Unit -5(PHP)

1. Variables, Data types, Operators, Expressions

2. Control Structures

3. Arrays, Strings

4. Functions

5. Reading data from web form controls like (text boxes, radio buttons, lists etc.,)

6. Handling File Uploads

7. Connecting to database (MySQL as reference), executing simple queries,handling results

8. Handling Sessions and Cookies

9. File Handling in PHP: File operations(opening, closing, reading, writing, appending, deleting etc.) on text and binary files

10. listing directories.

****

Introduction :

PHP was developed by Rasmus Lerdorf in 1994.

PHP stands for "**P**ersonal Home Page : **H**ypertext **P**reprocessor".

PHP is a server side scripting language that is embedded in HTML which is used to

manage dynamic content, databases, session tracking and even build entire e-commerce sites.

It is integrated with a number of popular databases, including MySQL, PostgreSQL,

Oracle, Sybase, Informix, and Microsoft SQL Server.

Common Uses of PHP :

- PHP performs system functions, i.e. from files on a system it can create, open, read, write,and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Eg:

```
<html>
<head>
        <title>Hello World</title>
<body>
        <?php   echo "Hello, World!";  ?>
</body>
</html>
```

1. Variables, Data types, Operators, Expressions

Variables :

In PHP, a Variable must be created by using $ as predecessor to variable name.

Eg:- $x, $val, $result, $UserName, $Val_1

There is no explicit Variable declaration in PHP.

Eg:-(stringVariables)

```
<?php              // test1.php
$name = "Deepu";
echo $name;
echo "<br>";
$current_user = $name;
echo $current_user;
?>
```

Numerical Variables :

```
$count = 17;
$count = 1.7;
```

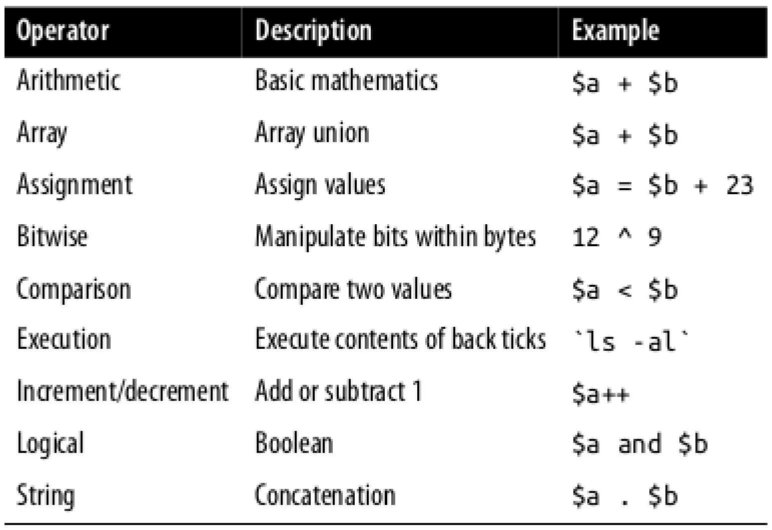
Global Variables :

```
global $is_logged_in;
```

Static Variables :

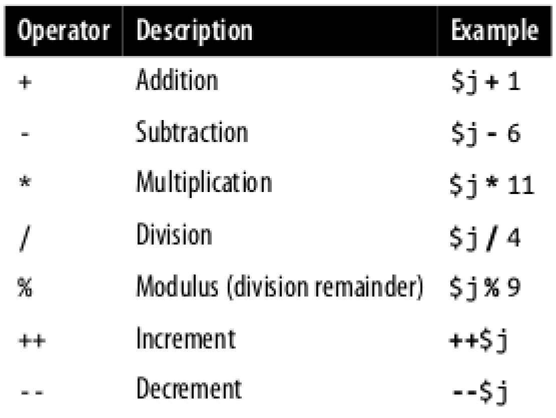
```
static $count = 0;
```

Operators :



Operator	Description	Example
Arithmetic	Basic mathematics	`$a + $b`
Array	Array union	`$a + $b`
Assignment	Assign values	`$a = $b + 23`
Bitwise	Manipulate bits within bytes	`12 ^ 9`
Comparison	Compare two values	`$a < $b`
Execution	Execute contents of back ticks	``` `ls -al` ```
Increment/decrement	Add or subtract 1	`$a++`
Logical	Boolean	`$a and $b`
String	Concatenation	`$a . $b`

Arithmetic operators

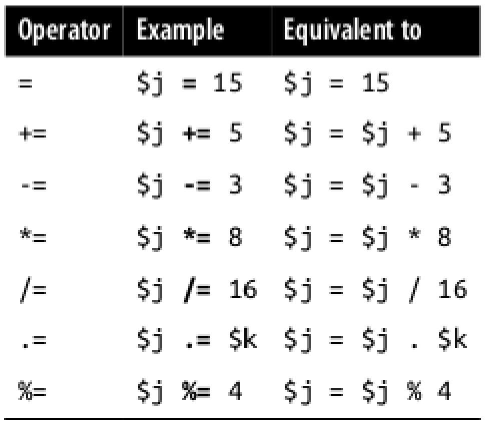


Operator	Description	Example
+	Addition	$j + 1
-	Subtraction	$j - 6
*	Multiplication	$j * 11
/	Division	$j / 4
%	Modulus (division remainder)	$j % 9
++	Increment	++$j
--	Decrement	--$j

Assignment operators(=)

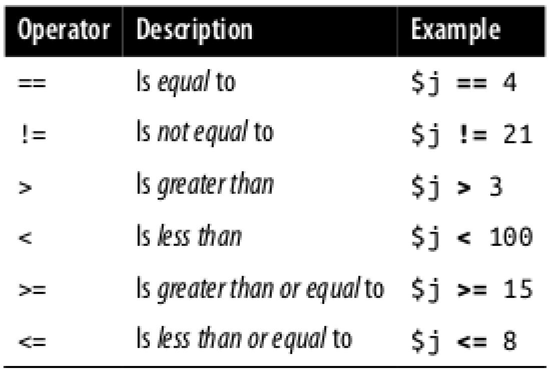
$count += 1;

$count = $count + 1;



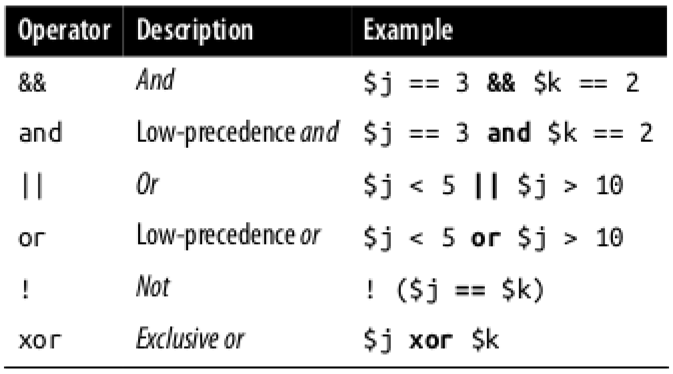
Operator	Example	Equivalent to
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j - 3
*=	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Comparison operators



Operator	Description	Example
==	Is *equal* to	$j == 4
!=	Is *not equal* to	$j != 21
>	Is *greater than*	$j > 3
<	Is *less than*	$j < 100
>=	Is *greater than or equal* to	$j >= 15
<=	Is *less than or equal* to	$j <= 8

Logical operators



Operator	Description	Example
&&	*And*	`$j == 3 && $k == 2`
and	Low-precedence *and*	`$j == 3 and $k == 2`
||	*Or*	`$j < 5 || $j > 10`
or	Low-precedence *or*	`$j < 5 or $j > 10`
!	*Not*	`! ($j == $k)`
xor	*Exclusive or*	`$j xor $k`

Expressions :

An expression is a combination of values, variables, operators, and functions that results in a value.

y = 3(abs(2x) + 4)

which in PHP would be

$y = 3 * (abs(2 * $x) + 4);

Automatic conversion from a number to a string

```
<?php
$number = 12345 * 67890;
echo substr($number, 3, 1);
?>
```

Automatically converting a string to a number

```
<?php
$pi= "3.1415927";
$radius = 5;
echo $pi * ($radius * $radius);
?>
```

Note :- print is also an alternative to echo that we can use. . The two commands are quite similar, but print is a function-like construct that takes a single parameter and has a return value (which is always 1), whereas echo is purely a PHP language construct. Since both commands are constructs,

2. Control Structures

if statement with curly braces

```
<?php
if ($bank_balance < 100)
{
	$money= 1000;
	$bank_balance += $money;
}
?>
```

if...else statement with curly braces

```
<?php
if ($bank_balance < 100)
{
	$money= 1000;
	$bank_balance += $money;
}
else
{
	$savings += 50;
	$bank_balance  -= 50;
}
?>
```

if...elseif...else statement with curly braces

```
<?php
if ($bank_balance < 100)
{
$money = 1000;
$bank_balance += $money;
}
elseif ($bank_balance > 200)
{
$savings += 100;
$bank_balance -= 100;
}
else
{
```

```
$savings += 50;
$bank_balance -= 50;
}
?>
```

switch statement

```
<?php
switch ($page)
{
case "Home":
        echo "You selected Home";
        break;
case "About":
        echo "You selected About";
        break;
case "News":
        echo "You selected News";
        break;
case "Login":
        echo "You selected Login";
        break;
case "Links":
        echo "You selected Links";
        break;
}
?>
```

3. Arrays, Strings

Arrays:

Adding items to an array

```
<?php
$paper[0] ="Copier";
$paper[1] ="Inkjet";
$paper[2] ="Laser";
$paper[3] ="Photo";
print_r($paper);
?>
```

Adding items to an array and retrieving them

```
<?php
$paper[] = "Copier";
$paper[] = "Inkjet";
$paper[] = "Laser";
$paper[] = "Photo";
        for ($j = 0 ; $j < 4 ; ++$j)
                echo "$j: $paper[$j]<br>";
?>
```

This example prints out the following:

```
0:Copier
1:Inkjet
2:Laser
3:Photo
```

Strings :

The various string handling functions are,

a. strlen() :- Returns the Length of a String

Eg:-

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

b. str_word_count() :- Count Words in a String

Eg:-

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

c. strrev() :- Reverse a String

Eg:-

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

d. strpos() :- Search For a Text Within a String

Eg:-

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

e. str_replace() - Replace Text Within a String

Eg:-

```
<?php
echo str_replace("world", "Dolly", "Hello world!");

// outputs Hello Dolly!
?>
```

4. Functions

Three inbuilt string functions

```
<?php
echo strrev(" .dlrow olleH"); // Reverse string
echo str_repeat("Hip ", 2);    // Repeat string
echo strtoupper("hooray!"); // String to uppercase
?>
```

Defining a User defined Function :

The general syntax for a function is as follows:

```
function function_name([parameter [, ...]])
{
// Statements
}
```

Eg:- Function without return

```
<?php
echo fix_names("WILLIAM", "henry", "gatES");

function fix_names($n1, $n2, $n3)
{
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
        return $n1 . " " . $n2 . " " . $n3;
}
?>
```

Eg:- Function with return

```
<?php
$names = fix_names("WILLIAM", "henry", "gatES");
```

```
echo $names[0] . " " . $names[1] . " " . $names[2];
function fix_names($n1, $n2, $n3)
{
$n1 = ucfirst(strtolower($n1));
$n2 = ucfirst(strtolower($n2));
$n3 = ucfirst(strtolower($n3));
return array($n1, $n2, $n3);
}
?>
```

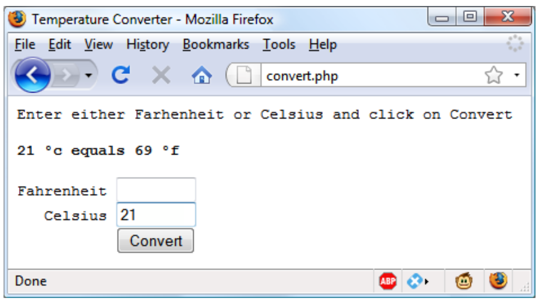
5. Reading data from web form controls like (text boxes, radio buttons, lists etc.,)

Eg:-convert.php(Text Box & Submit)

```
<?php
$f = $c = '';
if (isset($_POST['f']))
             $f = sanitizeString($_POST['f']);
if (isset($_POST['c']))
            $c = sanitizeString($_POST['c']);
if ($f != '')
{
$c = intval((5 / 9) * ($f - 32));
$out = "$f °f equals $c °c";
}
elseif($c != '')
{
$f = intval((9 / 5) * $c + 32);
$out = "$c °c equals $f °f";
}
else $out = "";
echo <<<_END
<html>
<head>
<title>Temperature Converter</title>
</head>
<body>
<pre>
Enter either Fahrenheit or Celsius and click on Convert
<b>$out</b>
```

```
<form method="post" action="convert.php">
Fahrenheit <input type="text" name="f" size="7">
Celsius <input type="text" name="c" size="7">
<input type="submit" value="Convert">
</form>
</pre>
</body>
</html>
_END;
function sanitizeString($var)
{
$var = stripslashes($var);
$var = strip_tags($var);
$var = htmlentities($var);
return $var;
}
?>
```

When you call up convert.php in a browser, the result is


Temperature Converter - Mozilla Firefox
File Edit View History Bookmarks Tools Help
convert.php
Done

```
Enter either Farhenheit or Celsius and click on Convert

21 °c equals 69 °f

Fahrenheit
   Celsius 21
           Convert
```

6. Handling File Uploads

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file_uploads directive, and set it to On:

```
file_uploads = On
```

Next, create an HTML form that allow users to choose the image file they want to upload:

```
<html>
<body>
```

```
<form action="upload.php" method="post" enctype="multipart/form-data">
  Select image to upload:
  <input type="file" name="fileToUpload" id="fileToUpload">
  <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form.

7. Connecting to database (MySQL as reference), executing simple queries,handling results

The process of using MySQL with PHP is as follows:

1. Connect to MySQL and select the database to use.
2. Build a query string.
3. Perform the query.
4. Retrieve the results and output them to a web page.
5. Disconnect from MySQL.

Connecting to a MySQL server with mysqli

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
?>
```

Querying a database with mysqli

```
<?php
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die($conn->error);
?>
```

Fetching results one cell at a time

```
<?php // query.php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
```

```
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die($conn->error);
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
$result->data_seek($j);
echo 'Author: '. $result->fetch_assoc()['author']. '<br>';
$result->data_seek($j);
echo 'Title: '. $result->fetch_assoc()['title']. '<br>';
$result->data_seek($j);
echo 'Category: ' . $result->fetch_assoc()['category']. '<br>';
$result->data_seek($j);
echo 'Year: '. $result->fetch_assoc()['year']. '<br>';
$result->data_seek($j);
echo 'ISBN: '. $result->fetch_assoc()['isbn']. '<br><br>';
}
$result->close();
$conn->close();
?>
```

8. Handling Sessions and Cookies :

Cookies :

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

A cookie is created with the setcookie()function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Retrieve Cookie :

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset()function to find out if the cookie is set

Eg:-

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
```

```
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Session :

A session is a way to store information (in variables) to be used across multiple pages.
Unlike a cookie, the information is not stored on the users computer.

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

Start a PHP Session
A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example :

```
<?php
// Start the session
session_start();
?>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] ="green";
$_SESSION["favanimal"] ="cat";
echo "Session variables are set.";
```

```
?>

</body>
</html>
```

9. File Handling in PHP: File operations(opening, closing, reading, writing, appending, deleting etc.) on text and binary files

1. fopen()

The PHP fopen() function is used to open a file.

Eg:-

```
<?php

$handle = fopen("c:\\folder\\file.txt", "r");

?>
```

2. fread()

The PHP fread() function is used to read data of the file. It requires two arguments: file resource and file size.

Eg:-

```
<?php

$filename = "c:\\file1.txt";

$fp = fopen($filename, "r");//open file in read mode


$contents = fread($fp, filesize($filename));//read file

echo "<pre>$contents</pre>";//printing data of file

fclose($fp);//close file

?>
```

3. fwrite()

The PHP fwrite() function is used to write content of the string into file.

Eg:-

```
<?php

$fp =fopen('data.txt', 'w');//opens file in write-only mode

fwrite($fp,'welcome');

fwrite($fp,'to php file write');

fclose($fp);


echo "File written successfully";

?>
```

4. Delete File
In PHP, we can delete any file using unlink() function.

Eg:-

```
<?php
$status=unlink('data.txt');
if($status){
echo "File deleted successfully";
}else{
echo "Sorry!";
}
?>
```

10. listing directories.

PHP Directory Functions

Function	Description
chdir()	Changes the current directory
chroot()	Changes the root directory
closedir()	Closes a directory handle
dir()	Returns an instance of the Directory class
getcwd()	Returns the current working directory
opendir()	Opens a directory handle
readdir()	Returns an entry from a directory handle
rewinddir()	Resets a directory handle
scandir()	Returns an array of files and directories of a specified directory