PHP

Dr K Chaitanya
Assistant Professor
Department of CSE
Dr YSR ANUCET
Acharya Nagarjuna University

- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).
- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"
- PHP can generate dynamic page content
- PHP can collect form data
- PHP can add, delete, modify data in your database
- With PHP you are not limited to output HTML. You can output images or PDF files. You can also output any text, such as XHTML and XML.
- PHP is compatible with almost all servers used today.
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. **For example** - Registration form.

Syntax

- Install WAMP software(windows + Apache server+PHP + My SQL)
- C:/wamp folder/www/new folder (php project)
- Save all files in this folder with .php extention
- Brackets is one editor for writing php programs.
- Open edge browser type localhost/new folder
- Ctrl + to increase the font size
- .(dot operator) is the concatenation operator
- echo statement is often used to output data to the screen.

second.php (php project) - Bracket

File Edit Find View Navigate Debug Help

Working Files

second.php

php project

first.php

second.php

```
<!DOCTYPE HTML>
<html>
<body>
<?php
echo " <h1>hello welcome</h1>";
?>
</body>
</html>
```

PHP Tutc | PHP Tutc

PHP | PHP Synt | PHP Exar | WampSe

localhost/php%20project/second.php

hello welcome

localhost/php%20project/

Index of /php project

Name	Last modified	Size	Description
Parent Directory		-	
first.php	2023-04-06 09:26	7	
second.php	2023-04-06 10:35	85	

Apache/2.4.54 (Win64) PHP/8.0.26 mod_fcgid/2.3.10-dev Server at localhost Port 80

variables

- Variables are "containers" for storing information.

Declaring PHP Variables:

- In PHP, a variable starts with the $ sign, followed by the name of the variable.

Rules for PHP variables:

- a variable starts with the $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive

- echo "x value is : $x"."
"; both are same
- echo "y value is : ".$y."
";

varibles1.php (php project)

File Edit Find View Navigate Debug Help

Working Files

second.php

varibles1.php

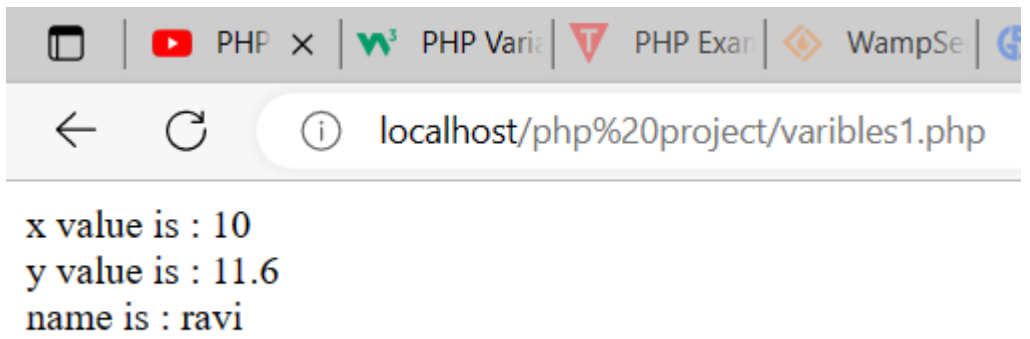
php project

first.php

second.php

varibles1.php

```php
<?php
$x=10;
$y=11.6;
$name="ravi";
echo "x value is : ".$x."<br>";
echo "y value is : ".$y."<br>";
echo "name is : ".$name;
?>
```



Types of variables

global variables :

- if we declare the variables outside of the function is called global variables.

local variables :

- if we declare the variables inside the function is called local variables.

static variable:

- if we declare the variables with static keyword.

```php
<?php
    $x=5;

    function myTest(){
        $y=10;
        static $z=15;
    }
    myTest();

    echo $x;
    echo $y;
    echo $z;
?>
```

5

Notice: Undefined variable: y in E:\PHP\wamp\www\youtube\variables2.php on line *11*

Call Stack

#	Time	Memory	Function	Location
1	0.0183	129392	{main}( )	...\variables2.php:0

Notice: Undefined variable: z in E:\PHP\wamp\www\youtube\variables2.php on line *12*

Call Stack

#	Time	Memory	Function	Location
1	0.0183	129392	{main}( )	...\variables2.php:0

```php
<?php
    $x=5;

    function myTest(){
        $y=10;
        echo $y."<br>";
        static $z=15;
        echo $z."<br>";
    }
    myTest();
    echo $x."<br>";
?>
```

```
10
155
```

Comments in PHP

- PHP Single Line Comments
- There are two ways to use single line comments in PHP.
- // (C++ style single line comment)
- # (Unix Shell style single line comment)
- PHP Multi Line Comments
- In PHP, we can comments multiple lines also. To do so, we need to enclose all lines within /* */.

```php
<?php
// this is C++ style single line comment
# this is Unix Shell style single line comment
echo "Welcome to PHP single line comments";
?>
```

```php
<?php
/*
Anything placed
within comment
will not be displayed
on the browser;
*/
echo "Welcome to PHP multi line comment";
?>
```

Data types in PHP

- PHP Data Types
- PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)
3. Special Types

- PHP Data Types: Scalar Types
- It holds only single value. There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

- PHP Data Types: Compound Types
- It can hold multiple values. There are 2 compound data types in PHP.

1. array
2. object

- PHP Data Types: Special Types
- There are 2 special data types in PHP.

1. resource
2. NULL

Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**. It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

```
1.<?php
2.    if (TRUE)
3.        echo "This condition is TRUE.";
4.    if (FALSE)
5.        echo "This condition is FALSE.";
6.?>
```

PHP Integer

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
- The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

```
1.<?php
2.    $dec1 = 34;
3.    $oct1 = 0243;
4.    $hexa1 = 0x45;
5.    echo "Decimal number: " .$dec1. "</br>";
6.    echo "Octal number: " .$oct1. "</br>";
7.    echo "HexaDecimal number: " .$hexa1. "</br>";
8.?>
```

PHP Float

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

```
1.<?php
2.    $n1 = 19.34;
3.    $n2 = 54.472;
4.    $sum = $n1 + $n2;
5.    echo "Addition of floating numbers: " .$sum;
6.?>
```

PHP String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.
String values must be enclosed either within **single quotes** or in **double quotes**. But both are treated differently.

```
1.<?php
2.    $company = “PHP";
3.    //both single and double quote statements will treat diff
erent
4.    echo "Hello $company";
5.    echo "</br>";
6.    echo 'Hello $company';
7.?>
```

Output:

```
Hello PHP
Hello $company
```

PHP Array

An array is a compound data type. It can store multiple values of same data type in a single variable.

```
1.<?php
2.    $bikes = array ("Royal Enfield", "Yamaha", "KTM");
3.    var_dump($bikes);   //the var_dump() function returns
the datatype and values
4.    echo "</br>";
5.    echo "Array Element1: $bikes[0] </br>";
6.    echo "Array Element2: $bikes[1] </br>";
7.    echo "Array Element3: $bikes[2] </br>";
8.?>
```

Output:

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }
Array Element1: Royal Enfield
Array Element2: Yamaha
Array Element3: KTM
```

PHP object

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared

```php
<?php
    class bike {
        function model() {
            $model_name = "Royal Enfield";
            echo "Bike Model: " .$model_name;
          }
    }
    $obj = new bike();
    $obj -> model();
?>
```

Output:

```
Bike Model: Royal Enfield
```

PHP Resource

Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources. **For example** - a database call. It is an external resource.

PHP Null

Null is a special data type that has only one value: **NULL**. There is a convention of writing it in capital letters as it is case sensitive.

```php
1.<?php
2.    $nl = NULL;
3.    echo $nl;   //it will not give any output
4.?>
```

constants

- PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

1. Using define() function:

define(name, value, **case**-insensitive)

1. **name:** It specifies the constant name.
2. **value:** It specifies the constant value.
3. **case-insensitive:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

```
<?php
define("MESSAGE","Hello PHP");
define(“A”,10);
echo MESSAGE;
?>
```

2. Using const keyword

constants are automatically global throughout the script.

Ex: const A=20;

```
1  <?php
2      define("MIN",35);
3      echo MIN."<br>";
4
5      MIN=MIN+5;
6      echo MIN;
7  ?>
```

Magic constants

- Magic constants are the predefined constants in PHP which get changed on the basis of their use. They start with double underscore (__) and ends with double underscore.
- Magic constants are case-insensitive.
- There are **nine** magic constants in PHP. In which eight magic constants start and end with double underscores (__).

__LINE__: It returns the current line number of the file, where this constant is used.

```
echo "You are at line number " . __LINE__ . "<br><br>";
```

__FILE__ : This magic constant returns the full path of the executed file. echo __FILE__ . "

"; (include file name also)

__DIR__ : It returns the full directory path of the executed file.(exclude file name). echo __DIR__ . "

"; echo dirname(__FILE__) . "

"; both are same.

__FUNCTION__: This magic constant returns the function name, where this constant is used. It will return blank if it is used outside of any function.

```
echo 'The function name is '. __FUNCTION__ . "<br><br>";
```

__CLASS__: It returns the class name, where this magic constant is used.

__TRAIT__: similar to single inheritance. This magic constant returns the trait name, where it is used.

__METHOD__: It returns the name of the class method where this magic constant is included.

__NAMESPACE__: It returns the current namespace where it is used.

ClassName::class : This magic constant does not start and end with the double underscore (__). It returns the fully qualified name of the ClassName.

Operators

- PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values.
- PHP Operators can be categorized in following forms:
- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Spaceship Operator
- We can also categorize operators on behalf of operands. They can be categorized in 3 forms:
- **Unary Operators:** works on single operands such as ++, -- etc.
- **Binary Operators:** works on two operands such as binary +, -, *, / etc.
- **Ternary Operators:** works on three operands such as "?:".

Arithmetic Operators

The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

```php
<?php
$x = 10;
$y = 4;
echo($x + $y); // Outputs: 14
echo($x - $y); // Outputs: 6
echo($x * $y); // Outputs: 40
echo($x / $y); // Outputs: 2.5
echo($x % $y); // Outputs: 2
?>
```

The exponentiation (**) operator has been introduced in PHP 5.6.

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

```php
$x = 20;
$x += 30;
echo $x; // Outputs: 50
```

Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

Comparison operators

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

The var_dump() function dumps information about one or more variables. The information holds type and value of the variable(s).

```php
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z);   // Outputs: boolean true
var_dump($x === $z);  // Outputs: boolean false
var_dump($x != $y);   // Outputs: boolean true
var_dump($x !== $z);  // Outputs: boolean true
var_dump($x < $y);    // Outputs: boolean true
var_dump($x > $y);    // Outputs: boolean false
var_dump($x <= $y);   // Outputs: boolean true
var_dump($x >= $y);   // Outputs: boolean false
?>
```

```php
<!DOCTYPE html>
<html>
<body>

<?php
$a = 32;
echo var_dump($a) . "<br>";

$b = "Hello world!";
echo var_dump($b) . "<br>";

$c = 32.5;
echo var_dump($c) . "<br>";

$d = array("red", "green", "blue");
echo var_dump($d) . "<br>";

$e = array(32, "Hello world!", 32.5, array("red", "green", "blue"));
echo var_dump($e) . "<br>";

// Dump two variables
echo var_dump($a, $b) . "<br>";
?>

</body>
</html>
```

int(32)
string(12) "Hello world!"
float(32.5)
array(3) { [0]=> string(3) "red" [1]=> string(5) "green" [2]=> string(4) "blue" }
array(4) { [0]=> int(32) [1]=> string(12) "Hello world!" [2]=> float(32.5) [3]=> array(3) { [0]=> string(3) "red"
int(32) string(12) "Hello world!"

Incrementing/Decrementing Operators

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

PHP Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

```php
<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))){
    echo "$year is a leap year.";
} else{
    echo "$year is not a leap year.";
}
?>
```

String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

```php
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!

$x .= $y;
echo $x; // Outputs: Hello World!
?>
```

Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

```php
<?php
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
$z = $x + $y; // Union of $x and $y
var_dump($z);
var_dump($x == $y);   // Outputs: boolean false
var_dump($x === $y);  // Outputs: boolean false
var_dump($x != $y);   // Outputs: boolean true
var_dump($x <> $y);   // Outputs: boolean true
var_dump($x !== $y);  // Outputs: boolean true
?>
```

PHP Spaceship Operator PHP 7

PHP 7 introduces a new spaceship operator (`<=>`) which can be used for comparing two expressions.

It is also known as combined comparison operator.

The spaceship operator returns `0` if both operands are equal, `1` if the left is greater, and `-1` if the right is greater.

```php
<?php
// Comparing Integers
echo 1 <=> 1; // Outputs: 0
echo 1 <=> 2; // Outputs: -1
echo 2 <=> 1; // Outputs: 1

// Comparing Floats
echo 1.5 <=> 1.5; // Outputs: 0
echo 1.5 <=> 2.5; // Outputs: -1
echo 2.5 <=> 1.5; // Outputs: 1

// Comparing Strings
echo "x" <=> "x"; // Outputs: 0
echo "x" <=> "y"; // Outputs: -1
echo "y" <=> "x"; // Outputs: 1
?>
```

Conditional statements

- PHP If Else
- PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

- if
- if-else
- if-else-if
- nested if

Example

```php
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

Output:

```
12 is less than 100
```

```php
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

Output:

```
12 is even number
```

```php
<?php
  $marks=69;
  if ($marks<33){
     echo "fail";
  }
  else if ($marks>=34 && $marks<50) {
     echo "D grade";
  }
  else if ($marks>=50 && $marks<65) {
    echo "C grade";
  }
  else if ($marks>=65 && $marks<80) {
     echo "B grade";
  }
  else if ($marks>=80 && $marks<90) {
     echo "A grade";
  }
  else if ($marks>=90 && $marks<100) {
     echo "A+ grade";
  }
  else {
     echo "Invalid input";
```

Example

```php
<?php
        $age = 23;
   $nationality = "Indian";
   //applying conditions on nationality and age
   if ($nationality == "Indian")
   {
      if ($age >= 18) {
         echo "Eligible to give vote";
      }
      else {
         echo "Not eligible to give vote";
      }
   }
?>
```

Output:

```
Eligible to give vote
```

```php
<?php
$num=20;
switch($num){
case 10:
echo("number is equals to 10");
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

Output:

```
number is equal to 20
```

```php
<?php
   $ch = 'U';
   switch ($ch)
   {
      case 'a':
         echo "Given character is vowel";
         break;
      case 'e':
         echo "Given character is vowel";
         break;
      case 'i':
         echo "Given character is vowel";
         break;
      case 'o':
         echo "Given character is vowel";
         break;
      case 'u':
         echo "Given character is vowel";
         break;
      case 'A':
         echo "Given character is vowel";
```

```php
<?php
  $ch = "B.Tech";
  switch ($ch)
  {
     case "BCA":
        echo "BCA is 3 years course";
        break;
     case "Bsc":
        echo "Bsc is 3 years course";
        break;
     case "B.Tech":
        echo "B.Tech is 4 years course";
        break;
     case "B.Arch":
        echo "B.Arch is 5 years course";
        break;
     default:
        echo "Wrong Choice";
        break;
  }
?>
```

```php
<?php
for($n=1;$n<=1(
echo "$n<br/>";
}
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

```php
<?php
$n=1;
do{
echo "$n<br/>";
$n++;
}while($n<=10);
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

```php
<?php
$season=array("summer","winter","spring","autumn");
foreach( $season as $arr ){
  echo "Season is: $arr<br />";
}
?>
```

Output:

```
Season is: summer
Season is: winter
Season is: spring
Season is: autumn
```

Expressions

Almost everything in a PHP script is an expression. Anything that has a value is an expression. In a typical assignment statement ($x=100), a literal value, a function or operands processed by operators is an expression, anything that appears to the right of assignment operator (=)

Syntax

```
$x=100; //100 is an expression
$a=$b+$c; //b+$c is an expression
$c=add($a,$b); //add($a,$b) is an expresson
$val=sqrt(100); //sqrt(100) is an expression
$var=$x!=$y; //$x!=$y is an expression
```

PHP String

- PHP string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set. There are 4 ways to specify a string literal in PHP.

1.single quoted

2.double quoted

3.heredoc syntax

4.newdoc syntax (since PHP 5.3)

Single Quoted

- We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.
- For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\). All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.

```php
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

```
Hello text within single quote
```

```php
<?php
$str1='Hello text
multiple line
text within single quoted string';
$str2='Using double "quote" directly inside single quoted string';
$str3='Using escape sequences \n in single quoted string';
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

```
Hello text multiple line text within single quoted string
Using double "quote" directly inside single quoted string
Using escape sequences \n in single quoted string
```

```php
<?php
$num1=10;
$str1='trying variable $num1';
$str2='trying backslash n and backslash t inside single quoted string \n \t';
$str3='Using single quote \'my quote\' and \\backslash';
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

```
trying variable $num1
trying backslash n and backslash t inside single quoted string \n \t
Using single quote 'my quote' and \backslash
```

Double Quoted

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

```php
<?php
$str="Hello text within double quote";
echo $str;
?>
```

```
Hello text within double quote
```

```php
<?php
$str1="Hello text
multiple line
text within double quoted string";
$str2="Using double \"quote\" with backslash inside double quoted string";
$str3="Using escape sequences \n in double quoted string";
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

```
Hello text multiple line text within double quoted string
Using double "quote" with backslash inside double quoted string
Using escape sequences in double quoted string
```

```php
<?php
$num1=10;
echo "Number is: $num1";
?>
```

```
Number is: 10
```

Heredoc

Heredoc syntax (<<<) is the third way to delimit strings. In Heredoc syntax, an identifier is provided after this heredoc <<< operator, and immediately a new line is started to write any text. To close the quotation, the string follows itself and then again that same identifier is provided. That closing identifier must begin from the new line without any whitespace or tab.

Naming Rules

The identifier should follow the naming rule that it must contain only alphanumeric characters and underscores, and must start with an underscore or a non-digit character.

```
<?php
   $str = <<<Demo
It is a valid example
Demo;    //Valid code as whitespace or tab is not valid before closing identifier
echo $str;
?>
```

```
It is a valid example
```

We cannot use any whitespace or tab before and after the identifier and semicolon, which means identifier must not be indented. The identifier must begin from the new line.

```
<?php
   $str = <<<Demo
It is Invalid example
      Demo;    //Invalid code as whitespace or tab is not valid before closing identifier
echo $str;
?>
```

```
Parse error: syntax error, unexpected end of file in C:\xampp\htdocs\xampp\PMA\heredoc.php on line 7
```

Heredoc is similar to the double-quoted string, without the double quote, means that quote in a heredoc are not required. It can also print the variable's value.

```
<?php
   $city = 'Delhi';
   $str = <<<DEMO
Hello! My name is Misthi, and I live in $city.
DEMO;
   echo $str;
 ?>
```

```
Hello! My name is Misthi, and I live in Delhi.
```

Newdoc

Newdoc is similar to the heredoc, but in newdoc parsing is not done. It is also identified with three less than symbols <<< followed by an identifier. But here identifier is enclosed in single-quote, **e.g. <<<'EXP'**. Newdoc follows the same rule as heredocs.

The difference between newdoc and heredoc is that - Newdoc is a **single-quoted string** whereas heredoc is a **double-quoted string**.

```php
<?php
echo <<<'EOD'
Example of string spanning multiple lines
using nowdoc syntax. Backslashes are always treated literally,
e.g. \\ and \'.
EOD;
```

```
Example of string spanning multiple lines
using nowdoc syntax. Backslashes are always treated literally,
e.g. \\ and \'.
```

String functions in PHP

PHP strtolower() function:
The strtolower() function returns string in lowercase letter.
string strtolower (string $string)

PHP strtoupper() function:
The strtoupper() function returns string in uppercase letter.
string strtoupper (string $string)

PHP ucfirst() function:
The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.
string ucfirst (string $str)

PHP lcfirst() function:
The lcfirst() function returns string converting first character into lowercase. It doesn’t change the case of other characters.
string lcfirst (string $str)

PHP ucwords() function:
The ucwords() function returns string converting first character of each word into uppercase.
string ucwords (string $str)

PHP strrev() function:
The strrev() function returns reversed string.
string strrev (string $string)

PHP strlen() function:
The strlen() function returns length of the string.
int strlen (string $string)

PHP bin2hex() Function: PHP bin2hex() function is used to convert string value of ASSCII characters to hexadecimal value.

```
$str ="Hello";
echo "<br>"."By using 'bin2hex()' Method your hexadecimal value is: ".bin2hex($str);
```

48656c6c6f

PHP string ltrim() function: PHP string ltrim() function is predefined function. It is often used to remove whitespace from both sides of a string or other character from the left side of a string.

PHP string md5() function: PHP string md5() is predefined function. It is used to calculate the MD5 hash of a string. It uses the RSA DATA security. It returns the hash as a 32 character hexadecimal number.

PHP string md5_file() Function: PHP string md5_file() function is in-built important function. It is used to calculate the MD5 hash of a file. It uses the RSA Data Security. It returns the md5 hash on success, or FALSE on failure.

PHP string md5_file() Function: PHP string md5_file() function is in-built important function. It is used to calculate the MD5 hash of a file. It uses the RSA Data Security. It returns the md5 hash on success, or FALSE on failure.

PHP String substr_replace() Function: The substr_replace is an in-built function of PHP, which replaces a part of the string with another text within a string.

PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

PHP Array Types:

There are 3 types of array in PHP.

1.Indexed Array
2.Associative Array
3.Multidimensional Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

```
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
```

OR

```
1.$season[0]="summer";
2.$season[1]="winter";
3.$season[2]="spring";
4.$season[3]="autumn";
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
```

Traversing PHP Indexed Array:

We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

```
1.<?php
2.$size=array("Big","Medium","Short");
3.foreach( $size as $s )
4.{
5.  echo "Size is: $s<br />";
6.}
7.?>
```

Length of PHP Indexed Array:

PHP provides count() function which returns length of an array.

```
1.$size=array("Big","Medium","Short");
2.echo count($size);
```

PHP array_reverse() function:

PHP array_reverse() function returns an array containing elements in reversed order.

```
1.$season=array("summer","winter","spring","autumn");
2.$reverseseason=array_reverse($season);
```

PHP sort() function

PHP sort() function sorts all the elements in an array.

```
1.$season=array("summer","winter","spring","autumn");
```

PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key or index if search is successful.

```
1.$season=array("summer","winter","spring","autumn");
2.$key=array_search("spring",$season);
```

PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

```
1.$name1=array("sonoo","john","vivek","smith");
2.$name2=array("umesh","sonoo","kumar","smith");
3.$name3=array_intersect($name1,$name2);
```

PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

```
"250000","Ratan"=>"200000");
```

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

PHP Associative Array:

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

```
$salary=array("Sonoo"=>"350000","John"=>"450000","Kranti"=>"200000");
1.echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
2.echo "John salary: ".$salary["John"]."<br/>";
3.echo "Kartik salary: ".$salary["Kranti"]."<br/>";
```

OR

```
1.$salary["Sonoo"]="350000";
2.$salary["John"]="450000";
3.$salary["Kranti"]="200000";
```

Traversing PHP Associative Array:

By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

```
1.$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
2.foreach($salary as $k => $v)
3.{
4.echo "Key: ".$k." Value: ".$v."<br/>";
5.}
```

PHP Multidimensional Array

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

```
1.<?php
2.$emp = array
3.  (
4.  array(1,"sonoo",400000),
5.  array(2,"john",500000),
6.  array(3,"rahul",300000)
7.  );
8.
9.for ($row = 0; $row < 3; $row++) {
10.  for ($col = 0; $col < 3; $col++) {
11.    echo $emp[$row][$col]."  ";
12.  }
13.  echo "<br/>";
14.}
15.?>
```

```
1 sonoo 400000
2 john 500000
3 rahul 300000
```

PHP Functions

- PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

PHP User-defined Functions

- We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

```
function functionname(){
//code to be executed
}
```

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**

```php
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

```php
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of call by reference in PHP.

File: functionref.php

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

```
Hello Rajesh
Hello Sonoo
Hello John
```

PHP Math

PHP provides many predefined math constants and functions that can be used to perform mathematical operations.

abs() :

The abs() function returns absolute value of given number. It returns an integer value but if you pass floating point value, it returns a float value.

```
echo (abs(-7)."<br/>"); // 7 (integer)
```

ceil() :

The ceil() function rounds fractions up.

```
echo (ceil(7.333)."<br/>");// 8
```

floor() :

The floor() function rounds fractions down.

```
echo (floor(7.333)."<br/>");// 7
```

sqrt() :

The sqrt() function returns square root of given argument.

```
1.echo (sqrt(7)."<br/>");// 2.64575131106 46
```

decbin() function

The decbin() function converts decimal number into binary. It returns binary number as a string.

```
echo (decbin(22) "<br/>");// 10110
```

dechex() :

The dechex() function converts decimal number into hexadecimal. It returns hexadecimal representation of given number as a string.

```
echo (dechex(22)."<br/>");// 16
```

decoct() :

The decoct() function converts decimal number into octal. It returns octal representation of given number as a string.

```
echo (decoct(22)."<br/>");// 26
```

base_convert() :

The base_convert() function allows you to convert any base number to any base number. For example, you can convert hexadecimal number to binary, hexadecimal to octal, binary to octal, octal to hexadecimal, binary to decimal etc.

```
echo (base_convert($n1,10,2)."<br/>");// 1010
```

bindec() :

The bindec() function converts binary number into decimal.

```
echo (bindec(1011)."<br/>");// 11
```

PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP super globals $_GET and $_POST. The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

File: form1.html

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```

File: welcome.php

```
<?php
$name=$_GET["name"];//receiving name field value in $name variable
echo "Welcome, $name";
?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

File: form1.html

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
```

File: login.php

```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password variable

echo "Welcome: $name, your password is: $password";
?>
```

What is Validation ?

Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows −

- **Client-Side Validation** − Validation is performed on the client machine web browsers.
- **Server Side Validation** − After submitted by data, The data has sent to a server and perform validation checks in server machine.

Form Validation in PHP

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

1. Empty String
2. Validate String
3. Validate Numbers
4. Validate Email
5. Validate URL
6. Input length

Empty String:

If the user leaves the required field empty, it will show an error message.

```php
if (empty($_POST["name"])) {
  $nameErr = "Name is required";
} else {
  $name = test_input($_POST["name"]);
}
```

Validate String

```php
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
    $ErrMsg = "Only alphabets and whitespace are allowed.";
            echo $ErrMsg;
} else {
    echo $name;
}
```

Validate Number

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
    $ErrMsg = "Only numeric value is allowed.";
    echo $ErrMsg;
} else {
    echo $mobileno;
}
```

Validate Email

```
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
    $ErrMsg = "Email is not valid.";
            echo $ErrMsg;
} else {
    echo "Your valid email address is: " .$email;
}
```

Input Length Validation

```php
$mobileno = ($_POST ["Mobile"]);
$length = strlen ($mobileno);

if ( $length < 10 && $length > 10) {
    $ErrMsg = "Mobile must have 10 digits.";
        echo $ErrMsg;
} else {
    echo "Your Mobile number is: " .$mobileno;
}
```

Button Click Validate

```php
if (isset ($_POST['submit']) {
    echo "Submit button is clicked.";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        echo "Data is sent using POST method ";
    }
} else {
    echo "Data is not submitted";
}
```

Create and validate a Registration form

Create a registration form using HTML and perform server-side validation using PHP.

```
<!DOCTYPE html>
<html>
<head>
<style>
.error {color: #FF0001;}
</style>
</head>
<body>

<?php
// define variables to empty values
$nameErr = $emailErr = $mobilenoErr = $genderErr = $websiteErr = $agreeErr = "";
$name = $email = $mobileno = $gender = $website = $agree = "";
```

```php
//Input fields validation
if ($_SERVER["REQUEST_METHOD"] == "POST") {

//String Validation
  if (emptyempty($_POST["name"])) {
     $nameErr = "Name is required";
  } else {
     $name = input_data($_POST["name"]);
       // check if name only contains letters and whitespace
       if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
          $nameErr = "Only alphabets and white space are allowed";
       }
  }
 //Email Validation
 if (emptyempty($_POST["email"])) {
       $emailErr = "Email is required";
 } else {
       $email = input_data($_POST["email"]);
       // check that the e-mail address is well-formed
       if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
          $emailErr = "Invalid email format";
       }
 }
```

```php
//Number Validation
if (emptyempty($_POST["mobileno"])) {
      $mobilenoErr = "Mobile no is required";
} else {
      $mobileno = input_data($_POST["mobileno"]);
      // check if mobile no is well-formed
      if (!preg_match ("/^[0-9]*$/", $mobileno) ) {
      $mobilenoErr = "Only numeric value is allowed.";
      }
   //check mobile no length should not be less and greator than 10
   if (strlen ($mobileno) != 10) {
      $mobilenoErr = "Mobile no must contain 10 digits.";
      }
}
```

```php
//URL Validation
if (emptyempty($_POST["website"])) {
   $website = "";
} else {
      $website = input_data($_POST["website"]);
      // check if URL address syntax is valid
                                    if    (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0
9+&@#\/%=~_|]/i",$website)) {
         $websiteErr = "Invalid URL";
      }
}
```

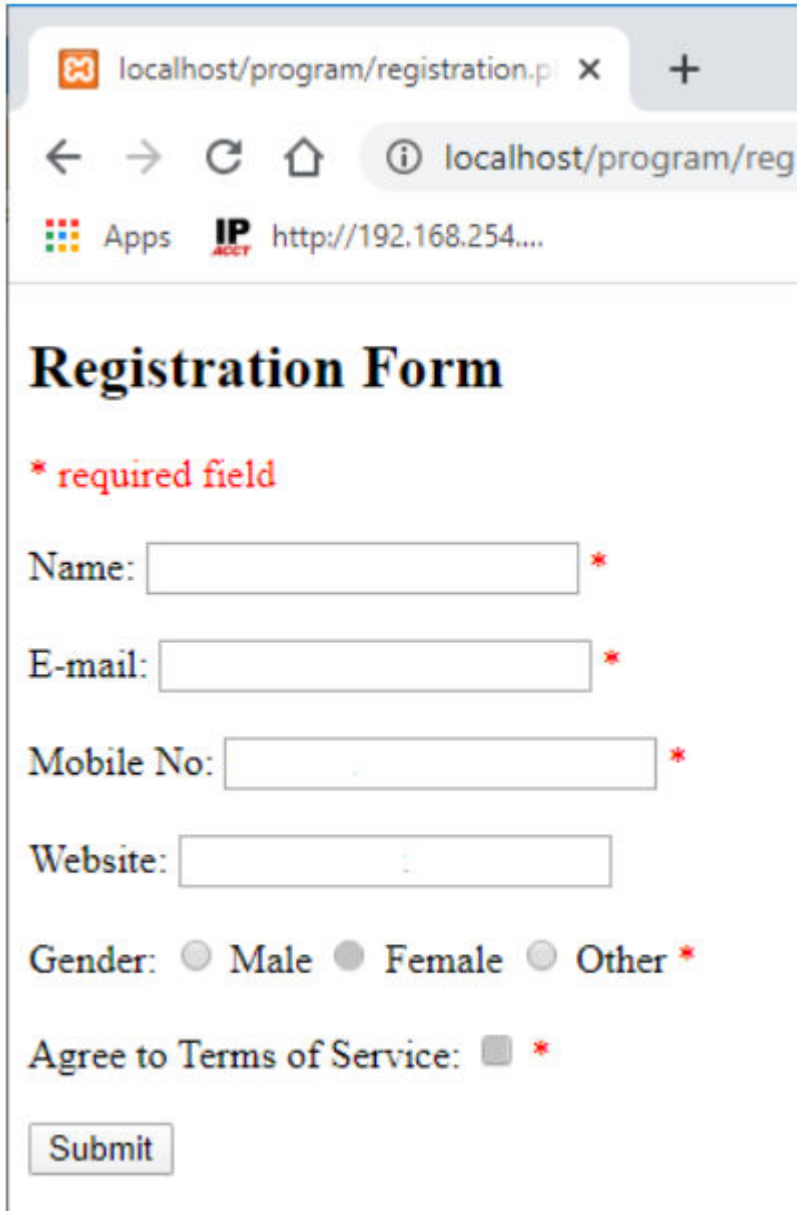
```php
    //Empty Field Validation
    if (emptyempty ($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = input_data($_POST["gender"]);
    }

    //Checkbox Validation
    if (!isset($_POST['agree'])){
        $agreeErr = "Accept terms of services before submit.";
    } else {
        $agree = input_data($_POST["agree"]);
    }
}
function input_data($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

```
<h2>Registration Form</h2>
<span class = "error">* required field </span>
<br><br>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" >
   Name:
   <input type="text" name="name">
   <span class="error">* <?php echo $nameErr; ?> </span>
   <br><br>
   E-mail:
   <input type="text" name="email">
   <span class="error">* <?php echo $emailErr; ?> </span>
   <br><br>
   Mobile No:
   <input type="text" name="mobileno">
   <span class="error">* <?php echo $mobilenoErr; ?> </span>
   <br><br>
   Website:
   <input type="text" name="website">
   <span class="error"><?php echo $websiteErr; ?> </span>
   <br><br>
   Gender:
   <input type="radio" name="gender" value="male"> Male
   <input type="radio" name="gender" value="female"> Female
   <input type="radio" name="gender" value="other"> Other
   <span class="error">* <?php echo $genderErr; ?> </span>
   <br><br>
   Agree to Terms of Service:
   <input type="checkbox" name="agree">
   <span class="error">* <?php echo $agreeErr; ?> </span>
   <br><br>
   <input type="submit" name="submit" value="Submit">
   <br><br>
</form>
```

```php
<?php
    if(isset($_POST['submit'])) {
    if($nameErr == "" && $emailErr == "" && $mobilenoErr == "" && $genderErr == "" && $websiteErr == "" && $agree
Err == "") {
        echo "<h3 color = #FF0001> <b>You have sucessfully registered.</b> </h3>";
        echo "<h2>Your Input:</h2>";
        echo "Name: " .$name;
        echo "<br>";
        echo "Email: " .$email;
        echo "<br>";
        echo "Mobile No: " .$mobileno;
        echo "<br>";
        echo "Website: " .$website;
        echo "<br>";
        echo "Gender: " .$gender;
    } else {
        echo "<h3> <b>You didn't filled up the form correctly.</b> </h3>";
    }
    }
?>

</body>
</html>
```



The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

PHP isset() function

The isset() function is a built-in function of PHP, which is used to determine that a variable is set or not. If a variable is considered set, means the variable is declared and has a different value from the NULL. In short, it checks that the variable is declared and not null.

This function returns **true** if the variable is not null, otherwise it returns **false**. Note that the null character ("**\0**") is considered different from the PHP **NULL** constant.

PHP preg_match() function

The preg_match() function is a built-in function of PHP that performs a regular expression match. This function searches the string for pattern, and returns true if the pattern exists otherwise returns false.

Generally, the searching starts from the beginning of $subject string parameter. The optional parameter $offset is used to start the search from the specified position.

PHP Include and Require

PHP allows us to create various elements and functions, which are used several times in many pages. It takes much time to script these functions in multiple pages. Therefore, use the concept of **file inclusion** that helps to include files in various programs and saves the effort of writing code multiple times.

"PHP allows you to include file so that a page content can be reused many times. It is very helpful to include files when you want to apply the same HTML or PHP code to multiple pages of a website." There are two ways to include file in PHP.

1. include
2. require

Both include and require are identical to each other, except failure.

- **include** only generates a warning, i.e., E_WARNING, and continue the execution of the script.
- **require** generates a fatal error, i.e., E_COMPILE_ERROR, and stop the execution of the script.

Advantage

Code Reusability: By the help of include and require construct, we can reuse HTML code or PHP script in many PHP scripts.