Basics

Hello World

echo function is used to display or print output

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

Comments

Commets are used to make the code more understandable for programmer, they are not executed by compiler or interpreter.

One Liner

This is a singleline comment

```
// Twinkle Twinkle Little Star
```

Another One Liner

This is a single-line comment

```
# Chocolate dedo mujhe yaar
```

Multiline

This is a multiline comment

```
/* Code With

Harry */
```

Vardump

This function dumps information about one or more variables.

```
<?php var_dump(var1, var2, ...); ?>
```

Variables

Variables are "containers" for storing information.

Defining Variables

```
$Title = "Welcome to php";
```

String methods

Method	Use
strlen(string)	To get length of string

$str_word_count(string , return (0,1,2))	To get word count of a string.
strrev(string)	To reverse the string.
echo str_replace("world", "dolly", "hello world!"); // outputs hello dolly!	To replace the string.
str_shuffle(string)	To shuffle the string characters.
stripslashes(string)	To remove slashes.
trim(string);	To remove white spaces.
strtolower(string) strtoupper(string)	To convert into lower/upper case.
ucfirst(string)	To convert first character upper case.
lcfirst(string)	To convert first character lower case.
ucwords()	To convert first character of words to upper case
Strip_tags(string , tags to allow (optional)	To remove the html tags from the string.

Constant variable

define(variable_name, value, case-**insensitive** (true/false);

define("GREETING", "Welcome to W3Schools.com!",true);

echo GREETING;

Constant array

```
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
```

Arrays

```
$cars = array("Volvo", "BMW", "Toyota");

echo $cars[0];

echo count($cars);
```

Method	Use
array_keys($x)	Return all the keys of an array
array_push($array,value to push)	Insert one or more at the end of array
array_pop($array)	Delet one element at the end.
Search in array	
array_search(“term to search”,array)	Search a value in array
in_array(43,$a)	Return 1 if value in the array else 0 rerturn.
array_search(43 , $a)	Return index or key if exist.
Adding deleting values from array	
array_shift(x)	Remove first element from an array and return the removed values
array_unshift($array, value to add)	Add element at the start.
array_merge($fruit,$veg)	Merge 2 arrays.
array_slice(array, start ,end)	Removes and replaces specified elements of an array
array_unique($variable_name)	Remove duplicate values.
array_values($variable name)	Return all the values of array.
Sorting	
sort($variable name)	Sorts indexed array
rsort($array)	Sort index aray descending order.
asort($variable name)	Sort associative array in ascending order

arsort($variable name)	Sort an associative array descending order.
ksort , krsort(array)	Sort keys of associative array and values accordingly to keys.
array_multisort(array1 , array2)	Sort multiple arrays simulataneosuly.
$newarray=array_reverse(array);	Reverse the array
Other array methods	
count($variable name)	Return number of elements in
array_sum(array)	To get sum of array
array_product(array)	To get product of array
array_flip(array);	Change keys into values and values into keys.
$newaray=array_column(array,"col umn name");	To get one whole column from the multidimensional array.
array_chunk(array, size of pair to make); array_chunk(array, 2,true);	To make paired array we use this. True is for associative array to print its keys also.
array_rand(array , number of values to get)	To get randomly specified number of values from array.
arra_walk(array, function , parameter	To run a function for every value of array
array_map(function , array1,2,3)	Same work like above but it ill return something but above array_walk method does not.
current(array);	To get current pointer value.
key(array)	To get key of current pointer.
next(array)	To move forword pointer.
prev(array)	To move backword pointer
end(array)	To move pointer at end.
range(start , end , steps to jump);	To find get specified range numbers like 1 to 10 or a to h.
explode(separator , string ,limit)	To convert string to array

Function square($n){

Return n$*$n;

}

$a=[1,2,3,4,5,];

$newArray=array_**map**(‘square’,$a);

Print_r($newArray);

Associative Array

```
$associative=[

    "bill"=>12,

    "joe"=>20,

    "peter"=>30

];          $associative.bill;  or $associative[‘property’]
```

Super global variables

Php has following super global variables.

$GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script.

Example:

```
$x = 75;

$y = 25;


function addition() {

  $GLOBALS['z'] = $GLOBALS['x'] +
$GLOBALS['y'];

}

addition();

echo $z;
```

$_SERVER:

Element/Code	**Description**
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script Better to use htmlspecialchars() method for php_self method <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?> ">
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)

$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

$_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

```
<html>

<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

if ($_SERVER["REQUEST_METHOD"] == "POST") {

$name = $_POST['fname'];

    if (empty($name)) {

    echo "Please Enter your name";

    } else {

    echo $name;

    }

}

?>

</body>

</html>
```

$_GET

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

```
<?php

echo "Hello" . $_GET[“name”];

    ?>
```

$_REQUEST

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

```
<html>

<body>
```

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

if ($_SERVER["REQUEST_METHOD"] == "POST") {

$name = $_REQUEST['fname'];

if (empty($name)) {

echo "Name is empty";

} else {

echo $name;

}
```

```
}

?>

</body>

</html>
```

```
    $_FILES

    if(isset($_FILES["image"])){

       print_r($_FILES);

       $name=$_FILES['image']['name'];

       $size=$_FILES['image']['size'];

       $temp_name=$_FILES['image']['tmp_name'];

       $type=$_FILES['image']['type'];



       // syntax

       // move_uploaded_file(temp_file_name ,
    destination);

       move_uploaded_file($temp_name ,
    "upload/images.$name");
```

}

$_COOKIE:

Create cookie:

setCookie(name,value, expire,path(/),

domain,secure (true/false (for http and https check)),httponly(true/false to get value of cookie in js make it true);

$name="zain";

$value="Ali";

setcookie(name,value,time()+(86400*30),"/
(to get it in all pages)";

View cookie name:

$cookie[name];   // output= Ali

To delete the cookie

setcookie(name,"",time()-(86400*30))";

$_SESSION:

Session_start();

$_SESSION[name]=value; //set session name and value.

Echo $_SESSION[name];   //get session value and print it.

Delete session:

session_unset();   // remove all session variables

session_destroy(); //destroy session

Encryption of data

Method

md5(String , raw)
true=raw 16 character binary format
false=default 32 character hex number

sha1(String , raw (true/false)

true=raw 20 character binary format
 	
false=default 40 character hex number

bin2hex($str);
hex2bin(hexadecimal string);

Math functions

Method	**Use**
pi();	Return Pi value
ceil(x); floor(x);	Rounds a number upwards/ lower to the nearest integer, and returns the
exp(x)	Returns the value of E^x.
log(x)	Returns the logarithmic value of x
pow(x,y)	Returns the value of x to the power y.
rand(lower limit, higher limit)	Returns a random number between 0 and 1.
sqrt(x)	Returns the square root of a number x
abs(x)	Returns the absolute value of x
cbrt(x)	Returns the cube root of x
cos(x) sin(x) tan(x)	Returns the cos/sin/tan of x
log(x)	Returns the log of value x
pow(whose power, how much power) pow(8,2)	Returns the power of x on y
round(x)	Returns the rounded value of x
cosh(x) sinh(x) tanh(x)	Returns the cosh /sinh /tanh of x
min / max(x , y ,z…)	Returns the maximum / minimum

filter_var(variable ,filter name, options/flag);

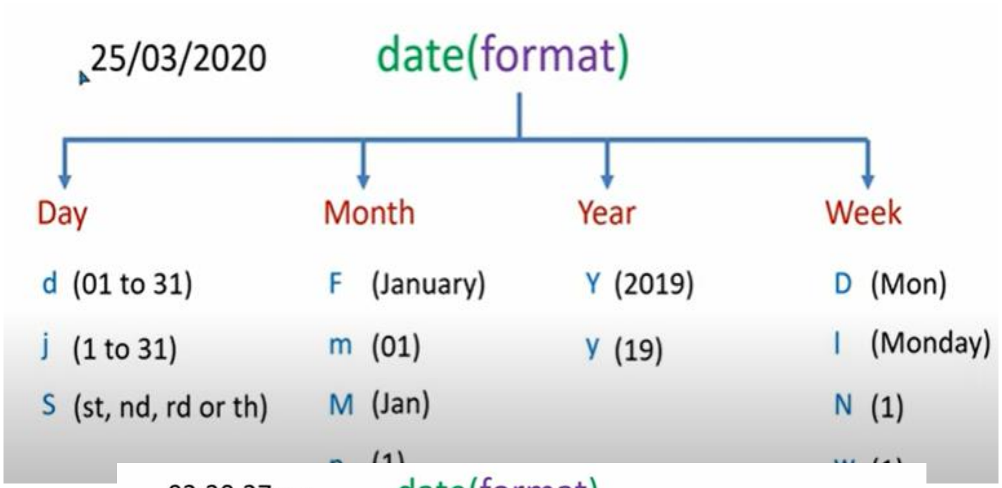
Different filters
FILTER_VALIDATE_INT
FILTER_VALIDATE_FLOAT
FILTER_VALIDATE_BOOLEAN
FILTER_VALIDATE_EMAIL
FILTER_VALIDATE_URL
FILTER_VALIDATE_IP
FILTER_VALIDATE_MAC
FILTER_VALIDATE_REGEXP

Sanitization	Use
FILTER_SANITIZE_EMAIL	Removes all illegal characters from an e-mail address
FILTER_SANITIZE_ENCODED	Removes/Encodes special characters
FILTER_SANITIZE_MAGIC_QUOTES	Apply addslashes()
FILTER_SANITIZE_NUMBER_FLOAT	Remove all characters, except digits, +- signs, and optionally .,eE
FILTER_SANITIZE_NUMBER_INT	Removes all characters except digits and + - signs
FILTER_SANITIZE_SPECIAL_CHARS	Removes special characters
FILTER_SANITIZE_STRING	Removes tags/special characters from a string
FILTER_SANITIZE_STRIPPED	Alias of FILTER_SANITIZE_STRING
FILTER_SANITIZE_URL	Removes all illegal character from a URL

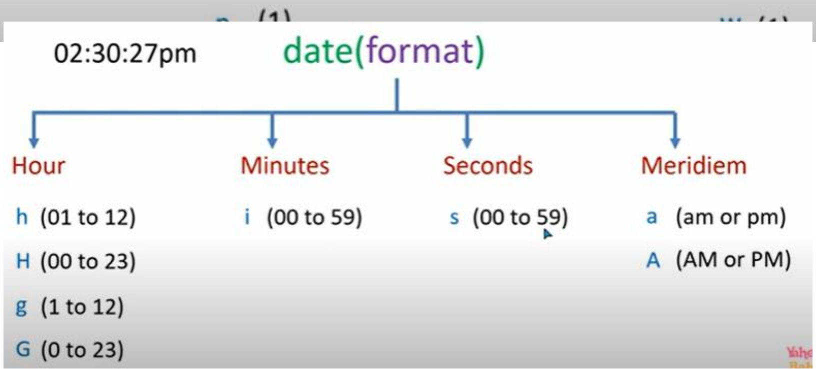
Flag	Use
FILTER_FLAG_ALLOW_OCTAL	Only for inputs that starts with a zero (0) as octal numbers. This only allows the succeeding digits to be 0-7
FILTER_FLAG_ALLOW_HEX	Only for inputs that starts with 0x/0X as hexadecimal numbers. This only allows succeeding characters to be a-fA-F0-9
FILTER_FLAG_STRIP_LOW	Strip characters with ASCII value lower than 32
FILTER_FLAG_STRIP_HIGH	Strip characters with ASCII value greater than 127
FILTER_FLAG_ENCODE_LOW	Encode characters with ASCII value lower than 32
FILTER_FLAG_ENCODE_HIGH	Encode characters with ASCII value greater than 127
FILTER_FLAG_ENCODE_AMP	Encode &
FILTER_FLAG_NO_ENCODE_QUOTES	Do not encode ' and "
FILTER_FLAG_EMPTY_STRING_NULL	Not in use
FILTER_FLAG_ALLOW_FRACTION	Allows a period (.) as a fractional separator in numbers
FILTER_FLAG_ALLOW_THOUSAND	Allows a comma (,) as a thousands separator in numbers
FILTER_FLAG_ALLOW_SCIENTIFIC	Allows an e or E for scientific notation in numbers
FILTER_FLAG_PATH_REQUIRED	The URL must contain a path part
FILTER_FLAG_QUERY_REQUIRED	The URL must contain a query string

FILTER_FLAG_IPV4	Allows the IP address to be in IPv4 format
FILTER_FLAG_IPV6	Allows the IP address to be in IPv6 format
FILTER_FLAG_NO_RES_RANGE	Fails validation for the reserved IPv4 ranges: 0.0.0.0/8, 169.254.0.0/16, 127.0.0.0/8 and 240.0.0.0/4, and for the reserved IPv6 ranges: ::1/128, ::/128, ::ffff:0:0/96 and fe80::/10
FILTER_FLAG_NO_PRIV_RANGE	Fails validation for the private IPv4 ranges: 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16, and for the IPv6 addresses starting with FD or FC
FILTER_FLAG_EMAIL_UNICODE	Allows the local part of the email address to contain Unicode characters

Date method



Time function:



Mktime(hour , minute , second , month , day , year);

Gmmktime(similar like above.) but sets according to Greenwich mean ti

Checkdate(month , day , year);

Date_add(date , interval (days to add);

Date_sub(date , interval)Date_difference(datetime1, datetime2 , absolute(true/false)

Date_add($date , date_inter_create_from_date_string(“30 days”);

date_modify($date, “-10 days”);

Date_format(date, “formate”);

$date=getDate();

$date[‘key]

$day=Gettimeofday();

Echo $day[key]

Date_parse(date);

Strtotime(string (like now ,next Monday or any date in string));

Strtotime(“2 week 3 days 7 hours 5 seconds”)

Date_sunrise(date); date_sunset(date);

Date_time_set(date, hour.minute,second)

Time zone functions

Date_default_timzone_get();	To get time of server.
Date_default_timzone_set();	To change the timezone of server.
Timezone_open();	After chaninging, to open the timezone.
Timezone_name_get()	To get the name of timezone.
Timezone_location_get()	To get the location of time zone.
Timezone_identifiers_list();	To get the list list of timezones.

Offer end code in php

```
$d1=strtotime("August 20");

$d2=ceil( ( $d1-time() ) / 60/ 60/ 24);

Echo $d2;
```

sFile handling

readfile(“file name/file path”);

File System

Modes	**Description**
R	**Open a file for read only**. File pointer starts at the beginning of the file
W	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
A	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
X	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

Function	**Description**
basename()	Returns the filename component of a path
chgrp()	Changes the file group
chmod()	Changes the file mode

chown()	Changes the file owner
clearstatcache()	Clears the file status cache
copy()	Copies a file
delete()	See unlink()
dirname()	Returns the directory name component of a path
disk_free_space()	Returns the free space of a filesystem or disk
disk_total_space()	Returns the total size of a filesystem or disk
diskfreespace()	Alias of disk_free_space()
fclose()	Closes an open file
feof()	Checks if the "end-of-file" (EOF) has been reached for an oper
fflush()	Flushes buffered output to an open file
fgetc()	Returns a single character from an open file
fgetcsv()	Returns a line from an open CSV file
fgets()	Returns a line from an open file
file()	Reads a file into an array
file_exists()	Checks whether or not a file or directory exists
file_get_contents()	Reads a file into a string
file_put_contents()	Writes data to a file
fileatime()	Returns the last access time of a file

filectime()	Returns the last change time of a file
filegroup()	Returns the group ID of a file
fileinode()	Returns the inode number of a file
filemtime()	Returns the last modification time of a file
fileowner()	Returns the user ID (owner) of a file
fileperms()	Returns the file's permissions
filesize()	Returns the file size
filetype()	Returns the file type
flock()	Locks or releases a file
fnmatch()	Matches a filename or string against a specified pattern
fopen()	Opens a file or URL
fpassthru()	Reads from the current position in a file - until EOF, and writes output buffer
fputcsv()	Formats a line as CSV and writes it to an open file
fputs()	Alias of fwrite()
fread()	Reads from an open file (binary-safe)
fscanf()	Parses input from an open file according to a specified format
fseek()	Seeks in an open file (move cursor position)
fstat()	Returns information about an open file
ftell()	Returns the current position in an open file

ftruncate()	Truncates (delete content) an open file to a specified length
fwrite($file ,"text")	Writes to an open file (binary-safe)
glob()	Returns an array of filenames / directories matching a specifie
is_dir()	Checks whether a file is a directory
is_executable()	Checks whether a file is executable (executable=.exe)
is_file()	Checks whether a file is a regular file
is_link()	Checks whether a file is a link
is_readable()	Checks whether a file is readable
is_uploaded_file()	Checks whether a file was uploaded via HTTP POST
is_writable()	Checks whether a file is writable
is_writeable()	Alias of is_writable()
lchgrp()	Changes the group ownership of a symbolic link
lchown()	Changes the user ownership of a symbolic link
link()	Creates a hard link
linkinfo()	Returns information about a hard link
lstat()	Returns information about a file or symbolic link
mkdir()	Creates a directory
move_uploaded_file()	Moves an uploaded file to a new location
parse_ini_file()	Parses a configuration file

parse_ini_string()	Parses a configuration string
pathinfo()	Returns information about a file path
pclose()	Closes a pipe opened by popen()
popen()	Opens a pipe
readfile()	Reads a file and writes it to the output buffer
readlink()	Returns the target of a symbolic link
realpath()	Returns the absolute pathname
realpath_cache_get()	Returns realpath cache entries
realpath_cache_size()	Returns realpath cache size
rename()	Renames a file or directory
rewind()	Rewinds a file pointer
rmdir()	Removes an empty directory
set_file_buffer()	Alias of stream_set_write_buffer(). Sets the buffer size for wri given file
stat()	Returns information about a file
symlink()	Creates a symbolic link
tempnam()	Creates a unique temporary file
tmpfile()	Creates a unique temporary file
touch()	Sets access and modification time of a file
umask()	Changes file permissions for files

unlink()	Deletes a file

Operators

Operators are symbols that tell the compiler or interpreter to perform specific mathematical or logical manipulations. These are of several types.

Modulus

The remainder of $x divided by $y

$x % $y

Exponentiation

Result of raising $x to the $y'th power

$x ** $y

PHP Comparison Operators

Not equal

Returns true if $x is not equal to $y

$x != $y

Not equal

Returns true if $x is not equal to $y

$x <> $y

PHP Logical Operators

And

True if both $x and $y are true

$x and $y

Or

True if either $x or $y is true

$x or $y

Xor

True if either $x or $y is true, but not both

$x xor $y

And

True if both $x and $y are true

```
$x && $y
```

Or

True if either $x or $y is true

```
$x || $y
```

Not

True if $x is not true

```
!$x
```

Operators

Ternary

```
$x = expr1 ? true statement : false statement
```

Conditional Statements

If..Elseif..Else

It executes different codes for more than two conditions

```
if (condition) {
// code to execute if condition is met
}
elseif (condition) {
// code to execute if this condition is met

} else {
// code to execute if none of the conditions are met
}
```

Switch Statement

It allows a variable to be tested for equality against a list of values (cases).

```
switch (n) {
case x:
code to execute if n=x;
break;
case y:

code to execute if n=y;

break;

// add more cases as needed

default:

code to execute if n is neither of the above;

}
```

Loops

Iterative statements or Loops facilitate programmers to execute any block of code lines repeatedly.

For Loop

It is used to iterate the statements several times. It is frequently used to traverse the data structures like the array and linked list.

```
for (starting counter value; ending counter value; increment by which  to increase) {

// code to execute goes here

}
```

Foreach Loop

The foreach loop loops through a block of code for each element in an array

```
foreach ($InsertYourArrayName as $value) {

// code to execute goes here

}     example

    $colors = array("red", "green", "blue", "yellow");



    foreach ($colors as $value) {

      echo "$value <br>";

    }
```

While Loop

It iterate the block of code as long as a specified condition is True or vice versa

```
while (condition that must apply) {

// code to execute goes here

}
```

Do-While Loop

This loop is very similar to the while loop with one difference, i.e., the body of the do-while loop is executed at least once even if the condition is False. It is an exit-controlled loop.

```
do {

// code to execute goes here;

} while (condition that must apply);
```

Variable-handling Functions

The PHP variable handling functions are part of the PHP core. No installation is required to use these functions.

isset

It is used to check whether a variable is empty. It also checks whether the variable is set/declared:

```
<?php

$x = 0;

// True because $x is set

if (isset($x)) {

echo "Variable 'x' is set";

}
```

unset

It unsets variables.

```
<?php

$a = "hello world!";

echo "The value of 'a' before unset: " . $a ;

unset($a);

echo "The value of 'a' after unset: " . $a;

?>
```

debug_zval_dump

debug_zval_dump is used to dump a string representation of an internal zval structure to output

```
<?php

$var1 = 'Hello';

$var1 .= ' World';
```

```
$var2 = $var1;

debug_zval_dump($var1);

?>
```

empty

Empty is used to check whether a variable is empty or not.

```
<?php

$var = 0;

// Evaluates to true because $var is empty

if (empty($var)) {

echo '$var is either 0, empty, or not set at all';

}

// Evaluates as true because $var is set

if (isset($var)) {

echo '$var is set even though it is empty';

}

?>
```

floatval

It returns the float value of different variables:

```
<?php

$var = '122.34343The';

$float_value_of_var = floatval($var);

echo $float_value_of_var; // 122.34343

?>
```

get_defined_vars

It returns all defined variables, as an array:

```
<?php

$b = array(1, 1, 2, 3, 5, 8);

$arr = get_defined_vars();

// print $b

print_r($arr["b"]);
```

```
/* print path to the PHP interpreter (if used as a CGI)  * e.g.
/usr/local/bin/php */

echo $arr["_"];

// print the command-line parameters if any

print_r($arr["argv"]);

// print all the server vars

print_r($arr["_SERVER"]);

// print all the available keys for the arrays of variables
print_r(array_keys(get_defined_vars()));

?>
```

gettype

It returns the type of different variables:

```
<?php

$a = 3;

echo gettype($a) ;

?>
```

intval

It returns the integer value of different variables:

```
<?php

echo intval(42); ?>
```

is_array

To check whether a variable is an array or not:

```
<?php

$a = "Hello";

echo "a is " . is_array($a) ;?>
```

OOP

Why OOP:

- code becomes more reuseable
- well organized code.
- Easier to debug.
- Best for medium and large size projects.

Classes

A class is a template for Classes

```
<?php
class car{
// code goes here...
}
?>
```

Object

An object is an instance of the class which make classes more specific.

```
$myBike = new Bike("red", "Honda");
```

Constructors / destructor

```
class Bike {
$color;
$model;
    public function __construct($color, $model) {
            $this->color = $color;
            $this->model = $model;
    }

       // destruct function call at the end of script
     function __destruct()  {
       echo "The fruit is {$this->name}.";
     }
    }

    $myBike = new Bike("red", "Honda");
echo $myBike -> message();
```

Access modifiers

Modifier	Access

private.	It can not be get able from any anywhere. To access it we have to use getter setters
Protected	To get one class variable into another but not allowed to use it from any other place
Public	To access variable function from any where

Functions

A function is a block of statements that can be used repeatedly in a program

Defining Functions

```
function NameOfTheFunction() {
//place PHP code here
}
```

Over riding function

2 functions with same name but parameters different, called overriding.

Static methods

```
class greeting {
        public static function welcome() {
          echo "Hello World!";
        }
}


                // Call static method
 greeting::welcome();
```

Calling one static function into another using self

```
      class greeting {
        public static function welcome() {
          echo "Hello World!";
        }
        public function __construct() {
          self::welcome();
        }
      }
      new greeting();
```

To call static variable of parent class into subclass

```
class pi {
  public static $value=3.14159;
}

class x extends pi {
  public function xStatic() {
    return parent::$value;
  }
}
```

// inheritance

```
Class Fruits extend vegetable{
}
```

Interfaces

```
interface A {
       function declaration like function sum();
}
Public Class B implements A,C{
       Function definition;
}
```

Abstraction

It is use to achieve privacy that we can not allow user to make object of that class directly. We have to make object of subclass to get it. Also we can not make function definition in that class. We only make function declaration and define it in subclass.

```
Abstract class a{

Function sum($a , $b);

}

Class b extends a{

    Function sum($a , $b){

    Return a+b;

    } }
```

Traits

To use one common function in different classes we use traits.

```
Trait trait_name{

    Function sum(){

            }

}
```

To access it in any class, we use the word “use” and trait name.

Use trait_name;

- If we have 2 classes and calling trait, first priority will be given to function of that sub class.
- If using trait in that subclass, then trait function will call.
- If there is no function here then super class function run. If not function defined there.

NameSpace

to use 2 classes with same name in one file is not possible. For this we have to use namespace.

Syntax:

Namespace namespace_name;

First.php	Second.php
Namespace first;	Namespace second;

Class test{
 }
	Class test{
	 }

Calling 2 classes in one class.
Require first.php;
$obj=new namespace_name \ class_name(); $obj=new first/test();

Method chaining

if we have multiple functions and we want to call them all then we can use this method chaining technique.

$obj->first()->second()->third();

For this we have to write " return $this" at the end of every function definition

__Autoload function

This function automatically run and use to include classes in the file.

```
Function __autoload($this){

Require "classes/".$this.".php";

}

$obj1=new first();

$obj2=new second();
```

__get function.

This function is use to get private variable values.This function automatically run when we access any private variable outside the class.

```
Function __get($property){

If (property_exsits(($property)){

Return $this->name;

} }

$obj=new a();

Echo "$obj->name";     // if we don't define __get function , this statement will give us fetal error.
```

__set function

This function run when we try to set value of any attribute of the class.

```
Function __set(property, $value){

If (property_exists($property){

        $this->property=$value;

} }
```

Method	Use
Get_class()	To getr class name
get_parent_class()	To get parent class name
get_class_methods()	To get class methods
get_class_vars()	To get class variables
get_object_vars()	To get object variables
get_called_class()	To get class name of which called
get_declared_classes()	To get declared classes
get_declared_interfaces()	To get interfaces
get_declared_traits()	To get traits
Class_alias	To get class aliases.

Data base

Method	**Use**
Mysqli (serverName, user name, password ,database);	To create connection with data base
Con->connect_error	To show error if not connecting to database.
Con->query(sql)	To execute query
Result->num_rows	For select query to check if data obtain or not.
$Row=Result->fetch_assoc() Getting with column $row [“column name ”]	To get data from the data base.
Con->close()	To Close connection

Constraints

```
CREATE TABLE Orders (

   OrderID int NOT NULL,

   OrderNumber int NOT NULL,

   PersonID int unique,

   PRIMARY KEY (OrderID),

   FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

);
```

Example:

```
$servername = "localhost";

$username = "username";

$password = "password";

// Create connection

$conn = new mysqli($servername, $username, $password);

// Check connection

if ($conn->connect_error) {

  die("Connection failed: " . $conn->connect_error);

}

// Create database

$sql = "CREATE DATABASE myDB";

if ($conn->query($sql) === TRUE) {

  echo "Database created successfully";

} else {

  echo "Error creating database: " . $conn->error;

} $conn->close();
```

Creating table:

```
// sql to create table

$sql = "CREATE TABLE MyGuests (

id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(30) NOT NULL,

email VARCHAR(50),

reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP

)";


if ($conn->query($sql) === TRUE) {

  echo "Table MyGuests created successfully";

} else {

  echo "Error creating table: " . $conn->error;
}

Insert query

$sql = "INSERT INTO table_name VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {

  echo "New record created successfully";

} else {

  echo "Error: " . $sql . "<br>" . $conn->error;

}
```

Inserting multiple records at a time

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";


$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";


$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
  echo "New records created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}
```

Inserting using prepared statements in mysqli

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?,
?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// Sss=this shows that parameters will be string

// For double values=d , int=i , blob for picture.

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
```

```
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
```

Select query and result->num_rows and result->fetch_assoc()

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
```

Alter command

it is use to modify existing table columns.

ALTER TABLE `movies` CHANGE `id` `movie_id` INT(11) NOT NULL AUTO_INCREMENT;

Select clause

Select * from table_name ;

Where clause:

SELECT column_name(s) FROM table_name WHERE column_name operator value

IN / NOT IN

Use is as similar as or operator

Select * from table_name where column_name in (condidtions);

Select * from students where age in (“18”, “20”);

Between / not between

Use to search values in between a range.

Select * from table_name where column_name first_condition AND second_condition;

Select * from students where age between 18 and 22;

Like operator:

Use where we want to get a value which start with specific character or end with specific character.

Select column name from table_name where column name like “pattern”;

Different patterns:

Patterns	Use
“a%”	Start with “a”
“%a”	End with “a”
“%am%”	Contain “am” at any position
“_a%”	Contain “ a “ at second position.
“__a%”	Contain “ a ” at third position.
“_ay”	Contain a at second and y at third position.

Select with Regular Expressions

Where column name regexp “expression”;

Expression:

Sign	pattern	Description
^	‘^ra’	Beginning of string
$	‘ans&’	End of string
[…]	‘[rms]’	Any character listed in brackets.
^[…]	^[rms]	Begin with any character listed in brackets.
[a-z],[0-9]	[a-h]e	Match with in range.
Patern1 | pattern 2	Ali | Awon | zain	Match any of pattern

Delete Date

DELETE FROM table_name WHERE some_column = some_value;

$sql = "DELETE FROM MyGuests WHERE id=3";

Update data

UPDATE table_name SET column1=value, column2=value2,...
WHERE some_column=some_value;

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

Limit function to get only limited records.

$sql = "SELECT * FROM Orders LIMIT 30";   //this will get 30 records only.

If we want to get records between 15 to 25 we can use offset method short hand.

$sql = "SELECT * FROM Orders LIMIT 15, 10";

Above will start from 16 and return until 10 records not completed.

Order By clause

Use to sort the data by specific column

SELECT column_name FROM table_name ORDER BY column_name(s) desc ;

// below one will sort the data according to column

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";

DISTINCT

To get different data.

Select distinct column_name from table_name;

Select distinct city from student;

is null / is not null

to get null and not null values. Syntax: Where column_name is null;

Having Clause

Where clause can not be used with aggregate function. So “ **having**“ clause can be used there.

SELECT COUNT(CustomerID), Country FROM Customers HAVING COUNT(CustomerID) > 5

Exist clause

The EXISTS operator is used to test for the existence of any record in a subquery.

Aggregate function:

select max("age") from dbo.customer;

select min("age") from dbo.customer;

select avg("age") from dbo.customer;

select sum("age") from dbo.customer;

select count("age") from dbo.customer;

Equi join

select ename,eadress,d.adress from employ e,dep d

where e.eid=d.did and

e.eadress=d.adress;

Natural join

It return common data in two tables show output

select * from employ e,dep d where e.eid=d.eid;

left outer join

it give matching rows and the rows which are in left table but not in right table. All left table data will come.

select * from emp left join dep on (emp.depNumber=dep.deptNumber)

Right outer join

it give matching rows and the rows which are in right table but not in left table. All right table data will show.

select * from emp right join dep on (emp.depNumber=dep.deptNumber);

Self join

Select * from student as t1 , student as t2 where t1.sid=t2.sid and t1.cid<> t2.cid;

MySQLi Functions

These functions allow you to access MySQL database server.

Method	Use
mysqli_connect($serverName, user name, password, database)	Use to connect with data base
mysqli_connect_error() ;	It shows the Error description for the connection error
Mysqli_query($con , $sql)	To execute query.
mysqli_affected_rows()	It returns the number of affected rows
mysqli_fetch_all()	It fetches all result rows as an array
mysqli_fetch_row()	It fetches one row from a result set and returns it as an enumerated array
Mysqli_fetch_assoc()	Fetch data as associative array
mysqli_kill()	It kills a MySQL thread
mysqli_close()	To close the connection