PHP BASICS| VARIABLES| ARRAYS|

Introduction to PHP

PHP started as a small open source project that evolved as more and more people found out that to be a quite useful language. ***Rasmus Lerdorf*** unleashed the first version of PHP way back in 1994.

- PHP stands for PHP Hypertext Pre-processor.
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, session tracking, databases, even build entire e-commerce sites.
- It is integrated into a number of popular databases, including Oracle, MySQL, Sybase, PostgreSQL, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.

Pre - request for installing PHP on Windows

- Download the PHP zip file from **https://www.apachefriends.org/download.html**

XAMPP Server Installation in Windows

Running a first hello world program

To understand PHP, first start with simple PHP scripts. First we will create a friendly little "Sakshi Education!" script.

PHP is embedded in HTML, Which means that in as same as your normal HTML (or XHTML if you're cutting-edge) you can use PHP statements like this:

Example:

```
<html>
<head>
<title>SakshiEducation</title>
```

```
</head>
<body>
<?php

echo "Sakshi Education!";

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

Result:

Sakshi Education!

PHP Basics:

The print() Statement

The print() statement displays data passed to it . All of the following are possible print() statements:

Example:

```
<?php
print ("<p>Sakshi Education.</p>");
?>
```

Output:

Sakshi Education.

Example:

```
<?php

$season = "Education";

print "<p>Sakshi $season.</p>";

?>
```

Output:

Sakshi Education.

In above program “$season “is a variable.

Example:

```
<?php

print "<p>SakshiEducation. </p>";
```

```
?>
```

Output:

Sakshi Education.

The echo() Statement

Both echo() and print() are similar, with some minuscule differences between echo() and print().

Syntax:

echo(string argument1 [, ...string argumentN])

echo() statements are used like print() statements..

Example:

```
<?php

echo "I love my india.";

?>
```

echo() can be used for displaying multiple strings values. echo() will give more performance.

Example:

```
<?php

$s = "Sakshi";

$e = "Education";

echo $s, " ", $e, " makes your future bright ";

?>
```

Output:

Sakshi Education makes your future bright

The printf() Statement:

The printf() statement is perfect when you need to mix of static content and dynamic data stored inside one or a few variables.

Syntax:

printf(string format [, mixed args])

Example:

```
<?php
printf("Sakshi: %d years celebrations.", 5);
?>
```

Output:

Sakshi: 5 years celebrations.

In above example, %d is known as a type specifier and'd' indicates an integer value will be placed in that position. When the printf() statement executes, the lone argument, 100, will be inserted into the placeholder. Remember that an integer is expected, so if you pass along a number including a decimal value (known as a float), it will be rounded down to the closest integer. If you pass along 100.2 or 100.6, then 100 will be output. Pass along a string value such as "one hundred", and 0 will be output, although if you pass along 123food, then 123 will be output.

Type	Description
%b	Argument considered an integer; presented as a binary number
%c	Argument considered an integer; presented as a character corresponding to that ASCII value
%d	Argument considered an integer; presented as a signed decimal number
%f	Argument considered a floating-point number; presented as a floating-point number
%o	Argument considered an integer; presented as an octal number
%s	Argument considered a string; presented as a string
%u	Argument considered an integer; presented as an unsigned decimal number
%x	Argument considered an integer; presented as a lowercase hexadecimal number

%X	Argument considered an integer; presented as an uppercase hexadecimal number

If you want to insert two specifiers into the string, you pass two values along as arguments.

Example:

```
<?php
printf("%d bottles of water cost Rs %f", 20, 400);
?>
```

Output:

20 bottles of water cost Rs 400

The sprintf() Statement

The sprintf() statement is similar to printf() statement, except that the output is assigned to a string instead of rendered to the browser.

Syntax:

```
stringsprintf(string format [, mixed arguments])
```

Example:

```
<?php
$cost = sprintf("$%.2f", 43.2);
?>
```

$cost will carry $43.20i.e $cost = $43.20

PHP's Supported Data Types

A data type is the generic name assigned to any data sharing a common set of characteristics. Data types include integer, Boolean, string, float, and array.

Scalar Data Types

Scalar data types are used to represent a single value. Several data types fall under this category, including Boolean, integer, float, and string.

Boolean

The Boolean data type is named after George Boole (1815–1864), a mathematician who is considered to be one of the founding fathers of

information theory. The Boolean data type represents truth, supporting only two values: TRUE and FALSE (case insensitive).

Alternatively, we can use zero to represent FALSE, and any nonzero value to represent TRUE.

Examples:

$alive = false; → $alive is false.

$alive = 1; → $alive is true.

$alive = -1; → $alive is true.

$alive = 5; →$alive is true.

$alive = 0; → $alive is false.

Integer

An integer is representative of any whole number or, in other words, a number that does not contain fractional parts. PHP supports integer values represented in base 10 (decimal), base 8 (octal), and base 16 (hexadecimal) numbering systems, although it's likely you'll only be concerned with the first of those systems.

Example:

21→ decimal

-234567→ decimal

0677→ octal

0xE4C→ hexadecimal

Float

Floating-point numbers, additionally referred to as floats, doubles, or real numbers, allow you to define numbers that contain fractional parts. Floats are utilized to represent monetary values, weights, distances, and an entire host of other representations in which a simple integer value won't suffice.

Example:

3.7868

4.2

3.4e7

3.21E+11

String

A string is a sequence of characters considered as a contiguous group. Strings are placed between single or double quotes.

Example:

"Sakshi Education"

'9*sakshi\n'

"786$^%123"

Compound Data Types

Compound data types allow multiple items of the same type to be aggregated under a single representative entity. The array and the object fall into this category.

Array

Arrays are used to aggregate similar items together, arranging and referencing them in a specific way. This data structure is known as an ***array.*** Arrays are formally known as an indexed collection of data values. Each member in array index references a corresponding value and can be a simple numerical reference to the value's position in the list, or it could have some direct correlation to the value.

Example:

$state[0] = "Andhra Pradesh";

$state[1] = "Arunachal Pradesh";

$state[2] = "Assam";

...

$state[29] = "West Bengal";

If we want to assign state to their capitals, Rather than base the keys on a numerical index, you might instead use an associative index, like this:

$state["Andhra Pradesh"] = "Hyderabad";

$state["Arunachal Pradesh"] = "Itanagar";

$state["Assam"] = "Dispur";

...

$state["West Bengal"] = "Kolkata";

Object

The object is a central concept of the object-oriented programming paradigm. The other compound data type supported by PHP is the object.

Objects are completely different from other data types in PHP. In PHP, an object must be explicitly declared. This declaration of an object's characteristics and behaviour takes place within a class.

Example:

```
<?php

classCycle{

var $tyre;

functioncount($tyre = "2") {

$this->tyre = $tyre;

function display(){

return $this->tyre;

}

}

}
```

Variables

A variable is a symbol that can store different values at different times or you can say that a variable is a named memory location that contains data and may be manipulated throughout the execution of the program.

Variable Declaration

A variable always begins with a dollar sign, $, in PHP which is then followed by the variable name. Variable names follow the same naming rules as identifiers. That is, a variable name can begin with either a letter or an underscore and can consist of letters, underscores, numbers, or other ASCII characters ranging from 127 through 255. The following some valid variables...

- $Sakshi
- $Sakshi_education
- $_Sakshi_post
- $sakshinews

Note that variables are case sensitive. For instance, the following variables bear no relation to one another:

- $sakshi
- $Sakshi
- $SAKSHI

In PHP variables do not have to be explicitly declared as in a language such as C. Rather, variables can be declared and assigned values simultaneously. Once you've declared your variables, you can begin assigning values to them.

Two methodologies are available for variable assignment:

- By value
- By reference.

By value

Assignment "by value" simply involves copying the value of the assigned expression to the variable assignee. This is the most common type of assignment.

Examples:

```
$colour = "blue";

$value = 27;

$cost = 27;

$total = 21 + "15"; // $total = 36
```

Each of the above variables passes a copy of the expression assigned to it. In above example, $value and $cost each possesses their own unique copy of the value 27. If you want two variables to carry the same copy of a value, you need to assign by reference.

By reference

In PHP 4 they introduced the ability to assign value to variables by reference, which means that you can create a variable that refers to the same

value as another variable does. You can assign variables by reference, by appending “&” to the equal sign.

```
<?php
    $first = "Good";
    $second =& $first; // $first and $second both equal "Good"
    $second = "Goodbye"; // $first and $second both equal "Goodbye"
?>
```

Variable Scope

You can declare variables anywhere in PHP script; however declaration of variables has to be done either by value or by reference. The location of the declaration greatly influences the realm in which a variable can be accessed, however. This accessibility domain is known as its scope.

PHP variables can be one of four scope types:

- Local variables
- Global variables
- Function parameters
- Static variables

Local Variables

If a variable is declared inside a function, then it is considered as local variable, i.e.., it can be referenced only in that function. When you exit the function in which a local variable has been declared, that variable and its corresponding value will be destroyed.

Example:

```
<?php
$y = 4;
function assigny () {
$y = 0;
printf("\$y inside function is %d <br />", $y);
}
assigny();
printf("\$y outside of function is %d <br />", $y);
?>
```

Output:

```
$y inside function is 0
$y outside of function is 4
```

In above example, two different values for $y are output, because the $y located inside the assigny() function is local. By changing the value of $y inside the function will not affect the value of outside the function. On the other hand, modifying the $y located outside of the function will not affect the value inside the function.

Function Parameters

In PHP, like in many other programming languages, any function that accepts arguments must declare those arguments in the function header. Although those arguments accept values, which come from outside of the function, they are no longer accessible once the function has exited.

Global Variables

Global variable can be accessed in any part of the program. To modify a global variable, it must be explicitly declared as *global* in the function in which it is to be modified. By placing *global* keyword in front of the variable then it should be considered as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name.

Example:

```
<?php

$varble = 15;

functionaddit() {

global $varble;

$varble++;

echo "Variable value is $somevar";

}

addit();

?>
```

An alternate method for declaring a *global* variable is to use PHP's $GLOBALS array. Reconsidering the preceding example, you can use this array to declare the variable $varble to be global.

Example:

```
<?php

$varble = 15;
```

```
functionaddit() {

$GLOBALS["varble "]++;

}

addit();

echo "Variable value is ".$GLOBALS["varble "];

?>
```

Output:

Variable value is 16

Function Parameters

In PHP, as in numerous other programming languages, any function that acknowledges arguments must declare those arguments in the function header. In spite of that, arguments accept values coming from outside of the function and they are no longer accessible once the function exits.

// multiply a value by 10 and return it to the caller

```
<?php
   function y ($v) {
      $v = $v * 10;
      return $v;
    }
?>
```

Static Variables

The final type of variable scoping is known as static. By placing the static keyword in front of the variable name we can make a variable as static. A static variable does not lose its value when the function exits and will still hold that value if the function is called again.

Example

```
<?php
STATIC $variable;
function count_static() {
static $count = 0;
$count++;
echo $count;
```

```
echo "<br />";
}

count_static ();
count_static ();
count_static ();

?>
```

$count is static; it retains its previous value each time the function is executed. Therefore, the outcome is the following:

Output:

```
1
2
3
```