

## Introduction php

PHP 1994 में **Rasmus Lerdorf** द्वारा develop की गयी थी। लेकिन market में यह **1995** में आयी थी , और इसी समय पर [Java](#) भी release की गयी थी। PHP के latest version 7. 4. 9 है , जो कि 16 Apr 2020 को release किया गया था। PHP एक Open Source , Interpreted , Object -Oriented scripting language है , जो कि सर्वर server पर run होती। PHP web development में सबसे ज्यादा use होने वाली language है , जो कि आसानी से web pages को dynamic बनाती है। और आज के समय में developers इसे ही prefer करते हैं , हालाँकि PHP के कुछ ऐसे features और functionalities हैं जो इसे दूसरी programming languages से अलग बनाती हैं।

### PHP Simple Example

```
<?php    $var = "Hello";  
    echo $var."<br>";  
    echo($var)."<br>";  
    print($var)."<br>";  
?>
```

### Why Use PHP ?

- Blogs को maintain करने के लिए WordPress भी PHP का ही use करती है।
- सबसे बड़ी सबसे Social मीडिया Platform Facebook भी PHP ही use की गई है।
- PHP से हम आसानी से Dynamic Web Pages से generate, और maintain कर सकते हैं।
- PHP Freely आपको मिल जाएगी इसे सीखने या testing के लिए आपको कोई pay नहीं करना पड़ेगा। PHP की [official Website](#) पर जाकर आप अपने System के according XAMPP , WAMPP या LAMPP download करके install कर सकते हैं।

## Advantages of PHP

- **Free & Open Source** : PHP free और open source है , मतलब आपको कोई Pay नहीं करना पड़ेगा आप आसानी से install और use कर सकते है ।
- **Object Oriented** : PHP , Object oriented programming language भी है , यानि आप PHP में Classes , Methods , members डिफाइन कर सकते हैं। Object-oriented programming एक approach है , जहा आप classes और Objects use करते हैं ,जैसे की आप JAVA या C++ में करते है।
- **Cross Platform** : PHP को आप किसी भी Operating System पर रन कर सकते है। चाहे बो Linux , Window , Unix किसी भी तरह के Operating System पर run कर सकते है।
- **Performance** : PHP की runtime performance बहुत अच्छी है , interpreted language होने की बजह से PHP Code Line BY Line execute होते है. हमें इसे compile जरूरत नहीं होती जैसा की C , JAVA में करते हैं।
- **Easy To Use** : PHP use करने के लिए बहुत ही easy language है , अगर आप थोड़ा सा भी C या जावा language के बारे में जानते हैं तो आप बहुत आसानी से इसे सीख सकते है ।
- **Powerful Library Support** : PHP के लिए functional library को आप आसानी से उसे कर सकते है और ये आसीन से आपको [GitHub](https://github.com) से आसानी से integrate कर सकते है , जैसे PDF , Excel etc . आप आसानी से integrate कर सकते हैं ।

## Disadvantages of PHP

- Security** : हालाँकि PHP Open Source है , इसलिए hackers आसानी से इसका source code देख सकते हैं ।
- Not suitable for large APP : PHP में बहुत ही ज्यादा large application को मेन्टेन कर पाना बहुत ही मुश्किल है।

## Syntax of php

Generally PHP file HTML, JavaScript, JQuery, aur PHP etc languages को contain कर सकती है , क्योंकि जब हम web page बनाते हैं तो जरूरत के हिसाब से हमें ये languages use करनी पड़ती हैं. PHP syntax बहुत ही आसान है हमें PHP Code को simply `<?php` से start करते हैं और `?>` से end करते हैं।

चूँकि PHP Script Server पर Run होती है और हमें plain HTML text as a result मिलता है। PHP file को हम `.php` extensionके साथ save करते हैं।

```
<?php
```

```
$var = "Hello";
```

```
echo $var;
```

```
?>
```

**Out put**

Hello

## PHP echo/print

PHP में हम value को print कराने के लिए **echo()** function का Use करते हैं , echo के बाद आप Parenthesis use कर भी सकते हैं और न भी करते हैं तो भी कोई problem नहीं है।

Example –

```
<?php
$name = "Bhuvanesh Kumar";
echo($name);
echo $name."</br>";
?>
```

Output

Bhuvanesh kumar  
Bhuvanesh kumar

Example में आप देख सकते हैं की value को हम दोनों तरह से Print करा सकते हैं। इसके अलावा PHP में **print()** function भी है जिसे हम value को print करने में ले सकते हैं।

Note - Example में '<br>' का use line break के लिए किया गया है और dot (.) का use [String Concatenation](#) के लिए।

```
<?php
$name = "Bhuvanesh Kumar";
print ($name);
print $name."</br>";
?>
```

## Difference Between Echo() and Print()

सबसे बड़ा difference यह है कि **echo()** function किसी value को print करने के बाद कुछ return नहीं करता जबकि **print()** function value को print करने के बाद 1 return करता है।

```
<?php
$name="Bhuvanesh";
echo print($name)."<br>";
?>
```

### Output

Bhuvanesh

1

Note - print() function हमेशा ही 1 return करता हो चाहे value print हो या न हो , variable Define किया गया हो या न किया गया हो।

## Php types

Type / Data type का simply मतलब है किसी variable का type जिससे हमें पता चल सके कि कोई variable किस type की value hold किये हुए है , या in future किस type की value hold करेगा।

Data Type के case में PHP बहुत ही **flexible** language है , अगर आपको Java या C की थोड़ी भी knowledge है तो आपको याद होगा कि किसी भी तरह के variable को define करने से पहले हमें उस variable का Data Type define करना पड़ता था। मतलब हमें variable define करते समय ये बताना पड़ता था कि वह variable किस type की value store करने जा रहा है।

किसी भी variable का type किसी programmer द्वारा नहीं किया जा सकता है , यह runtime में ही assign की गयी value के according decide होता है कि variable किस type का है।

PHP 10 primitives types को support करती है।

1.PHP scalar types :boolean : True , false

integer : 1 2 3 4....float : 1.3 , 0.9..string : 'single qoutes' or "double qoutes"

2.PHP compound types :[array](#) : array(value1,value2) or [value1, value2] >or ['key'=> 'value']object : class Object Ex- \$x = new ClassObj();callable : will know about iterable : like array or traversable Object

3.PHP special types :resources : file object When we opwn file to read or write.NULL : null type की value होती है जो , जो represent करती है कि variable में कोई value नहीं है।

Note - किसी भी variable का type check करने के लिए var\_dump() function का use किया जाता है , हालाँकि हम gettype() फंक्शन भी use कर सकते हैं।

## Boolean

Boolean Type में maximum दो value हो सकती हैं - True or False . Boolean variable define करने के लिए just True या False Assign कर दीजिये। लेकिन जब आप Boolean value true print कराओगे तो 1 print होगा जबकि false के लिए कुछ भी print नहीं होगा।

```
<?php
```

```
/* define boolean */
```

```
    $a_bool = true;
```

```
    $b_bool = False;
```

```
    echo "Boolean Value For True : ".$a_bool."<br>";
```

```
    echo "Boolean Value For False : ".$b_bool."<br>";
```

```
?>
```

Output

**Boolean Value For True : 1**

**Boolean Value For False :**

Exaple में देख सकते हैं कि True के लिए 1 है जबकि false के लिए कुछ भी print नहीं हुआ है।

## Integer

Integer को आप -1 , -2 . . . . 1 2 3 से डिफाइन कर सकते हैं। इसके अलावा Integer Decimal Numbers (base 10 ), Hexadecimal Numbers (base 16 ), Octa Numbers (base 8 ) और binary Numbers (base 2 ) से define कर सकते हैं।

```
<?php /* decimal number */           $dec = 1234;
/* octal number (equivalent to 83 decimal) */
$oct = 0123;
/* hexadecimal number (equivalent to 26 decimal) */
$hex = 0x1A;
/* binary number (equivalent to 255 decimal) */
$bin = 0b11111111;
/*Now Print these values*/
echo "Hexa Deciaml Value : ".$hex ." And Type :".gettype($hex)."<br>";
echo "Deciaml Value : ".$dec ." And Type :".gettype($dec)."<br>";
echo "Octa Value : ".$oct ." And Type :".gettype($oct)."<br>";
echo "Binary Value : ".$bin ." And Type :".gettype($bin)."<br>";
?>
```



## Output

Hexa Deciaml Value : 26 And Type :integer

Deciaml Value : 1234 And Type :integer

Octa Value : 83 And Type :integer

Binary Value : 255 And Type :integer

## Float

Floating points numbers को real numbers और **double** और **float** भी कहते हैं।

## Type Casting Php

किसी दी गयी value / variable को दूसरे type के variable में convert करना Type Casting कहते हैं।

## Boolean Type Casting

boolean में convert करने के लिए simply value के आगे (bool) या (boolean) prepend करना पड़ता है  
वैसे boolean के लिए हम prepend न भी करें तो भी यह value के according boolean treat करता है।  
जब हम Boolean में Convert करते हैं तो नीचे लिखी गयी value false होती हैं।

.boolean FALSE integer या float 0.

- null empty string ' ' or '0'.
- any undefined variable.
- empty array . Example \$x = array(); or \$x = [];
- empty Object

```
<?php
echo var_dump((bool) "")."<br>";
echo var_dump((bool) 1)."<br>";
echo var_dump((bool) -2)."<br>";
echo var_dump((bool) "foo")."<br>";
echo var_dump((bool) 2.3e5)."<br>";
echo var_dump((bool) array(12))."<br>";
echo var_dump((bool) array())."<br>";
echo var_dump((bool) "false")."<br>";
?>
```

Output

```
bool(false)
bool(true)
bool(true)
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)
```

## Integer Type Casting

किसी दी गयी value को Integer में convert करने के लिए (int) या (integer) prepend करना पड़ता है।

```
<?php
    echo var_dump((int) "1")."<br>";
    echo var_dump((int) 1)."<br>";
    echo var_dump((int) '1')."<br>";
    echo var_dump((int) 2.5)."<br>";
?>
```

Output

```
int(1)
int(1)
int(1)
int(2)
```

## Float Type Casting

किसी दी गयी value को Integer में convert करने के लिए (float) prepend करना पड़ता है।

```
<?php
    echo var_dump((float) "1.5")."<br>";
    echo var_dump((float) 1)."<br>";
    echo var_dump((float) '1')."<br>";
?>
```

**Output**

```
float(1.5)
float(1)
float(1)
```

## String Type Casting

किसी दी गयी value को Integer में convert करने के लिए (string) prepend करना पड़ता है।

```
<?php
```

```
    echo var_dump((string) true)."<br>";  
    echo var_dump((string) false)."<br>";  
    echo var_dump((string) 0)."<br>";  
    echo var_dump((string) 2.5)."<br>";
```

```
?>
```

Output

```
string(1) "1"
```

```
string(0) ""
```

```
string(1) "0"
```

```
string(3) "2.5"
```

## Array Type Casting

किसी दी गयी value / variable को Array में convert करने के लिए (Array) prepend करना पड़ता है।

```
<?php
    /*define a variable*/
    $int_var = (array)10;
    $string_var = (array)"String";
    $bool_var = (array)True;
    var_dump($int_var);
    echo "<br>";
    var_dump($string_var);
    echo "<br>";
    var_dump($bool_var);
?>
```

Output

```
array(1) { [0]=> int(10) }
array(1) { [0]=> string(6) "String" }
array(1) { [0]=> bool(true) }
```

## PHP Variables

variable किसी memory को दिया गया नाम है जो कि किसी value को Hold करती है या store करती है। मतलब जब हम Variable define कर रहे हैं तो हम memory में value को store करके उसे नाम दे रहे होते हैं। PHP में Variables \$ sign के साथ define किये जाते हैं , Valid PHP Variables को letter और underscore के साथ define किया जा सकता है।

```
<?php
$_4site = 'Valid Variable';
// valid; starts with an underscore
echo $_4site."<br>";
$4site = 'Invalid variable';
// invalid; starts with a number
echo $4site;
?>
```

Output

**Parse error:** syntax error, unexpected '4' (T\_LNUMBER), expecting variable (T\_VARIABLE) or '{' or '\$' in **C:\xampp\htdocs\test\variables.php** on line 5

अगर आप किसी variable को number के साथ start करते हैं तो Error आएगी।

PHP Variables case sensitive होते हैं , this means अगर आप एक ही name के Variables को small और capital letters में define करते हैं तो दोनों ही Variables different होंगे , और different value hold करेंगे।

```
<?php
$var = 'bhuvanesh kumar';
$Var = 'Bhuvanesh kumar';
echo "$var, $Var";
?>
```

Output

bhuvanesh kumar | Bhuvanesh kumar

ऊपर दिए गए example में आप देख सकते हैं , कि same name के define किये गए हैं लेकिन दोनों ही variables different value Hold किये हुए हैं।

PHP Variables define करते समय हमें कोई type करने की जरूरत भी नहीं पड़ती है , जैसे कि JAVA या C में कोई variable define करते समय इसका टाइप define करना पड़ता है। आप जिस type की value को assign कराओगे variable उसी type की value को hold कर लेगा।

```
<?php
```

```
$a = "String";  
$b = 'String';  
$c = 0;  
$d = 2.5;  
$e = 'C';  
$f = true;  
$g = "";  
$h = null;  
echo var_dump($a)."<br>";  
echo var_dump($b)."<br>";  
echo var_dump($c)."<br>";  
echo var_dump($d)."<br>";  
echo var_dump($e)."<br>";  
echo var_dump($f)."<br>";  
echo var_dump($g)."<br>";  
echo var_dump($h)."<br>";
```

Output

```
string(6) "String"  
string(6) "String"  
int(0)  
float(2.5)  
string(1) "C"  
bool(true)  
string(0) ""  
NULL
```

```
?>
```

## PHP Variable Scope

Variable Scope , किसी भी **Variable** के लिए एक accessible area / portion होता है जहाँ किसी **Variable** को define करने के बाद उसे access किया जा सके। PHP Variable Scope तीन तरह के होते हैं - Local Variable , Global Variable And Static Variable .

1.Local Variable

2.Global Variable

3.Static Variable

जयादातर PHP Variable single scope ही होते हैं और यह variables किसी included File से भी access किये जा सकते हैं।

### PHP Local Variable

Local Variables वो variables होते हैं जो किसी function के अंदर ही define किये जाते हैं और उसी function के अंदर ही accessible होते हैं , और न ही उस variable को उस function के बाहर access कर पाएंगे।

```
<?php
$x = "External Variable";
function test()
{
    $x = "Variable Inside Function";
    echo $x;
}
test();
?>
```

Output

Variable Inside Function

Note - अगर किसी फंक्शन के अंदर और बाहर एक ही नाम के variable define करेंगे तो function के अंदर वाले variable की priority ज्यादा होगी।



## PHP Global Variable

Global Variables वो variables होते हैं जो किसी function के बाहर **define** किये जाते हैं और लेकिन function के अंदर भी **access** कर सकते हैं। हालांकि इन्हें access करने के लिए हमें या तो उस variable को globally define करना पड़ता है या तो [Super Global variable](#) के through access करना पड़ता है।

```
<?php
$a = 1;
$b = 2;
function variable()
{
    /* first access using $GLOBALS variables */
    echo "Access Variables Using Super GLOBALS variable: ".$GLOBALS['a'].'.'.$GLOBALS['b'];
    /* now first make these variables global*/
    global $a, $b;
    echo "Access Global Variables : $a , $b";
}
variable();
?>
```

### Output

```
Access Variables Using Super GLOBALS variable: 1,2
Access Global Variables : 1 , 2
```

### Output

```
Access Variables Using Super GLOBALS
variable: 1,2
Access Global Variables : 1 , 2
```

**Note** - अगर किसी function के अंदर हमें बाहरी variable को access करना है तो उस variable को बिना Super GLOBAL variable के access नहीं कर पाएंगे और अगर हम ऐसा करते हैं तो Error आएगी।

## PHP Static Variable

PHP में Static Variables हमें एक feature provide कराते हैं , Normally जब हम किसी function में variable को define करते हैं , और उसे कुछ value assign करने के बाद उस variable पर कुछ task perform करते हैं , तो Function Run होने के बाद उस variable में initial value ही रहती है।

```
<?php
function static_variable()
{
    $x = 1;
    echo "Before Increment : ".$x."<br>";
    /* increment by one */
    $x++;
    echo "After Increment : ".$x;
}
/* Call the function */
static_variable();
/* Again Call the function */
static_variable();
?>
```

### Output

Before Increment :1

After Increment :2

Before Increment :1

After Increment :2

तो आप जैसा देख सकते हैं जीतनी बार आप function को call कराएँगे variable \$x initial value 1 ही रहेगी।

इसी कमी को static variable दूर करते हैं , means function के run होने के बाद last value variable को assign होगी तो दुबारा function call होने पर processing उसी value से start होती है।

Static variable define करने के लिए simply static keyword साथ define करते हैं।

## PHP Super Global Variables

PHP Super Global Variables कुछ predefined variables होते हैं , जिन्हे हम as a global variable की तरह PHP Script में कहीं भी use कर सकते हैं , PHP Super Global Variables को हमें define करने की जरूरत नहीं पड़ती है।

PHP में Super Global Variables कुछ इस प्रकार हैं-

- \$GLOBALS
- \$\_SERVER
- [\\$\\_GET](#)
- [\\$\\_POST](#)
- [\\$\\_REQUEST](#)
- [\\$\\_FILES](#)
- [\\$\\_SESSION](#)
- \$\_ENV
- [\\$\\_COOKIE](#)

### PHP \$GLOBALS

**\$GLOBALS** super global variable , सभी global scope में define किये global variable को reference करता है , या हम कह सकते हैं कि user द्वारा define किये गए global variables को **\$GLOBALS** variable के through access कर सकते हैं।

```
<?php
    $test_var = "Variable value";
    $x = "Variable Other value";
    global $test_var;
    echo $GLOBALS['test_var']."<br>";
    echo $GLOBALS['x']."<br>";
?>
```

Output

Variable value

Variable Other value

दिए गए example में आप देख सकते हैं कि किस तरह से हम global variables को **\$GLOBALS** super global variable के through access कर सकते हैं।

## PHP \$\_SERVER

**\$\_SERVER** super global variable हमें root path , current file , header info etc . provide करता है।

## PHP Constants

PHP में Constants, Identifier होते हैं जिन्हें हम variables की तरह ही define करते हैं , हालाँकि variables की तरह इनकी value mutable नहीं होती , means हम script के execution के समय इनकी value को change नहीं कर सकते हैं। जैसे हम variable को Define करके हम Runtime पर भी increment या decrement करके value change करते हैं।

**Understanding Identifier** - identifier का use program में किसी entity को Identify करने के लिए किया जाता है, ये unique होते हैं, define करने के बाद हम इन्हें program में कहीं भी use कर सकते हैं।

PHP में constants define करने के लिए हम define('var\_name', 'value') function का use करते हैं।

Note - हालाँकि Class में हम **constants** define करने के लिए **define()** function नहीं use कर सकते हैं यह सिर्फ Out Of The Class ही काम करता है।

## **Why Use Constants In PHP ?**

अब जबकि constant variable को डिफाइन करने के बाद हम इसकी value को change नहीं कर सकते हैं तो सवाल अतः है कि इन्हें हम use क्यों करते हैं , PHP में constant को use करने के कुछ reasons हैं

- जब हमें यह मालूम हो कि program में variable की value change नहीं होगी वहाँ **constant** use करते हैं , जैसे जब हम PHP में **MySQL** Connection establish करते हैं तो वहाँ पर use होने वाली value complete Project के लिए Immutable होती है।

Note - PHP में Constants Global होते हैं , this means हम function के बाहर Constants को define करके function के अंदर एक्सेस कर सकते हैं। इसके अलावा हम Constants variable में array भी store कर सकते हैं।

```
<?php      /*defining constant array variable outside the function*/
define('arr_var', ["value1", "value2"]);

function fun_name()
{
    /*access array variable*/

    foreach (arr_var as $value)
    {
        echo $value."<br>";
    }
}

/*call the function*/
fun_name();

?>
```

Output

Value1

Value2

constants variable में single value के अलावा array भी स्टोर कराकर use में ले सकते हैं।

## PHP Magic Constants

Magic Constants, PHP कुछ predefined constants जिनका Output उनके use किये गए place के according change होता रहता है।

### PHP \_\_LINE\_\_

\_\_LINE\_\_ PHP constant current line number return करता है , जिस line में यह use हो रहा है।

```
<?php /* Know the line number*/  
echo '__LINE__ variable tells the current line number : '.__LINE__;  
?>
```

Output

\_\_LINE\_\_ variable tells the current line number : 3

Example में आप देख सकते हैं कि \_\_LINE\_\_ current line return करता है।

### PHP \_\_FILE\_\_

```
<?php /* Know full path of file*/  
echo '__FILE__ variable tells the full path : '.__FILE__;  
?>
```

Output

\_\_FILE\_\_ variable tells the full path : C:\xampp\htdocs\test\test.php

## PHP \_\_DIR\_\_

**\_\_DIR\_\_** PHP constant current file के साथ full path return करता है।

```
<?php /* know the directory name*/
```

```
    echo '__DIR__ variable tells directory name of current file : '__DIR__';  
?>
```

Output :

**\_\_FILE\_\_ variable tells the full path : C:\xampp\htdocs\test\test.php**

## PHP \_\_DIR\_\_

**\_\_DIR\_\_** PHP constant current file के साथ full path return करता है।

```
<?php /* know the directory name*/
```

```
    echo '__DIR__ variable tells directory name of current file : '__DIR__';  
?>
```

Output :

**\_\_DIR\_\_ variable tells directory name of current file : C:\xampp\htdocs\test**



## \_\_FUNCTION\_\_

\_\_FUNCTION\_\_ PHP constant current function का नाम return करता है, जिस function में यह use हो रहा है । हालाँकि \_\_FUNCTION\_\_ की जगह आप \_\_METHOD\_\_ constants भी use कर सकते हैं।

```
<?php  /* define the function first*/  
function test_fun() {  
    echo '__FUNCTION__ variable tells the current function name : '.__FUNCTION__;  
}  
// call the function.  
test_fun();  
?>
```

Output:

\_\_FUNCTION\_\_ variable tells the current function name : test\_fun

## PHP Operators

**Operator** एक symbol होते हैं , जो process / operation represent करते हैं करते हैं , या हम कह सकते हैं की **Operators** को कोई process / या operation को perform करने के लिए use करते हैं।

PHP हमें different - different **Operators** provide कराती है different - different action perform करने के लिए। PHP में normally use होने वाले Operators कुछ इस प्रकार हैं -

- 1.Arithmetic Operators
- 2.Assignment Operators
- 3.Comparison Operators
- 4.Incrementing/Decrementing Operators
- 5.Logical Operators
- 6.String Operators
- 7.Array Operators
- 8.Conditional / Ternary Operator

## **PHP Arithmetic Operators**

Arithmetic Operators simple calculation में use होने wale Operators होते हैं जैसे Addition , Subtraction etc . Now assume हमने \$x और \$y दो variables define किये हैं तो arithmetic operator कुछ इस तरह से operation perform करेंगे।

Operator	Name	Example	Explanation
+	Addition	$\$x + \$y$	( + ) plus operator uses to add two or more numbers / values
-	Subtraction	$\$x - \$y$	( - ) minus operator uses to subtract one numbers / values from another numbers / values or finds the difference
/	Division	$\$x / \$y$	quotient of $\$x$ and $\$y$
*	multiplication	$\$x * \$y$	product of $\$x$ and $\$y$
%	Modulus	$\$x \% \$y$	remainder of operands
**	Exponentiation	$\$x ** \$y$	$\$x$ raised to the power $\$y$

Note - Exponential ( \*\* ) operator PHP Version 5.6 में introduce किया गया था।

## PHP Assignment Operators

Assignment Operator को ( = ) से represent करते हैं , जो कि value को किसी variable में assign करने के लिए use किया जाता है। हालाँकि इसे Arithmetic Operators के साथ भी use करते हैं। नीचे दिए गए Example में आप देख सकते हैं।

Operator	Name	Example	Explanation
=	Assign	$\$x = \$y$	value of $\$y$ assigned to $\$x$
+=	Addition then Assign	$\$x += \$y$	First add and assign it to $\$x$ . (It is same as $\$x = \$x + \$y$ )
- =	Subtract then Assign	$\$x -= \$y$	get the difference and assign it to $\$x$ . (It is same as $\$x = \$x - \$y$ )
/ =	Divide then assign quotient	$\$x /= \$y$	quotient of $\$x$ and $\$y$ then assign it to $\$x$ . (It is same as $\$x = \$x / \$y$ )
* =	Multiplication then assign the product	$\$x * = \$y$	product of $\$x$ and $\$y$ then assign it to $\$x$ . (It is same as $\$x = \$x * \$y$ )
% =	Divide then assign remainder	$\$x \% = \$y$	remainder of operands then assign it to $\$x$ . (It is same as $\$x = \$x \% \$y$ )

## PHP Comparison Operators

Comparison Operators दी हुई दो values को compare करके **Boolean (1 for true and for false It returns nothing)** value return करते हैं according to condition . PHP में Comparison Operators को कुछ इस तरह से use कर सकते हैं।

Operator	Name	Example	Explanation
<b>==</b>	<b>Equal</b>	<b>\$x == \$y</b>	<b>checks if \$x is equal to \$y.</b>
<b>===</b>	<b>Identical</b>	<b>\$x === \$y</b>	<b>checks if \$x is equal to \$y with their data types.</b>
<b>!=</b>	<b>Not equal</b>	<b>\$x != \$y</b>	<b>checks if \$x is not equal to \$y.</b>
<b>&lt;&gt;</b>	<b>Not equal</b>	<b>\$x &lt;&gt; \$y</b>	<b>checks if \$x is not equal to \$y.</b>
<b>!==</b>	<b>Not identical</b>	<b>\$x !== \$y</b>	<b>checks if \$x is not equal to \$y with data types.</b>
<b>&lt;</b>	<b>Less than</b>	<b>\$x &lt; \$y</b>	<b>checks if \$x is less than \$y.</b>
<b>&gt;</b>	<b>Greater than</b>	<b>\$x &gt; \$y</b>	<b>checks if \$x is greater than \$y.</b>
<b>&lt;=</b>	<b>Less than or equal</b>	<b>\$x &lt;= \$y</b>	<b>checks either \$x is less than or equal to \$y.</b>
<b>&gt;=</b>	<b>Greater than or equal</b>	<b>\$x &gt;= \$y</b>	<b>checks either \$x is greater than or equal to \$y.</b>

## PHP Incrementing/Decrementing Operators

**Incrementing/Decrementing Operators** को किसी variable को 1 से increase या decrease करने के लिए use किया जाता है। हालाँकि इसमें Addition और Subtraction operation ही होते हैं , इसलिए इसमें ( ++ ) या ( -- ) sign ही use होते हैं। नीचे table में आप देख सकते हैं कि सा तरह से इन्हें उसे करते हैं और क्या Output generate हैं।

Operator	Name	Explanation
++\$a	Pre Increment	first increment by 1 then return the value of \$a
\$a++	post Increment	first return the value of \$a then increment by 1.
--\$a	Pre Decrement	first decrement by 1 then return the value of \$a
\$a--	Post Decrement	first return the value of \$a then decrement by 1.

## PHP Logical Operators

Logical Operators , एक या एक से अधिक expression के according Boolean value return करते हैं। जैसे –

Operator	Name	Example	Explanation
and	And	\$x and \$y	Returns True if Both operands (\$x and \$y) are true;
or	Or	\$x or \$y	Returns True if either \$x or \$y is true;
xor	Xor	\$x xor \$y	Returns True if either \$x or \$y is true but not both.
&&	And	\$x && \$y	same as and, returns True if Both operands (\$x and \$y) are true;
!	Not	!\$x	Returns True if \$x is not true;
	Or	\$x    \$y	same as or , returns True if either \$x or \$y is true;

## PHP String Operators

PHP में दो String Operators introduce किये ,

- 1 . concatenation operator ( . ) string concatenation के लिए और
- 2 . concatenating assignment - जो string को concat करने के बाद assign भी कर देता है।

Operator	Name	Example	Explanation
.	Concatenation	\$x . \$y	Concatenate \$x and \$y.
or	Assignment Concatenation	\$x .= \$y	Concatenate \$x and \$y and then assign it to \$x.



## PHP Array Operators

PHP हमें array operators भी provide कराती है जिसकी help से हम directly array पर action perform कर सकते हैं। माना हमारे पास दो array variable `$x = array('key1'=>'value1', 'key2'=>'value2')` और `$y = array('key1'=>'value1', 'key2'=>'value2')` हैं , तो array operators को कुछ इस तरह से use करेंगे।

Operator	Name	Explanation
<code>\$x+\$y</code>	Union	union of \$x and \$y .
<code>\$x == \$y</code>	Equality	यह true return करता है अगर दोनों array में same key value हैं , हालाँकि ( == ) में value का type match नहीं होता।
<code>\$x === \$y</code>	Identity	यह true return करता है अगर दोनों array में same key value हैं , और values का type भी same है।

Example :

```
<?php
```

```
    /*define array*/
```

```
    $x = array('key1'=>'value1', 'key2'=>'value2', 'key3'=> 'value3');
```

```
    $y = array('key1'=>'value1', 'key2'=>'value2');
```

```
    echo "Union of x and y :";
```

```
    echo "<pre>";
```

```
    print_r($x+$y);
```

```
    echo "</pre>";
```

```
    /*again define array with same key value*/
```

```
    $a = array('key1'=>'value1', 'key2'=>'value2');
```

```
    $b = array('key1'=>'value1', 'key2'=>'value2');
```

```
    echo "Equality of x and y :". ($a==$b). "<br>";
```

```
    echo "Identity of x and y :". ($b=== $b);
```

```
?>
```

## Output

Union of x and y :

Array ( [key1] => value1 [key2] => value2 [key3] => value3 )

Equality of x and y :1

Identity of x and y :1

तो आपने देखा कि किस तरह से array operators को use करते हैं।

## PHP Ternary Operator

**Conditional / Ternary Operator** में हम ( ? : ) use करते हैं , सबसे पहले दिए गए expression / condition को evaluate करते हैं अगर expression **true** है तो question mark ( ? ) के बाद का statement run होगा और अगर expression / condition **false** है तो **colon ( : )** के बाद वाला statement run होगा। ।

जैसे –

```
<?php
```

```
$age = 17;
```

```
$res = ($age >= 18) ? "You are able to vote" : "You are not able to vote";
```

```
echo $res;
```

```
?>
```

## Output

You are not able to vote

## PHP If Else

If Else statement किसी भी programming language का सबसे important feature है , PHP If Else conditions के according किसी Code Of Block को run करने के लिए use होता है। Means जब हमें Condition True होने पर कोई दूसरा code run करना हो Condition गलत होने पर कुछ और तब हम If Else का use करते हैं।

### PHP IF Else Syntax

```
if(condition)
{
// write your logic for true condition
} else
{
//write logic if condition false
}
```

Example: ifelse

```
<?php
    $age = 19;
    if($age > 18)
    {
        echo 'You are eligible to vote';
    }
    else
    {
        echo 'You are not eligible to vote';
    }
?>
```

Output

You are able to vote

PHP में if या else के साथ use होने वाले curly brackets {} की जगह आप colon : भी use कर सकते हैं।

Example –

```
<?php
    $age = 19;
    if($age > 18) :
        echo 'You are eligible to vote';
    else:
        echo 'You are not eligible to vote';
    endif
?>
```

Output:-

You are eligible to bote

### If Else With HTML

जब हम किसी project / website develop करते हैं , तो हमें PHP को दूसरी languages (HTML, CSS , JavaScript , j Query etc .) के साथ एक्सेक्यूटे करना पड़ता है , इस कंडीशन में कुछ इस तरह से हम If Else use करते हैं।

Example -

```
<?php
    if($condition) :
?>
        Your HTML Content
<?php
    else:
?>
        Another HTML Content
<?php
    endif
?>
```

### Output

Another HTML Content

तो इस तरह से हम If Else Statement के साथ HTML को use करते हैं।  
Example में मैंने एक undefined variable \$condition check किया है  
हालाँकि यह undefined है इसलिए condition False है।

## PHP Elseif / Else

**elseif / else if** जैसा की नाम से पता चलता है कि अगर upper condition condition **false** होती है और elseif में दी हुयी condition **true** return करती है तो elseif code of block run होगा। हालाँकि if के बाद यह जरूरी नहीं की एक ही elseif use हो Condition के according एक से ज्यादा भी use हो सकते हैं।

## PHP elseif / else if Syntax

```
if(condition)
{
// write your logic for true condition
} elseif(another condition)
{
//write logic if first condition gets false.
} else
{
//if both conditions get false
}
```

Example –

```
<?php
$marks = 70;
if($marks >= 80)
{
    echo "Amazing Grade : A";
}
elseif($marks >= 70)
{
    echo "Grade B";
}
elseif($marks >= 50)
{
    echo "Grade C";
}
```

```
elseif ($marks >= 33)
{
    echo "Echo D";
}
else
{
    echo "Failed !";
}
?>
```

**Output**

Grade B

दिए गए Example में देख सकते हैं कि किस तरह से elseif Use कर सकते हैं। हालाँकि elseif को एक साथ लिखने की जगह else if दो शब्दों में अलग अलग भी लिख सकते हैं।

**Note -** Else If statement में आप **curly brackets {}** के साथ ही else if अलग अलग लिख सकते हैं , अगर आप else if **colon :** के साथ use करते हैं तो Error आएगी।



Example -

```
<?php
    $a = 14;
    $b = 11;

    /*Correct way to use elseif with colon(:) */
    if($a > $b):
        echo "a is greater than b";
    elseif($a==$b):
        echo "both are equal";
    endif;

    /*Incorrect way*/
    if($a>$b):
        echo "a is greater than b";
    else if($a==$b):
        echo "It'll generate error";
    endif;
?>
```

## Output

**Parse error:** syntax error, unexpected 'if' (T\_IF), expecting ':' in **C:\xampp\htdocs\test\ifelse3.php** on line **15**

clearly देख सकते हैं कि line 15 पर else if दो शब्दों में लिखा होने की वजह से fatal error generate हुए हैं। PHP में Else If statement use करते हैं यह बात ध्यान में रखें कि **colon (:)** के साथ कभी भी else if अलग अलग न हो।

## PHP while Loop

Actually किसी code of block को repeatedly run करने का सबसे easy way looping है , **PHP** में different - different Looping Statements हैं -

- while Statement
- [do while Statement](#)
- [for Statement](#)
- [foreach Statement](#)
- [switch Statement](#)

while Loop:-

PHP में While Loop ठीक वैसे ही काम करते हैं जिस तरह से **C** या **JAVA** में करते हैं। While Loop Simply Nested Statements को Execute करता है , जब तक कि दी हुई Condition **False** न हो। जब तक Condition **True** है तब तक Loop Execute होगा। While Loop को **Entry Control Loop** भी कहते हैं क्योंकि loop को Execute करने से पहले दी हुई condition Check होती है , condition True होने पर ही Loop में Entry होती है।

## PHP while Loop Syntax

```
while(condition / expression)
{
    //write your logic
}
```

## PHP While Loop Example

```
<?php
/*Example 1 to print print 1 to 10*/
$x = 1;
while ($x <= 10)
{
    echo $x.", ";
    /*increment by 1*/
    ++$x;
}
```

```
/*break the line*/
echo "<br>";
/*Example 2 to print print 1 to 10*/
$x = 1;
while ($x <= 10) :
    echo $x.", ";
    /*increment by 1*/
    ++$x;
endwhile;
?>
```

### Output

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

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

PHP में हम colon (:) का use भी कर सकते हैं इसलिए ऊपर दिए गए example में दो बार 1 से 10 तक print हुआ।

## PHP Nested While Loop

PHP में Nested While Loop का भी use कर सकते हैं , means While Loop के अंदर एक और While Loop.

Example –

```
<?php
$x = 1;
while($x <= 10)
{
    $y = 1;
    echo $y." ";
    while($y < $x)
    {
        $y++;
        echo $y." ";
    }

    ++$x;
    echo"<br>";
}
?>
```

### Output

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

## PHP do while Loop

Do While Loop , जैसा कि नाम से समझ आ रहा है कि पहले कोई code of block run हो रहा है। PHP में While Loop और Do While Loop में यही main difference भी है।

while loop में सबसे पहले condition ही check होती है उसके बाद ही code of block run होता है , अगर condition **false** है तो loop में entry ही नहीं होगी , उसके उलट do while loop में सबसे पहले code of block run होगा और सबसे end में condition check होती है , इससे कोई फर्क नहीं पड़ता कि condition सही है या गलत , loop को एक बार run होना ही है।

## PHP do while Loop Syntax

```
do {  
    //code of block  
}  
while(condition / expression);
```

## PHP do while Loop Example

```
<?php
$x = false;
do
{
    echo"Run This Code Of Block First <br>";
    var_dump($x);
}
while($x === true);
?>
```

### Output

Run This Code Of Block First  
bool(false)

सबसे पहले loop के अंदर code run हुआ उसके बाद condition Check हुई। मैंने variable \$x में boolean value **false** initialize की , और end में Condition Check की।

while loop के same example को do while loop के though किया है , आप output में देख सकते हैं कि output **same** ही आया है , बस program का structure change हो गया है। well , **do while loop** हम वहां पर use करते हैं , जब हमें किसी loop के अंदर का code of block कम से कम एक बार तो code of block run करना ही हो।

Note - हम जानते हैं , कि PHP में Alternative Syntax use कर सकते हैं means Control Statement में **Curly braces {}** की जगह colon : use कर सकते हैं , लेकिन Do While Loop में Alternative Syntax use नहीं कर सकते हैं , अगर आप ऐसा करते हैं तो PHP Fatal Error Generate करती है।

Example –

```
<?php
/* It will work */
$a = 0
while ($a < 10) :
    echo $a;
    $a++;
endwhile;
```

```
/* While it will not work and PHP generates fatal Error*/
$a = 0;
do :
    $a++;
    echo $a;
while ($a <= 10);
?>
```

## Output

PHP Parse error: syntax error, unexpected ':' in **C:\xampp\htdocs\test\do\_while2.php** on line **11**

While Loop Alternative Syntax के साथ कोई Error generate नहीं करता , वहीं Do While Loop Alternative Syntax के साथ Error Generate करता है। इसलिए Do While Loop के साथ कभी भी Alternative Syntax use न करें हमेशा Standard Syntax ही use करें।

## PHP for Loop

PHP में For Loop हम तब use करते हैं जब हमें पता हो कि Loop को कितने बार रन करना है , और यही main Difference है While Loop और For Loop में।

While Loop हम तभी use करते थे , जब हमें यह नहीं पता होता था कि Loop कितनी बार रन होना है। PHP में For Loop लगभग C / Java language की तरह ही काम करते हैं। जहां हम सभी condition / expression एक ही जगह रखते हैं।

## PHP for Loop Example

print table of 2.

```
<?php
$number = 2;
for($x=1; $x<=10; $x++)
{
    echo $number.' x '.$x.' = '. $x*$number."<br>";
}
?>
```

## Output

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```



## Iterating Letters In PHP For Loop

ऊपर दिया गया example में तो सिर्फ numbers iterate किये गए हैं , हालांकि For Loop के through हम letters भी iterate कर सकते हैं।

Example :

```
<?php
for($let = 'A'; $let != 'AA'; $let++)
{
    echo $let." , ";
}
?>
```

### Output

A , B , C , D , E , F , G , H , I , J , K , L , M , N , O , P , Q , R , S , T , U , V , W , X , Y , Z ,

For Loop के through हम letters भी iterate कर सकते हैं। Example में मैंने second expression में 'AA' check किया है , क्योंकि A से Z के बाद For Loop में AA start होता है , और यह इसी तरह से चलता रहता है AA , AB , AC .... etc.

**Note** - Letters iterate करते यह ध्यान में रखे की Condition में कभी भी Greater Than (>) या Less Than (<) का use न करें, हमेशा Equals To (==) या Non Equals To (!=) का use करें। क्योंकि Greater Than (>) या Less Than (<) का use हम numeric calculations में करते हैं।

## PHP Nested For Loop

For Loop के अंदर एक और For Loop use करना ही Nested For Loop कहते हैं। PHP में Nested For Loop कुछ इस तरह से use करते हैं।

Example :

```
<?php
for($x=1;$x<=5; $x++)
{
    for($y=1; $y <= $x; $y++)
    {
        echo $y." ";
    }

    /*for line break*/
    echo"<br>";
}
?>
```

### Output

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

## PHP foreach Loop

PHP में foreach Loop के through किसी Array या Object को iterate करने का सबसे अच्छा तरीका है। Foreach Loop सिर्फ Array या Object को Iterate करता है , अगर आप किसी simple variable के साथ उसे करेंगे तो Error आएगी। PHP में foreach Loop को दो तरह से use कर सकते हैं। Syntax कुछ इस तरह है -

### PHP foreach Loop Syntax

1. `foreach($array_expr as $value)`
2. `{`
3. `//code of block`
4. `}`
5. `2. foreach($array_expr as $key => $value)`
6. `{`
7. `//code of block`
8. `}`

First Syntax में किसी Array या object को iterate करते समय value , \$value variable में automatically assign हो जाती है, और pointer 1 से increment हो जाता है।

जबकि Second Type के foreach loop में एक additional parameter \$key लिया है , जो current array store करता है।

## PHP foreach Loop Example

```
<?php
$arr = array(10, 20 , 30, 40);
foreach($arr as $value)
{
    echo $value."<br>";
}
?>
```

### Output

10  
20  
30  
40

Note - Example में '  
' का use line break के लिए किया गया है और dot (.) का use [String Concatenation](#) के लिए।

### Another Example :

```
<?php
$arr = array(10, 20 , 30, 40);
foreach($arr as $key=> $value)
{
    echo "Key : ".$key." Value : ".$value."<br>";
}
?>
```

### Output

```
Key : 0 Value : 10
Key : 1 Value : 20
Key : 2 Value : 30
Key : 3 Value : 40
```

### PHP Nested foreach Loop

For Loop की तरह ही हम Foreach Loop में भी हम Loop के अंदर Loop use कर सकते हैं।

## PHP Nested foreach Loop

Example :

```
<?php
/*defining double dimentional array*/
$arr = array(
    array(10, 20, 30, 40),
    array(50, 60, 70, 80),
);
foreach($arr as $inner_arr)
{
    foreach($inner_arr as $value)
    {
        echo $value." , ";
    }
}
?>
```

### Output

10 , 20 , 30 , 40 , 50 , 60 , 70 , 80 ,

## PHP continue

PHP में **continue** का use किसी दी गयी condition के according current loop iteration को skip करने के लिए किया जाता है। और skip करने के बाद Loop next iteration से Start हो जाता है। simply हम कह सकते हैं कि PHP में **continue** का use हम वहाँ करते हैं जब हमें किसी condition पर loop execution को skip करना हो।

PHP में continue statement JavaScript से थोड़ा अलग होता है क्योंकि [JavaScript में continue statement](#) में optional parameter की value हम किसी loop (for या while loop ) को identify करने के लिए define किये गए label का name provide कराते हैं। जिसे [Labeled Statement](#) भी कहते हैं।

### PHP continue Example

```
<?php
for($x=1; $x <= 5 ; $x++)
{
    if($x==3)
    {
        continue;
    }
    echo $x."<br>";
}
?>
```

### Output

1  
2  
4  
5

Example में आप देख सकते हैं कि \$x की value 3 होते ही वो level skip हो गयी।

Break की तरह ही continue भी एक Optional parameter accept करता है जो कि बताता है की कितने loop iterate levels skip करने हैं , default parameter 1 set होता है।

## PHP continue Example With Optional Parameter

```
<?php
for ($i = 1; $i <= 5; ++$i)
{
    for ($y = 1; $y <= 5; ++$y)
    {
        if ($y == 2)
            continue 2;

        print "$i $y <br>";
    }
}
```

?>

### Output

```
1 1
2 1
3 1
4 1
5 1
```

तो Example में आप देख सकते कि Outer Loop तो 5 बार execute हुआ है , but inner loop हर बार एक ही बार execute हुआ है क्योंकि inner loop में variable की value 2 होते ही , execution दोनों ही loop से skip हो जाता है।



PHP में **continue** use करते समय कुछ बातें ध्यान में रखें।

1.Optional parameter positive integer ही होना चाहिए , String , null , 0 , floating point , negative integer नहीं होना चाहिए , नहीं तो Fatal Error Generate होगी।

2.Optional Parameter की value हम use किये जाने वाले Loops से ज्यादा नहीं होनी चाहिए , means single loop में Optional Parameter की value 1 ही होगी और 1 level nested Loop में Optional Parameter की value 2 से ज्यादा नहीं होगी।

3.Continue को सभी तरह के Loops जैसे [For Loop](#) , [Foreach Loop](#) , [While Loop](#) , [Do While Loop](#) , [switch loop](#) में use कर सकते हैं , But [If Else](#) में use नहीं कर सकते हैं।

Examples में आप देख सकते हैं कि PHP में Continue Syntax कौन सा valid है और कौन सा invalid .

```
<?php
```

```
/*Invalid It Generates : PHP Fatal error: 'continue' not in the 'loop' or 'switch'*/
```

```
if ($y == 2)
```

```
    continue 2;
```

```
/*Invalid It Generates : PHP Fatal error: Cannot 'continue' 2 levels */
```

```
for ($y = 1; $y <= 5; ++$y)
```

```
{
```

```
    if ($y == 2)
```

```
        continue 2;
```

```
    print "$y";
```

```
}
```

```
/*Invalid It Generates : PHP Fatal error: 'continue' operator accepts only positive numbers*/
```

```
for ($y = 1; $y <= 5; ++$y)
```

```
{
```

```
    if ($y == 2)
```

```
        continue 's';
```

```
print "$y";  
}
```

/\*Invalid It Generates : PHP Fatal error: 'continue' operator accepts only positive numbers\*/

```
for ($y = 1; $y <= 5; ++$y)  
{  
    if ($y == 2)  
        continue 4.5;
```

```
    print "$y";  
}
```

/\*Invalid It Generates : PHP Fatal error: 'continue' operator accepts only positive numbers\*/

```
for ($y = 1; $y <= 5; ++$y)  
{  
    if ($y == 2)  
        continue 0;
```

तो ये कुछ wrong syntax थे जिन्हे कुछ developers miss कर देते हैं इसलिए Continue को use करते समय ध्यान रखें।

```
    print "$y";  
}
```

?>

## PHP goto Statement

PHP में goto keyword का use हम Program में किसी दूसरे section पर move करने के लिए करते हैं। Targeted Section को हम label के through define करते हैं और उसी Specified label को goto keyword के साथ target करते हैं।

## PHP goto Example

```
<?php
goto my_level;
my_level:
{
    echo "This is simple goto test";
}
?>
```

## Output

This is simple goto test

Note - Targeted label same file और same context में होना चाहिए , ऐसा नहीं हो सकता कि आप किसी function के अंदर define किये गए label को function के बाहर Access करें। और न ही [For Loop](#) या [While Loop](#) को jump कर सकते हैं

```
<?php
goto end;
for($i=0; $i<10; $i++)
{
    end:
    echo 'Not working';
}
?>
```

### Output

PHP Fatal error: 'goto' into loop or switch statement is disallowed in **C:\xampp\htdocs\test\goto2.php** on line 6

Example में देख सकते हैं कि जब हम [For Loop](#) के अंदर define किये गए label को access करते हैं तो PHP Fatal Error Generate करती है।

हम [For Loop](#) या [While Loop](#) के अंदर से बाहर define किये गए labels को आसानी से Access कर सकते हैं।

For Example :

```
<?php
for($i=0; $i<10; $i++)
{
    if($i==3)
        goto end;
}

end:
echo 'Awesome it works';
?>
```

### Output

Awesome it works

One more thing , जब हम [For Loop](#) या [While Loop](#) के अंदर से बाहर define किये गए labels को Access करते हैं तो जैसे ही goto statement execute होता है तो loop भी break हो जाता है , means हम पूरी तरह से Loop को exit कर देते हैं।

For Example ऊपर दिए गए example में ही अगर हम \$i की value print करें तो ये values तभी तक print होंगी जब तक कि goto statement Execute नहीं हो जाता।

```
<?php
for($i=0; $i<10; $i++)
{
    if($i==3)
        goto end;

    /*printing values*/
    echo $i."<br>";
}

end:
echo 'Awesome it works';
?>
```

## Output

0

1

2

Awesome it works

## PHP break Statement

PHP में **break** का use current [For Loop](#) , [Foreach Loop](#) , [While Loop](#) , [Do While Loop](#) और [switch loop](#) execution को terminate करता है , means break keyword का use करके हम loop को terminate कर सकते हैं।

**break** एक optional numeric integer parameter accept करता है जो बताता है कि कितने nested loops structure को break out करना है। Parameter की Default value 1 set होती है।

## PHP break Example

```
<?php
for($x=1; $x <= 5; $x++)
{
    if ($x == 3)
    {
        break; /* You could also write 'break 1;' here. */
    }

    echo "$x <br>";
}
?>
```

## Output

1  
2



अब अगर हम Optional parameter की value nested loop के according बढ़ाते हैं , तो हम सभी loops को break out देंगे।

### PHP break Example With Optional Parameter

```
<?php
$x=1;
while($x <= 6)
{
    echo "Outer Loop : $x <br>";
    $y=1;
    while($y <= $x)
    {
        if($x==4)
            break 2;
        echo " inner loop : $y <br>";
        $y++;
    }
    echo "<br>";
    $x++;
}
?>
```

#### Output

Outer Loop : 1  
inner loop : 1

Outer Loop : 2  
inner loop : 1  
inner loop : 2

Outer Loop : 3  
inner loop : 1  
inner loop : 2  
inner loop : 3

Outer Loop : 4

## PHP Switch Statements

PHP में Switch Loop , किसी matched expression के लिए code of block Run करता है , यह Else If की तरह ही work करता है जहा हम कई सारी Conditions में से True Condition वाला statement ही Run होता था, और अगर एक भी condition match नहीं होती तो else part (default) run होता था। वहीं Switch में हम cases use करते हैं , और जो case match करता है वही statement execute करता है। और कोई case match न होने पर default statement execute होता है।

### PHP Switch Loop Example

```
<?php
$x = "apple";
switch ($x)
{
    case "apple":
        echo "x is apple";
        break;
    case 3:
        echo "x is bar";
        break;
    case 9:
        echo "x is cake";
        break;
    default:
        echo "default case running";
} ?>
```

### Output

x is apple

Note - जैसा कि आप example में देख रहे होंगे कि हर case statement के end में [break keyword](#) का use किया गया है , जिससे कि matched case statement ही रन हो अगर हम ये [break](#) remove तो जिस case के साथ value match करती है वहाँ से सभी statements (cases) switch Loop खत्म होने तक execute होंगे।

Example :

```
<?php
    $x = 3;
    switch ($x)
    {
        case "apple":
            echo "x is apple";
        case 3:
            echo "x is bar <br>";
        case 9:
            echo "x is cake <br>";
        default:
            echo "default case running";
    }
?>
```

### Output

x is bar

x is cake

default case running

[break keyword](#) remove करने पर जो case match होता है वहाँ से सभी statements (default statement) execute होते हैं। अगर हम case के बाद use किये जाने वाले **colon ( : )** की जगह **semicolon ( ; )** use करते हैं तो भी कोई problem नहीं है।

A Very Important Note - हो सके तो switch loop में mixed type value जैसे 1str , str324 use करने से बचें , इससे problem हो सकती है।

For Example :

```
<?php
$string="1string";
switch($string)
{
    case 1:
        echo "this is 1";
        break;
    case 2:
        echo "this is 2";
        break;
    case '1string':
        echo "this is a string";
        break;
    default:
        echo "Default case";
}
?>
```

**Output**

this is 1

variable में 1string assign किया गया है ,  
और यह case 1 , से ही match कर ले रहा है  
वजाय case '1string' के , इसलिए ये Switch  
Loop में mixed type value use करने से बचें।

! Important

ये Examples **PHP Version 7.4** पर run किये गए थे , हालाँकि **PHP Version 8** में matching rules को **strict** किया है , अब आप **mixed type** भी देंगे तब भी कोई problem नहीं है because **PHP Version 8** में pass की गयी value के **type** के according ही **case match** होता है।

इसके अलावा PHP Version 8 में एक और Match expression add किया गया है जो switch Loop की तरह ही work करता है , but switch loop से काफी sort & easy to understand है।

### PHP Match expression

```
<?php
    /*try it on PHP 8*/
    echo match(true)
    {
        0 => 'this is zero',
        false => 'False',
        true => 'True',
        '1' => 'This is string 1',
        1 => 'This is int 1',
    };
?>
```

Output:-

True

## PHP include and include\_once

PHP में **include** and **include\_once** का use हम files को include करने के लिए करते हैं। क्योंकि जब हम किसी Project पर काम करते हैं तो कई जगह पर हमें same data चाहिए होता है , उसके लिए हम हर एक File में same data न लिखकर किसी दूसरी file में लिखकर include कर लेते हैं।

जैसे - ज्यादातर Websites में Header और Footer same ही रहते हैं , या कई ऐसे functionality होती हैं , जिन्हे हमें बार बार लिखने की जरूरत होती है, तो इस तरह के content को use करने का सबसे अच्छा तरीका यही है कि इन्हे आप दूसर file में लिखकर include करें।

### PHP Include Example

```
<?php
    $x = 'x defined in file1';
?>
```

```
<?php
    include('file1.php');
    echo $x;
?>
```

### Output

x defined in file1

Note - [Print और Echo](#) की तरह ही अगर हम file को बिना parenthesis ( ) के बिना भी Path देते हैं तब भी File include होगी। जैसे - **include'filepath';** and **include\_once'filepath';**

## Why we use include or include\_once ?

### 1.Code Re-Usability

**2.include** या **include\_once** सबसे बड़ा Code Re-usability ही है , जैसा मैं पहले भी बता चुका हूँ कि same content हमें बार-बार लिखने की जरूरत नहीं इसके लिए PHP ने files को include कराने के लिए features दिए हैं।

### 3.Easy To Update

4. चूंकि अब कई जगह पर use होने वाला Same Code एक जगह पर है , तो In Future अगर हमें उस Code को update करने की जरूरत पड़ती है तो simply हम एक जगह से ही Code को Update कर सकते हैं , और जहां - जहां वह file include होगी automatically सब जगह change आ जायेगा।

## Difference Between include And include\_once

**include** and **include\_once** में सबसे बड़ा difference यह है की जब हम include का use करते हैं तो ये हर बार file को include करता है चाहे same file पहले से ही क्यों न include कर ली गयी हो , जिस वजह से PHP fatal error generate करती है same file दो या दो से ज्यादा बार include करने पर।

See Example

```
<?php
function test()
{
    echo "function inside a.php";
}
?>
```

```
<?php
include('a.php');
/*again include it so that we can differentiate */
include('a.php');
/*call the function*/
test();
?>
```

## Output

**Fatal error:** Cannot redeclare test() (previously declared in C:\xampp\htdocs\test\a.php:4)  
in **C:\xampp\htdocs\test\a.php** on line 5

Example में देख सकते हैं कि किस तरह से same file बार - बार **include** करने पर fatal error generate हुई।

## Important

ऐसा इसलिए होता है क्योंकि PHP में variables कितनी ही बार **re-declare** कर सकते हैं (**constant** variables को छोड़कर ) लेकिन functions नहीं , और same file को बार - बार include करने पर functions की वजह से ही PHP Fatal Error Generate करती है। इसी problem से बचने के लिए include\_once() का use करते हैं।

**include\_once** का use करके files **include** करने पर same file include होने पर भी कोई Error Generate नहीं होगी। क्योंकि **include\_once** file को तभी include करता है अगर same file पहले include नहीं गयी है।

See Example :

```
<?php
include_once('a.php');
/*again include */
include_once('a.php');
/*call the function*/
test();
?>
```

## Output

function inside a.php



## Best Practice With include and include\_once

अगर include or include\_once का use करके हम किसी file को function के अंदर include करते हैं तो , उस file में define किये गए code (variables etc. ) का [scope](#) उस function में ही होगा , उस file में define किये गए variables को हम function के बाहर access नहीं कर पाएंगे।

और अगर आप उस file में define किये गए variables को access करना चाहते हैं तो define किये गए variables को [global](#) बनाना पड़ेगा।

See Example :

File : [a.php](#)

```
<?php
    $x = 10;
?>
```

## Output

value inside function of x : 10

**Notice:** Undefined variable: x in **C:\xampp\htdocs\test\b.php** on line **10**

value out of the function :

File : [b.php](#)

```
<?php
    function test()
    {
        include_once('a.php');
        echo "value inside function of x : ".$x;
    }

    /*call the function and print both variables*/
    test();
    echo "value out of the function : $x
";
?>
```

देख सकते हैं कि , test() function में a.php file include की गयी है जिसमे \$x variable define किया गया है , और जब इस variable को function के अंदर access करते हैं तो हमें इसकी value मिलती है but जब इसे out of the function access करते हैं तो PHP Warning Generate करती है। अब , अगर आप function के अंदर include की गयी file में define किये गए variables को उस function के बाहर access करना चाहते हैं तो उन variables को file include से पहले [global](#) बनाना पड़ेगा।

See Example :

File : [b.php](#)

```
<?php
function test()
{
    global $x;
    include_once('a.php');
    echo "value inside function of x : $x <br>";
}
/*call the function and print variable*/
test();
echo "value out of the function : $x";
?>
```

**Output**

value inside function of x : 10  
value out of the function : 10

Note - है, हमेशा variables को file include करने से पहले ही [global declare](#) करें file include के बाद नहीं , otherwise global [declare](#) किये गए variables के लिए PHP Warning तो नहीं Generate करेगी But हमें उस variable की value नहीं मिल सकेगी ।

## PHP require and require\_once

require and require\_once का use file को include करने के लिए किया जाता है। हालाँकि file को include करने के लिए PHP में और भी functions जैसे include और include\_once भी होते हैं।

require , require\_once , [include, include\\_once](#) इन सभी का main purpose file को include करना ही होता है , but इनमें कुछ difference होता जो इन्हें एक दूसरे से अलग बनाता है।

### PHP Difference Between include And require

बैसे तो require , include दोनों ही file को include करते हैं , But इन दोनों में सबसे बड़ा difference यह है कि अगर दी गयी location पर File नहीं

है तो include सिर्फ Warning Generate करता है और script execution continue कर देता है , जबकि require Fatal Error Generate करता है और execution को terminate कर देता है।

Example :

```
<?php
/*include a file that doesn't exist */
include("file.php");
echo "Execution is continue";

require("file.php");
echo "It will not print";
?>
```

## Output

**Warning:** include(file.php): failed to open stream: No such file or directory in **C:\xampp\htdocs\test\test.php** on line **3**

**Warning:** include(): Failed opening 'file.php' for inclusion (include\_path='C:\xampp\php\PEAR') in **C:\xampp\htdocs\test\test.php** on line **3**  
Execution is continue

**Warning:** require(file.php): failed to open stream: No such file or directory in **C:\xampp\htdocs\test\test.php** on line **6**

**Fatal error:** require(): Failed opening required 'file.php' (include\_path='C:\xampp\php\PEAR') in **C:\xampp\htdocs\test\test.php** on line **6**

## PHP Difference Between require And require\_once

require and require\_once में सबसे बड़ा difference यह है की जब हम require का use करते हैं तो ये हर बार file को include करता है चाहे same file पहले से ही क्यों न include कर ली गयी हो , जिस वजह से PHP fatal error generate करती है same file दो या दो से ज्यादा बार include करने पर।

Example :

File : test2.php

```
<?php
function test()
{
    echo "function inside test2.php";
}
?>
```

File : test.php

```
<?php
require('test2.php');
/*again include it so that we can differentiate */
require('test2.php');
/*call the function*/
test();
?>
```

## Output

**Fatal error:** Cannot redeclare test() (previously declared in C:\xampp\htdocs\test\test2.php:4)  
in **C:\xampp\htdocs\test\test2.php** on line 5

Example में आप देख सकते हैं कि किस तरह से same file बार - बार include करने पर fatal error generate हुई।

वहीं अगर हम file को include करने के लिए require\_once का use करते हैं तो same file include होने पर भी कोई Error Generate नहीं होगी। क्योंकि require\_once file को तभी include करता है अगर same file पहले include नहीं गयी है।

Example :

File : test.php

```
<?php
require_once('test2.php');
/*again include it so that we can differentiate */
require_once('test2.php');
/*call the function*/
test();
?>
```

### Output

function inside test2.php

### PHP Array

PHP में Array एक ordered map है। और map एक type है जो value को key के साथ associate करता है means key value pair में data store करता है।

PHP में **Array** दो तरह से define कर सकते हैं।

1.By using array() construct

2.By using Brackets

और दोनों में ही हम comma ( , ) separated values pass करते हैं।

array define करने के लिए हम दोनों में से कोई भी method use कर सकते हैं।

## PHP Array Example

```
<?php
$arr1 = array('Rahul', 'Raju', 'Shyam');
echo "<pre>";
print_r($arr1);
?>
```

### Output

```
localhost/test/array.php
```

```
Array
(
    [0] => Rahul
    [1] => Raju
    [2] => Shyam
)
```

जहाँ **print\_r()** एक predefined PHP function है जिसका use array variables में stored data को display करने के लिए किया जाता है। PHP Array display करने का यह सबसे अच्छा method है। example में देख सकते हैं कि जो values हमने जिस order में store की थी उसी order में values automatically एक key के साथ associate हो गई है।

Note - PHP में array define करने के लिए हमें array का type define करने की जरूरत नहीं होती है। जैसे कि JAVA या C में हमें array define करते समय array का type define करने की जरूरत पड़ती है और वह array define किये गए type की value को accept करता है।

यही reason है कि PHP में array किसी भी तरह की value (string , integer , float , boolean etc. ) को store कर सकते हैं।

Example :

```
<?php
$arr1 = array('Rahul', 10, 56.56, true);
echo "<pre>";
print_r($arr1);
?>
```

### Output

```
Array
(
    [0] => Rahul
    [1] => 10
    [2] => 56.56
    [3] => 1 )
```

localhost/test/array2.php

तो देखा आपने कि किस तरह से PHP Array में हम किसी भी तरह की values store करा सकते हैं।



## Advantage of Array :

### 1.No Need to Define Multi Variables :

2.PHP में array use करने का सबसे बड़ा advantage यही है की हमें variables को बार बार define नहीं करना पड़ता है , हम एक साथ कई variables define करके use में ले सकते हैं।

### 3.Easy To Manage :

4. चूंकि अब सभी variables की values हमें एक single variable में मिल रही है तो हम इसा आसानी से manage कर सकते हैं।

## Types Of Array

### PHP Indexed Array

PHP में indexed array , store की गयी values को index number के रूप में associate करता है जो कि 0 start होता है।

### PHP Indexed Array Example

```
<?php
$arr1 = array('Student', 'Job', 'Study');
echo "<pre>";
print_r($arr1);
?>
```

### Output

```
Array
(
    [0] => Student
    [1] => Job
    [2] => Study
)
```

Example में simple indexed array define किया गया है।

PHP में हम इसके अलावा एक और तरीके से indexed array define कर सकते हैं , जिसमें हमें values को एक साथ नहीं रखते , need के according single value को array में store करते हैं।

Example :

```
<?php
$arr[ ] = "value1";
$arr[ ] = "value2";
$arr[ ] = "value3";
$arr[ ] = "value4";

echo "<pre>";
print_r($arr);
?>
```

### Output

```
Array
(
    [0] => value1
    [1] => value2
    [2] => value3
    [3] => value4 )
```

इस तरह से भी हम PHP में indexed array define कर सकते हैं , जहां index number automatically previous index के accordingly update होकर value को store करता रहता है।

अब अगर हमें array से सिर्फ single value को get करना है तो simply index number के साथ access कर सकते हैं। ऊपर दिए गए example में से अब अगर हमें first और last values चाहिए हो तो कुछ इस तरह से access करेंगे

Example :

```
<?php
$arr[ ] = "value1";
$arr[ ] = "value2";
$arr[ ] = "value3";
$arr[ ] = "value4";
echo "first value : ". $arr[0]." , last value :". $arr[3];
?>
```

### Output

first value : value1 , last value :value4

### Modifying Indexed Array Values

PHP में हम आसानी से array values को change / modify कर सकते हैं।

Example :

```
<?php
$arr = array('John', 'Tom');
echo 'Before change : <pre>';
print_r($arr);
$arr[0] = 'Hugo';
$arr[1] = 'Diego';
echo '</pre> After Change : <pre>';
print_r($arr);
?>
```

## Output

Before change :

Array

```
(  
  [0] => John  
  [1] => Tom  
)
```

After Change :

Array

```
(  
  [0] => Hugo  
  [1] => Diego  
)
```

## Traversing Using foreach Loop

array को foreach loop के through आसानी से traverse किया जा सकता है।

```
<?php
$arr[0] = 'Hugo';
$arr[1] = 'Diego';

foreach ($arr as $key=> $value)
{
    echo $key." = ".$value."<br>";
}
?>
```

### Output

```
0 = Hugo
1 = Diego
```

इस तरह से हम array को [foreach loop](#) के through traverse करते हैं , हालाँकि \$key variables optional होता है , अगर हम ye नहीं चाहते तो इसे remove करके direct values को use कर सकते हैं।

Example :

```
<?php
$arr[0] = 'Hugo';
$arr[1] = 'Diego';

foreach ($arr as $value)
{
    echo $value."<br>";
}
?>
```

## Output

Hugo  
Diego

## PHP Associative Array

Associative Array , एक प्रकार का array ही है जो कि array values को key के साथ associate करता है , means Associative Array में हम values के साथ-साथ key भी define करते हैं। Array values को key के साथ associate करने के लिए हम => का use करते हैं।

## PHP Associative Array Syntax

```
$x = array('key1'=>'value1', 'key2'=>'value2');
```

```
$x = ['key1'=>'value1', 'key2'=>'value2'];
```

ऊपर define किये गए किसी भी method के through हम PHP में Array Define कर सकते हैं।

## Difference Between Indexed Array And Associative Array

PHP में [Indexed Array](#) और Associative Array में सबसे बड़ा difference यही की [Indexed Array](#) में हमें सिर्फ values को insert करना होता है जिससे value indexed number के साथ associate (bind) होती थी , जबकि Associative Array में हम key के साथ value insert करते हैं जिससे value दी गयी key के साथ ही associate होती है।

## PHP Associative Array Example

```
<?php
```

```
$arr = ['key1'=> 'value1', 'key2'=> 'value2'];
```

```
echo'<pre>';
```

```
print_r($arr);
```

```
?>
```

### Output

```
Array
```

```
(
```

```
[key1] => value1
```

```
[key2] => value2
```

```
)
```

इसके आलावा आप [Indexed Array](#) की तरह key value को एक साथ न रखकर अलग -अलग भी assign कर सकते हैं , और उस पर [foreach loop](#) का use करके traversal भी कर सकते हैं।

Example

```
<?php
```

```
$arr['name'] = 'Rahul Rajput';
```

```
$arr['age'] = 25;
```

```
$arr['Skills'] = 'PHP, Laravel, JS, JQuery, Ajax, JSON etc.';
```

```
$arr['Designation'] = 'Web Developer';
```

```
/*Now print display these values using foreach loop*/
```

```
foreach($arr as $key => $value)
```

```
{
```

```
    echo $key." = ". $value."<br>";
```

```
}
```

```
?>
```

### Output

name = Rahul Rajput

age = 25

Skills = PHP, Laravel, JS, JQuery, Ajax, JSON etc.

Designation = Web Developer



इस तरह से भी हम values को key के साथ associate कर सकते हैं और [foreach loop](#) use कर सकते हैं। और अगर आप चाहें तो [Indexed Array](#) की तरह सिर्फ key के नाम के साथ भी single value को access कर सकते हैं।

For Example

```
<?php
```

```
$arr['name'] = 'Rahul Rajput';
```

```
$arr['age'] = 25;
```

```
$arr['Skills'] = 'PHP, Laravel, JS, JQuery, Ajax, JSON etc.';
```

```
$arr['Designation'] = 'Web Developer';
```

```
echo $arr['name']; /*It will print Rahul Rajput*/
```

```
echo $arr['age']; /*It will print 25*/
```

```
?>
```

Associative Array में key Define करते समय कुछ बातें ध्यान में रखें।

1. key का type आप string, integer, boolean, float numbers भी दे सकते हैं, But PHP दी गयी key के type के हिसाब से automatically दूसरे type में convert कर देता है जिसे type casting कहते हैं।
2. हालाँकि array और object को आप as a key नहीं दे सकते हैं। Otherwise PHP warning generate करती है।

## Associative Array Key Type Casting

1. string जो कि valid integer number या floating point number contain करती है PHP Floating numbers से Fractional part remove करके उसे Int में Cast कर देती है।
2. Boolean Values (true , false) में true को 1 और false को empty key में cast करती है। But इन values को access करते समय true key को true or 1 और false को false से ही access कर सकते हैं।

Example:-

```
<?php
$arr_var = ['str_key'=> 'value1', 23 => 45, 56.58=>56, true=>false];
echo"<pre>";
print_r($arr_var);
?>
```

### Output

```
Array
(
    [str_key] => value1
    [23] => 45
    [56] => 56
    [1] =>
)
```

String जो कि Boolean value (true , false) या 1 or 1.5 है को Boolean value True या 1 में ही cast करती है। और PHP Associative Array में अगर हम same name की key देते हैं तो हमेशा last key => value ही associate (bind) करती है।

See Example

```
<?php
$arr_var = ['1'=> '1.5', true => 45, 1.7=>56, 1=>'value 1'];
echo"";
print_r($arr_var);
?>
```

## Output

```
Array
(
    [1] => value 1
)
```

तो देखा की किस तरह से [PHP type casting](#) करती है , और same key होने की वजह से हमेशा last key=>value ही accept करती है।

Note - Empty String (") और Null हमेशा null ही होंगे और इन values को आप null या " से Access कर सकते हैं।

Example:-

```
<?php
$arr_var = ["=> '1.5', null => 45];
echo $arr_var["]; /*It will print 45*/
echo $arr_var[null]; /*It will also print 45*/
?>
```

### PHP Multidimensional Array

Multidimensional Array का Simply मतलब होता है Array inside Array , PHP में हम Multidimensional Array भी create कर सकते हैं , Multidimensional Array ऐसे दोनों तरह के हो सकते हैं Indexed Multidimensional Array , या Associative Multidimensional Array . ये हमारी need पर depend करत है कि हमें किसका use करना है।

Multidimensional Array में हम हर एक key या index के लिए value की जगह एक array देते हैं , और फिर उस array के अंदर हम अपनी जरूरत के हिसाब से data store करते हैं।

Note - यह जरूरी नहीं एक array के अंदर सिर्फ एक ही array हो आप अपनी need के accordingly कितने ही nested array create कर सकते हैं।

## PHP Multidimensional Array Example

```
<?php
    $students = [
        'student1' => ['name' => 'Raja', 'age'=>21, 'class'=> 9],
        'student2' => ['name' => 'Mohan', 'age'=>25, 'class'=> 10],
        'student3' => ['name' => 'Ram', 'age'=>19, 'class'=> 12],
    ];

    /* now it's time to access them */
    echo "student 1 : ".$students['student1']['name']." and class : ".$students['student1']['class']."<br>";
    echo "student 3 : ".$students['student3']['name']." and class : ".$students['student3']['class'];
?>
```

### Output

```
student 1 : Raja and class : 9
student 3 : Ram and class : 12
```

दिया गया example Two Multidimensional Associative Array है , आप Two Multidimensional Indexed Array भी इसी तरह से create कर सकते हैं , और access करने के लिए key की जगह पर index number (0 , 1 , 2 etc.. ) use करते हैं।

## PHP Traversing Multidimensional Array

पिछले topic में पढ़ा कि normal array (single dimensional array) traverse करने की लिए हमने [foreach loop](#) use किया था , अब दिए गए Two Multidimensional Array से data display करने के लिए हम [nested foreach loop](#) का use करेंगे।

Example:-

```
<?php
$students = [
    'student1' => ['name' => 'Raja', 'age'=>21, 'class'=> 9],
    'student2' => ['name' => 'Mohan', 'age'=>25, 'class'=> 10],
    'student3' => ['name' => 'Ram', 'age'=>19, 'class'=> 12],
];
foreach($students as $key => $student)
{
    echo 'data of '.$key.'<br>';
    /*here $student is an array */
    foreach($student as $key2 => $value)
    {
        echo $key2.' : '.$value.'<br>';
    }
}
?>
```

### Output

```
data of student1
name : Raja
age : 21
class : 9
data of student2
name : Mohan
age : 25
class : 10
data of student3
name : Ram
age : 19
class : 12
```

हम PHP में Multidimensional Array [foreach loop](#) का use करके traversal करते हैं , हालाँकि यह जरूरी नहीं कि आप [foreach loop](#) का ही use करें आप [for loop](#) भी use कर सकते हैं।

### PHP Traversing Multidimensional Array Using [for loop](#)

[for loop](#) के through traverse करने के लिए हम Two Multidimensional Indexed Array का example लेंगे , वैसे तो आप Associative Array का भी [foreach loop](#) के through traversal कर सकते हैं but फिर हमें array keys को extract करना पड़ेगा।

```
<?php
```

```
$students = [  
    ['Raja',21,9],  
    ['Mohan',25,10],  
    ['Ram',19, 12],  
];  
  
$size = count($students);  
for($s=0; $s<$size; $s++)  
{  
    echo 'data of studnet'.($s+1).'  
    /*now get inner array size */
```

```
$in_size = count($students[$s]);  
for($i=0; $i<$in_size; $i++)  
{  
    echo $students[$s][$i).'}  
}  
?>
```

### Output

```
data of studnet1  
Raja  
21  
9  
data of studnet2  
Mohan  
25  
10  
data of studnet3  
Ram  
19  
12
```

Note - **count()** एक predefined function है जिसका use array length जानने के लिए किया जाता है

## PHP Array Functions

PHP हमें कई सारे array functions provide कराती है जिनकी help से हम Array create , update , array merge और बहुत से Operations perform कर सकते हैं। कुछ array functions इस प्रकार हैं -

### PHP is\_array

is\_array( ) function check करता है कि दिया गया variable का type ARRAY है या नहीं।

```
<?php
```

```
    $arr = ['Girish', 'Mohit', 'Rahul'];
```

```
    if(is_array($arr))
```

```
        echo '$arr is an array';
```

```
    else
```

```
        echo '$arr is not an array';
```

```
?>
```

Output:-

\$arr is an array

### PHP array\_push

यह new element को array में सबसे last में insert करता है।



```
<?php
    $arr = ['Girish', 'Mohit', 'Rahul'];
    array_push($arr, 'Rohit');
    echo "after inserting new item";
    echo "<pre>";
    print_r($arr);
?>
```

### Output:-

```
after inserting new item
Array
(
    [0] => Girish
    [1] => Mohit
    [2] => Rahul
    [3] => Rohit
)
```

### PHP array\_pop

यह array में से सबसे last element को remove करता है।

```
<?php
$arr = ['Girish', 'Mohit', 'Rahul'];
array_pop($arr);
echo "after removing the last item";
echo "<pre>";
print_r($arr);
?>
```

## Output

after removing the last item

```
Array
(
    [0] => Girish
    [1] => Mohit
)
```

## PHP array\_shift

यह function array में से सबसे पहले element को remove करता है।

```
<?php
$arr = ['Girish', 'Mohit', 'Rahul'];
array_shift($arr);
echo "removes first value from the array";
echo "<pre>";
print_r($arr);
?>
```

## Output

removes first value from the array

```
Array
(
    [0] => Mohit
    [1] => Rahul
)
```

## PHP array\_unshift

यह array में सबसे पहले index पर new element को insert करता है।

```
<?php
$arr = ['Mohit', 'Rahul'];
array_unshift($arr, 'Girish');
echo "insert new value at the beginning of the array";
echo "<pre>";
print_r($arr);
?>
```

## Output

insert new value at the beginning of the array

Array

```
(
[0] => Girish
[1] => Mohit
[2] => Rahul
)
```

## PHP array\_sum And array\_product

array\_sum() function array में मौजूद सभी numeric values का Addition return करता है। जबकि array\_product() array में मौजूद सभी numeric values का Product return करता है।

```
<?php
$arr = [10, 20, 30];
echo "Sum of array elements : ". array_sum($arr);
echo "<br>Product of array elements : ". array_product($arr);
?>
```

### Output

Sum of array elements : 60

Product of array elements : 6000

array में numeric value के साथ mixed type value भी होती है तो PHP array\_sum() / array\_product() function run कराते समय उन्हें 0 मानती है।

## PHP String

String series of characters या group of characters होती है , PHP में मुख्यता चार तरह से हम string define करते हैं -

- 1.Single Quoted
- 2.Double Quoted .
- 3.heredoc syntax .
- 4.newdoc syntax

### PHP Single Quoted String

Single Quoted string PHP में String define करने सबसे आसान तरीका है। जिसके कुछ example आप नीचे देख सकते हैं।

```
<?php
```

```
$x = "bhuvanesh kumar";
```

```
echo 'hello';
```

```
echo 'hello $x';
```

```
/*It doesn't print the value of $x */
```

```
echo 'hello {$x}';
```

```
?>
```

### Output:-

```
hello
```

```
hello $x
```

```
hello {$x}
```

## PHP Double Quoted String

Double Quoted String के लिए " " use करते हैं इसके अंदर हम PHP variables भी print करा सकते हैं।

```
<?php
$x = "Bhuvanesh kumar";
echo "Hello";
echo "Hello : $x";
?>
```

### Output:-

```
Hello
Hello : Bhuvanesh kumar
```

## PHP Heredoc String

PHP में String define करने का ये तीसरा method है , heredoc syntax में string को define करने के liye three times less than ( <<<str ) और उसके बाद एक identifier provide किया जाता है , but string हम new line से ही start करते हैं। same identifier का use string quotation end करने जाता है।

```
$x = <<<START
first line content
another content
START;
```

close करते समय ये ध्यान रहे कि identifier newline के first column में ही हो , उससे पहले न कोई space , न variable कुछ भी नहीं होना चाहिए नहीं तो PHP Fatal Error Generate करेगी।

## PHP Heredoc String Example

```
<?php
$x = 10;
/*valid*/
echo <<<START
this is simple example
START;
```

```
/* valid (you can assign it to any variable */
$str = <<<START
this is simple example
START;
echo $str;
```

```
/* valid (you can print variable inside it just like double
quoted string "" */
$str = <<<START
this is simple example $x
START;
echo $str;
```

```
/*Invalid because before the end of identifier there is
some space */
echo <<<START
this is simple example
START;
```

```
/*Invalid , because there is a comment in same line */
echo <<<START
this is simple example
START; /*even you can't write comment here*/
```

```
/*Invalid , because you have left some space befoer
semocolon */
echo <<<START
this is simple example
START ;
?>
```

## PHP Newdoc String

Newdoc String Single quoted string होती है। और यही main difference है heredoc string में और Newdoc string में। heredoc string के अंदर हम variables print करा सकते हैं जबकि Newdoc string के अंदर variables की value की जगह variable का नाम print हो जाएगा।

Means Newdoc String में parsing नहीं होती है , इसके अलावा एक और difference है यहां हम identifier को single quote में लिखते हैं start करते समय।

```
echo <<<'START'  
write string here  
START;
```

```
<?php  
$x = 10;  
/*valid*/  
echo <<<'START'  
this is simple example <br>  
START;
```

```
/* valid (you can assign it to any variable just like heredoc string */  
$str = <<<'START'  
you can assign it to variable <br>  
START;  
echo $str;
```



```
/* valid but variable value will not be print */  
$str = <<<'START'  
variable value doesn't print here $x  
START;  
echo $str;  
?>
```

## Output

this is simple example  
you can assign it to variable  
variable value doesn't print here \$x

## PHP Function

Function reusable piece of code या block of code होता है जो कि कोई specific task perform करता है। एक बार define करने के बाद हम इन्हें कितनी ही बार use कर सकते हैं। PHP में 1000+ predefined useful function हैं जिन्हें हम Built In Functions कहते हैं।

हालाँकि PHP हमें ये facility provide करती है कि user खुद के function define कर सके जिनहे **User Defines Functions** कहते हैं । Web page load होते समय कोई भी function automatically run नहीं होता है जब तक कि हम उसे call न करें।

## PHP Function Example

```
<?php
/*define the function */
function myfun()
{
    echo "Hell ! this is my first user defined function.";
}

/*now it's time to call */
myfun();
?>
```

### Output:-

Hell ! this is my first user defined function.

## PHP Function : Important Tips

1. आप एक ही name के function define नहीं कर सकते , अगर आप ऐसा करते हैं तो PHP Fatal Error Generate कर देगी।
2. PHP में Functions नाम ASCII characters A to Z के लिए **case insensitive** होते हैं , means myfun() define किये जाने के बाद इसे आप Myfun() या MYFUN() name से भी call करा सकते हैं।

**Note** - PHP में functions और methods दोनों अलग - अलग हैं -

Methods किसी Class के अंदर define किये जाते हैं इन्हें हमेशा Class Name या Class Object के साथ access किया जाता है हालाँकि यह method के type पर depend करता है , अगर method static हुआ तो Class Name के साथ access किया जायेगा और अगर non-static हुआ तो Class Object के साथ access किया जायेगा।

जबकि function out of the class define किया जाते हैं , ये Class / Object के context में नहीं होते इसलिए इन्हें कहीं भी access किया जा सकता है , class के अंदर भी और class के बाहर भी।

### PHP Function Advantages

- 1. Code Re-usability** : functions use करने का सबसे बड़ा advantage यही है कि , हम code को reuse कर सकते हैं। same processing के लिए एक बार function define करने के बाद उसे हम कहीं भी और कितनी बार भी use कर सकते हैं।
- 2. Less Code** : चूंकि हम same code के लिए functions use करते हैं जिससे Program की length कम जाती है।
- 3. Reduce Coding Time** : Function use करने से coding time reduce होता है , जो कि किसी भी developer के लिए important है।
- 4. Easy To Understand** : Code को समझना आसान हो जाता है।
- 5. Easy To Maintain** : समय के साथ - साथ हमें code update करना पड़ जाता है जिसे हम function की help से आसानी से maintain कर सकते हैं।

## PHP Types Of Function

PHP में हम लगभग 7 type के functions use करते हैं।

### PHP Function With Return Value

किसी भी function return statement यह function का end indicate करता है। means function के अंदर जहां भी हम return statement लिखते हैं function वही तक execute होता है , value return करने के बाद execution उसी line से start होता है

PHP User Define Functions में return statement Optional होता है , आप चाहे तो function में value return करा सकते हैं और नहीं भी।

### PHP Function With Return Value Syntax

```
function function_name()
{
    return value;
}
```

### PHP Function With Return Value Example

```
<?php
function test()
{
    return echo "Hello Test";
}

echo test();
?>
```

### Output

Hello Test

PHP में Function names A to Z के लिए case insensitive होते हैं means , example में define किये गए function को आप test(), Test() या tesT() से भी call करा सकते हैं।

## PHP Function : Return Type Declaration

Older Versions में हम किसी भी तरह की value return करा सकते हैं। **PHP 7** में function के लिए एक नया update आया **return type declaration**. means function define करते समय हम return type भी define करते हैं जो कि ensure करता है कि वह function सिर्फ define किये गए type की value ही return करेगा।

```
<?php
function test1() : string
{
    return "Hello Test";
}

echo Test1();
?>
```

### Output

Hello Test

## PHP Function : Return Type With Strict Mode

अगर हम define किये गए type की value return नहीं करते हैं तो वह function कुछ भी return नहीं करता है , या हो सकता है कि **Warning generate** करे और अगर **Strict Mode On** हुई तो PHP Fatal Error Generate कर सकती है।

```
<?php
declare(strict_types=1);
function test1() : int
{
    return "Hello Test";
}

echo Test1();
?>
```

### Output

PHP Fatal error: Uncaught TypeError: Return value of test1() must be of the type integer, string returned in **C:\xampp\htdocs\test\param\_fun3.php:5**

Example में function का return type integer define किया गया है जबकि function string value return कर रहा है , इसलिए fatal error generate हुई।

## PHP Parameterized Functions

Parameterized Function वो function होते हैं जो कि **parameter** accept करते हैं , इन functions को define करते समय हम parameter भी define करते हैं जो कि ensure करते हैं कि call करते समय हम कितने argument pass करने वाले हैं।

function में हम अपनी need के according कितने ही parameter pass कर सकते हैं। और यही parameter उस function के लिए as a **variable** work करते हैं।

Technically देखा जाए तो किसी function के लिए **Parameter** और **Argument** दोनों अलग अलग हैं। **Parameters** function definition के समय define किये गए variables होते हैं , जबकि **function call** करते समय pass की गयी values / variables **Arguments** होते हैं।

### PHP Parameterized Function Example

```
<?php
```

```
/*function that returns sum of two numbers */
```

```
function add($num1, $num2)
```

```
{
```

```
    return $num1+$num2;
```

```
}
```

```
echo 'Sum of 234 and 123 is :'. add(234, 123). '<br>';
```

```
echo 'Sum of 546 and 23 is :'. add(546, 23);
```

```
?>
```

### Output

Sum of 234 and 123 is :357

Sum of 546 and 23 is :569

```
<?php
/*define function with the parameter type */
function add(int $num1, int $num2)
{
    return $num1+$num2;
}

echo ' Sum : ' .add(56, 123);
?>
```

### Output

Sum : 179

## PHP Function With Optional Parameter

PHP में आप function को **optional parameter** के साथ भी define कर सकते हैं , इसके लिए function declaration के समय null assign कर दिया जाता है , फिर function समय value देते हैं तब भी function run होगा और value pass न करें तो भी Run होगा।



```
<?php
/*define function with optional parameter */
function print_name($first_name, $last_name=null)
{
    echo $first_name.' '.$last_name;
}

/*call function with both values */
print_name('bhuvanesh', 'Kumar').'\n';
/*now call function with only first value */
print_name('Hello');
?>
```

### Output

Bhuvanesh Kumar  
Hello

### PHP Function With Default Parameter

PHP हमें Function में **default parameter** भी set करने की facility provide करती है , कई जगह हमें ऐसी जरूरत पड़ती है जिसमे हमें कुछ parameters default रखने होते हैं , अगर उस parameter की value pass की जाती है तब passed value use होगी otherwise **default** value use होगी।

```
<?php
/* define function with default parameter*/
function print_table(int $number=2)
{
    for($n=1; $n <= 10; $n++)
    {
        echo $n*$number.' , ';

    }
}

/* first call function withowt argument */
print_table();
echo'<br>';
/* now pass any number so that we can diferenciate*/
print_table(5);
?>
```

### Output

```
2 , 4 , 6 , 8 , 10 , 12 , 14 , 16 , 18 , 20 ,
5 , 10 , 15 , 20 , 25 , 30 , 35 , 40 , 45 , 50 ,
```

## PHP Function Call By Value and Call By Reference

### PHP Difference Between Call By Value And Call By Reference

**Call By Value** और **Call By Reference** Functions में सबसे बड़ा difference भी यही है , **Call By Value** में हम सिर्फ value को accept करते हैं , इसलिए इस value को हम function के अंदर ही manipulate कर सकते हैं function के बाहर नहीं जबकि **Call By Reference** में हम function में value की जगह value का reference (जो कि & ampersand से access करते हैं ) use करते हैं।

### PHP Function Call By Value

अभी तक हम जो functions define और use कर रहे थे ये **Call By Value** functions थे । जिसमे हम simply value को pass करते थे।

```
<?php
$str = 'Hello';
function test($str)
{
    return $str.=' Welcome';
}

echo test($str)."<br>";
echo $str;
?>
```

### Output

Hello Welcome  
Hello

Example में आप देख सकते हैं कि Function के अंदर value को manipulate किया गया है , but function call होने के बाद define किये गए external variable की value पर कोई असर नहीं पड़ा।

## PHP Function Call By Reference

इस तरह के functions में function define करते समय parameter से पहले ampersand (&) का use करते हैं . **Reference** means variable content को different नाम के साथ access करना , हालाँकि यह Pointers की तरह नहीं है जैसे C में होता है।

```
<?php
$str = 'Hello';
function test(&$str)
{
    return $str.=' Welcome';
}

echo test($str)."<br>";
echo $str;
?>
```

### Output

```
Hello Welcome
Hello Welcome
```

Same Example में value की जगह variable से पहले value का reference (& ampersand) use किया गया है , इसलिए function के अंदर value को modify करते ही उसकी actual value भी चेंज हो गयी।

## PHP Variable Length Argument Functions

मतलब हमें ये पता होता था कि कितने variables pass किये जायेंगे उसके according हम function के अंदर logic लिखते थे , अब अगर हमें ये नहीं मालूम हो कि function call करते समय कितने variables pass हो रहे हैं , इस तरह के functions को handle करने के लिए हमें PHP ने facility provide की है **Variable Length Argument Functions** . जिसकी help से हम pass किये गए सभी arguments / values को easily handle कर सकते हैं। इसके लिए हमें function define करते समय variable से पहले सिर्फ **triple dots ( ... )** prepend करने होते हैं।

### PHP Variable Length Argument Function Example

```
<?php
function print_vars(...$vars)
{
    echo 'Numbers of variables : '. count($vars).'<br>';
    foreach($vars as $var)
    {
        echo $var.' , ';
    }
}

print_vars(12, 34, 56);
?>
```

#### Output

```
Numbers of variables : 3
12 , 34 , 56 ,
```

जब हम Variable Length Argument Handle करते हैं तो जितने भी variables pass करते हैं function call करते समय हमें उन Variables का [Array](#) मिलता है , जिसे **count()** function के through total passed variables पता कर सकते हैं , साथ ही [Foreach Loop](#) या [For Loop](#) का use भी कर सकते हैं।

## PHP func\_get\_args function

अगर हम function define करते समय कोई भी parameter define नहीं करना चाहते और pass किये गए सभी arguments handle भी करना चाहते हैं , तो आप predefine function **func\_get\_args()** का भी use कर सकते हैं। हालाँकि **func\_get\_args()** भी हमें pass किये गए सभी arguments का [Array](#) ही return करता है।

```
<?php
function test_fun()
{
    $arguments = func_get_args();
    echo '<pre>';
    print_r($arguments);
}
test_fun(45, 34, 56, 67, 78, 34);
?>
```

### Output

```
Array
(
    [0] => 45
    [1] => 34
    [2] => 56
    [3] => 67
    [4] => 78
    [5] => 34
)
```

## PHP Recursive Functions

C /C++ की तरह ही PHP भी हमें recursive functions define करना allow करती है। Function को उसी Function के अंदर एक specified condition तक **call** करने को ही **recursive function** कहते हैं , और इस process को **recursion** कहते हैं।

### PHP Use Of Recursive Function

Recursive Functions का use tree structure (read या tree structure बनाने में ) में सबसे ज्यादा होता है , जहां पर हमें ये नहीं मालूम होता की Node का कोई children है या नहीं अगर है तो call the function.

2. दूसरा File Directories को **read** करने के लिए , क्योंकि हमें नहीं मालूम होता है कि एक directory के अंदर सभी files होंगी या sub directory, और फिर उसके अंदर sub-directories और फिर उसके अंदर etc.

### PHP Recursive Functions Example

```
<?php
```

```
function test_rec($number)
```

```
{
```

```
    echo $number.' , ';
```

```
    $number++;
```

```
    if($number <= 100)
```

```
    {
```

```
        test_rec($number);
```

```
    }
```

```
}
```

```
test_rec(1);
```

```
?>
```

### Output

```
1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 ,  
18 , 19 , 20 , 21 , 22 , 23 , 24 , 25 , 26 , 27 , 28 , 29 , 30 , 31 ,  
32 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42 , 43 , 44 , 45 ,  
46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 58 , 59 ,  
60 , 61 , 62 , 63 , 64 , 65 , 66 , 67 , 68 , 69 , 70 , 71 , 72 , 73 ,  
74 , 75 , 76 , 77 , 78 , 79 , 80 , 81 , 82 , 83 , 84 , 85 , 86 , 87 ,  
88 , 89 , 90 , 91 , 92 , 93 , 94 , 95 , 96 , 97 , 98 , 99 , 100 ,
```

ऊपर दिए गए example में 100 बार same function call हुआ है। example को आप ध्यान से देखेंगे तो पायेंगे कि एक **condition** दी हुई है कि अगर variable की value 100 से कम या बराबर है तभी function को call किया गया है।

Recursive Functions का work flow same as [For Loop](#) की तरह ही होता है , क्योंकि [For Loop](#) में भी execution एक specified condition तक repeat होता है। Condition false होने पर Loop end हो जाता था।

Recursive Functions को और अच्छे से समझने के लिए हम एक और common example देखते हैं - get factorial of any number .

```
<?php
function find_fact($n)
{
    if ($n == 0)
    {
        return 1;
    }
    echo $n.' * ';
    /* Recursion */
    $result = ( $n * find_fact( $n-1 ) );
    return $result;
}
```

### Output

5 \* 4 \* 3 \* 2 \* 1 \* The factorial of 5 is: 120

10 \* 9 \* 8 \* 7 \* 6 \* 5 \* 4 \* 3 \* 2 \* 1 \* The factorial of 10 is: 3628800

पिछले example की तरह इसमें भी एक **condition** दी है कि variable की value 0 होने पर सिर्फ 1 ही **return** करता है , otherwise same function call होता रहेगा।

Recursive Function में Loop की तरह ही **condition** देना mandatory है , otherwise function infinite call करता रहेगा और page breach हो सकता है।

```
echo "The factorial of 5 is: " . find_fact( 5 ).'<br>';
echo "The factorial of 10 is: " . find_fact( 10 );
?>
```



## PHP Disadvantages Of Recursive Function

Recursive Functions , iterative program (like [For Loop](#) / [While Loop](#) ) के तुलना में काफी **memory** और time **consume** करते हैं। इसलिए ज्यादा जरूरी होने पर ही **Recursive Functions** का use करें।

## PHP Anonymous Functions

Anonymous Functions , जैसा कि नाम से ही ही मालूम होता है ऐसे functions जिनका कोई name नहीं है , Yes PHP हमें ये facility provide करती है कि हम without name के functions भी define कर सके। Anonymous Functions Closure Functions भी कहते हैं।

Anonymous Functions internally Closure class का use करते हैं।

## PHP Anonymous Function Example

```
<?php
$var = function()
{
    echo"Print Inside Anonymous Function";
};
$var();
?>
```

## Output

Print Inside Anonymous Function

**Note** - Anonymous Functions define करते समय curly brackets के बाद semicolon ( ;) लगाना न भूलें otherwise PHP **Fatal Error** Generate कर देगी।

Normal Functions की तरह ही Anonymous Functions में भी हम arguments pass कर सकते हैं। पिछले Topic में हमने [Variable Length Arguments Functions](#) के बारे में पढ़ा , उसी तरह से इसमें भी variable Length Arguments handle कर सकते हैं।

```
<?php
$my_fun = function()
{
    echo"This is normal anonymous function.";
};
$my_fun();
echo"<br>";
/* function to handle variable list arguments */
$list_fun = function(...$x)
{
    echo"<pre>";
    print_r($x);
};
$list_fun(23,45,67,889);
?>
```

## Output

This is normal anonymous variable.

```
Array
(
    [0] => 23
    [1] => 45
    [2] => 67
    [3] => 889
)
```

## PHP Arrow Functions

PHP में Arrow Functions **PHP version 7.4** में introduce किये गए थे। [Anonymous Functions](#) और Arrow Functions दोनों ही functions को Closure Class implement करके बनाया गया है। Arrow Functions भी same functionalities support करता है जो कि [Anonymous Functions](#) करता है , सिर्फ एक difference कि , Arrow Functions में external variables के लिए हमें **use keyword** का use नहीं करना पड़ता है। ये variables हम directly access कर सकते हैं।

**Note** - Arrow Functions पढ़ने से पहले यह check कर लें कि आपका PHP Version 7 . 4 या इससे ज्यादा है , otherwise fatal error generate होगी।

### PHP Arrow Function Example

```
<?php
$y = 1;
$fn1 = fn($x) => $x+$y;
echo $fn1(10);
?>
```

### Output

11

Simply Arrow ( => ) के बाद जो भी हम value लिखते हैं Arrow Function वो return कर देता है। हालाँकि इसके लिए हमें return statement लिखने की जरूरत नहीं पड़ती है।

## What Does Arrow Function ?

1. callback function की तुलना में Arrow Function short होते हैं , जिससे performance fast और execution time कम हो जाता है।
2. External Variables ( जो कि function के बाहर define किये गए हैं ) को access करने के लिए use keyword नहीं use करना पड़ता है।

## PHP Form Handling

PHP में हम form को submit कर सकते हैं जिससे user data को process कर सकें , या need के according database में save कर सकें।

PHP में generally हम form को submit / handle करने के लिए हम generally three [Predefined Super Global Variables](#) का use करते हैं।

1. \$\_GET
2. \$\_POST
3. \$\_REQUEST

इन variables का use हम submit किये गए form के according use करते हैं।

## \$\_GET

PHP \$\_GET Super Global Variable है means \$\_GET के through form data access करने के लिए हमें इस global variable बनाने की जरूरत नहीं होती , PHP Script में इसे कहीं भी access कर सकते हैं।

**\$\_GET** variable का use हम get method type के साथ submit किये गए form को handle करने के लिए use करते हैं । किसी भी form को जब हम get method के साथ submit करते हैं तो उस form का data **query string** के रूप में URL के साथ key=value (जहां key name submit किये गए form का **input name** होता है और **value**, उस input में enter की गयी value ) pair में append हो जाता है , जिसे हम अपने **URL** में आसानी से देख सकते हैं।

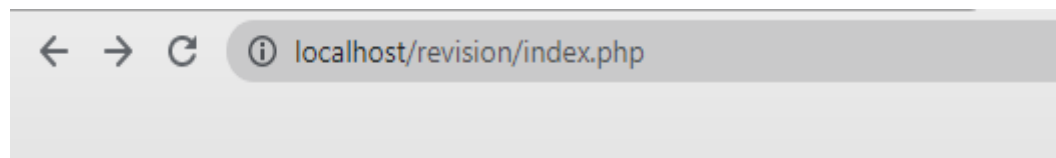
\$\_GET Submitted Form Data को एक Array में store करता है , जहां key name input field का नाम और value input field में enter की गयी value होती है। जिसे हम input field name के साथ आसानी से access कर सकते हैं।

## PHP Handle Form Using \$\_GET

```
<!DOCTYPE html>
<html>
  <head>
    <title>PHP Form Handling Using $_GET</title>
  </head>
  <body>
    <form action="get.php" method="get">
      <p>Name : <input name="name" type="text" placeholder="Enter Your Full Name"></p>
      <p>Age : <input name="age" type="number" min="18" max="120" placeholder="Enter Your Age"></p>
      <p><button type="submit">Submit</button></p>
    </form>
  </body>
</html>
```

```
<?php
/* here we will handle submitted form*/
echo 'Entered All Values : <pre>';
print_r($_GET);
echo '</pre>';
echo 'Your Full Name : ' . $_GET['name'] . "<br>";
echo 'Your Age : ' . $_GET['age'];
?>
```

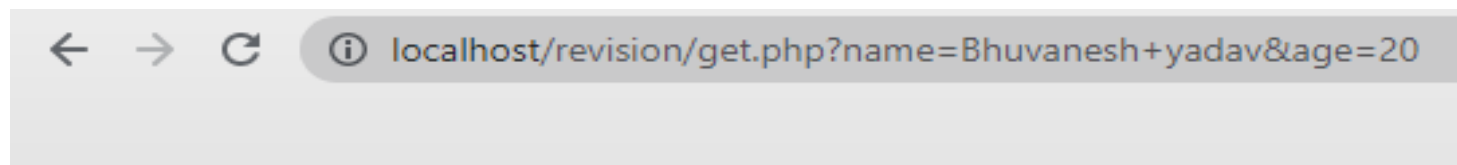
## Output:-

A screenshot of a web browser's address bar. It contains navigation icons (back, forward, refresh) and an information icon. The address is 'localhost/revision/index.php'.

Name :

Age :

After submit:-

A screenshot of a web browser's address bar. It contains navigation icons (back, forward, refresh) and an information icon. The address is 'localhost/revision/get.php?name=Bhuvanesh+yadav&age=20'.

Entered All Values :

```
Array
(
    [name] => Bhuvanesh yadav
    [age]  => 20
)
```

Your Full Name : Bhuvanesh yadav

Your Age : 20

Example को अब हम step by step समझेंगे -

- 1.सबसे पहले हमने एक form बनाया जिसका **submit method get / GET** है जिससे हम **\$\_GET** के through form data access कर सके , अगर आप Form का **method define** नहीं भी करते तो automatically **get** ही होता है। form action **get.php** है means जब हम form submit करेंगे तो वह form हम **get.php** file में handle करेंगे।
- 2.उसके बाद दो input field name और age बनाये जिससे हम value Enter कर सके । age field के लिए हमने HTML Validation लगाएं हैं जिससे end user 0 से काम और 120 से ज्यादा age enter न कर पाए। और एक Button जिसका type submit है।
- 3.अब get.php file में हमने सबसे पहले तो **\$\_GET** Super Global variable को print कराया।
- 4.और फिर submit की गयी values को एक एक करके access किया।
- 5.Note - अगर आप URL देखेंगे तो पाएंगे कि submit की गयी values URL के साथ append हो गयी है।

### When Should We Use It ?

**\$\_GET** use करने से पहले इसके बारे में और अधिक जानते हैं जिससे यह clear हो सके कि कब हमें ये use करना चाहिए और कब नहीं।

हालाँकि get request के through हम limited data ही send कर सकते हैं , क्योंकि यह URL में visible होता है और हर कोई इसे देख सकता है। get request के through लगभग हम 2000 characters ही send कर सकते हैं।

**\$\_GET** का use कभी भी important data send करने के लिए नहीं करना चाहिए , जैसे - User Authentication या Payment Process etc...



## PHP \$\_POST

**\$\_POST** [Super Global Variable](#) है means \$\_POST के through form data access करने के लिए हमें इस global variable बनाने की जरूरत नहीं होती , PHP Script में इसे कहीं भी access कर सकते हैं। \$\_POST का use PHP में Post / post Method के साथ submit किये गए Form को handle ( Extracting Form Input Data) करने के लिए use किये जाता है। Form को Handle करने का यह सबसे reliable और अच्छा method है।

\$\_GET की तरह ही \$\_POST भी Submitted Form Data को एक [Array](#) में store करता है , जिसे हम input field name के साथ आसानी से access कर सकते हैं।

## **\$\_GET vs \$\_POST**

**\$\_POST** और **\$\_GET** में सबसे बड़ा difference यही कि \$\_GET का use हम **get / Get** method के साथ submit हुए form या किसी **URL** में query string से data को access करने के लिए किया जाता है , जबकि **\$\_POST** का use हम **Post / post** Method के साथ submit किये गए Form को handle ( Extracting Form Input Data) करने के लिए use किये जाता है।

## \$\_POST Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Form Handling Using $_POST</title>
```

```
</head>
```

```
<body>
```

```
<form action="post.php" method="post">
```

```
    <p>Full Name : <input type="text" name="full_name" placeholder="Enter Full Name"
```

```
/></p>
```

```
    <p>Age : <input type="number" name="age" min="0" max="120" placeholder="Enter
```

```
Address" /></p>
```

```
    <p>Address : <input type="text" name="address" placeholder="Enter Address" /></p>
```

```
    <p><button type="submit">Submit</button></p>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
<?php
```

```
    /* here we will handle submitted form*/
```

```
    echo 'Entered All Values : <pre>';
```

```
    print_r($_GET);
```

```
    echo '</pre>';
```

```
    echo 'Your Full Name :
```

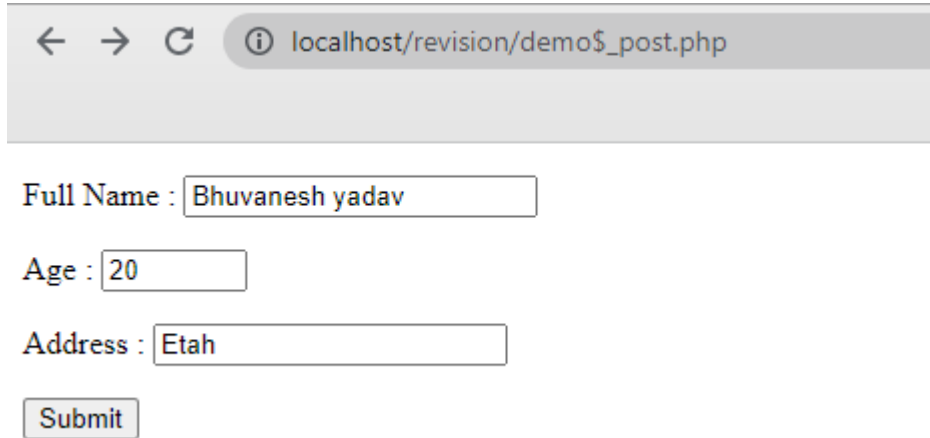
```
    '$_POST['full_name']. "<br>";
```

```
    echo 'Your Age : ' . $_POST['age']. "<br>";
```

```
    echo 'Your Age : ' . $_POST['address'];
```

```
?>
```

## Output:-



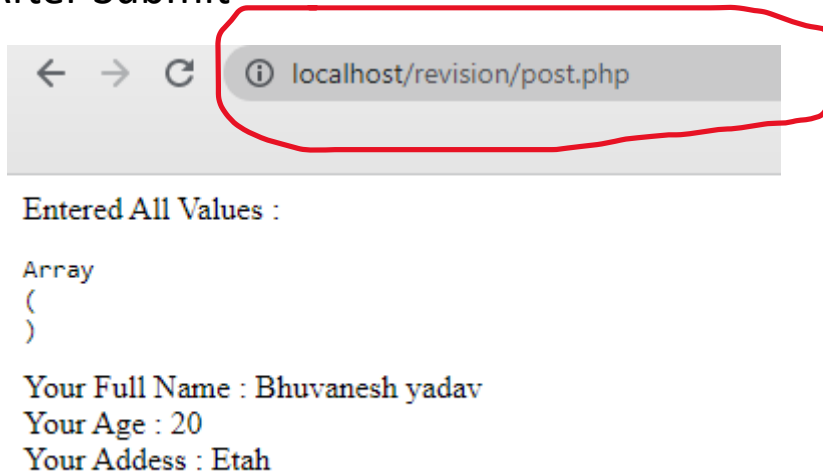
← → ↻ ⓘ localhost/revision/demo\$\_post.php

Full Name :

Age :

Address :

After Submit



← → ↻ ⓘ localhost/revision/post.php

Entered All Values :

Array  
(  
)

Your Full Name : Bhuvanesh yadav  
Your Age : 20  
Your Address : Etah

दूसरी image में आप साफ़ देख सकते हैं कि जिस Form Action में Form को handle करने के लिए जिस file का नाम दिया था , Form Submit होने के बाद Data **get / Get** method की तरह **query string** में append नहीं होता है।

**Note** - \$\_POST के through हम सिर्फ Post / post method के साथ Submit किये गए Form Data को ही access कर सकते हैं। Get / get method के साथ Submit किये गए Form Data को \$\_POST के through कभी भी access नहीं कर सकते हैं।

## PHP \$\_REQUEST

PHP में **\$\_REQUEST** , **\$\_GET** और **\$\_POST** की तरह ही [Super Global Variable](#) है means इसे भी हमें define करने की जरूरत नहीं पड़ती और PHP Script में इसे हम कहीं भी access कर सकते हैं।

**\$\_REQUEST** का use हम get और post दोनों तरह के method के साथ submit किये गए Form Data को access करने के लिए किया जाता है। means हम URL में **query string** में आ रहे data को भी **\$\_REQUEST** के through access कर सकते हैं। **\$\_REQUEST** एक [Associative Array](#) होता है जो कि **\$\_GET** और **\$\_POST** दोनों variables का data store करता है।

## **\$\_REQUEST vs \$\_POST vs \$\_GET**

हम जानते हैं कि जब कोई Form get method के साथ submit किया जाता है या data URL में query string के रूप में आ रहा है तो उसे हम [\\$\\_GET Super Global Variable](#) से ही access कर सकते हैं। और जब कोई Form post method के साथ submit किया जाता है तो उसे हम [\\$\\_POST Super Global Variable](#) से access कर सकते हैं। But **\$\_REQUEST** के through हम **\$\_GET** और **\$\_POST** दोनों तरह का data access कर सकते हैं। और इन तीनों में यही main difference भी है।

एक example के through समझेंगे कि किस तरह से **\$\_REQUEST** Variable , **\$\_GET** और **\$\_POST** दोनों Variable का data contain करता है।

## PHP \$\_REQUEST Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Form Handling Using $_Request </title>
</head>
<body>
<form action="request.php?name=Bhuvanesh Kumar&age=20" method="post">
    <p>Type :
        <select name="student">
            <option value="">Select Type</option>
            <option value="Student">Student</option>
            <option value="Teacher">Teacher</option>
            <option value="Staff">Staff</option>
            <option value="Other">Other</option>
        </select>
    </p>
    <p>Address : <input type="text" name="address" placeholder="Enter Full Address" /></p>
    <p><button type="submit">Submit</button></p>
</form>
</body>
</html>
```

```
<?php
```

```
/* here we will print submitted form values with $_GET , $_POST , $_REQUEST*/
```

```
echo '$_GET Values : <pre>';
```

```
print_r($_GET);
```

```
echo '</pre>';
```

```
echo '$_POST Values : <pre>';
```

```
print_r($_POST);
```

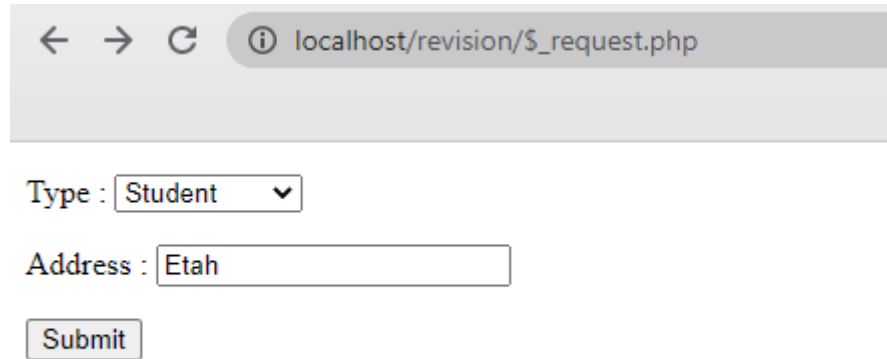
```
echo '</pre>';
```

```
echo 'All Values : <pre>';
```

```
print_r($_REQUEST);
```

```
?>
```

## Output:-

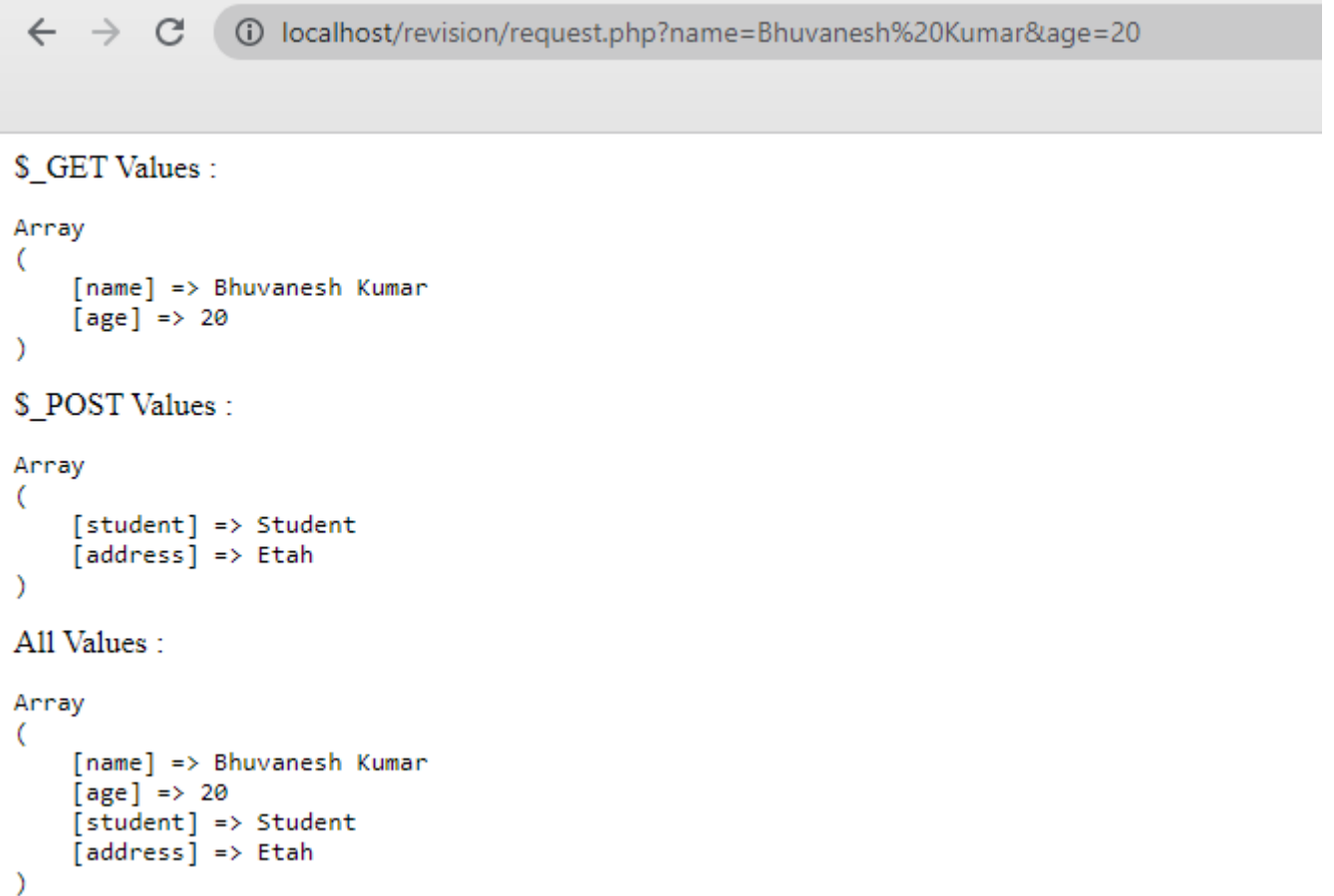


← → ↻ ⓘ localhost/revision/\$\_request.php

Type :

Address :

## After Submit:-



← → ↻ ⓘ localhost/revision/request.php?name=Bhuvanesh%20Kumar&age=20

\$\_GET Values :

```
Array
(
    [name] => Bhuvanesh Kumar
    [age] => 20
)
```

\$\_POST Values :

```
Array
(
    [student] => Student
    [address] => Etah
)
```

All Values :

```
Array
(
    [name] => Bhuvanesh Kumar
    [age] => 20
    [student] => Student
    [address] => Etah
)
```

## PHP File Uploading

PHP में आप easily file upload कर सकते हैं , साथ साथ ये भी देखेंगे कि कैसे uploaded file को validate करते हैं file के type के according .

PHP में File upload करने के लिए Form Method 'post' ही होना चाहिए , 'get' method के साथ आप file upload नहीं कर सकते हैं। इसके अलावा Form tag में एक और attribute add करना होता है enctype="multipart/form-data" .

## PHP File Uploading Example

PHP में File को handle करने के लिए हम **\$\_FILES** Super Global Variable का use करते हैं। क्योंकि file को हम [\\$\\_GET](#) , [\\$\\_POST](#) या [\\$\\_REQUEST](#) से handle नहीं कर सकते हैं। PHP , File Input Field के name का [Associative Array](#) create करता है जिसमें File से related info contain करता है जिसमे file का actual name , file type ,file size etc .. information होती है जिससे हम uploading के समय ही file rename करके save करें।

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP FileUploading</title>
    </head>
    <body>
        <form action="file_upload.php" method="post" enctype="multipart/form-data">
            <p>Upload File : <input type="file" accept="image/*" name="myimage"
                placeholder="Upload File" /></p>
            <p><button type="submit">Submit</button></p>
        </form>
    </body>
</html>
```



```
<?php
/* first check file */
if(isset($_FILES['myimage']['name']))
{
    echo 'File Info : <pre>';
    print_r($_FILES['myimage']);
    echo '</pre>';
    /* before upload the file we should rename it */
    $tmp = $_FILES["myimage"]["tmp_name"];
    $extension = explode("/", $_FILES["myimage"]["type"]);
    $new_name = time()." ".$extension[1];

    if(move_uploaded_file($tmp, $new_name))
        echo "Your file is uploaded with ".$new_name." name";
    else
        echo 'Error In Uploading';
}
else
{
    echo 'File Upload Field Is Required';
}
?>
```

## Output

File Info :Array

```
(
  [name] => 1.png
  [type] => image/png
  [tmp_name] => C:\xampp\tmp\phpCA27.tmp
  [error] => 0
  [size] => 4913
)
```

Your file is uploaded with 1600055903.png name

## PHP Multiple File Uploading

हम single file upload करते हैं तो file field नाम का **One Dimensional Array** मिलता है और उसी field name में जब हम multiple files upload करते हैं तो हमें **Two Dimensional Array** मिलता है जिसे हम Foreach Loop का use करके आसानी एक - एक करके file को upload कर सकते हैं। इसके अलावा Multiple File upload करने के लिए हमें file input tag में '**multiple**' attribute add करना पड़ता है। और file name को as a Array variable define ( myimage[] ) करना पड़ता है।

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP FileUploading</title>
    </head>
    <body>
        <form action="multi_file_upload.php" method="post" enctype="multipart/form-data">
            <p>Upload File : <input type="file" accept="image/*" name="myimage[]"
placeholder="Upload File" multiple="multiple" /></p>
            <p><button type="submit">Submit</button></p>
        </form>
    </body>
</html>
```

```

<?php
    echo 'File Info : <pre>';
    print_r($_FILES['myimage']);
    echo '</pre>';
    $total_files = count($_FILES['myimage']['name']);
    for($n=0; $n<$total_files; $n++)
    {
        /* before upload the file we should rename it */
        $tmp = $_FILES["myimage"]["tmp_name"][$n];
        $extension = explode("/", $_FILES["myimage"]["type"][$n]);
        $new_name = time()." ".$extension[1];

        if(move_uploaded_file($tmp, $new_name))
            echo "Your file is uploaded with $new_name name <br>";
        else
            echo 'Error In Uploading';
    }
?>

```

## Output

```

File Info :Array
(
    [name] => Array
    (
        [0] => open-admin.png
        [1] => open-xampp.png
    )
    [type] => Array
    (
        [0] => image/png
        [1] => image/png
    )
    [tmp_name] => Array
    (
        [0] => C:\xampp\tmp\php825E.tmp
        [1] => C:\xampp\tmp\php828E.tmp
    )
    [error] => Array
    (
        [0] => 0
        [1] => 0 ) [size] => Array ( [0] => 57716 [1] => 42910 )

```

## PHP Session

Session का use information को temporary store करने के लिए जाता है। या हम कह सकते हैं कि Session user data को temporary store करने का एक Alternative way है। Session में store data सभी pages पर accessible होता है , जिसे हम सभी pages पर access या modify कर सकते हैं।

### Starting PHP Session

PHP में Session start करने के लिए हम predefined function **session\_start()** use किया जाता है। **session\_start()** existing session return करता है यदि session already start हो चुका है , यदि session start नहीं हुआ है तो new session start कर देता है ।

**session\_start()** हमेशा page के starting में ही call करना चाहिए। और अगर हमने session\_start() call नहीं किया तो उस page पर हम Session data access नहीं कर सकते हैं।

### When Session Starts First Time

- जब पहली बार Session start होता है तो PHP particular Session के लिए एक **32 Hexadecimal Number** की Unique **Session ID** generate करता है।
- इस **Unique Session ID** को store करने के लिए एक '**PHPSESSID**' नाम की Cookie user browser पर automatically create कर दी जाती है। फिर इसी **Unique Session ID** का use **existing session** data को retrieve करने लिए किया जाता है।
- फिर server पर एक temporary फाइल generate होती है जिसका नाम इसी Unique Session ID के आगे sess\_ prepend कर दिया जाता है।

## PHP \$\_SESSION To Access Session Data

PHP में **\$\_SESSION** Super Global Variable का use Session data को retrieve करने के लिए करते हैं। जब अभी हम Session में data रखते हैं तो **\$\_SESSION** एक **Associative Array** create करता है जिसमे key => value pair में data को store होता है।

## PHP Session Example

```
<?php
session_start();
if(isset($_SESSION['num']))
    $_SESSION['num'] +=1;
else
    $_SESSION['num'] = 1;
echo 'You have visited this page : ' . $_SESSION['num'] . ' times';
?>
```

## Destroying PHP Session

PHP में Session को destroy / finish करने के लिए हम predefined function **session\_unset()** और **session\_destroy()** use करते हैं। **session\_unset()** function पूरी तरह से Current Session Variables को unset / remove करता है। जबकि **session\_destroy()** Current Session finish करता है। और अगर आपको सिर्फ particular variable को Session से remove करना हो तो हम **unset()** function का use करते हैं।

```

<?php
session_start();
$_SESSION['session_start'] = date('h:i:s a');
$_SESSION['userid'] = 'unique_user_hash_id';
$_SESSION['user_email'] = 'user@gmail.com';

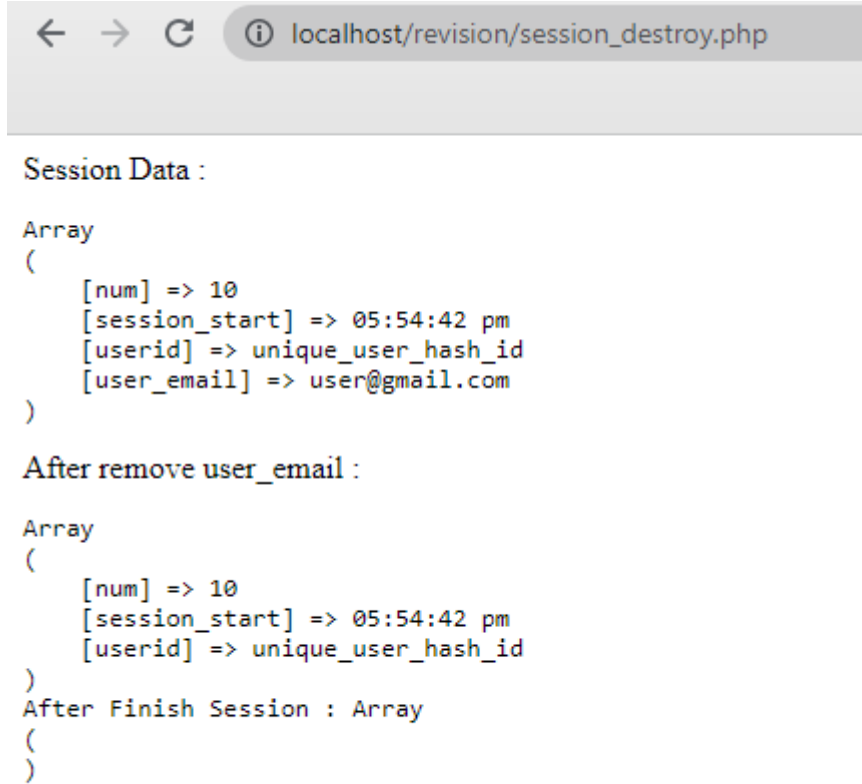
echo 'Session Data : <pre>';
print_r($_SESSION);
echo '</pre>';

/*now remove one value*/
unset($_SESSION['user_email']);
echo 'After remove user_email : <pre>';
print_r($_SESSION);

/* now completely finish session */
session_unset(); /* remove all session variables */
session_destroy(); /* destroy session*/
echo 'After Finish Session : ';
print_r($_SESSION);
?>

```

Output:-



```

Session Data :
Array
(
    [num] => 10
    [session_start] => 05:54:42 pm
    [userid] => unique_user_hash_id
    [user_email] => user@gmail.com
)

After remove user_email :
Array
(
    [num] => 10
    [session_start] => 05:54:42 pm
    [userid] => unique_user_hash_id
)
After Finish Session : Array
(
)

```

## PHP Session Functions

PHP हमें कुछ Important Session Functions Provide करती है जिनकी help से Current Session / New Session को और अच्छी तरह से use कर सकें और अपनी need के According modify कर सकें।

### PHP session\_start

**session\_start()** function existing session return करता है यदि session already start हो चुका है , यदि session start नहीं हुआ है तो new session start कर देता है ।

### PHP session\_status

**session\_status()** function current Session status return करता है , यह तीन तरह की value return करता है -

**PHP\_SESSION\_DISABLED or 0** if sessions are disabled.

**PHP\_SESSION\_NONE or 1** if sessions are enabled , but none exists.

**PHP\_SESSION\_ACTIVE or 2** if sessions are enabled, and one exists.

```
<?php
echo "Before Start Session : ".session_status( );
session_start( );
echo "After Start Session : ".session_status( );
?>
```

### Output

Before Start Session : 1  
After Start Session : 2

## PHP session\_name

**session\_name()** function का use current session का name return करता है , और अगर कोई new name pass किया गया है function call करते समय तो session\_name() pass किये गए name को update करके old session name return करता है।

## PHP session\_name

**session\_name()** function का use current session का name return करता है , और अगर कोई new name pass किया गया है function call करते समय तो session\_name() pass किये गए name को update करके old session name return करता है।

```
<?php
    echo "Current Session Name : ".session_name('MySession');
    echo "After Updtae : ".session_name();
?>
```

## Output

Current Session Name : PHPSESSID  
After Updtae : MySession

## PHP session\_id

**session\_id()** function current session ID get / update करता है , और अगर session start नहीं हुआ है तो empty string return करता है।



```
<?php
    session_start();
    echo "Current Session ID : ".session_id();
    session_id('mysessionid1234567890');
    echo "After Change Session ID : ".session_id();
?>
```

### Output

Current Session ID : mdq92a4rgi3trjbprh8djmc3bk  
After Change Session ID : mysessionid1234567890

### PHP session\_regenerate\_id

**session\_regenerate\_id()** function current session ID को new generated key से update करता है ।

```
<?php
    session_start();
    echo "Current Session ID : ".session_id();
    session_regenerate_id();
    echo "After Regenerate Session ID : ".session_id();
?>
```

### Output

Current Session ID : hvmgsc37cmljre6hmjht19hgdf  
After Regenerate Session ID : 5dt68i1ej9qqsh28vemtau83k9

## PHP session\_reset

**session\_reset()** function current session को original values के साथ reinitialize करता है , means अगर हम same key name के साथ session में value रख दें फिर **session\_reset()** function call करें तो हमें first value ही मिलेगी।

```
<?php
    session_start();
    $_SESSION["A"] = "Some Value";
    echo 'Before Reset : ' . $_SESSION["A"];

    $_SESSION["A"] = "New Value"; /* set new value */
    session_reset(); /* old session value restored */

    echo 'After Reset : ' . $_SESSION["A"];
?>
```

## Output

Before Reset : Some Value  
After Reset : Some Value

## PHP session\_unset

Function current session variables जो unset / free करता है , successfully free / unset होने पर **True** otherwise **False** return करता है।

```
<?php
    session_start();
    $_SESSION["A"] = "Some Value";
    $_SESSION["B"] = "Other Value";
    echo 'Before Unset : <pre>';
    print_r($_SESSION);

    session_unset();
    echo 'After Unset : ';
    print_r($_SESSION);
?>
```

## PHP session\_destroy

function current session को destroy करता है , हालाँकि यह session variables को unset / free नहीं करता है , इसलिए ध्यान रखें कि session destroy करना हो तो सबसे पहले **session\_unset()** फिर **session\_destroy()** function call करें।

## PHP Cookie

Cookie small piece of data होती है जो कि client / user browser पर store की जाती है। जब भी कोई user request send की जाती है तो cookie request के साथ embed होती है। Cookies को server द्वारा create , access या send किया जा सकता है।

client browser पर Cookies JavaScript के through भी create / access किया जा सकता है। Browser में set की गयी cookies को देखने के लिए आप Inspect Element (**Press F12**) करके **Storage: section** में **Cookies menu** में site listing में देख सकते हैं।

Client Browser में Cookie store करने का main reason end user को track करना होता है। जिससे उस information को future में use किया जा सके। Cookie का सबसे अच्छा example 'Remember Password' Mechanism है जिसमे किसी user के username , password cookie के रूप में browser में store कर दिया जाता है , और जब हम उस login पेज दुबारा visit करते हैं तो हमें username , password fields fill मिलते हैं।

## PHP Set Cookies

PHP में Cookie set करने के लिए predefined function **setcookie()** use करते हैं जिसमे required parameter , **key name** और **value** होते हैं। हालाँकि आप optional parameter के रूप में **time** भी set कर सकते है जीतनी देर तक cookie store करना चाहते हैं , दिया गया time expire होने पर cookie data भी expire हो जाती है।

## PHP setcookie() function

**setcookie()** function 7 parameter accept करता है

1. **string \$name | required** cookie name , जिस name से cookie data store करना है। इसी name से आप set की गयी value को access करेंगे।
2. **\$value | optional** वो value जो हमें cookie में set / update करनी है , अगर cookie को delete करना है तो आप **empty string** pass कर सकते हैं।
3. **\$expires\_or\_options | optional** यह expired time है जितने time तक cookie data **accessible** रहेगा। यहाँ हमें **UNIX timestamp** value provide करनी होती है। **by default** 0 set रहता है जिसका मतलब कि जब तक browser **open** है cookie को access कर सकते हैं। browser close हो जाने के बाद आप उस cookie को **access** नहीं कर पाएंगे।
4. **\$path | optional** वो path / URL जिस पर आपको cookie data access करना है। मतलब हम किसी **particular URL** के लिए भी cookie set कर सकते हैं। फिर cookie data सिर्फ उसी URL पर **accessible** होगा। अगर आप slash ( / ) set करते हैं तो cookie data प्रत्येक URL के लिए **accessible** होगा है।
5. **\$domain | optional** जिस domain के लिए cookie set करनी है , by default **current domain** ही set होता है। और अगर आप cookie data को सभी **subdomain** के लिए accessible रखना चाहते हैं तो **.example.com** करके set करना पड़ेगा।
6. **bool \$secure | optional** cookie data **http** या **https** से access होगा , by default **false** होता है मतलब data दोनों type से access होगा।
7. **bool \$httponly | optional** मतलब cookie को सिर्फ http request से access किया जा सकता या नहीं। by default **false** होता है जिसका मतलब है कि cookie data हर तरह की request के लिए **accessible** रहेगा। अगर true set है तो फिर data की script request like : **JS AJAX** request के लिए accessible नहीं होगा।

## PHP cookie Example

```
<?php
// 60 : 1 minute
// 60 * 60 : 1 hour
// 60 * 60 * 24 : 24 hours (1 day)
// 60 * 60 * 24 * 365 : 1 year
setcookie("key1", "Some Value");
setcookie("key2", "other Value", time()+3600); /* expire in 1 hour */
?>
```

## PHP Accessing Cookies

PHP में Cookies को access करने के लिए simply **\$\_COOKIE** Super Global Variables का use किया जाता है। किसी particular key check करने के लिए isset() function use कर सकते हैं।

```
<?php
setcookie('name', 'Bhuvanesh kumar');
if(isset($_COOKIE['name']))
    echo $_COOKIE['name'];
?>
```

**Output:-**

**Bhuvanesh kumar**

## PHP Deleting Cookies

PHP में Cookies delete करने के लिए **setcookie()** function को name argument के साथ expired time set कर दिया जाता है ।

```
<?php
/* expire 60 seconds ago */
setcookie('name', '', time()-60);
?>
```

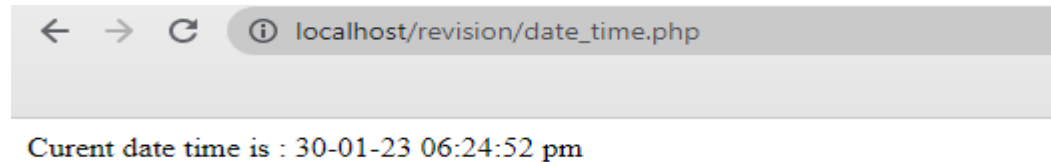
## PHP Date And Time

PHP date() function हमें readable form में timestamp (date with time) value return करता है according to parameter . जिसे हम अपनी need के according customize भी कर सकते हैं।

### PHP Date And Time Example

date() function का use करके current date और time print कराने की कोशिश करते हैं।

```
<?php
echo 'Curent date time is : '. date('d-m-y h:i:s a');
?>
```



**Output:-**

PHP में किस तरह से Date / Time को handle करते हैं।

Example:-

```
<?php
echo 'Curent Date : '. date('d/m/y');
echo 'Curent Date : '. date('D-M-Y');
echo 'Curent Date : '. date('d.M.Y');
echo 'Curent Date : '. date('d/m/y');
echo 'Curent Time : '. date('h:i:s a');
echo 'Curent Time : '. date('H:i:s A');
echo 'Curent Time : '. date('h:i:s A');
echo 'Curent Date Time : '. date('d/M/Y h:i:s A');
?>
```

### PHP strtotime Function

इसके अलावा आप predefined date को customize भी कर सकते हैं , इसके लिए PHP ने हमें **strtotime()** function provide किया है जिसकी help से हम दी गयी date / time को need के according get कर सकते हैं।

```
<?php
$date = '1990-11-23';
echo "Date : ".date('d/M/Y', strtotime($date));
echo "Day : ".date('d', strtotime($date));
echo "Day Name : ".date('D', strtotime($date));
?>
```



## PHP Date And Time Functions

PHP हमें ऐसे बहुत से predefined functions provide करती है जिनका use करके हम date / time को आसानी से handle कर सकते हैं।

### PHP date\_default\_timezone\_set Function

date\_default\_timezone\_set() function default timezone set करने के लिए उसे किया जाता है , आप अपने timezone के हिसाब से सेट कर सकते हैं।

```
<?php
    date_default_timezone_set('Asia/Kolkata');
    echo "Current Date Time : ".date("d/M/Y h:i:s A");
?>
```

**Output:-**

### PHP checkdate Function

checkdate() function check करता है कि दी गयी date valid है या नहीं।

```
<?php
    if(checkdate(12,31, 2303))
        echo "Date is Valid";
    else
        echo "Date is Not valid";
?>
```

**Output**

Date is Valid

## PHP strtotime Function

strtotime() function , किसी English textual date time को UNIX timestamp में convert करता है ।

```
<?php
$timestamp = strtotime('2020-12-12');
echo "Timestamp Format : $timestamp";
echo "After Change : ".date('Y-M-d', $timestamp);
/* we can also use now inside strtotime() function*/
echo "Current Date Time : " .date('d-M-Y h:i:s A', strtotime('now'));
?>
```

### Output

Timestamp Format : 1607711400

After Change : 2020-Dec-12

Current Date Time : 11-Sep-2020 06:15:19 PM

## PHP date\_default\_timezone\_get Function

date\_default\_timezone\_get() function , current default timezone return करता है।

```
<?php
echo date_default_timezone_get();
?>
```

Output:-

Current time zoon.

## PHP MYSQL Introduction

MySQL एक **RDBMS** (Relational Database System) है जो कि fast , reliable , flexible and use करने में बहुत ही आसान है, हालाँकि यह structured query language (**SQL**) पर based है। और SQL एक **standard language** है जिसका use database से data fetch करने के लिए किया जाता है।

MySQL records को **Tables & rows** में contain करता है। **attribute** से मिलकर एक **row** बनती है जिसे Entity भी कहते हैं , और Rows से मिलकर एक **Table** बनती है , और Tables से मिलकर **Database** बनता है। तो हम कह सकते हैं -

- Collection of attributes / columns - Row
- Collection of rows / entity set - Table
- Collection of tables - Database .

## PHP MySQL Connect

PHP में MySQL Database से 3 method से connect कर सकते हैं।

- 1.mysql procedural
- 2.mysql Object Oriented
- 3.PDO

main difference इनमे यही है कि PDO 12 different database driver support करता है जबकि mysqli सिर्फ MySQL को ही support करता है।

## PHP mysqli\_connect()

Database connect करने के लिए PHP Predefined Function **mysqli\_connect()** का use करते हैं।

Syntax :

```
mysqli_connect(hostname, username, password, database_name);
```

- hostname **required** होता है ,यह host name या IP Address हो सकता है ।
- username आपके database का username होता है , default **root** होता है।
- password आपके database का password होता है , अगर password set नहीं है तो इसे **blank** छोड़ दें।
- database\_name database का name होता है , हालाँकि यह **optional** होता है।

## PHP MySQL Create Or Delete Database

### PHP MySQL Create Database:-

```
<?php
```

```
$connection = mysqli_connect('localhost', 'root', null);
```

```
if(! $connection)
```

```
    die('Database connection error : '.mysqli_connect_error());
```

```
/* write query to create a new database */
```

```
$query = 'create database db_test';
```

```
if(mysqli_query($connection, $query))
```

```
    echo 'Database created successfully.';
```

```
else
```

```
    echo 'Error : '.mysqli_error($connection);
```

```
?>
```