookie can be created in two steps:

#### **a)Setting a cookie:**

A cookie is set in PHP by using the setcookie() function.

-If a cookie was set with name “user”,a variable will be created with name \$user, containing the cookie value.

-setcookie() function has six arguments and should be return before <html>tag.

-For each cookie this function has to be called separately.

#### **Syntax: setcookie(name,value,expire,path,domain,security);**

- **Name** –Sets the name of the cookie and is stored in an environment variable called HTTP\_COOKIE\_VARS. This variable is used while accessing cookies.
- **Value** – Sets the value of the named variable and is the content that actually need to store.
- **Expiry** –Specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** –Specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

- **Domain** – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

### Sample program:

Create two cookies name and age.

```
<?php
    Setcookie("name","john",time()+3600,"",0);
    Setcookie("age","36",time()+3600,"",0);
?>
```

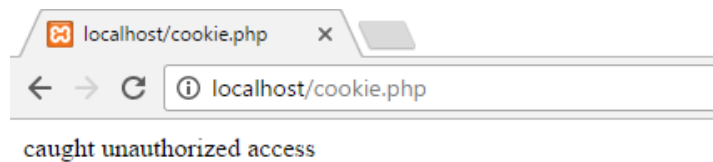
### b)Getting a cookie:

PHP provides two ways to access cookies. By using \$\_COOKIE or \$HTTP\_COOKIE\_VARS variable.

**Syntax:** \$\_COOKIE["cookie name"].

### Sample program:

```
<?php
    $str="caught unauthorized access";
    setcookie("mycookie",$str);
    setcookie("mycookie",$str,time()+60);
    echo $_COOKIE['mycookie'];
?>
```



### Deleting a cookie:

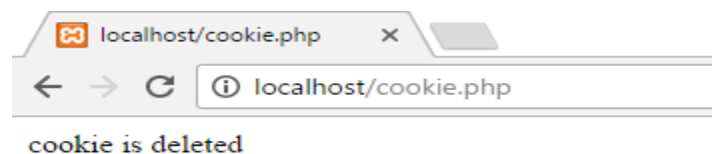
Deleting a cookie is a difficult task. A cookie can be deleted in three ways.

- 1.) A cookie can be deleted by itself by assigning a time to die.
- 2.) If the time to die is not assigned, the cookie will be deleted automatically when the browser gets closed.
- 3.) The cookie gets deleted when the user performs the functionality of "logout" through a user interface.

The following example shows how the user deletes the cookie by performing "logout" functionality.

### Sample program:

```
<?php
    setcookie("user","",time()-60);
    echo "cookie is deleted";
?>
```



### 5.10.4 Query string to pass data:

The information can be sent across the web pages. This information is called query string.

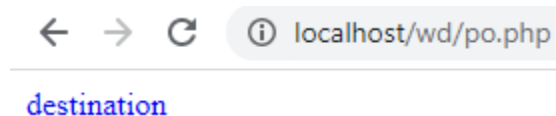
There are two ways to pass the query string

1. By using url
2. By using urlencode()

#### By using url

```
<html>
<body>
<?php

$name="sai";
$age=18;
$email="abc@gmail.com";
$str="name={ $name }&age={ $age }&email={ $email } ";
?>
```



```
<a href="dest.php?<?php echo "$str"; ?>">destination </a>
</body>
</html>
```

### Dest.php

```
<html>
<body>
<?php
$name=$_GET['name'];
$age=$_GET['age'];
$email=$_GET['email'];
echo "name={ $name}<br>";
echo "age={ $age}<br>";
echo "email={ $email}<br>";
?>
</body>
</html>
```

```
name=sai
age=18
email=abc@gmail.com
```

### By using urlencode():-

- A query string can be created by URL encoding the keys and values that are to be encoded.
- The URL will be embedded in query string when it will be passed to another web page. For this purpose URL must be converted into hexadecimal characters.
- A URL is encoded by using urlencode() function of PHP.
- This function will accept string as parameter and will return the encoded form in it.

**Syntax:** urlencode("url");

### Sample program:

```
<html>
<body>
<?php
$hobbies="palying and reading";
$age=18;
?>
<a href="encode.php?hobbies=<?php echo urlencode($hobbies); echo urlencode($age);?>">gmail</a>
</body>
</html>
```

[gmail](#)

### Encode.php

```
<html>
<body>
<?php
echo "hobbies: ".$_GET['hobbies'];
?>
</body></html>
```

hobbies: palying and reading18

### 5.10.5,5.10.6 Understand session function and use session variable:

In PHP, session will start with the session\_start() function with a sessionID and records this ID on the first call to this function in a session. A session ID is a unique identifier for each session.

**Syntax:** session\_start();

## **Setting accessing session ID**

PHP allows to set and get the session ID manually. This is done using the session\_id() function.

### **Syntax:**

**string session\_id([string sid])**

The sid is an optional parameter. If it is not specified then the session\_id() function returns the current SID. If it is specified then it sets a new value to the current SID.

### **Example:**

```
<?php
    session_start();
    print "current sid=".session_id();
?>
```

## **Creating session variables:**

A session variable is created just like all variables. \$\_SESSION is used as the context which is a global array.

### **Sample program:**

```
<?php
$_SESSION['user']="CME";
$_SESSION['user1']="DCME";
echo $_SESSION['user'];
echo $_SESSION['user1'];
?>
```

## **Deleting session variables:** A session can be deleted in two ways

A session variable is deleted using unset() function.

### **Syntax:**

unset(list of session variables);

#### **1)session unset()function:**

It deletes all the session variables stored in a session. However, it does not completely delete the session from the storage mechanism.

### **Syntax:**

unset(list of session variables);

### **sample program:**

```
<?php
    session_start();

    $_SESSION['login']="hai";
    Unset($_SESSION['login']);
    print "session deleted" ;
?>
```

#### **2)session destroy()function:**

It deletes the current session completely from the storage mechanism. It does not delete the cookies that are stored on the browser.

### **Syntax:**

session\_destroy()

### **Example:**

```
<?php
    session_start();
    session_destroy();
?>
```

### **5.11.Process of debugging PHP code:**

**Debugging:** Debugging means to get rid of the errors, which occurred due to design, coding or logical errors.

#### **Process of debugging is as follows:**

- (a) Execute PHP scripts through URL
- (b) Clear vision regarding the version
- (c) Always verify HTML code
- (d) Always keep display errors ON
- (e) Need not ignore error messages

#### **1)Execute PHP scripts through URL:**

- The execution of PHP scripts must be done only through URL.
- The scripts must be run through the web server application such as <http://URL>.

#### **2) Clear vision regarding the version:**

- The user must have a clear idea about the version of PHP.
- To enable the version of PHP the user need not to execute a file named as phpinfo() after execution of this file ,it confirms the version of PHP which will be used.

#### **3)Always verify HTML code:**

- The user must be aware of errors that are to be occurred in HTML page because the error are not visible sometimes.
- Therefore,it is necessary for the user to check the HTML source page repeatedly.

#### **4)Always keep display errors ON:**

- The user need to confirm that the configuration setting display errors is active or not.
- This setting debugs many problems by displaying all the error messages.
- The configuration setting display-errors is enabled when the user executes a function phpinfo().
- If this setting was not active,the error messages are not displayed.Instead of error messages,blank pages are displayed when an error occurs.

#### **5)Need not ignore error messages:**

- The user need to concentrate on all the error messages that are generated. Sometimes, users ignore error messages that are reported by PHP.
- It is necessary for the users to read the messages. The user must trust those messages. Then the user can debug those errors and mean while logical errors are also rectified.