

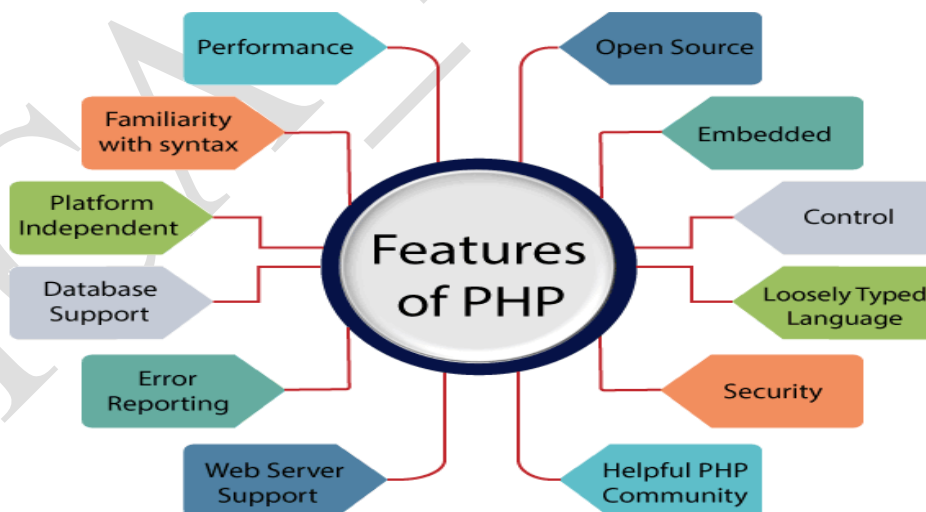
**INTRODUCTION:** PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).

PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995. **PHP 7.4.0** is the latest version of PHP, which was released on **28 November**. Some important points need to be noticed about PHP are as followed:

**Some facts about PHP:**

- PHP has many syntaxes similar to C, Java, and Perl, and has many unique features and specific functions.
- PHP page is a file with a .php extension can contain a combination of HTML Tags and PHP scripts.
- PHP is Server-side scripting language: Server-side scripting means that the PHP code is processed on the web server rather than the client machine.
- PHP supports many databases (MySQL and PHP combination is widely used).
- PHP is an open-source scripting language.
- PHP is an object-oriented language.
- PHP is free to download and use.
- PHP is simple and easy to learn language.

**CHARACTERISTICS OF PHP(OR)FEATURES OF PHP:** PHP is very popular language because of its simplicity and open source. There are some important features of PHP given below:



**1. Performance:** PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

- 2. Open Source:** PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.
- 3. Familiarity with syntax:** PHP has easily understandable syntax. Programmers are comfortable coding with it.
- 4. Embedded:** PHP code can be easily embedded within HTML tags and script.
- 5. Platform Independent:** PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- 6. Database Support:** PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
- 7. Error Reporting :** PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E\_ERROR, E\_WARNING, E\_STRICT, E\_PARSE.
- 8. Loosely Typed Language:** PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.
- 9. Web servers Support:** PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
- 10. Security:** PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threats and malicious attacks.
- 11. Control:** Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.
- 12. A Helpful PHP Community:** It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

**APPLICATIONS OF PHP:** PHP is one of the most widely used language over the web. The following are the major applications of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.

- It can encrypt data.

## PHP SYNTAX

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

### Basic PHP Syntax

A PHP script can be placed anywhere in the document. A PHP script starts with `<?php` and ends with `?>`

```
<?php
```

```
// PHP code goes here
```

```
?>
```

The default file extension for PHP files is ".php". A PHP file normally contains HTML tags, and some PHP scripting code. PHP statements end with a semicolon (;). Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

### Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>My first PHP page</h1>
```

```
  <?php
```

```
    echo "Hello World!";
```

```
  ?>
```

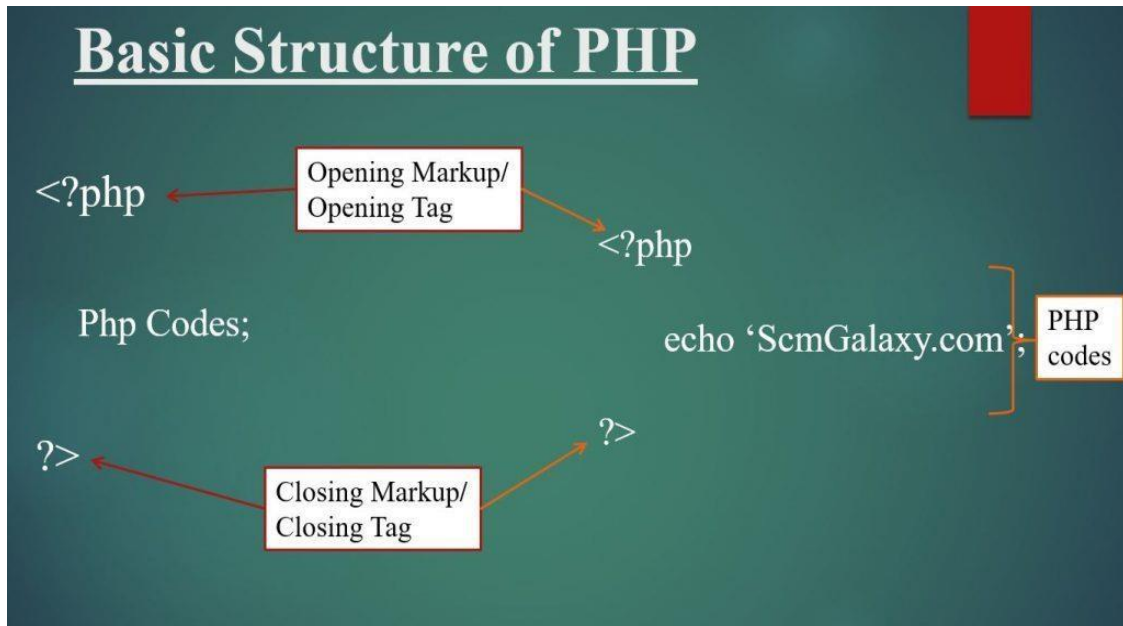
```
</body>
```

```
</html>
```

### Output: My first PHP page

Hello World!

**STRUCTURE OF PHP:** PHP is an embedded scripting language when used in Web pages. This means that PHP code is embedded in HTML code. You use HTML tags to enclose the PHP language that you embed in your HTML file — the same way that you would use other HTML tags. We create and edit Web pages containing PHP the same way that you create and edit regular HTML pages.

**Examples:-**

```
<!DOCTYPE html>

<html>
  <body>
    <h1>ScmGalaxy</h1>
    <?php
      echo 'Hello\n';
      print "World";
    ?>
  </body>
</html>
```

**2. Commenting**

# and // are used to comment out a single line of code, while /\* and \*/ indicate the start and end of a commented block of code.

**1. HTML tags**

There are HTML tags for PHP code to indicate the start and end of PHP code in an HTML file. Any one of the following 4 tags can be used:

1. `<?php php-code-here ?>`
2. `<SCRIPT LANGUAGE="php"> php-code-here </SCRIPT>`
3. `<? php-code-here ?>`
4. `<% php-code-here %>`

The first and second tags are the ones most recommended and most widely used. Using a tag which is rarely used may result in a web-server being unable to detect the start and end of the PHP code.

**VARIABLES:** Variables in a program are used to store some values or data that can be used later in a program.

In PHP, a variable is declared using a **\$ sign** followed by the variable name. Here, some important points to know about variables:

**Rules for declaring PHP variable:**

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- A variable must start with a dollar (\$) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, \_).
- A variable name must start with a letter or underscore (\_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so \$name and \$NAME both are treated as different variable.

**Syntax for creating variable:**

\$variablename=value;

**Example**

```
<?php
```

```
$txt = "Hello world!";
```

```
$x = 5;
```

```
$y = 10.5;
```

```
?>
```

**DATA TYPES:** PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (Predefined)
2. Compound Types (User-defined)
3. Special Types

**1. Scalar Types (Predefined):** It holds only single value. There are 4 scalar data types in PHP.

- a. boolean
  - b. integer
  - c. float
  - d. string
-

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

**Example:**

```
<?php
if (TRUE)
echo "This condition is TRUE.";
if (FALSE)
echo "This condition is FALSE.";
?>
```

**Output:**

This condition is TRUE.

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

**Rules for integer:**

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

**Example:**

```
<?php
$n1 = 10;
$n2 = 10;
$sum = $n1 + $n2;
echo "Addition of floating numbers: " . $sum;
?>
```

**Output:**

Addition of floating numbers: 20

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

**Example:**

```
<?php
```

```
$n1 = 19.34;  
$n2 = 54.472;  
$sum = $n1 + $n2;  
echo "Addition of floating numbers: " . $sum;  
?>
```

**Output:**

Addition of floating numbers: 73.812

**d. string:** A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within **single quotes** or in **double quotes**. But both are treated differently. To clarify this, see the example below:

**Example:**

```
<?php  
$college = "DNR";  
//both single and double quote statements will treat different  
echo "Hello $college";  
?>
```

**Output:**

Hello DNR

**2. Compound Types (User-defined):** It can hold multiple values. There are 2 compound data types in PHP.

- a) array
- b) object

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

**Example:**

```
<?php  
$season[0]="summer";  
$season[1]="winter";  
$season[2]="spring";  
$season[3]="autumn";
```

```
echo  
"Season  
are:  
$season[0]  
,  
$season[1]  
,  
$season[2]  
and  
$season[3]  
";  
?>
```



**output:**

Season are: summer, winter, spring and autumn

**b) object:** An object is an instance of a class. It's a central concept in object-oriented programming.

An object has properties. For example, a person object may have the first name, last name, and age properties.

An object also has behaviours, which are known as methods. For example, a person object can have a method called `getFullName()` that returns the full name.

**Example:**

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

**Output:**

Bike Model: Royal Enfield

**3. Special Types:** There are 2 special data types in PHP.

- a) resource
- b) NULL

**a) resource: resources:** Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource. Resource variables hold special handles for files and database connections.

**b) NULL:** Null is a special data type which can have only one value: NULL. If a variable is created without a value, it is automatically assigned a value of NULL. Variables can also be emptied by setting the value to NULL:

**Example:**

MCA - ATMECE

```
<?php
$nm = NULL;
echo $nm; // this will return no output
// return data type
var_dump($nm);
?>
```

**Output**

NULL

**PHP OPERATORS:** Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values.

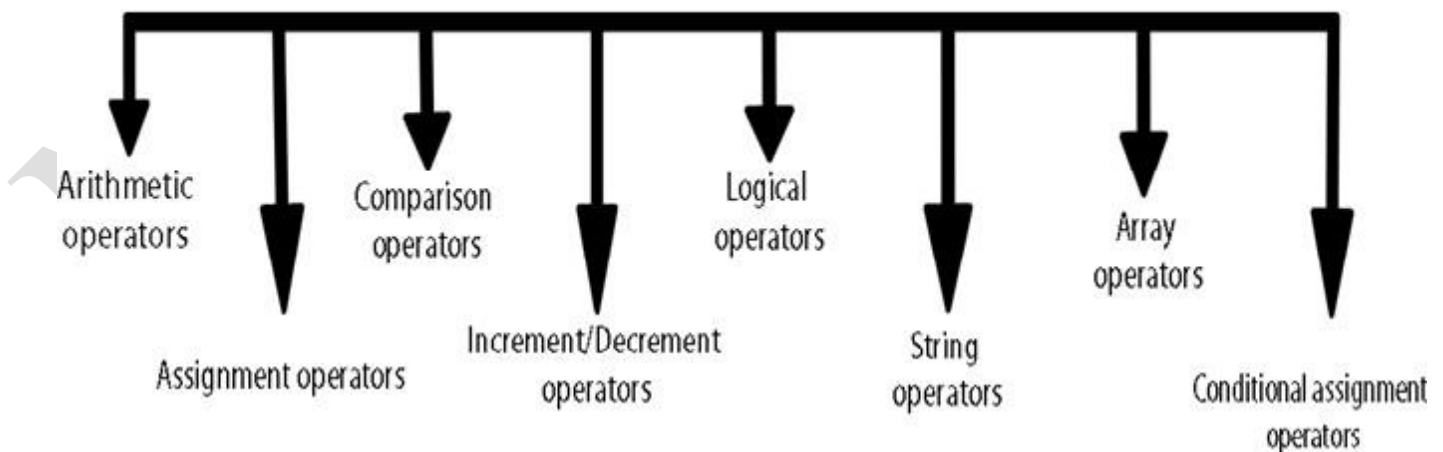
**For example:**

\$num=10+20;//+ is the operator and 10,20 are operands

PHP divides the operators in the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators
7. Array operators
8. Conditional assignment operators

## PHP Operators



## 1. Arithmetic Operators:

The arithmetic operators are used to perform simple mathematical operations like addition, subtraction, multiplication, etc. Below is the list of arithmetic operators along with their syntax and operations in PHP.

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

Example:

```
<?php
$x = 4; // Variable 1
$y = 2; // Variable 2
// Some arithmetic operations on
// these two variables
echo ($x + $y), "\n";
echo($x - $y), "\n";
echo($x * $y), "\n";
echo($x / $y), "\n";
echo($x % $y), "\n";
echo( $x ** $y);
?>
```

**Output:**

```
6
2
8
```

2

0

16

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

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

**Example:**

```
<?php
$x = 10;
echo $x . "<br>"; // Outputs: 10
$x = 20;
$x += 30;
echo $x . "<br>"; // Outputs: 50
$x = 50;
$x -= 20;
echo $x . "<br>"; // Outputs: 30
$x = 5;
$x *= 25;
```

```
echo $x . "<br>"; // Outputs: 125
```

```
x = 50;
```

```
$x /= 10;
```

```
echo $x . "<br>"; // Outputs: 5
```

```
$x = 100;
```

```
$x %= 15;
```

```
echo $x . "<br>"; // Outputs: 10
```

**OUTPUT:**

```
10
```

```
50
```

```
30
```

```
125
```

```
5
```

```
10
```

**3. Comparison operators:** These operators are used to compare two elements and output the result in boolean form.

Operator	Name	Example	Result
==	Equal	\$x == \$y	Returns true if \$x is equal to \$y
===	Identical	\$x === \$y	Returns true if \$x is equal to \$y, and they are of the same type
!=	Not equal	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Not equal	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Not identical	\$x !== \$y	Returns true if \$x is not equal to \$y, or they are not of the same type

>	Greater than	$x > y$	Returns true if $x$ is greater than $y$
---	--------------	---------	---

<	Less than	$\$x < \$y$	Returns true if \$x is less than \$y
>=	Greater than or equal to	$\$x \geq \$y$	Returns true if \$x is greater than or equal to \$y
<=	Less than or equal to	$\$x \leq \$y$	Returns true if \$x is less than or equal to \$y
<=>	Spaceship	$\$x \lt=> \$y$	Returns an integer less than, equal to, or greater than zero, depending on if \$x is less than, equal to, or greater than \$y. Introduced in PHP 7.

**4. Increment/Decrement operators:** The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Description
<b>++\$x</b>	<b>Pre-increment</b>	<b>Increments \$x by one, then returns \$x</b>
<b>\$x++</b>	<b>Post-increment</b>	<b>Returns \$x, then increments \$x by one</b>
<b>--\$x</b>	<b>Pre-decrement</b>	<b>Decrements \$x by one, then returns \$x</b>
<b>\$x--</b>	<b>Post-decrement</b>	<b>Returns \$x, then decrements \$x by one</b>

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

Operator	Name	Example	Result
and	And	$\$x \text{ and } \$y$	True if both \$x and \$y are true



or	Or	\$x or \$y	True if either \$x or \$y is true
xor	Xor	\$x xor \$y	True if either \$x or \$y is true, but not both
&&	And	\$x && \$y	True if both \$x and \$y are true
	Or	\$x    \$y	True if either \$x or \$y is true
!	Not	!\$x	True if \$x is not true

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

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

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

Operator	Name	Example	Result
+	Union	\$x + \$y	Union of \$x and \$y

MCA-ATMECE

==	Equality	\$x == \$y	Returns true if \$x and \$y have the same key/value pairs
===	Identity	\$x === \$y	Returns true if \$x and \$y have the same key/value pairs in the same order and of the same types
!=	Inequality	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Inequality	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Non-identity	\$x !== \$y	Returns true if \$x is not identical to \$y

**8. Conditional assignment operators:** The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	\$x = expr1 ? expr2 : expr3	Returns the value of \$x. The value of \$x is expr2 if expr1 = TRUE. The value of \$x is expr3 if expr1 = FALSE

### FLOW CONTROL FUNCTIONS IN PHP

**Flow Control Functions in PHP:** PHP supports a number of traditional programming constructs for controlling the flow of execution of a program. Control statements are conditional statements that execute a block of statements if the condition is correct. The statement inside the conditional block will not execute until the condition is satisfied.

**CONDITIONAL STATEMENTS:** PHP provides us with five conditional statements:

1. if statement
2. if...else statement

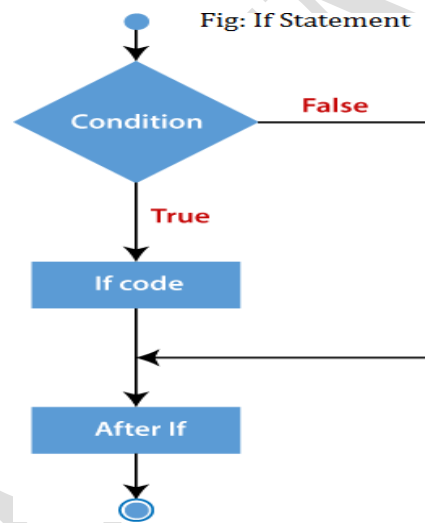
3. if...elseif...else statement
4. nested if
5. switch statement

**1. if statement:** If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

**Syntax:**

```
if(condition)
{
    //code to be executed
}
```

**Flowchart:**



**Example**

```
<?php
$num=12;
if($num<100)
{
```

```
        echo "$num is less than 100";  
    }  
    ?>
```

**Output:** 12 is less than 100

**2. if...else statement:** PHP if-else statement is executed whether condition is true or false.

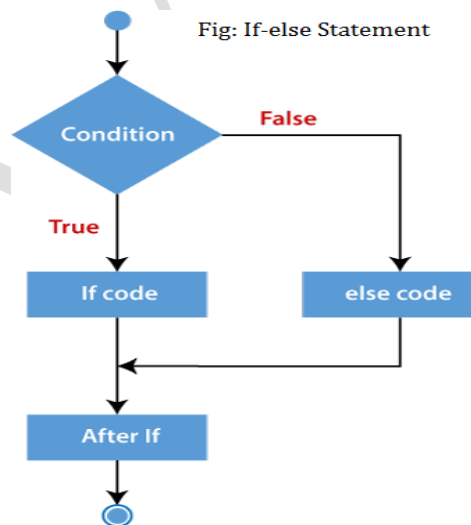
If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.

---

#### Syntax:

```
if(condition)  
{  
    //code to be executed if true  
}  
else  
{  
    //code to be executed if false  
}
```

#### Flowchart



#### Example:

```
<?php  
$num=12;  
if($num%2==0)
```

```
{  
    echo "$num is even number";  
}  
else  
{  
    echo "$num is odd number";  
}  
?>
```

---

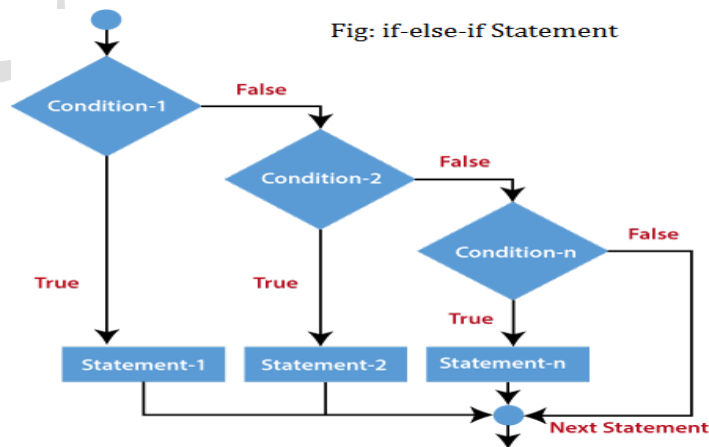
**Output:**

12 is even number

**3. If...else if...else statement:** The PHP if-else-if is a special statement used to combine multiple if else statements. So, we can check multiple conditions using this statement.

**Syntax:**

```
if (condition1)
{
    //code to be executed if condition1 is true
}
elseif (condition2)
{
    //code to be executed if condition2 is true
}
elseif (condition3)
{
    //code to be executed if condition3 is true
    ....
}
else
{
    //code to be executed if all given conditions are false
}
```

**Flowchart:**

MCA - ATMECE



**Example**

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

**Output:**

B Grade

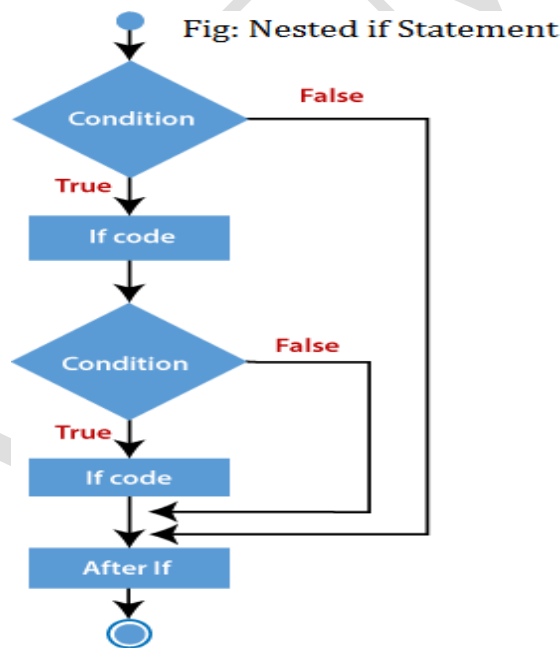
**4. Nested if:** The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is **true**.

**Syntax**

```

if (condition)
{
    //code to be executed if condition is true
}
if (condition)
{
    //code to be executed if condition is true
}
}

```

**Flowchart****Example**

```

<?php
    $age = 23;
    $nationality = "Indian";
    //applying conditions on nationality and age
    if ($nationality == "Indian")
    {

```

```
    if ($age >= 18)
    {
        echo "Eligible to give vote";
    }
    else
    {
        echo "Not eligible to give vote";
    }
}
?>
```

**Output:**

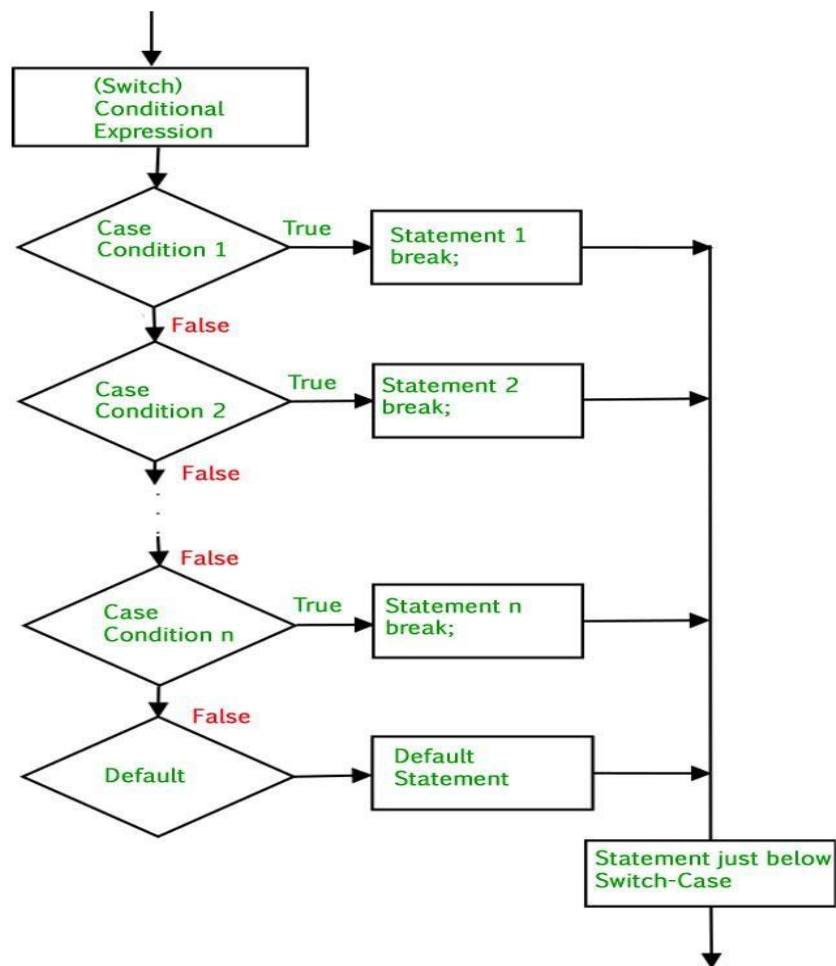
Eligible to give vote

**5. switch statement:** The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, **break** and **default**.

1. The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
2. The **default** statement contains the code that would execute if none of the cases match.

**Syntax:**

```
switch(expression)
{
    case value1:
        //code to be executed
        break;
    case value2:
        //code to be executed
        break;
    .....
    default:
        code to be executed if all cases are not matched;
}
```

**Flowchart:****Example:**

```
<?php
$favcolor = "red";
switch ($favcolor)
{
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
```

```
        echo "Your favorite color is green!";  
        break;  
    default:  
        echo "Your favorite color is neither red, blue, nor green!";  
    }  
?>
```

**LOOPING STATEMENTS:** Like any other language, loop in PHP is used to execute a statement or a block of statements, multiple times until and unless a specific condition is met. This helps the user to save both time and effort of writing the same code multiple times.

PHP supports four types of looping techniques;

1. for loop
2. while loop
3. do-while loop
4. foreach loop

**1. for loop:** PHP for loop can be used to execute same block of statements for the specified number of times. It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.

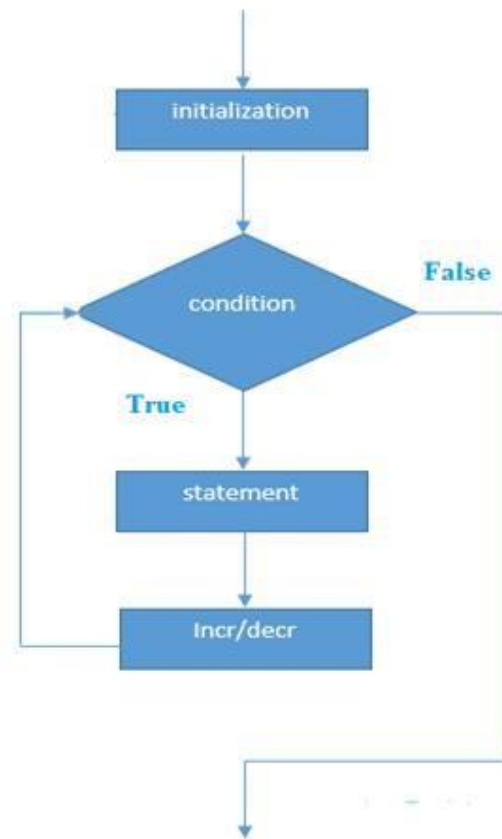
**Syntax:**

```
for (initialization expression; test condition; update expression)  
{  
    // code to be executed  
}
```

In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than counter value. If statement is true, then loop body is executed and loop variable gets updated. Steps are repeated till exit condition comes.

- **Initialization Expression:** In this expression we have to initialize the loop counter to some value. for example: \$num = 1;
- **Test Expression:** In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: \$num <= 10;
- **Update Expression:** After executing loop body this expression increments/decrements the loop variable by some value. for example: \$num += 2;

MCA - ATMECE

**Flowchat****Example:**

```
<?php
// code to illustrate for loop
for ($num = 1; $num <= 5; $num += 1)
{
    echo "$num \n";
}
?>
```

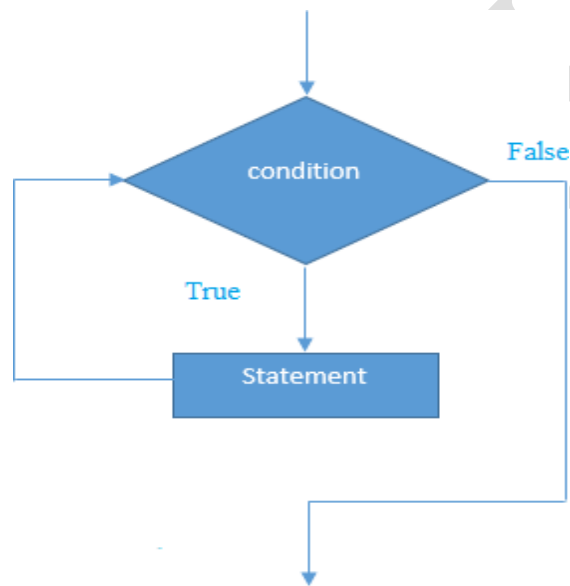
**Output:** 1  
2  
3  
4  
5

2. **while loop:** The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if it is true then it enters the loop and executes the block of statements until particular condition is false. Syntax:

while (if the condition is true)

```
{  
    // code is executed  
    Inc/dec;  
}
```

**Flowchart:**



**Example:**

```
<?php  
$n=1;  
while($n<=5)  
{  
  
}
```

1  
2  
3  
4  
5

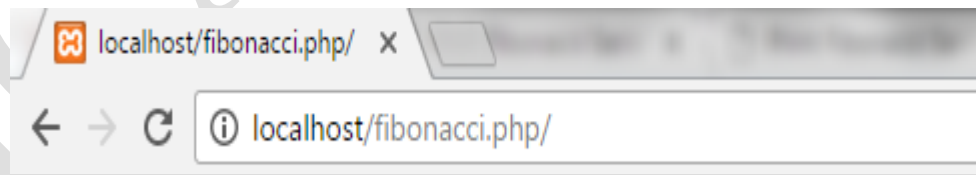
**Output:**



```
echo  
"$n<br/>";  
  
$n++;  
  
}
```

**Example: WRITE PHP PROGRAM TO DISPLAY FIBONACCI SERIES**

```
<!DOCTYPE html>
<html>
<body>
<?php
    $num = 0;
    $n1 = 0;
    $n2 = 1;
    echo "<h3>Fibonacci series for first 12 numbers: </h3>";
    echo "\n";
    echo $n1.' '.$n2.' ';
    while ($num < 10 )
    {
        $n3 = $n2 + $n1;
        echo $n3.' ';
        $n1 = $n2;
        $n2 = $n3;
        $num = $num + 1;
    }
?>
</body>
</html>
```

**Output:**

**Fibonacci series for first 12 numbers:**

0 1 1 2 3 5 8 13 21 34 55 89

MCA - ATMECE

**Example: WRITE PHP PROGRAM TO CHECK THE GIVEN NUMBER IS PALINDROME OR NOT**

```
<!DOCTYPE html>
<html>
<body>
<?php
    $num = 121;
    $sum = 0;
    $n=$num;
    while(floor($num))
    {
        $r = $num%10;
        $sum = $sum * 10 + $r;
        $num = $num/10;
    }
    if($n==$sum)
    {
        echo "$n is a Palindrome number.";
    }
    else
    {
        echo "$n is not a Palindrome number.";
    }
?>
</body>
</html>
```

**Output:** 121 is a Palindrome number.

3. **do-while loop:** This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do...while loop. After executing once, the program is executed as long as the condition holds true.

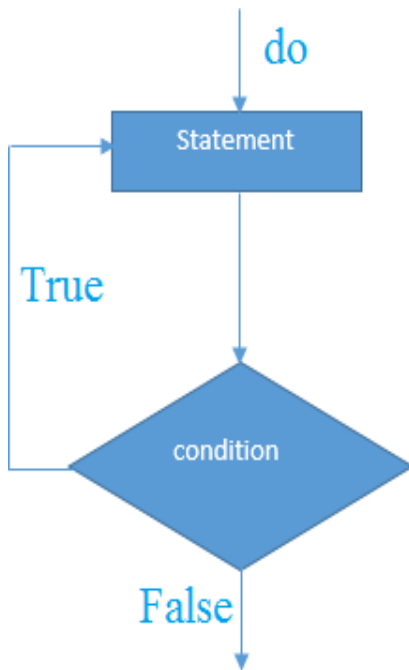
**Syntax:**

```
do
{
```

```
//code to be executed
```

```
}while(condition);
```

**Flowchart:**



**Example**

```
<?php
$n=1;
do
{
    echo "$n<br/>";
    $n++;
} while($n<=5);
?>
```

**Output:**

```
1
2
3
4
5
```

## WORKING WITH FUNCTIONS

**Function:** A function is a block of code written in a program to perform some specific task.. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

PHP provides us with two major types of functions:

- **Built-in functions** : PHP provides us with huge collection of built-in library functions. These functions are already coded and stored in form of functions. To use those we just need to call them as per our requirement like, `var_dump`, `fopen()`, `print_r()`, `gettype()` and so on.
- **User Defined Functions** : Apart from the built-in functions, PHP allows us to create our own customised functions called the user-defined functions.

Using this we can create our own packages of code and use it wherever necessary by simply calling it.

### Advantage of PHP Functions

**Code Reusability:** PHP functions are defined only once and can be invoked many times, like in other programming languages.

**Less Code:** It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

**Easy to understand:** PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

### Creating a Function

While creating a user defined function we need to keep few things in mind:

1. Any name ending with an open and closed parenthesis is a function.
2. A function name always begins with the keyword *function*.
3. To call a function we just need to write its name followed by the parenthesis
4. A function name cannot start with a number. It can start with an alphabet or underscore.
5. A function name is not case-sensitive.

### Syntax:

```
function function_name()
{
    executable code;
}
```

### PHP Functions Example:

```
<?php
function sayHello()
{
```

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

**Output:**

Hello PHP Function

**PHP FUNCTION ARGUMENTS:** We can pass the information in PHP function through arguments which is separated by comma. PHP supports 4 types of function arguments

1. Call by Value (default),
2. Call by Reference
3. Default argument values
4. Variable-length argument list.

**1. Call by Value:** Call by value means passing the value directly to a function. The called function uses the value in a local variable; any changes to it do not affect the source variable. Let's see the example to pass two argument in PHP function.

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

**Output:**

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

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

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

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

**Example:**

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

**Output:**

Hello Call By Reference

**3. Default Argument Value:** We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument.

Let's see a simple example of using default argument value in PHP function.

**Example:**

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

**Output:**

Hello Rajesh

Hello Sonoo

Hello John

**4. Returning Value**

Let's see an example of PHP function that returns value.

**Example:**

```
<?php
```



```
function cube($n)
{
    return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

**Output:**

Cube of 3 is: 27

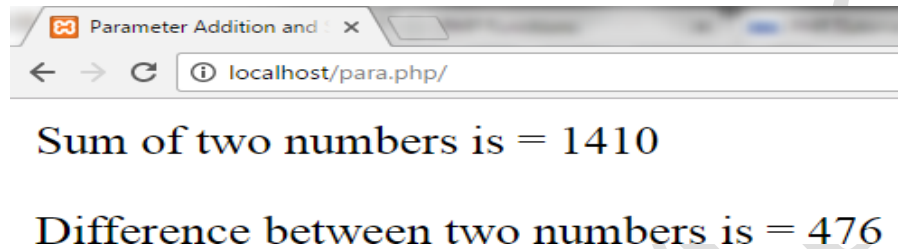
**PHP PARAMETERIZED FUNCTION:** PHP Parameterized functions are the functions with parameters. You can pass any number of parameters inside a function. These passed parameters act as variables inside your function. They are specified inside the parentheses, after the function name. The output depends upon the dynamic values passed as the parameters into the function.

**PHP Parameterized Example 1:****Addition and Subtraction**

In this example, we have passed two parameters **\$x** and **\$y** inside two functions **add()** and **sub()**.

```
<!DOCTYPE html>
<html>
<head>
    <title>Parameter Addition and Subtraction Example</title>
</head>
<body>
<?php
    //Adding two numbers
    function add($x, $y)
    {
        $sum = $x + $y;
        echo "Sum of two numbers is = $sum <br><br>";
    }
    add(467, 943);
    //Subtracting two numbers
    function sub($x, $y)
    {
        $diff = $x - $y;
```

```
        echo "Difference between two numbers is = $diff";
    }
    sub(943, 467);
?>
</body>
</html>
```

**Output:**

**TYPES OF VARIABLES:** In PHP, variables can be declared anywhere in the script. The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

**1. Local variable:** The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

**Example:**

```
<?php
function local_var()
{
    $num = 45; //local variable
    echo "Local variable declared inside the function is: ". $num;
}
local_var();
?>
```

**Output:**

Local variable declared inside the function is: 45

---

## 2. Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword

keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

Let's understand the global variables with the help of an example:

### Example:

```
<?php
$name = "DNR";    //Global Variable

function global_var()
{
    global $name;
    echo "Variable inside the
function: ". $name; echo
"</br>";
}
global_var();
echo "Variable outside the function: ". $name;
?>
```

### Output:

```
Variable inside
the function:
DNR Variable
outside the
function: DNR
```

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

### Example

```
<?php
function myTest()
{
    static
    $count = 0;
}
```

```
myTest();
```

```
myTest();
```

```
?> i
```

```
c
```

```
$
```

```
x
```

```
=
```

```
0
```

```
;
```

```
e
```

```
c
```

```
h
```

```
o
```

```
$
```

```
x
```

```
;
```

```
$
```

```
x
```

```
+
```

```
+
```

```
;
```

```
}
```

---

**Output:** 0

1

2

## ARRAYS

**Arrays:** An array is a collection of similar types of data. It is used to hold multiple values of similar type in a single variable.

### **Advantage of PHP Array:**

1. **Less Code:** We don't need to define multiple variables.
2. **Easy to traverse:** By the help of single loop, we can traverse all the elements of an array.
3. **Sorting:** We can sort the elements of array.

**TYPES OF ARRAYS IN PHP:** There are 3 types of array in PHP.

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

**1. Indexed Array:** PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.

PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as numeric array. There are two ways to define indexed array:

#### **1st way:**

##### **Example:**

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

#### **Output:**

Season are: summer, winter, spring and autumn

#### **2nd way:**

##### **Example:**

```
<?php
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

**Output:**

Season are: summer, winter, spring and autumn

**2. Associative Array:** PHP allows you to associate name/label with each array elements in PHP using => symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number. There are two ways to define associative array:

**1st way:**

```
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

**Example**

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

**Output:**

Sonoo salary: 550000

Vimal salary: 250000

Ratan salary: 200000

**2nd way:**

```
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
```

**Example:**

```
<?php
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

**Output:**

Sonoo salary: 550000

Vimal salary: 250000

Ratan salary: 200000

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

**Example:**

```
<?php
$emp = array
(
    array(1,"sonoo",400000),
    array(2,"john",500000),
    array(3,"rahul",300000)
);
for ($row = 0; $row < 3; $row++)
{
    for ($col = 0; $col < 3; $col++)
    {
        echo $emp[$row][$col]. " ";
    }
    echo "<br/>";
}
?>
```

**Output:**

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

## WORKING WITH OBJECTS

### PHP WITH OOPS CONCEPT

Object-oriented programming is a programming model organized around Object rather than the actions and data rather than logic.

**Class:** A class is an entity that determines how an object will behave and what the object will contain. In

other words, it is a blueprint or a set of instruction to build a specific type of object.



This is the blueprint of the Vehicle that is class, and the Box truck, Sports car, Sedan car and Pickup truck are the objects of Vehicle.

In PHP, declare a class using the class keyword, followed by the name of the class and a set of curly braces ({}).

#### Syntax to Create Class in PHP

```
<?php
class MyClass
{
    // Class properties and methods go here
}
?>
```

In PHP, to see the contents of the class, use var\_dump(). The var\_dump() function is used to display the structured information (type and value) about one or more variables.

#### Syntax:

```
var_dump($obj);
```

**Object:** Objects are real time entities. Objects are determined from classes in Object-Oriented Programming like PHP. When a class is specified, we can create any number of objects out of the class.

**Example:** If Vehicle is the class, then Box truck, Sports car, Sedan car and Pickup truck are the objects of the Class Vehicle.





**Creating an object:** Following is an example of how to create object using **new** operator.

**Syntax:**

```
<?php
class MyClass
{
    // Class properties and methods go here
}
$obj = new MyClass;
var_dump($obj);
?>
```

**Example of class and object:**

```
<?php
class SayHello
{
    function hello()
    {
        echo "Hello World";
    }
}
$obj=new SayHello;
$obj->hello();
?>
```

**Output:**

Hello World

**Output:**

**INHERITANCE:** It is a concept of accessing the features from one class to another class. If we inherit the class features into another class, we can access both class properties. We can extend the features of a class by using 'extends' keyword.

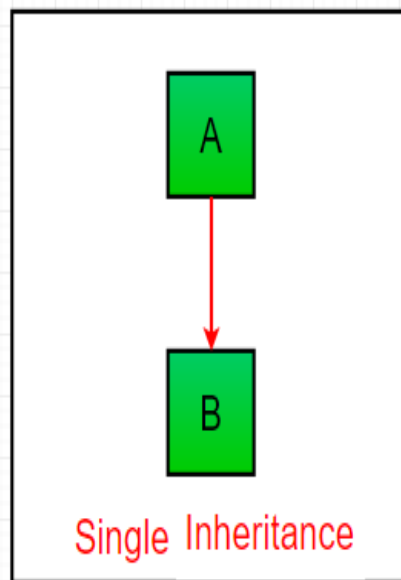
- It supports the concept of hierarchical classification.
- Inheritance has three types, single, multiple and multilevel Inheritance.
- PHP supports only single inheritance, where only one class can be derived from single parent class.
- We can simulate multiple inheritance by using interfaces.

**TYPES OF INHERITANCES:** PHP offers mainly three types of inheritance based on their functionality.

These three types are as follows:

**1. Single Inheritance:** Deriving a class from only one base class(super class) is known as Single inheritance.

In below image, the class A serves as a base class for the derived class B.

**Example:**

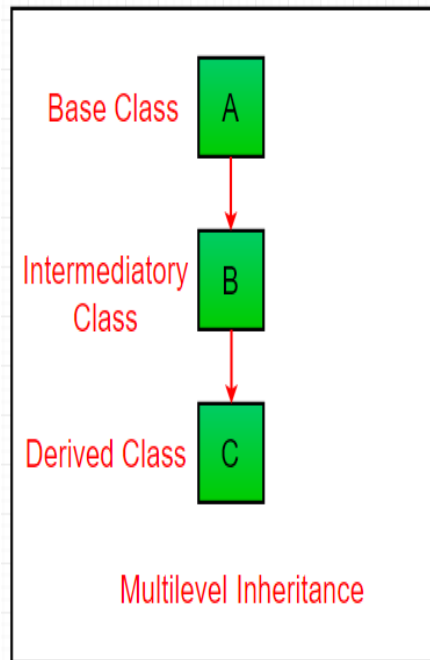
```
<?php
class demo
{
    public function display()
```

```
{
    echo "Example of inheritance ";
}
}
class demo1 extends demo
{
    public function view()
    {
        echo "in php";
    }
}
$obj= new demo1();
$obj->display();
$obj->view();
?>
```

**Output:**

Example of inheritance

**2. Multilevel Inheritance:** Deriving a class from another derived class is known as Multi level inheritance. In below image, the class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C.



**Example:**

```
<?php
class A
{
    function showA()
    {
        echo "I am show method in A Class<br>";
    }
}

class B extends A
{
    function showB()
    {
        echo "I am show method in B Class<br>";
    }
}

class C extends B
{
    function showC()
```

```

    {
        echo "I am show method in C Class<br>";
    }
}

$obj=new C;
$obj->showA();
$obj->showB();
$obj->showC();
?>

```

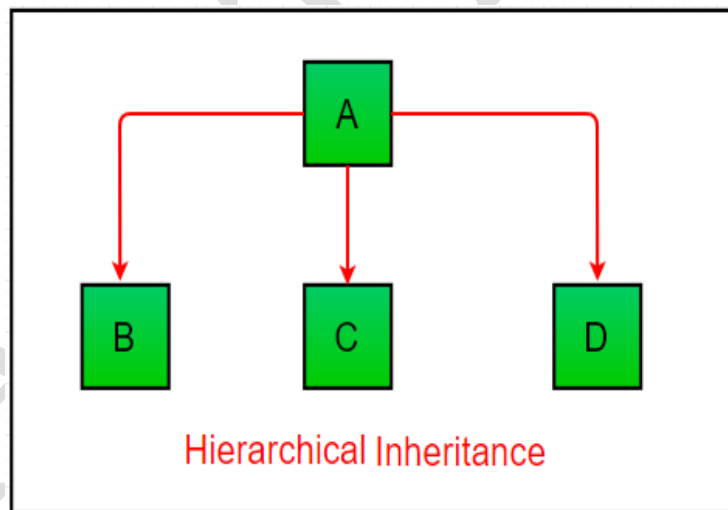
**Output:**

```

I am show method in A Class
I am show method in B Class
I am show method in C Class

```

**Hierarchical Inheritance:** Deriving several classes from one base class is known as Hierarchical inheritance. In below image, the class A serves as a base class for the derived class B,C and D.



**Example:**

```

<?php
// base class named "Jewellery"
class Jewellery
{
    public function show()
    {
        echo "This class is Jewellery ";
    }
}

```

```
    }  
}  
  
// Derived class named "Necklace"  
class Necklace extends Jewellery  
{  
    public function show()  
    {  
        echo "This class is Necklace ";  
        echo "<br>";  
    }  
}  
  
// Derived class named "Necklace"  
class Bracelet extends Jewellery
```

```

    {
        public function show()
        {
            echo 'This class is Bracelet ';
        }
    }

// creating objects of derived classes "Necklace" and "Bracelet"
$n = new Necklace();
$n->show();
$b = new Bracelet();
$b->show();
?>

```

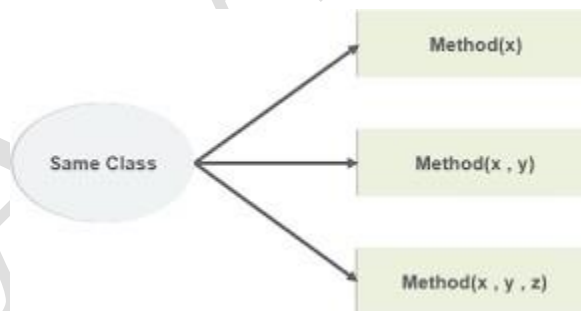
**Output:**

This class is Necklace

This class is Bracelet

**FUNCTION OVERLOADING AND OVERRIDING:**

**Function Overloading:** “If a class contains multiple methods with same name but different parameters is known as Method overloading.”



```

<?php
class Shape
{
    const PI = 3.142 ;
    function __call($name,$arg)
    {
        if($name == 'area')
        switch(count($arg))

```

```

        {
            case 0 : return 0 ;
            case 1 : return self::PI * $arg[0] ;
            case 2 : return $arg[0] * $arg[1];
        }
    }
}

$circle = new Shape();
echo $circle->area(3);
$rect = new Shape();
echo $rect->area(8,6);
?>

```

**Output:** 9.42648

**Function Overriding:** Function overriding is same as other OOPs programming languages. In function overriding, both parent and child classes should have same function name with and number of arguments. It is used to replace parent method in child class. The purpose of overriding is to change the behavior of parent class method. The two methods with the same name and same parameter is called overriding.

**Example:**

```

<?php
class Vehicle
{
    function run()
    {
        echo "Vehicle is running<br>";
    }
}
class Bike extends Vehicle
{
    function run()
    {
        echo "Bike is running";
    }
}

$v = new Vehicle;
$b= new Bike;
$v->run();
$b->run();

```



?>

**Output:**

Vehicle is running

Bike is running

**INTERFACE**

- An interface is similar to a class except that it cannot contain code.
- An interface can define method names and arguments, but not the contents of the methods.
- Any classes implementing an interface must implement all methods defined by the interface.
- A class can implement multiple interfaces.
- An interface is declared using the "interface" keyword.
- Interfaces can't maintain Non-abstract methods.

**Example**

```
<?php
```

```
interface circle
```

```
{
```

```
    public function draw1();
```

```
}
```

```
interface rectangle
```

```
{
```

```
    public function draw2();
```

```
}
```

```
class demo implements circle,rectangle
```

```
{
```

```
    public function draw1()
```

```
{
```

```
    echo "Drawing Circle";
```

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

```
}
```

```
    public function draw2()
```

```
{
```

```
    echo "Drawing Rectangle";
```

```
}
```

```
}
```

```
$obj= new demo();  
$obj->draw1();  
$obj->draw2();  
?>
```

**Output:**

Drawing Circle

Drawing Rectangle

**CONSTRUCTORS:** Constructor is a method which is defined inside a class. Constructor is called automatically at the time of creation of object. Purpose of a constructor method is to initialize the object. In PHP, a method of special name **\_\_construct** acts as a constructor. Constructors start with two underscores and generally look like normal PHP functions.

**Syntax:**

```
__construct():  
function __construct()  
{  
    // initialize the object and its properties by assigning  
    //values  
}
```

The constructor is the most useful because it allows us to send parameters along when creating a new object, which can then be used to initialize variables on the object. Whenever, you use **\_\_construct()** function. PHP automatically executed this function When you create an object from a class.

**TYPES OF CONSTRUCTORS:** There are two types of constructors

- 1) Default Constructor
- 2) Parameterized Constructor

**1. Default Constructor:** It has no parameters, but the values to the default constructor can be passed dynamically.

**Example**

```
<?php  
class Person  
{  
    function CanSpeak()  
    {  
        echo " Not a constructor method " . '<br>';  
    }  
}
```

```

    }
    function __construct()
    {
        echo " In the constructor method " . '<br>';
    }
}

//Object of class calling the constructor internally
$p = new Person();
// Object of class calling the normal method
$p->CanSpeak();
?>

```

Output:

In the constructor method

Not a constructor method

**2) Parameterized Constructor:** A constructor can be with or without arguments. The constructor with arguments is called the parameterized constructor and the constructor without arguments is called the zero-argument constructor.

### Example

```

<?php
class Employee
{
    public $name;
    public $position;
    function __construct($name,$position)
    {
        // This is initializing the class properties
        $this->name=$name;
        $this->profile=$position;
    }
    function show_details()
    {
        echo $this->name." : ";
        echo "Your position is ".$this->profile."\n";
    }
}

```

```

    }
}

$employee_obj= new Employee("Rakesh","developer");
$employee_obj->show_details();
$employee2= new Employee("Vikas","Manager");
$employee2->show_details();
?>

```

**Output:**

Rakesh : Your position is developer

Vikas : Your position is Manager

**DESTRUCTOR:** The Destructor is automatically invoked before an object is deleted. It happens when the object has no reference or when the script ends. Use the `__destruct()` to define a destructor for a class. PHP automatically invokes the destructor when the object is deleted or the script is terminated.

**PHP STRING:** String is a sequence of characters i.e., used to store and manipulate text. There are four ways of creating strings in PHP

1. Single quoted
2. Double quoted
3. Heredoc
4. Nowdoc

**1. Single quoted:** We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.

**Example 1:**

```

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

```

**Output:**

Hello text within single quote

**Example 2**

```

<?php
    $str1='Hello text
multiple line
text within single quoted string';

```

```

$str2='Using double "quote" directly inside single quoted string';
$str3='Using escape sequences \n in single quoted string';
echo "$str1 <br/> $str2 <br/> $str3";
?>

```

**Output:**

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

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

**Example 1**

```

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

```

**Output:**

Hello text within double quote

Now, you **can't use double quote directly** inside double quoted string.

**Example 2**

```

<?php
$str1="Using double "quote" directly inside double quoted string";
echo $str1;
?>

```

**Output:**

Parse error: syntax error, unexpected 'quote' (T\_STRING) in C:\wamp\www\string1.php

**Example 3:** We can store multiple line text, special characters and escape sequences in a double quoted PHP string.

```

<?php
$str1="Hello text
multiple line
text within double quoted string";
$str2="Using double \"quote\" with backslash inside double quoted string";

```

```
$str3="Using escape sequences \n in double quoted string";  
echo "$str1 <br/> $str2 <br/> $str3";  
?>
```

**Output:**

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

**3. Heredoc:** The syntax of Heredoc (<<<) is another way to delimit PHP strings. An identifier is given after the heredoc (<<< ) operator, after which any text can be written as a new line is started. To close the syntax, the same identifier is given without any tab or space.

**Example:**

```
<?php  
//Print the first heredoc document  
echo <<< HERE  
  
<pre>  
PHP is a general-purpose scripting language especially suited to web  
development.  
It was created by Rasmus Lerdorf in 1994.  
HERE;  
//Print the second heredoc document  
echo <<< DOC  
  
<pre>  
www.google.com  
www.bing.com  
www.ask.com  
www.yahoo.coms  
</pre>  
DOC;  
?>
```

**Output:**

PHP is a general-purpose scripting language especially suited to  
web development.

It was created by Danish-Canadian programmer Rasmus Lerdorf in 1994.

www.google.com

www.bing.com

www.ask.com

www.yahoo.coms

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

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

**STRING FUNCTIONS OR MANIPULATING STRINGS WITH PHP:** PHP provides various string functions to access and manipulate strings. A list of PHP string functions are given below.

**1. strlen() Function:** It returns the length of the string i.e. the count of all the characters in the string including whitespaces characters.

**Syntax:**

strlen(string or variable name)

**Example:**

```
<?php
    $str = "Hello World!";
    // Prints 12 as output
    echo strlen($str);
    // Prints 13 in a new line
    echo "<br>" . strlen("GeeksForGeeks");
?>
```

**Output:**

12

13

**2. strrev() Function:** It returns the reversed string of the given string.

**Syntax:**

strrev(string or variable name)

**Example:**

```
<?php
```

---

MCA-ATMECE



```
$str = "Hello World!";  
echo strrev($str);  
?>
```

**Output:**

!dlroW olleH

**3. trim(), ltrim(), rtrim(), and chop() Functions:** It remove white spaces or other characters from the string. They have two parameters: one string and another charList, which is a list of characters that need to be omitted.

- **trim()** – Removes characters or whitespaces from both sides.
- **rtrim()** & **chop()** – Removes characters or whitespaces from right side.
- **ltrim()** – Removes characters or whitespaces from the left side.

**Syntax:**

```
rtrim(string, charList)  
ltrim(string, charList)  
trim(string, charList)  
chop(string, charList)
```

**Example:**

```
<?php  
$str = "\nThis is an example for string functions.\n";  
// Prints original string  
echo $str. "<br>";  
// Removes whitespaces from right end  
echo chop($str) . "<br>";  
// Removes whitespaces from both ends  
echo trim($str) . "<br>";  
// Removes whitespaces from right end  
echo rtrim($str) . "<br>";  
// Removes whitespaces from left end  
echo ltrim($str);  
?>
```

**Output:**

This is an example for string functions.

This is an example for string functions.

This is an example for string functions.

This is an example for string functions.

This is an example for string functions.

**4. strtoupper() Function:** The strtoupper() function returns string in uppercase letter.

**Syntax:**

strtoupper (string)

**Example:**

```
<?php
$str="Dnr College";
$str=strtoupper($str);
echo $str;
?>
```

**Output:**

DNR COLLEGE

**5. strtolower() function:** The strtolower() function returns string in lowercase letter

**Syntax:**

strtolower(string)

**Example:**

```
<?php
$str="Dnr College";
$str=strtolower($str);
echo $str;
?>
```

**Output:**

dnr college

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

**Syntax**

ucfirst(strin )

**Example**

```
<?php
$str="dNR College";
```

```
$str=ucfirst($str);  
echo $str;  
?>
```

**Output:**

DNR College

**7. lcfirst() function:** The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

**Syntax**

string lcfirst ( string \$str )

**Example**

```
<?php  
$str="Dnr College";  
$str=lcfirst($str);  
echo $str;  
?>
```

**Output:**

dnr College

**8. ucwords() function:** The ucwords() function returns string converting first character of each word into uppercase.

**Syntax**

string ucwords ( string \$str )

**Example**

```
<?php  
$str="dnr college bhimavaram";  
$str=ucwords($str);  
echo $str;  
?>
```

**Output:**

Dnr College Bhimavaram

**9. strcmp() Function:** The strcmp() function is used to compare two strings. It returns true if two strings are equal otherwise returns false.

**Syntax:**

strcmp(string1,string2)

---

MCA-ATMECE

**Example:**

```
<html>
<body>
    <?php
        echo strcmp("Hello world!","Hello world!");
    ?>
<p>If this function returns 0, the two strings are equal.</p>
</body>
</html>
```

**Output:**

0

If this function returns 0, the two strings are equal.

**DATE AND TIME FUNCTIONS IN PHP:** The date/time functions allow you to get the date and time from the server where your PHP script runs. You can then use the date/time functions to format the date and time in several ways. Some of the predefined functions in PHP for date and time are discussed below.

**PHP date() Function:** The PHP date() function converts timestamp to a more readable date and time format.

**Syntax:**

```
date(format, timestamp)
```

**Explanation:**

The format parameter in the date() function specifies the format of returned date and time.

The timestamp is an optional parameter, if it is not included then the current date and time will be used.

**Example:**

```
<?php
echo "Today's date is :";
$today = date("d/m/Y");
echo $today;
?>
```

**Output:**

Today's date is :05/12/2017

**Example 2:**

```
<?php
```

---

MCA-ATMECE

```
echo "Today's date in various formats:" . "\n";
echo date("d/m/Y") . "\n";
echo date("d-m-Y") . "\n";
echo date("d.m.Y") . "\n";
echo date("d.M.Y/D");
?>
```

**Output:**

```
Today's date in various formats:
05/12/2017
05-12-2017
05.12.2017
05.Dec.2017/Tue
```

**PHP time() Function:** The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1, 1970, 00:00:00 GMT).

Example: The below example explains the usage of the time() function in PHP.

```
<?php
$timestamp = time();
echo($timestamp);
echo "\n";
echo(date("F d, Y h:i:s A", $timestamp));
?>
```

**Output:**

```
1512486297
December 05, 2017 03:04:57 PM
```

---

MCA-ATMECE



## WORKING WITH FORMS

**HTML FORM:** **HTML form** is a section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

`<form>` is a HTML element to collect input data with containing interactive controls. It provides facilities to input text, number, values, email, password, and control fields such as checkboxes, radio buttons, submit buttons, etc., or in other words, form is a container that contains input elements like text, email, number, radio buttons, checkboxes, submit buttons, etc.

### **Syntax:**

`<form>`

`<!--form elements-->`

`</form>`

The `<form>` element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

### **The `<input>` Element**

The HTML `<input>` element is the most used form element.

An `<input>` element can be displayed in many ways, depending on the type attribute.

Here are some examples:

Type	Description
<code>&lt;input type="text"&gt;</code>	Displays a single-line text input field
<code>&lt;input type="radio"&gt;</code>	Displays a radio button (for selecting one of many choices)
<code>&lt;input type="checkbox"&gt;</code>	Displays a checkbox (for selecting zero or more of many choices)
<code>&lt;input type="submit"&gt;</code>	Displays a submit button (for submitting the form)
<code>&lt;input type="button"&gt;</code>	Displays a clickable button

## Example:

```
<!Doctype Html>
```

```
<Html>
```

```
<Head>
```

```
<Title>
```

```
Create a Login form
```

```
</Title>
```

```
</Head>
```

```
<Body>
```

The following tags are used in this Html code for creating the Login form:

```
<form>
```

```
<label>User Id: </label> <br>
```

```
<input type="text"> <br> <br>
```

```
<label>Password:</label> <br>
```

```
<input type="password"> <br> <br>
```

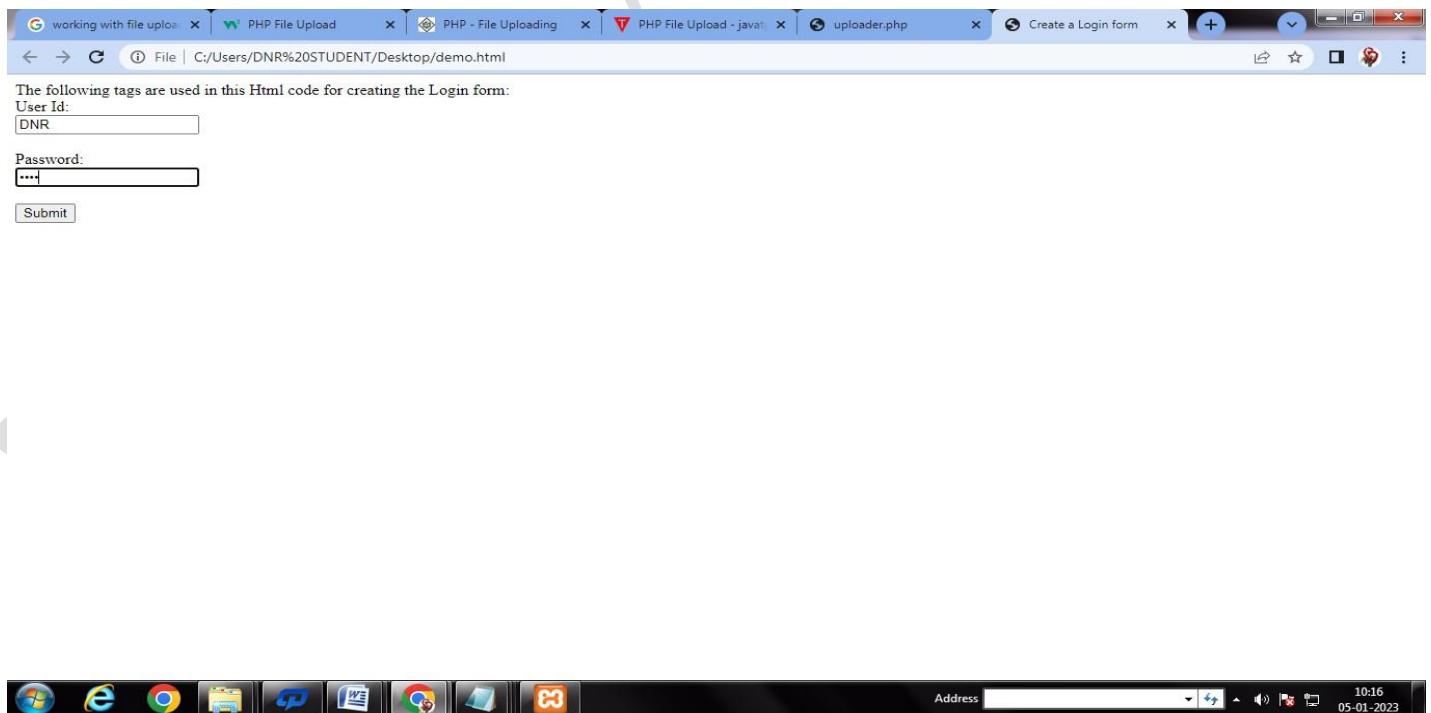
```
<input type="submit" value="Submit">
```

```
</form>
```

```
</Body>
```

```
</Html>
```

## Output:



**COMBINING HTML AND PHP CODE ON A SINGLE PAGE:** PHP code is normally mixed with HTML tags. PHP is an embedded language, meaning that you can jump between raw HTML code and PHP without sacrificing readability.

In order to embed PHP code with HTML, the PHP must be set apart using PHP start and end tags. The PHP tags tell the web server where the PHP code starts and ends.

**Creating Forms:** To create a form, you use the <form> element as follows:

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

The <form> element has two important attributes:

- **action:** specifies the URL that processes the form submission. In this example, the form.php will process the form.
- **method:** specifies the HTTP method for submitting the form. The most commonly used form methods are POST and GET. In this example, the form method is post.

The form method is case-insensitive. It means that you can use either post or POST. If you don't specify the method attribute, the form element will use the get method by default. To retrieve data from get request, we need to use \$\_GET, for post request \$\_POST.

Typically, a form has one or more input elements including text, password, checkbox, radio button, select, file upload, etc. The input elements are often called form fields.

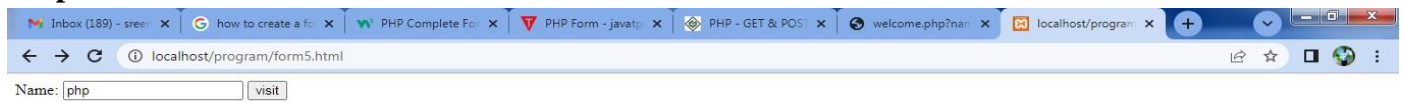
**PHP Get Form:** Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

**File: Save this file as form1.html**

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

---

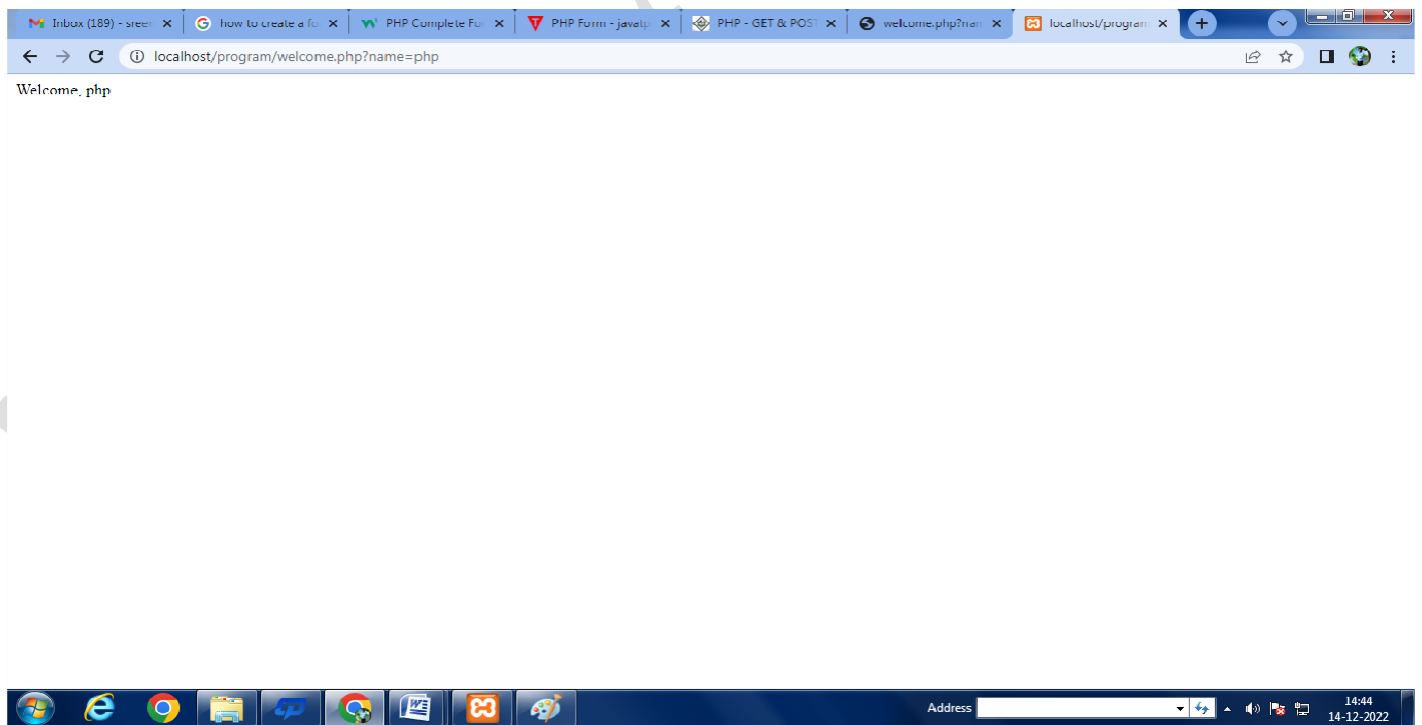
## Output:



## File: Save this file as welcome.php

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

## Output:



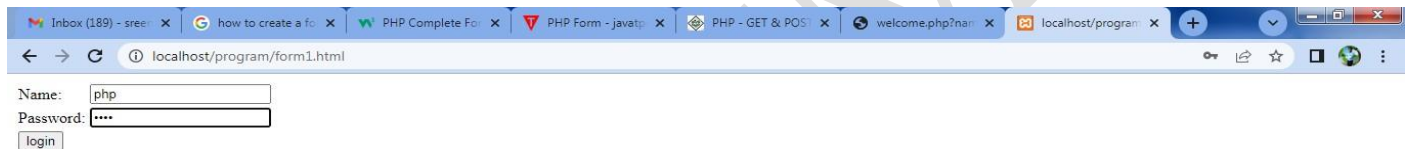
**PHP Post Form:** Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

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

Let's see a simple example to receive data from post request in PHP.

**File: Save this file as form1.html**

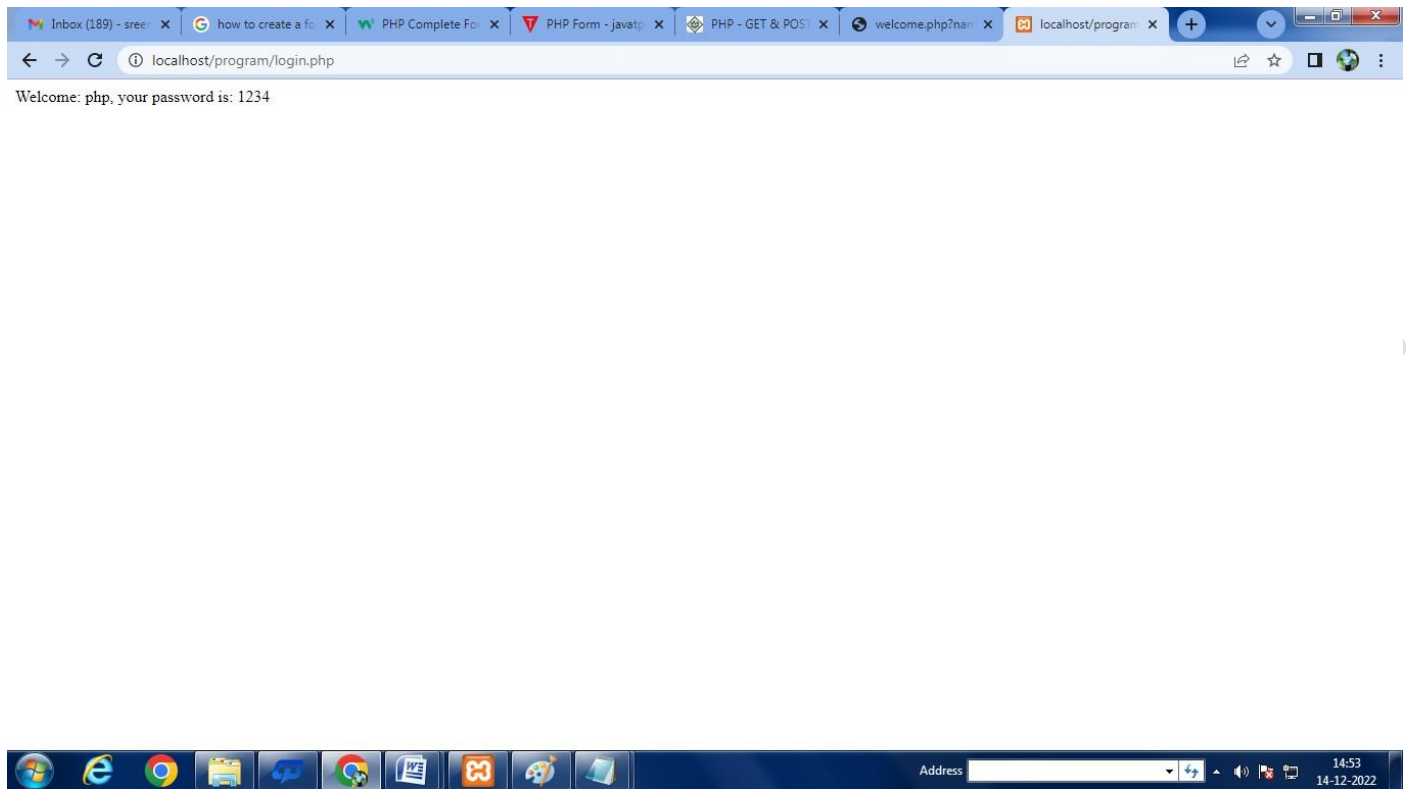
```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/> </td></tr>
</table>
</form>
```



**File: Save this file as login.php**

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

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



**SENDING MAIL ON FORM SUBMISSION:** PHP makes use of mail() function to send an email. This function requires three mandatory arguments that specify the recipient's email address, the subject of the message and the actual message additionally there are other two optional parameters.

**Syntax:** mail( to, subject, message, headers, parameters );

Sr.No	Parameter & Description
1	to Required. Specifies the receiver / receivers of the email
2	subject Required. Specifies the subject of the email. This parameter cannot contain any newline characters
3	message Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
4	headers

	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)
5	parameters Optional. Specifies an additional parameter to the send mail program

As soon as the mail function is called PHP will attempt to send the email then it will return true if successful or false if it is failed. Multiple recipients can be specified as the first argument to the mail() function in a comma separated list.

**WORKING WITH FILE UPLOADS:** PHP allows you to upload single and multiple files through few lines of code only. PHP file upload features allow you to upload binary and text files both.

**PHP \$\_FILES:** The PHP global \$\_FILES contains all the information of file. By the help of \$\_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

PHP would create following five variables –

- **\$\_FILES['file']['tmp\_name']** – the uploaded file in the temporary directory on the web server.
- **\$\_FILES['file']['name']** – the actual name of the uploaded file.
- **\$\_FILES['file']['size']** – the size in bytes of the uploaded file.
- **\$\_FILES['file']['type']** – the MIME type of the uploaded file.
- **\$\_FILES['file']['error']** – the error code associated with this file upload.

The process of uploading a file follows these steps –

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.
- The full path to the selected file appears in the text filed then the user clicks the submit button.
- The selected file is sent to the temporary directory on the server.
- The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- The PHP script confirms the success to the user.

## PHP File Upload Example

**File: uploadform.html**

```
<form action="uploader.php" method="post" enctype="multipart/form-data">
```

Select File:

```
<input type="file" name="fileToUpload"/>
```

```
<input type="submit" value="Upload Image" name="submit"/>
```

</form>

### File: uploader.php

<?php

```
$target_path = "e:/";
```

```
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);
```

```
if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {
```

```
    echo "File uploaded successfully!";
```

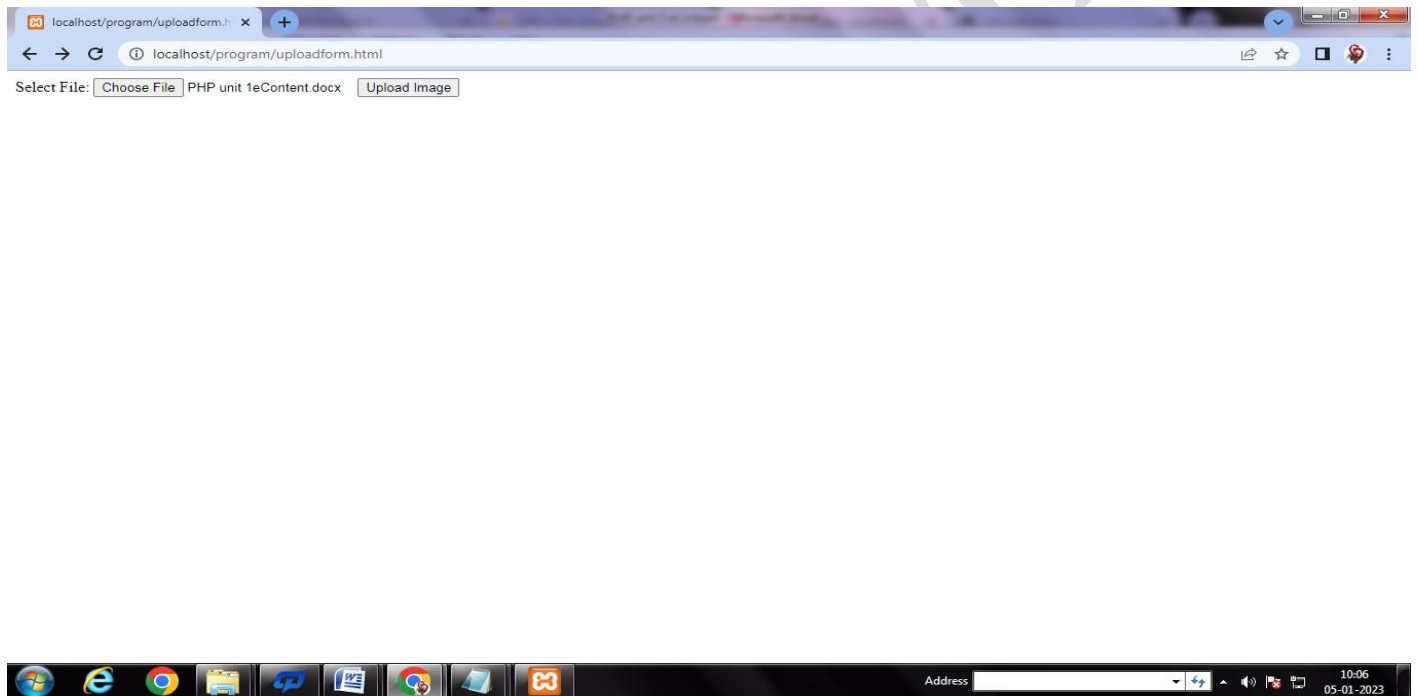
```
} else{
```

```
    echo "Sorry, file not uploaded, please try again!";
```

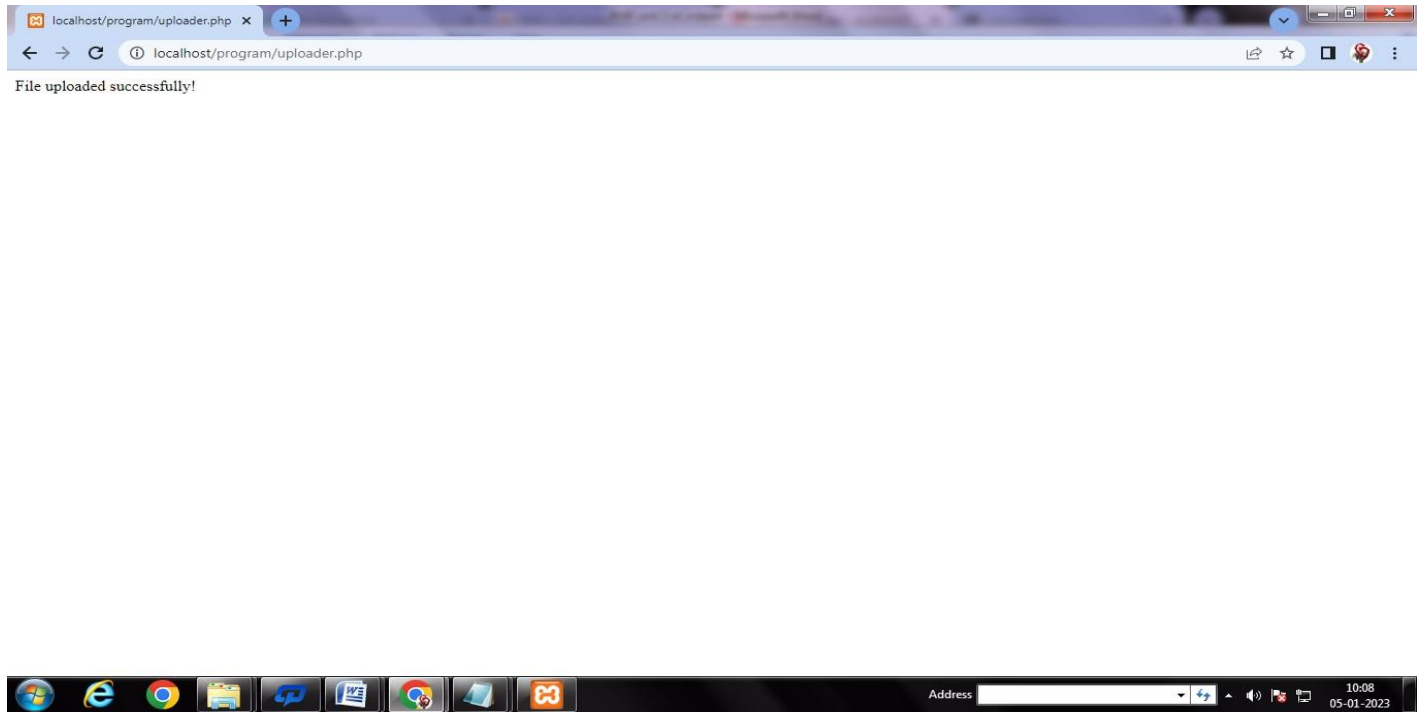
```
}
```

```
?>
```

### Output:



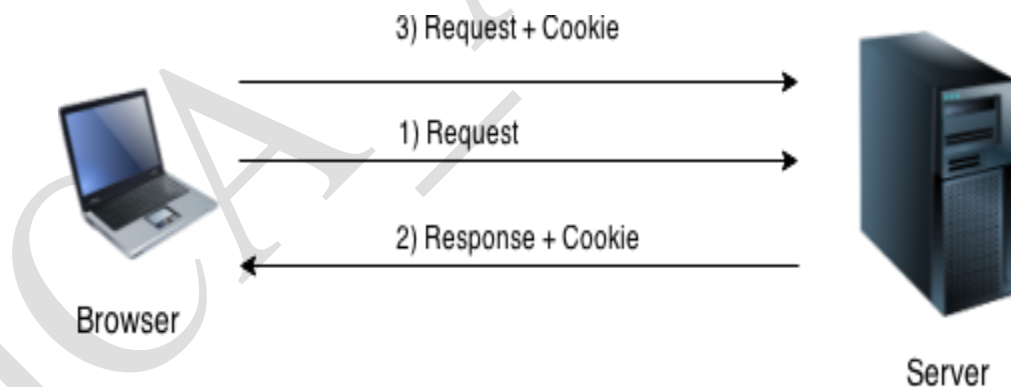




## WORKING WITH COOKIES AND USER SESSIONS

**INTRODUCING COOKIES:** PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



There are three steps involved in identifying returning users –

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
  - Browser stores this information on local machine for future use.
  - When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.
-

**SETTING A COOKIE WITH PHP:** PHP provided **setcookie()** function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

Syntax: setcookie(name, value, expire, path, domain, security);

Here is the detail of all the arguments –

- **Name** – This sets the name of the cookie and is stored in an environment variable called HTTP\_COOKIE\_VARS. This variable is used while accessing cookies.
- **Value** – This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** – This specifies a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which means cookie can be sent by regular HTTP.

### PHP Cookie Example

**File: cookie1.php**

```
<?php
setcookie("user", "Sreenu");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"]))
{
    echo "Sorry, cookie is not found!";
}
else
{
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
```

---

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

**Output:**

Sorry, cookie is not found!

Firstly cookie is not set. But, if you refresh the page, you will see cookie is set now.

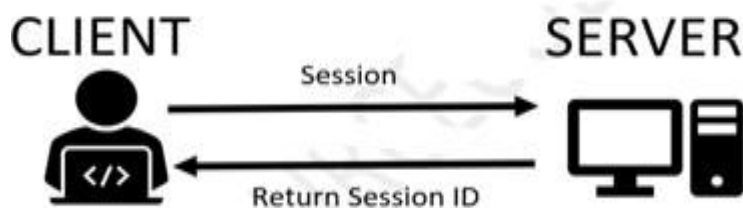
**Output:**

Cookie Value: Sreenu

**SESSION INTRODUCTION:** When we work with an application, we open it, do some changes, and then we close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser. So Session variables hold information about one single user, and are available to all pages in one application.

**PHP SESSION:** PHP session is used to store and pass information from one page to another temporarily (until user close the website). Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc. PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

**PHP session\_start() function**

PHP session\_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

**Syntax**

```
bool session_start (void)
```

---

## Example

```
session_start();
```

**PHP \$\_SESSION:** PHP \$\_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

### Example: Store information

```
$_SESSION["user"] = "Sachin";
```

### Example: Get information

```
echo $_SESSION["user"];
```

## PHP Session Example

//Save file as session1.php

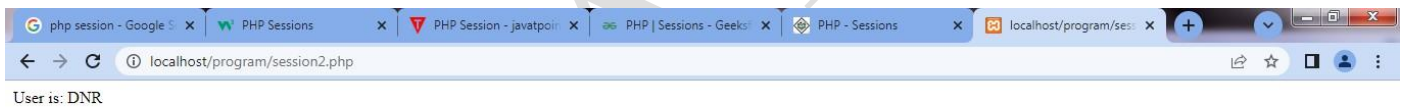
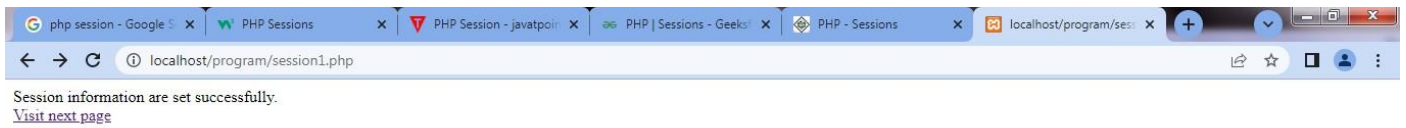
```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "DNR";
echo "Session information are set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>
</body>
</html>
```

//Save file as session2.php

```
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: " . $_SESSION["user"];
?>
</body>
</html>
```

---

## OUTPUT:



## WORKING WITH FILES AND DIRECTORIES

**Files:** A File is a collection of data stored in the secondary memory. Files are used for storing information that can be processed by the programs. Files are not only used for storing the data, programs are also stored in files.

**Directories:** A directory is a unique type of file that contains only the information needed to access files or other directories. A directory occupies less space than other types of files.

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

1. include
2. require

**1. PHP include:** PHP include is used to include a file on the basis of given path. You may use a relative or absolute path of the file.

### **Syntax**

There are two syntaxes available for include:

**include** 'filename ';

Or

**include** ('filename');

### **Example**

//Save file as menu.html

```
<html>
<body>
<a href="http://www.javatpoint.com">Home</a> |
<a href="http://www.javatpoint.com/php-tutorial">PHP</a> |
<a href="http://www.javatpoint.com/java-tutorial">Java</a> |
<a href="http://www.javatpoint.com/html-tutorial">HTML</a>
</body>
</html>
```

//Save file as include1.php

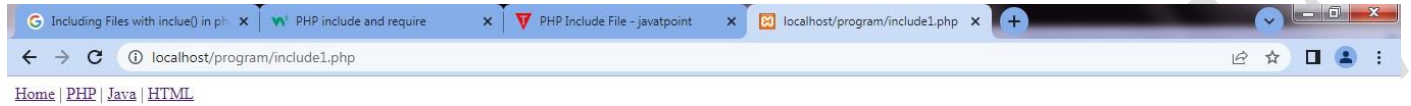
```
<?php
```

```
include("menu.html");
```

```
?>
```

```
<h1>This is Main Page</h1>
```

## Output:



**This is Main Page**



**PHP REQUIRE:** PHP require is similar to include, which is also used to include files. The only difference is that it stops the execution of script if the file is not found whereas includes doesn't.

## Syntax

There are two syntaxes available for require:

```
require 'filename';
```

or

```
require ('filename');
```

## Example

//Save file as menu2.html

```
<html>
```

```
<body>
```

```
<a href="http://www.javatpoint.com">Home</a> |
```

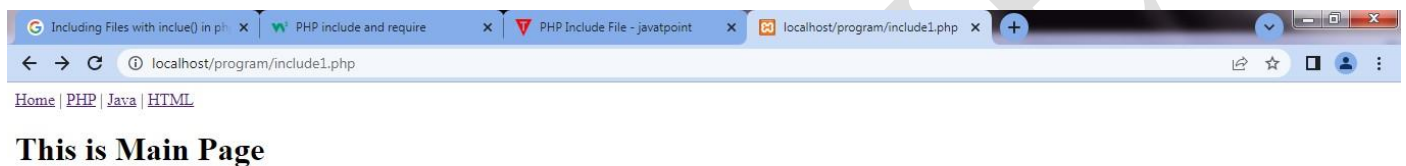
```
<a href="http://www.javatpoint.com/php-tutorial">PHP</a> |
```

```
<a href="http://www.javatpoint.com/java-tutorial">Java</a> |  
<a href="http://www.javatpoint.com/html-tutorial">HTML</a>  
</body>  
</html>
```

//Save file as require1.php

```
<?php  
require("menu.html");  
?  
<h1>This is Main Page</h1>
```

### Output:



**PHP FILE HANDLING FUNCTIONS:** PHP File System allows us to create file, read file, write file, append file, delete file and close file.

**1. PHP Create File - fopen():** The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

### Example

```
<?php
```



```
$file = fopen("demo.txt","w");
```

```
?>
```

**2. PHP Write to File - fwrite():** The fwrite() function is used to write to a file. The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

#### Syntax

```
int fwrite ( resource $handle , string $string [, int $length ] )
```

#### Example

```
<?php
$fp = fopen('data.txt', 'w');//open file in write mode
fwrite($fp, 'hello ');
fwrite($fp, 'php file');
fclose($fp);
echo "File written successfully";
?>
```

#### OUTPUT

File written successfully

**3. PHP Read File - fread():** The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

#### Syntax

```
string fread ( resource $handle , int $length )
```

#### Example

```
<?php
$filename = "c:\\myfile.txt";
$handle = fopen($filename, "r");//open file in read mode
$contents = fread($handle, filesize($filename));//read file
echo $contents;//printing data of file
fclose($handle);//close file
?>
```

#### OUTPUT

hello php file

**4. PHP Delete File - unlink():** The PHP unlink() function is used to delete file.

#### Syntax

```
bool unlink ( string $filename [, resource $context ] )
```

MCA-ATMECE

### Example

```
<?php
unlink('data.txt');
echo "File deleted successfully";
?>
```

**5. PHP Append Text:** We can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file. In the example below we open our existing file "newfile.txt", and append some text to it.

Syntax:

resource fopen ( string filename, string mode )

### Example

```
<?php
$fp = fopen('data.txt', 'a');//opens file in append mode
fwrite($fp, ' this is additional text ');
fwrite($fp, 'appending data');
fclose($fp);
echo "File appended successfully";
?>
```

### OUTPUT:

File appended successfully

**6. fclose()** – file is closed using fclose() function. Its argument is file which needs to be closed.

### Syntax

fclose(filepointer)

### Example:

```
<?php
$file = fopen("demo.txt", 'r');
//some code to be executed
fclose($file);
echo "File closed successfully"
?>
```

### OUTPUT:

File closed successfully

---

MCA - ATMECE

**PHP DIRECTORIES:** PHP directory functions as their name suggests are a set of functions used in retrieving details, modifying them and fetching information on various file system directories and their specific contents. A lot of operations can be performed on the directories like creating, deleting, changing the present working directory, listing files present in the directory and so on. These basic functions are listed below.

- `mkdir()`: To make new directory.
- `opendir()`: To open directory.
- `readdir()`: To read from a directory after opening it.
- `closedir()`: To close directory with resource-id returned while opening.
- `rmdir()`: To remove directory.

**1. Creating New Directory:** For creating a new directory using PHP programming script, `mkdir()` function is used and the usage of this function is as follows.

```
<?php
mkdir($directory_path, $mode, $recursive_flag, $context);
?>
```

This function accepts four arguments as specified. Among them, the first argument is mandatory, whereas, the remaining set of arguments is optional.

- **\$directory\_path:** By specifying either relative and absolute path, a new directory will be created in such location if any, otherwise, will return an error indicating that there are no such locations.
- **\$mode:** The mode parameter accepts octal values on which the accessibility of the newly created directory depends.
- **\$recursive:** This parameter is a flag and has values either true or false, that allow or refuse to create nested directories further.
- **\$context:** As similar as we have with PHP `unlink()` having a stream for specifying protocols and etc.

**2. Listing Directory Content in PHP:** For listing the contents of a directory, we require two of the above listed PHP directory functions, these are `opendir()` and `readdir()`. There are two steps in directory listing using the PHP program.

- Step 1: Opening the directory.
- Step 2: Reading content to be listed one by one using a PHP loop.

**Step 1: Opening Directory Link:** `opendir()` function is used to perform this step. And, it has two arguments, one is compulsory for specifying the path of the directory, and the other is optional, expecting stream context if any. The syntax will be,

```
<?php
opendir($directory_path, $context);
```

?>

**Step 2: Reading Directory Content:** PHP `readdir()` will return string data on each iteration of the loop, and this string will be the name of each item stored in the directory with its corresponding extension.

**3. Closing Directory Link:** Once the directory link is opened to perform a set of dependent operations like reading directory content, we need to close this link after completing the related functionalities required. For example,

```
<?php
$directory_handle = opendir($directory_path);

...

closedir($directory_handle);

?>
```

**4. Removing Directory:** PHP `unlink()` function is used to delete a file from a directory. Similarly, for removing the entire directory, PHP provides a function named as `rmdir()` which accepts the same set of arguments, as `mkdir()`. These are, the `$directory_path` and `$context(Optional)` as stated below.

```
<?php
rmdir($directory_path, $mode, $recursive_flag, $context);

?>
```

**READING DATA FROM FILES:** PHP provides various functions to read data from file. There are different functions that allow you to read all file data, read data line by line and read data character by character.

The available PHP file read functions are given below.

1. `fread()`
2. `fgets()`
3. `fgetc()`

**1. PHP Read File - `fread()`:** The PHP `fread()` function is used to read data of the file. It requires two arguments: file resource and file size.

**Syntax:**

string `fread` (resource `$handle` , int `$length` )

**`$handle`** represents file pointer that is created by `fopen()` function.

**`$length`** represents length of byte to be read.

**Example:**

```
<?php
```

```
$filename = "c:\\file1.txt";  
$fp = fopen($filename, "r");//open file in read mode  
$contents = fread($fp, filesize($filename));//read file  
echo "<pre>$contents</pre>";//printing data of file  
fclose($fp);//close file  
?>
```

#### OUTPUT:

```
this is first line  
this is another line  
this is third line
```

**2. PHP Read File - fgets():** The PHP fgets() function is used to read single line from the file.

#### Syntax

```
string fgets ( resource $handle [, int $length ] )
```

#### Example

```
<?php  
$fp = fopen("c:\\file1.txt", "r");//open file in read mode  
echo fgets($fp);  
fclose($fp);  
?>
```

#### OUTPUT

```
this is first line
```

**3. PHP Read File - fgetc():** The PHP fgetc() function is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

#### Syntax

```
string fgetc ( resource $handle )
```

#### Example

```
<?php  
$fp = fopen("c:\\file1.txt", "r");//open file in read mode  
while(!feof($fp))  
{  
    echo fgetc($fp);  
}
```

```
fclose($fp);>
```

## OUTPUT

this is first line this is another line this is third line

## WORKING WITH IMAGES

**PHP IMAGECREATE( ) FUNCTION:** Image create ( ) function is another inbuilt PHP function mainly used to create a new image. The function returns the given image in a specific size. We need to define the width and height of the required image. Instead of the image create ( ) function, we can also use other creative functions like imagecreatetruecolor( ), which is a better alternative as it will return a better image quality.

**Syntax:** In PHP, imagecreate( ) function follows the following syntax.

```
imagecreate( $width, $height )
```

S.No	Parameter	Description	Optional / mandatory
1	\$ width	This parameter is used to define the image's width that we want to display.	Mandatory
2	\$ height	This parameter is used to define the height of the image that we want to display	Mandator

The image creates ( ) function returns the resource identifier of an image on successful execution of the program and FALSE on a failed attempt.

### Example:

```
<?php
// to define the size of the image
$img = imagecreate(500, 300);
// to define the background color of the image
$bgcolor = imagecolorallocate($img, 150, 200, 180);
// to define the text color of the image
$fontcolor = imagecolorallocate($img, 120, 60, 200);
imagestring($img, 12, 150, 120, "Demo Text1", $fontcolor);
```

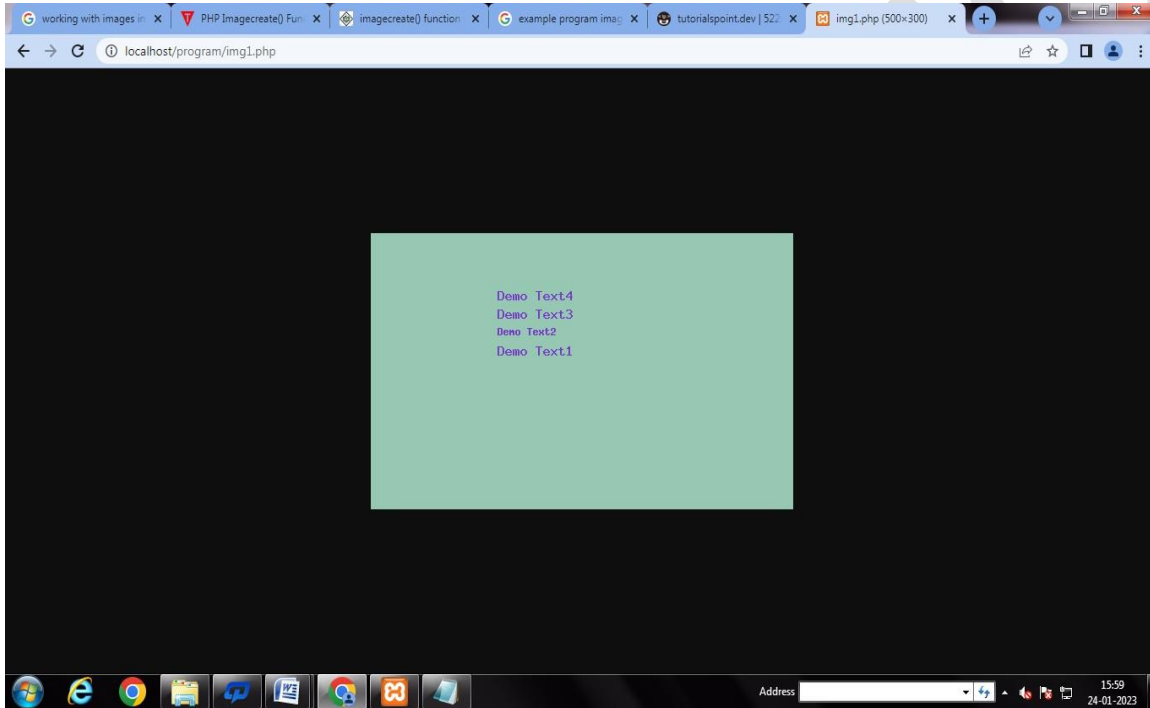


```

imagestring($img, 3, 150, 100, "Demo Text2", $fontcolor);
imagestring($img, 9, 150, 80, "Demo Text3", $fontcolor);
imagestring($img, 12, 150, 60, "Demo Text4", $fontcolor);
header("Content-Type: image/png");
imagepng($img);
imagedestroy($img);
?>

```

## OUTPUT:



Here in this program, we have declared various variables like **\$img** to define the size of the image that we require, **\$bgcolor** to define the color of background we require, **\$fontcolor** to define the color of text we require. We have used the **image string ( )** function to declare the string we want to display as an image. To display the output of the image, we have used an inbuilt PHP command **header** and **imagepng** to display on the browser.

## WRITE A PHP PROGRAM TO DRAW DIFFERENT IMAGES.

```

<?php
header("Content-type: image/png");
$img_width = 800;
$img_height = 600;
$img = imagecreatetruecolor($img_width, $img_height);

```

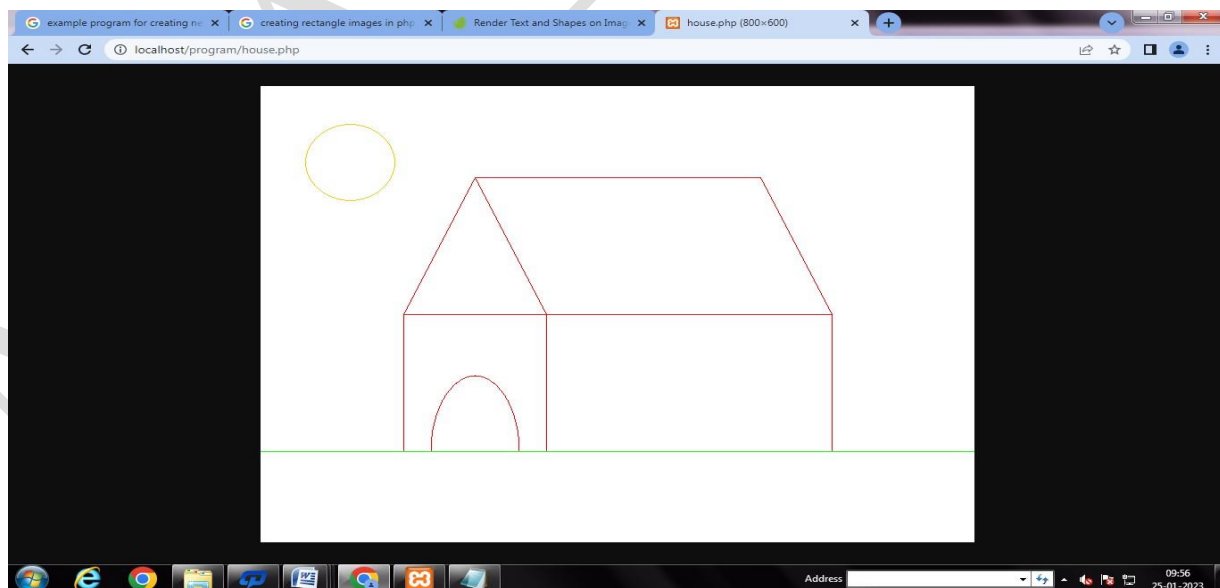
```

$black = imagecolorallocate($img, 0, 0, 0);
$white = imagecolorallocate($img, 255, 255, 255);
$red  = imagecolorallocate($img, 255, 0, 0);
$green = imagecolorallocate($img, 0, 255, 0);
$blue  = imagecolorallocate($img, 0, 0, 255);
$orange = imagecolorallocate($img, 255, 200, 0);
imagefill($img, 0, 0, $white);
imagerectangle($img,$img_width*2/10, $img_height*5/10, $img_width*4/10, $img_height*8/10,
$red);
imagerectangle($img,$img_width*4/10, $img_height*5/10, $img_width*8/10, $img_height*8/10,
$red);
imagepolygon($img,[$img_width*3/10,$img_height*2/10,$img_width*2/10,$img_height*5/10,$img
_width*4/10, $img_height*5/10], 3, $red);
imageopenpolygon($img,[$img_width*3/10,$img_height*2/10,$img_width*7/10,$img_height*2/10,
$img_width*8/10, $img_height*5/10], 3, $red);
imageellipse($img, 100, 100, 100, 100, $orange);
imagearc($img, $img_width*3/10, $img_height*8/10, 100, 200, 180, 360, $red);
imageline($img, 0, $img_height*8/10, $img_width, $img_height*8/10, $green);
imagepng($img);

```

?>

## OUTPUT



---

MCA - ATMECE

**CREATE AN IMAGE USING PHP FUNCTIONS:** PHP provides many functions to draw lines, rectangles, polygons, arcs, ellipses, and much more. The GD(Graphics Draw) library is utilized for dynamic picture creation. In PHP, we can easily use the GD library to make GIF, PNG, or JPG pictures quickly from our code. So there is no need to write HTML and CSS code. We can easily handle this using the PHP programming language. There are some PHP predefined functions used for generating images.

**1. PHP imagecreatetruecolor() function:** This function is used to create an image of the specified size. It takes width and height as parameters.

**Syntax:**

```
imagecreatetruecolor(int $width, int $height);
```

**2. PHP imagecolorallocate() function:** This function is used to fill the background colour of the image, and it returns an identifier for that particular colour. It takes four parameters. Image resource in the first parameter, and the other three parameters contain colour code in RGB component form.

**Syntax:**

```
imagecolorallocate(resource $image, int $red, int $green, int $blue);
```

**3. PHP imagefilledrectangle() function:** This function is used to draw a filled rectangle with the given start and end coordinates. It takes six parameters. In the first parameter, it takes image resource and in the next four parameters, it takes the rectangle coordinates and in the last parameter, it takes the color code to fill the rectangle.

**Syntax:**

```
imagefilledrectangle(resource $image, int $x1, int $y1, int $x2, int $y2, $fillcolor );
```

**4. PHP imagestring() function:** The imagestring() function is used to write text to the image. In the first parameter, it takes the image source, second parameter is the font size, which can be from 1 to 5 (smaller to bigger), third and fourth parameters are the coordinates from the left corner and the fifth parameter is the string to be written and the sixth parameter contains string color.

**Syntax:**

```
imagestring(resource $image, int $font , int $x , int $y , string $string , int $color );
```

**5. PHP imagesetthickness() function:** The imagesetthickness() function is used to set the thickness of line drawing. In the first parameter, it takes an image resource, and in the second parameter it takes the thickness size in pixels.

**Syntax:**

```
imagesetthickness(resource $image, int $size);
```

**6. PHP imagepng() function:** The imagepng() function is used to generate image in png format.

**Syntax:**

MCA-ATMECE

```
imagepng(resource $image);
```

**7. PHP imagedestroy() function:** The imagedestroy() function is used at last to free the allocated memory associated with the image.

**Syntax:**

```
imagedestroy();
```