INTRODUCTION TO PHP

BY LECTURER SURAJ PANDEY CCT COLLEGE

INTRODUCTION TO PHP

2.1 Introduction to PHP Scripting Language

2.2 PHP vs JSP vs ASP server-side programming

2.3 Server-Side Scripting vs Client-Side Scripting

2.4 Installing XAMPP or WAMP or other web server

2.5 Setting server environment

2.6 Configuration and Adjusting setting in PHP.ini and httpd.conf

2.7 Running PHP scripts

2.8 Formatting outputs

2.9 Working with variables, global variables and constants

210 Logical, concatenation, mathematical and relational operators

2.11 Escape Sequences

WHAT IS PHP?

PHP stands for **Hypertext Pre-Processor**. PHP is a scripting language used to develop static and dynamic webpages and web applications. Here are a few important things you must know about PHP:

1. PHP is an Interpreted language, hence it doesn't need a compiler.
2. To run and execute PHP code, we need a Web server on which PHP must be installed.
3. PHP is a server side scripting language, which means that PHP is executed on the server and the result is sent to the browser in plain HTML.
4. PHP is open source and free.

USES OF PHP

To further fortify your trust in PHP, here are a few applications of this amazing scripting language:

1. It can be used to **create Web applications** like Social Networks(Facebook, Digg), Blogs(Wordpress, Joomla), eCommerce websites(OpenCart, Magento etc.) etc.
2. **Comman Line Scripting**. You can write PHP scripts to perform different operations on any machine, all you need is a PHP parser for this.
3. **Create Facebook applications** and easily integrate Facebook plugins in your website, using Facebook's PHP SDK.
4. **Sending Emails** or building email applications because PHP provides with a robust email sending function.
5. Wordpress is one of the most used blogging(CMS) platform in the World, and if you know PHP, you can try a hand in **Wordpress plugin development**.

WHAT SHOULD I KNOW BEFORE LEARNING PHP

Before learning PHP you should have a basic understanding of:

HTML - it is strongly recommended to have a deep understanding of the HTML markup

CSS - it is recommended to know some CSS basics before learning PHP. This will help you on your way to developing cool and dynamic PHP websites

JavaScript - it would be nice of you to recognize and have a basic reading of the JavaScript programming language

Programming Logic - This is something that always helps. Any knowledge of another programming language will help you to learn and apply the logic to the new programming language, in this case PHP

WHO IS USING PHP?

There are lots of big site using PHP:

Facebook

Wikipedia

Yahoo

Flickr

Wordpress

ADVANTAGES OF PHP OVER OTHER LANGUAGES

There are several advantages why one should choose PHP.

1. **Easy to learn:** PHP is easy to learn and use. For beginner programmers who just started out in web development, PHP is often considered as the preferable choice of language to learn.
2. **Open source:** PHP is an open-source project. It is developed and maintained by a worldwide community of developers who make its source code freely available to download and use.
3. **Portability:** PHP runs on various platforms such as Microsoft Windows, Linux, Mac OS, etc. and it is compatible with almost all servers used today such Apache, IIS, etc.
4. **Fast Performance:** Scripts written in PHP usually execute or runs faster than those written in other scripting languages like ASP, Ruby, Python, Java, etc.
5. **Vast Community:** Since PHP is supported by the worldwide community, finding help or documentation related to PHP online is extremely easy.

Php Syntax

```
<?php
        echo ‘Hello world’;
?>
```

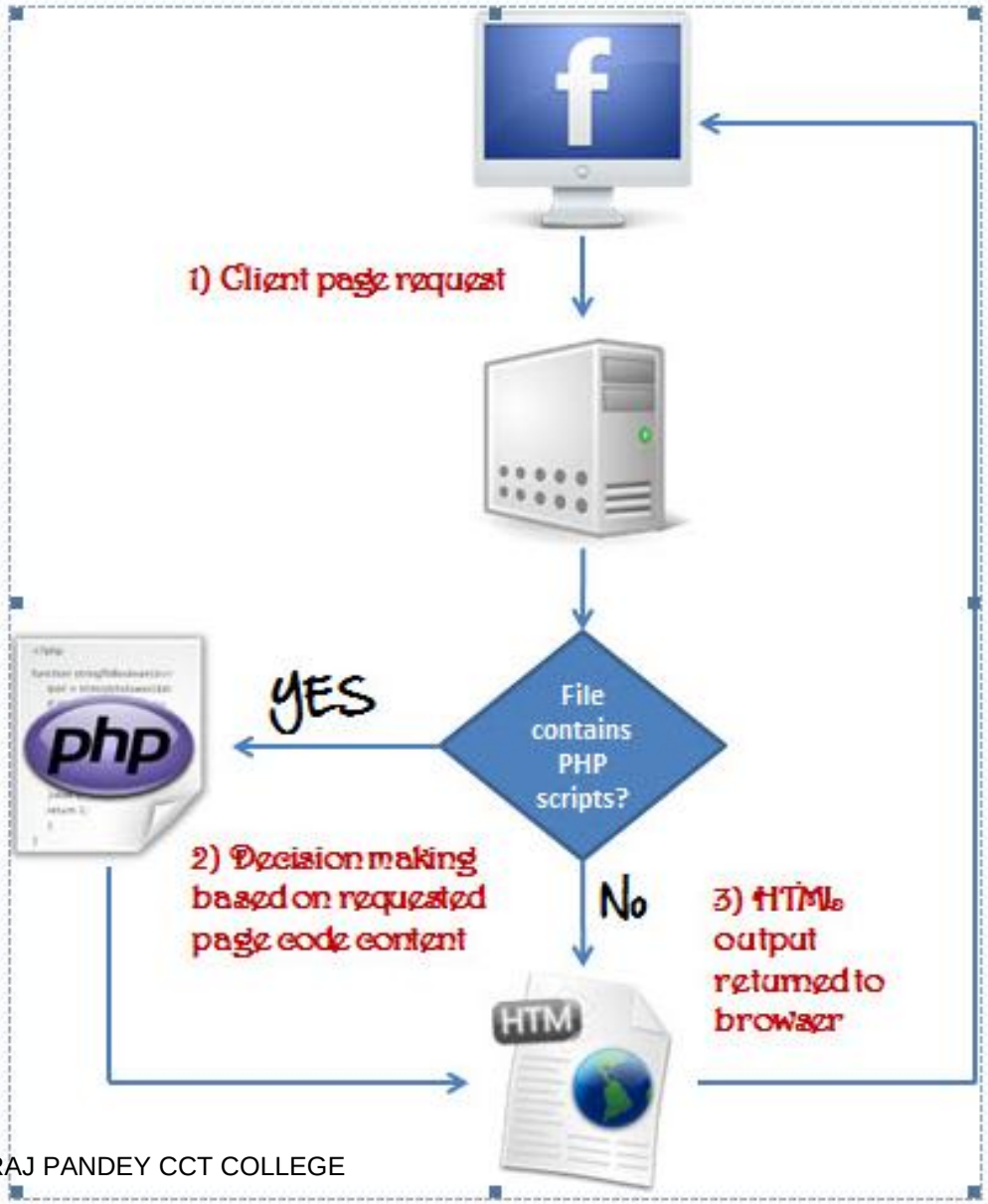
A PHP file can also contain tags such as HTML and client side scripts such as JavaScript.

HTML is an added advantage when learning PHP Language. You can even learn PHP without knowing HTML but it’s recommended you at least know the basics of HTML.

Database management systems DBMS for database powered applications.

For more advanced topics such as interactive applications and web services, you will need JavaScript and XML.

The flowchart diagram shown below illustrates the basic architecture of a PHP web application and how the server handles the requests.


1) Client page request
File contains PHP scripts?
YES
No
2) Decision making based on requested page code content
3) HTML output returned to browser
php
HTM

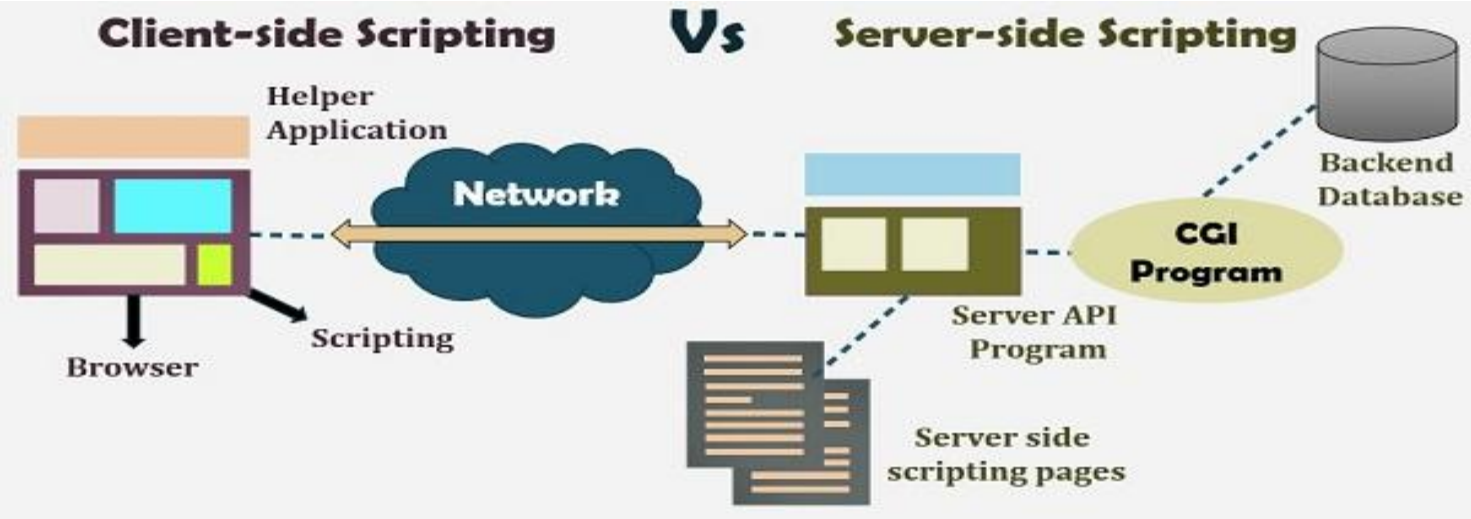
PHP VS JSP VS ASP SERVER-SIDE PROGRAMMING

The table below compares the various server side scripting languages with PHP

JSP	PHP
JSP require more and complex code.	PHP is basic and require less lines of code.
Server-side programming technology.	Server-side scripting dialect made by Rasmus Lerdorf.
Web applications with dynamic substance.	Little to medium measured web arrangements.
JSP facilitating isn't much expensive than PHP.	PHP facilitating is exceptionally cheap.
Common to JSPs since they are	Way less characteristic than JSP.
Require Servlet holder like Tomcat.	Run on its possess as a CGI motor.
JSP permits to characterize custom tags.	PHP won't permit custom tags.

FEATURE	PHP	ASP	JSP
Learning curve	short	Longer than PHP	Longer than PHP
Web hosting	Supported by almost all hosting servers	Needs dedicated server	Fairly supported
Open source	Yes	No	Yes
Web services support	Built in	Uses the .NET framework	Uses add on libraries
Integration with HTML	Easy	Fairly complex	Fairly complex
MySQL support	Native	Needs third party drivers	Needs third party drivers
Easily extended by other languages	Yes	No	Extended using Java classes and libraries.

SERVER-SIDE SCRIPTING VS CLIENT-SIDE SCRIPTING



The scripts can be written in two forms, at the server end (back end) or at the client end (server end). The main difference between server-side scripting and client-side scripting is that the server side scripting involves server for its processing. On the other hand, client-side scripting requires browsers to run the scripts on the client machine but does not interact with the server while processing the client-side scripts.

BASIS FOR COMPARISON	SERVER-SIDE SCRIPTING	CLIENT-SIDE SCRIPTING
Basic	Works in the back end which could not be visible at the client end.	Works at the front end and script are visible among the users.
Processing	Requires server interaction.	Does not need interaction with the server.
Languages involved	PHP, ASP.net, Ruby on Rails, ColdFusion, Python, etcetera.	HTML, CSS, JavaScript, etc.
Affect	Could effectively customize the web pages and provide dynamic websites.	Can reduce the load to the server.
Security	Relatively secure.	Insecure

INSTALLING XAMPP OR WAMP OR OTHER WEB SERVER

Requirements for PHP

To run PHP scripts, we need the following services:

PHP Parser: To execute PHP scripts, PHP installation is required.

Web Server: Because PHP is mostly used to develop websites, hence most of its implementations comes bundled with Apache Web Server, which is required to host the application developed in PHP over HTTP.

Database: Any one database management system, which is generally MySQL, as PHP comes with a native support for MySQL

Now, you can install all the 3 services separately yourself, or you can simply download and install softwares which automatically installs all the above services.

The most popular such software package is **XAMPP**.

WHAT IS XAMPP?

XAMPP stands for:

X: Cross Platform, as it supports all the moder operating systems like Windows, Mac OSX, Linux etc.

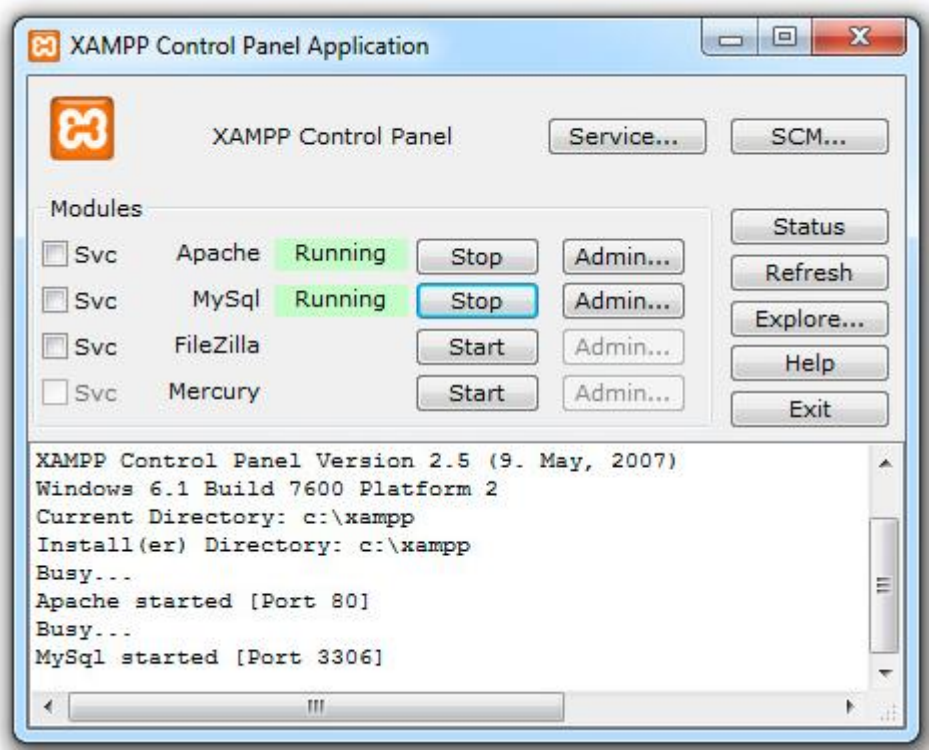
A: Apache Web Server

M: MySQL database management system.

P: PHP installation

P: Perl scripting language

You can easily download and install XAMPP from this link. The installation process is simple and once installed you will see a small window like this, showing the status of various services which are running.



You can easily control, stop and restart, various services using the XAMPP Control Panel.

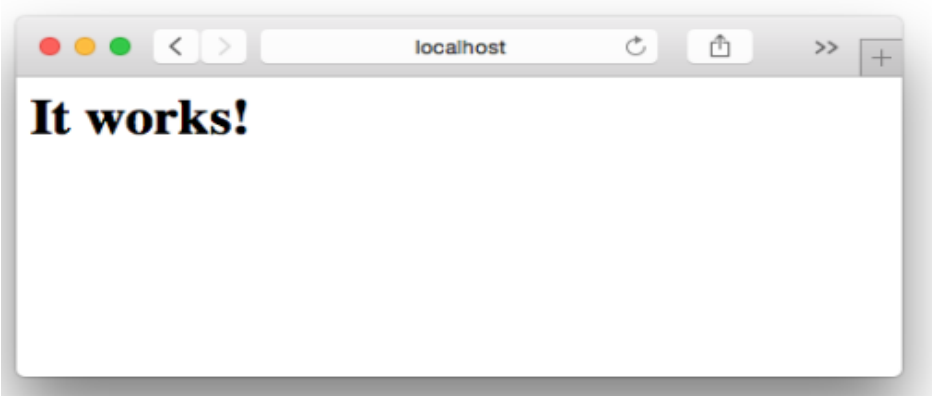
Upon successful installation, a folder with name **xampp** will be created in the C drive(by default). In the folder **xampp** there are many sub-folders like apache, cgi-bin, FileZillaFTP etc, but the most important sub-folders are:

htdocs: This is the folder in which we will keep all our PHP files.

mysql: This folder contains all the files for the MySQL database. By default the MySQL databse runs on port number 3306.

php: This folder holds all the installation files for PHP. All the configurations for the current PHP installation is saved in **php.ini** file which is stored in this folder.

If everything seems fine and the apache server is running(check in the XAMPP control panel), open your **Web browser** and enter localhost in the address bar and hit enter. You will see the default page of Apache web server.



Other Software Packages

XAMPP is not the only available package, although it is the most widely used one. Here are a few other Operating System specific options that you can download and install:

WAMP: It's for Windows (Window, Apache, MySQL and PHP)

MAMP: It's for Mac OSX (Macintosh, Apache, MySQL and PHP)

LAMP: It's for Linux (Linux, Apache, MySQL and PHP)

IDE or Editor for writing PHP code

As PHP code is not compiled, you can even use a simple editor like Notepad to create PHP code files. But it's always good to have a nice looking Editor which can highlight the code, provide you suggestions while coding, and even analyze your code for syntax errors as your write it.

So here are a few good IDEs and Editors:

Netbeans IDE

Eclipse IDE

PyStorm IDE

Atom Editor

Brackets Editor

Notepad ++

Sublime Text

FIRST PHP EXAMPLE

Hello World script in PHP

All the php code is written inside php code tags which is <?php and ?>. And the file with php code is saved with an extension .php. Here is a simple example:

<?php

***//* code statements**

?>

Hence a simple **Hello, World!** program in PHP will be:

<?php

echo "Hello, World!";

?>

output

Hello, World!

echo is a command used in PHP to display anything on screen.

If we want to include this php code in an HTML document, we can do so like this:

```
<!DOCTYPE>
 <html>
<body>
<?
php echo "<h1>Hello, World!</h1>";
?>
</body>
 </html>
```

Now copy the above code and paste it in you Code Editor or IDE and save the file with the name **hello_world.php**

Now go to your C directory where XAMPP was installed, and open **xampp** → **htdocs** and create a new folder with name **studytonight** and put your php code file **hello_world.php** in the **studytonight** folder.

Now visit the following link in your browser: **localhost/studytonight/hello_world.php** and you would see **Hello, World!** written on the screen. Notice that *Hello, World!* is written as a heading because in the HTML code we specified that *Hello, World!* should be printed as a heading as put it inside the heading tag <h1>

php code file

```
<!DOCTYPE>
<html>
    <body>
    <?php
        echo "<h1>Hello, World!</h1>";
    ?>
    </body>
</html>
```

PHP code is executed and the result in plain HTML is sent to the browser

plain HTML code

```
<!DOCTYPE>
<html>
    <body>
        <h1>Hello, World!</h1>
    </body>
</html>
```

Sent to browser over HTTP

PHP SYNTAX RULES

Below we have a listed down the main syntax rules that you must follow while writing php code.

1. All the php code in a php script should be enclosed within <?php and ?>, else it will not be considered as php code. Adding php code inside the PHP tags is known as **Escaping to php.<?php ... ?>**

   Apart from the standard <?php and ?>, you can also use

   the **Short-open tags**:

   <? ... **?>**Or use the HTML script tags, like we do for adding javascript code in HTML document:

<script language="PHP"> ... </script>

2. Every expression in PHP ends with a **semicolon ;**

3. **Commenting PHP code**: Both single line and multi-line comments are supported in PHP. For single line comment, we can either use # or // before the comment line. For example,

```
<?php
# This is also a single line comment
# another line of comment
// This is a single line comment
echo "Example for Single line Comments";
?>
```

And for multi-line comments, we use /* ... */. For example,

```
<?php
/* This is also a single line comment another line of comment */
 echo "Example for Multi-line Comments";
?>
```

4. PHP is case sensitive, which means that a variable **$tiger** is not same as **$Tiger.** Both of these, represent two different variables here.But all the predefined keywords and functions like if, else, echo etc are case insensitive.

```
<?php
echo "Hello, World!";
ECHO "Hello, World!";
?>
```

OUTPUT

Hello, World! Hello, World!

5. PHP uses **curly braces to define a code block**.

```
<?php
if($zero == 0)
{
echo "If condition satisfied";
echo "This is a code block";
}
?>
```

Don't worry we will learn about if...else conditions in details, this is just to demonstrate the use of curly braces.

PHP - VARIABLES

PHP variables are just a name for a value and you can simply use it like this *$var_name = value;*. They usually store sting characters and numeric values for later use in the programming block. Here is a simple example:

PHP

```php
<?php
$a      = "Welcome";
$b      = "Hello";
$first  = "John";
$last   = "Doe";
$c      = ", how are you today?";

echo "$a $first $last <br />";  // Returns: Welcome John Doe
echo "$b $first $c <br />";    // Returns: Hello John, How are you today?

$x=5;
$y=10.1;
echo $x+$y;  // Returns: 15.1
?>
```

WHAT IS A VARIABLES SCOPE TYPES IN PHP?

In PHP, the scope of a variable is the part of the script where a variable was defined and can be referenced. There are three types of scopes in PHP.

local - declared **within** a function and can only be accessed within that specific function

global - declared **outside** a function and can only be accessed outside a function. There is an exception for this when the *global* keyword is used.

static - declared **withing** a function and maintaining his value in that function

SPECIAL KEYWORDS FOR DECLARING VARIABLES SOCPES

There are cases when you declare a variable outside of a function and you want to use it inside and the other way around.

This is were the ***global*** keyword enters.

PHP

```php
<?php
$x = 1;
$y = 2;

function sum() {
    global $x, $y;
    $result = $x + $y;
}

sum();
echo $result; // Outputs: 3
?>
```

On the other side ***static*** keyword is used to store a value of a varible declared inside a function, for consecutive cals. Normally all the local variable are resseted when the fnction closes, unless static keyword is used.

```php
<?php
function count_table() {
    static $x = 0;
    echo "Value of x is now $x <br />\n";
    return $x++;
}

while(count_table() < 10){
    // do some other stuff
}


/* Outputs:
Value of x is now 0
Value of x is now 1
Value of x is now 2
Value of x is now 3
Value of x is now 4
Value of x is now 5
Value of x is now 6
Value of x is now 7
Value of x is now 8
Value of x is now 9
Value of x is now 10
*/
?>
```

RULES FOR CREATING VARIABLES IN PHP

Here we have a few basic rules that you must keep in mind while creating variables in PHP. All the rules are explained with help of simple examples.

A variable name will always start with a $ sign, followed by the variable name.

A variable name should not start with a numeric value. It can either start with an alphabet or an underscore sign _.

```php
<?php
    $var = "I am a variable";
    $_var = 5;
    // Invalid variable name
    $7var = "I am a variable too";
?>
```

A variable name can only contain alphabets, numbers and underscore symbol _.

Variable names in PHP are case-sensitive, which means $love is not same as $Love.

```php
<?php
    $car = "Jaguar E-Pace";
    echo "My favorite car is $car";
    // below statement will give error
    echo "I love $Car";
?>
```

OUTPUT:

```
My favorite car is Jaguar E-Pace

Notice: Undefined variable: Car
```

PHP VARIABLES

Variables are "containers" for storing information. A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

1. **A variable starts with the $ sign, followed by the name of the variable**
2. **A variable name must start with a letter or the underscore character**
3. **A variable name cannot start with a number**
4. **A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)**
5. **Variable names are case-sensitive ($age and $AGE are two different variables)**

ADD TWO NUMBER

```
<html>
<body>
<?php
$a =5 ;
$b=6;
$c=$a+$b;
echo "The sum of  $a  and  $b is  $c";
?>
</body>
</html>
```

Output:

The sum of 5 and 6 is 11

PHP VARIABLES SCOPE

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be used.

PHP has three different variable scopes:

1. **local**
2. **global**
3. **static**

GLOBAL AND LOCAL SCOPE

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<html>
<body>
<?php
$x = 5; // global scope
function myTest() {
echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
</body>
</html>
```

Output:

Variable x inside function is:

Variable x outside function is: 5

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

```
<html>
<body>
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
</body>
</html>
```

Output:

Variable x inside function is: 5

Variable x outside function is:

PHP THE STATIC KEYWORD

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.To do this, use the static keyword when you first declare the variable:

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

```
Output:
0
1
2
```

PHP THE GLOBAL KEYWORD

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

```
<html>
<body>
<?php
$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();  // run function
echo $y; // output the new value for variable $y
?>
</body>
</html>
```

Output:
15

PHP ECHO AND PRINT FUNCTIONS

PHP Echo

echo() function is used to print or output one or more strings. We have specifically mentioned string here because, the syntax of the echo function is:

echo(string)

Although you can use echo() function to output anything, as PHP parser will automaticallly convert it into string type.

echo doesn't need parenthesis, although you can use parenthesis if you want.

```
<?php
echo "I am open";
echo ("I am enclosed in parenthesis");
?>
```

I am open I am enclosed in parenthesis

EXAMPLE

Printing a basic sentence

```
<?php
echo "I am a sentence";
 ?>
```

I am a sentence

Printing multiple strings using comma ,

Here is how you can use multiple strings as parameters for echo.

```
<?php
 echo 'This','is','a','broken','sentence';
?>
```


This is a broken sentence

Printing a multiline text(string)

```
<?php
echo "This is a multiline sentence example";
?>
```

This is a multiline sentence example

Printing a string variable

```
<?php
$str = "I am a string variable";
echo $str;
?>
```

I am a string variable

Printing a string variable with some text

we use a string variable with some plain text in echo.

```
<?php
$weird = "Stupid";
echo "I am $weird";
echo 'I am $weird'; ?>
```

I am Stupid

I am $weird

As you can see, when we use **double quotes** the value of the string variable gets printed, while if we use **single quotes**, the variable is printed as it is.

PHP Print

The PHP print is exactly the same as echo, with same syntax and same usage. Just replace echo with print in all the above examples, and they will work just fine.

PHP - CONSTANTS

A constant is a type of variable that cannot be changed. A constant cannot be undefined and once his value has been set, it can not change his value along the script.

This is how you define a constant in PHP:

```php
<?php

        // site name
        define('SITE_NAME', 'my cool site name', flase);

        // database conection
        define('DB_INFO', array(
                'host'=> 'localhost',
                'usr' => 'my_user',
                'pass'=> 'my_database_pass',
                'db'  => 'my_database')
        );


        echo "Wellcome to " SITE_NAME;
        echo "<br />"
        var_dump(DB_INFO);
?>
```

Let's take a closer look at the examples above. We used the *define()* function to create the constant and with the following params:
name - the defined name for our constant
value - the defined value for our constant
case-sensitive - this is an optional param and it is true by default. This means that our constant is case sensitive. Set it to false if you need it

Naming constants

Just like a normal variable a valid constant name starts with a letter or underscore. On the other hand, a PHP constant doesn't need the "$" sign before the constant name.

Although it is not a must, there is a convention that constant names to be written in uppercase letters. This makes it easy for the identification and distinguishes them from other variables when reviewing coding documents.

Constants scope

Constants are automatically defined as globals and can be directly used anywhere around the script without further action.

```php
<?php
        // database connection
        define('DB_INFO', array(
                'host'=> 'localhost',
                'usr' => 'my_user',
                'pass'=> 'my_database_pass',
                'db'  => 'my_database')
        );

        // simple function too print data
        function nice_printing(){
                echo "<pre>";
                print_r(DB_INFO);
                echo "</pre>"
        }

        // function execution
        nice_printing();
?>
```

Check if a constant has been defined

Although we have not wet talked about the if/else tag yet, this is a good place to mention that constants have a specific function to check if they are defined.

For variables we will use *isset()* while for constants we will use ***defined()***. Here is an example using it on a constant:

```php
<?php
        define('DEBUG_LEVEL', 3);

        if(defined(DEBUG)) {
                // do something if debug is defined
        }
?>
```

DATA TYPES IN PHP

PHP Data types specify the different types of data that are supported in PHP language. There are total 8 data types supported in PHP, which are categorized into 3 main types. They are:

Scalar Types: boolean, integer, float and string.

Compound Types: array and object.

Special Types: resource and NULL.

PHP Boolean

A boolean data type can have two possible values, either **True** or **False**.

$a = true;

$b = false;

NOTE: Here the values true and false are not enclosed within quotes, because these are not strings.

PHP Integer

An Integer data type is used to store any non-decimal numeric value within the range -2,147,483,648 to 2,147,483,647.

An integer value can be negative or positive, but it cannot have a decimal.

$x = -2671;

$y = 7007;

PHP Float

Float data type is used to store any decimal numeric value.

A float(floating point) value can also be either negative or positive.

$a = -2671.01;

$b = 7007.70;

PHP String

String data type in PHP and in general, is a sequence of characters(or anything, it can be numbers and special characters too) enclosed within quotes. You can use single or double quotes.

```
$str1 = "Hello";
 $str2 = "What is your Roll No?";
$str3 = "4";
echo $str1;
echo "<br/>";
echo $str2;
 echo "<br/>";
echo "Me: My Roll number is $str3";
```

OUTPUT
Hello What is your Roll No?
Me: My Roll number is 4

PHP NULL

NULL data type is a special data type which means **nothing**. It can only have one value, and that is NULL.

If you create any variable and do not assign any value to it, it will automatically have NULL stored in it.

Also, we can use NULL value to empty any variable.

```
// holds a null value

$a;

$b = 7007.70;

 // we can also assign null value

$b = null;
```

PHP resource data type

A resource is a special variable, holding a reference to an external resource like a database connection for example.

PHP$conn = mysqli_connect(localhost,"root","admin","users");

PHP OPERATORS

Operators are used to perform operations on PHP variables and simple values.

In PHP there are total 7 types of operators, they are:

Arithmetic Operators

Assignment Operators

Comparison Operators

Increment/Decrement Operators

Logical Operators

String Operators

Array Operators

There are a few additional operators as well like, Type operator, Bitwise operator, Execution operators etc.

Based on how these operators are used, they are categorised into 3 categories:

Unary Operators: Works on a single operand(variable or value).

Binary Operators: Works on two operands(variables or values).

Ternary Operators: Works on three operands.

PHP Arithmetic Operators

These operators are used to perform basic arithmetic operations like addition, multiplication, division, etc.

Name	Operator	What does it do?	Example
Addition	`+`	It is used to perform normal addition.	$a + $b
Subtraction	`-`	It is used to perform normal subtraction.	$a - $b
Multiplication	`*`	It is used to perform multiplication.	$a * $b
Division	`/`	It is used to perform division.	$a / $b
Exponent	`**`	It returns the first operand raised to the power the second operand. `$a ** $b = $a^$b`	$a ** $b
Modulus(or, Remainder)	`%`	It returns the remainder of first operand divided by the second operand	$a % $b

PHP Assignment Operators

Assignment operators are used to assign values to variables, either as it is or after performing some arithmetic operation on it. The most basic assignment operator is **equal to**=.

Operator	Usage
`=`	`$a = $b`, will save the value of variable `$b` to the variable `$a`
`+-`	`$a += $b` is same as `$a + $b`
`-=`	`$a -= $b` is same as `$a - $b`
`*=`	`$a *= $b` is same as `$a * $b`
`/=`	`$a /= $b` is same as `$a / $b`
`%=`	`$a %= $b` is same as `$a % $b`

So basically, the assignment operator provides us with shorthand techniques to perform arithmetic operations.

PHP Comparison Operators

As the name suggest, these are used to compare two values.

Name	Operator	What does it do?	Example
Equal	`==`	It returns `true` if left operand is equal to the right operand.	$a == $b
Identical	`===`	It returns `true` if left operand is equal to the right operand and they are of the same type.	$a === $b
Not Equal	`!=`	It returns `true` if left operand is not equal to the right operand.	$a != $b
Not Identical	`!==`	It returns `true` if left operand is not equal to the right operand, and they are of different type as well.	$a !== $b
Greater than	`>`	It returns `true` if left operand is greater than the right operand.	$a > $b
Less than	`<`	It returns `true` if left operand is less than the right operand.	$a < $b
Greater than or equal to	`>=`	It returns `true` if left operand is greater than or equal to the right operand.	$a >= $b
Less than or equal to	`<=`	It returns `true` if left operand is less than or equal to the right operand.	$a <= $b

PHP Increment/Decrement Operators

These operators are **unary operators**, i.e they require only one operand.

Operator	Usage
`++$a`	**Pre Increment**, It will first increment the operand by `1` (add one to it) and then use it or return it.
`$a++`	**Post Increment**, It will first return the operand and then increment the operand by `1`.
`--$b`	**Pre Decrement**, It will first decrement the operand by `1` (subtract one from it) and then use it or return it.
`$b--`	**Post Decrement**, It will first return the operand and then decrement the operand by `1`.

These operators are very useful and handy when use loops or when we have simply increment any value by one in our program/script.

PHP Logical Operators

Logical operators are generally used when any action depends on two or more conditions.

Name	Operator	What does it do?	Example
And	`and` or `&&`	It returns true if both the operands(or expressions) returns true.	$a && $b
Or	`or` or `||`	It returns true if any one out of the two operands(or expressions) returns true, or both return true.	$a || $b
Xor	`xor`	It returns true if any one out of the two operands(or expressions) returns true, but not when both return true.	$a xor $b
Not	`!`	This is a **unary operator**. It returns true, if the operand(or expression) returns false.	!$a

PHP String Operators

String operators are used to perform operations on string. There are only two string operators, generally PHP built-in functions are used to perform various operations on strings,

Name	Operator	What does it do?	Example
Concatenation	`.` (a dot)	It is used to concatenate(join together) two strings.	$a.$b
Concatenation Assignment	`.=`	It is used to append one string to another.	$a .= $b

Here we have a simple example to demonstrate the usage of both the string operators.

```php
<?php
    $a = "study";
    $b = "tonight";
    // concatenating $a and $b
    echo $a.$b;

    // appending $b to $a
    $a .= $b
    echo $a;
?>
```

PHP Array Operators

These operators are used to compare arrays.

Name	Operator	What does it do?	Example
Equal	`==`	It returns `true` if both the arrays have same key/value pairs.	$a == $b
Identical	`===`	It returns `true` if both the arrays have same key/value pairs, in same order and of same type.	$a === $b
Not Equal	`!=`	It returns `true` if both the arrays are not same.	$a != $b
Not Identical	`!==`	It returns `true` if both the arrays are not identical, based on type of value etc.	$a !== $b
Union(Join)	`+`	It joins the arrays together	$a + $b

ESCAPE SEQUENCE

Escape sequences are used for escaping a character during the string parsing. It is also used for giving special meaning to represent line breaks, tabs, alert and more. The escape sequences are interpolated into strings enclosed by double quotations or heredoc syntax.

Escape sequences are started with the escaping character backslash (\) followed by the character which may be an alphanumeric or a special character.

If it is an alphanumeric character, it gives special meaning to represent the line breaks \n, carriage return \r and more.

If it is a special character, then it will be considered as it is during the string parsing.

Escape Sequence Example

This code shows a simple PHP example program to distinguish the behavior of the escape sequences with alphanumeric and non-alpha numeric characters. In this example, I used the escape sequence \n to add a line break after label and I have used \\ to escape \ character and print it to the browser.

```
<?php

 echo "PHP Escape Sequences:\n Backslash \\ is used as an escaping character.";

?>
```

Widely used Escape Sequences in PHP

\’ – To escape ‘ within single quoted string.

\” – To escape “ within double quoted string.

\\ – To escape the backslash.

\$ – To escape $.

\n – To add line breaks between string.

\t – To add tab space.

\r – For carriage return.