UNIT-1

Topics:

Introduction to PHP: Declaring variables, data types, arrays, strings, operators, expressions, control structures, functions, Reading data from web form controls like text boxes, radio buttons, lists etc., Handling File Uploads. Connecting to database (MySQL as reference), executing simple queries, handling results, Handling sessions and cookies

File Handling in PHP: File operations like opening, closing, reading, writing, appending, deleting etc. on text and binary files, listing directories.

Introduction

What is PHP? What are the common uses of PHP? (2 marks May 2017)

- PHP stands for "PHP: Hypertext Preprocessor". Earlier it was called Personal Home Page.
- PHP is a server side scripting language that is embedded in HTML.
- The default file extension for PHP files is ".php".
- PHP supports all the databases that are present in the market.
- PHP applications are platform independent. PHP application developed in one OS can be easily executed in other OS also.

Common uses of PHP:

- PHP performs system functions, i.e. Using System function we can create, open, read, write & close Files.
- Using PHP we can create Dynamic Page Content i.e. anew Feature is added without disturbing the Old feature.
- Collect FORM data- User data is collected to Databasei.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete and modify elements within your database through PHP.
- Control Access- Using PHP, you can restrict users to access some pages of your website.
- Cookies- We can create,access cookies variables and set cookies.
- PHP can encrypt data.

Characteristics/Features of PHP:

- Cross Platform-PHP code will be run on every platform, Linux, Unix, Mac OS X, Windows.
- Cross Server- We can access data from different servers on our php application.
- Cross database- PHP supports all the database(Oracle, MySQL, SQLite etc.) that are present in the market.
- Interpreted- It is an interpreted language, i.e. there is no need for compilation.
- Open Source-Open source means you no need to pay for use php, you can free download and use.

PHP Basic Syntax-

Universal Style Tag A PHP script starts with **<?php** and ends with **?>**	<?php // PHP code goes here ?>
Comments in PHP	// This is single-line comment # This is also a single-line comment /* This is a multiple-lines comment block that spans over multiple lines */

Output Functions in PHP

In PHP there are two basic ways to get output: echo and print.

1. **The PHP echo Statement**

   The echo() function outputs one or more strings.

   The echo statement can be used with or without parentheses: echo or echo().

   Syntax: echo(*strings*)

   Example

   ```
   <?php
   echo "Hello world!";
   echo ("Hello world!");
   ?>
   ```

2. **PHP print() Function :**

   Syntax: print(*strings*)

   Example:

   ```
   <?php
   print "Hiee!";
   ?>
   ```

   echo and print are used to output data to the screen. The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

Printing Some Text

```
<html>
<head><title>Displaying TEXT from PHP</title></head>
<body>
<?php
echo 'Welcome to PHP.'; //echo statment using single quote
echo "Welcome to PHP."; //echo statment using double quote
echo ("Welcome to PHP."); //We can pass text inside parenthesis
?>
</body>
</html>
```

Mixing HTML and PHP

```
<html>          // Page starts with standard <html> <head> <title> section.
<head>
<title>
Using PHP and HTML together
</title>
</head>

<body>                  // <body> section
<h1> Using PHP and HTML together</h1>  //Contain <h1> header and some text
Here is PHP info: </br></br>
<?php      // <?php starts a PHP section
phpinfo(); //PHP function phpinfo(); displays information about the PHP installation, Every PHP statement ends with semicolon
?>
</body>
</html>
```

When this page is run by PHP engine on the server HTML will be passed through browser and PHP part will be executed

---

Variables :

Variables are "containers" that are used for storing information.
In PHP, a variable starts with the $ sign, followed by the name of the variable.

What are the rules for naming a variable in PHP?

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

Storing Data in Variables

PHP variables can hold numbers or strings of characters. We can store information in variables with an assignment operator/ single equal sign (=).

```
<?php
$name="ACE";
$age = 21;
?>
```

After the execution of the statements above, the variable $name will hold the value ACE**,** the variable $age will hold the value **21.**

Destroying Variable

Want to uncreate a variable? We can indeed uncreate a variable in PHP using unset() function. The unset() function destroys a given variable.

```
<?php
$car="TATA";
echo "Before unset(), My car is $car";  //Interpolating String: PHP will place the value held in $car, inside a double-quoted text String
unset($car);
echo "After unset(), My car is ", $car;
?>
```

Creating Variable Variables ($ and $$)

Lets create a variable named $name and set the value='Siri'. $name="Siri"
$$name uses the value of the variable $name
$$var is known as reference variable where as $var is normal variable.

```
<?php
$name="Siri";
$$name="Siri CSEA";
echo $name."<br/>";
echo $$name."<br/>";
echo $Siri;
?>
```

Data Types

Explain about various data types in PHP (3 marks)

In other programming language C, C++, JAVA, we need to specify the data type for each variable, but In PHP we don't have to specify data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value.

PHP supports the following data types:

1) Boolean Holds true/false values
2) Integer Holds non decimal numbers like -1, 0 5 and so on (Positive or negative)
3) String Holds text like "Welcome to PHP" (text inside quotes)
4) Float Holds floating point numbers (also called double) like 3.14159 or 2.7128 with decimal point
5) Array Holds array of data items
6) Object Holds Programming Objects
7) Resource Holds a data resource
8) NULL Holds a value of NULL

Program Example:

```
<?php
$x = "Hello world!";  //String in double Quote
$y = 'Hello world!';  //String in single quote
echo $x;
echo "<br>";
echo $y;
$num = 5985;          //Integer
var_dump($num);       //var_dump() Prints the data type and value
$fnum = 10.365;             //float
var_dump($fnum);      //var_dump() Prints the data type and value
$boovar = true;             //Boolean represents two possible states: TRUE or FALSE
echo $boovar;
$cars = array("Volvo","BMW","Toyota");   //array stores multiple values in one single variable
var_dump($cars);                         //var_dump() Prints the data type and value
$myval = null;              //NULL has no value assigned to it
var_dump($myval);
?>
```

Arrays

- An array is collection of multiple values stored under a single variable name.
- In PHP, we have 3 types of arrays:
    1. Indexed arrays - Arrays with a numeric index
    2. Associative arrays - Arrays with named keys
    3. Multidimensional arrays - Arrays containing one or more arrays

Creating an Array in PHP

We use the array construct to create an array in PHP.

$actor=array("Pavan","kalyan","mahesh","babu");

Modifying the data in Arrays

We have seen how to create arrays, Next how can we modify the data in the arrays?

Example: We have created an array:

$actor=array("Pavan", "kalyan", "mahesh", "babu");

Now we want to change $actor[0] from Pavan to Pawan

We do : **$actor[0] ="Pawan";**

If we want to add new actor, "Prabhas"

We do : **$actor[]="Prabhas"**

Deleting Array Elements:

we can remove an element from an array using unset function.

unset($actor[0]);

Handling Arrays with Loops:

1) **The for Loop**
2) **The print_r Function**
3) **The foreach Loop**
4) **The while Loop**

The for Loop

- The for loop is used to iterate over the elements of an array **when we know in advance the number of elements of an Array.**
- The **count() function** is used to return the length (the number of elements) of an array & this number is useful when we want to set the number of times the for loop to repeat.

Syntax

for (*init counter; test counter; increment/decrement counter*) {

code to be executed;

}

Program Example:

```
<html>
<head>
<title>
Using a for loop to loop over an Array
</title>
</head>
<body>
<h1>Using a for loop to loop over an Array</h1>
</br></br>

<?php
$cities=array("Hyderabad","Chennai","Delhi","Mumbai","Pune","Kolkata","Lucknow","Chandigarh");
$arrayLength=count($cities);

for($i=0; $i<$arrayLength;$i++)
{
        echo $cities[$i],"<br>";
}

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

The print_r Function

There is simple way to print the contents of an array using print_r function.

For example:

```
<?php
$cities=array("Hyderabad","Chennai","Delhi","Mumbai","Pune","Kolkata","Lucknow","Chandigarh");
print_r($cities);
?>
```

The foreach Loop

- The for each loop is designed to iterate over an elements of an Array/Associative array.
- We have two syntax to use foreach loop:
    1. **foreach (array as $value) statement**
    2. **foreach (array as $key => $value) statement**

The following example, puts the foreach loop to work, looping over the $cities array like:

```
<html>
<head>
<title>
Using a foreach loop to loop over an Array
</title>
</head>

<body>
<h1>Using a foreach loop to loop over an Array</h1>
</br></br>
<?php

$cities=array("Hyderabad","Chennai","Delhi","Mumbai","Pune","Kolkata","Lucknow","Chandigarh");

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

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

when foreach starts executing, the array pointer is automatically set to the first element of the array. On each iteration, the value of the current element is assigned to $value and the array pointer is incremented by one.

The second form of foreach loop lets us to work with keys as well as values.

- Element is a combination of element key and element value.
- PHP also support slightly different type of array in which index numbers are replaced with user defined string keys. This type of array is known as **Associative Array.**
- The keys of the array must be unique, each key references a single value. The key value relationship is expressed through **=> symbol.**

For example:

```
<html>
<head>
<title>
Using a foreach loop with keys and values in an Array
</title>
</head>

<body>
<h1>Using a foreach loop with keys and values in n Array</h1>
</br></br>
<?php

$details=array("name"=>"Siri","Type"=>"College","Place"=>"Hyderabad");

foreach ($cities as $key=>$value)
{
        echo "$key : $value<br>";
}

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

Note: If we don't provide an explicit key to an array element, The new key value depends upon the previous key.

```
<?php
$a=array(10,100=”ACE”,30);          //[0]=>10      [100]=>ACE    [101]=30
print_r($a);
?>
```

The while loop

- We can use while loop to iterate over an array.
- It tells PHP to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE.
- The value of the expression is checked each time at the beginning of the loop.
- if the while expression evaluates to FALSE the execution of nested statement will stop.

To handle multiple-item return value from the each function, we can use list() function .
list() function assign the two return value from each separate variable.

Example of using list() function:

```
<?php
$Language=array("PHP","XML","Servlet","JSP","JavaScript");
list($x, $y, $z)=$Language;
echo "We have covered $x, $y and $z.";
?>
```

Example (Displaying array using while):

```
<html>
<head>
<title>
Using a while loop with keys and values in an Array
</title>
</head>

<body>
<h1>Using a while loop with keys and values in n Array</h1>
</br></br>
<?php

$details=array("name"=>"ACE","Type"=>"College","Place"=>"Hyderabad");

while (list($key, $value)=each($details))
{
        echo "$key : $value<br";
}

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

PHP Array Function

array() - function creates and returns an array.
array_change_key_case() - function changes the case of all key of an array.
array_chunk() - function splits array into chunks. we can divide array into many parts.
count() - function counts all elements in an array.
sort() - function sorts all the elements in an array.
array_reverse() - function returns an array containing elements in reversed order.
array_search() - function searches the specified value in an array.
array_intersect() - function returns the intersection of two array. It returns the matching elements of two array.
array_push() — function pushes one or more elements onto the end of array.
array_pop() — function pops the element off the end of array.
array_merge() — function merge one or more arrays.
array_sum() — function calculate the sum of values in an array.

Converting Between String and Arrays Using implode and explode

- **The implode function** is used to "join elements of an array with a string".
- **implode()** accepts two argument, first argument the separator which specifies what character to use between array elements, and second one is array.

```
<?php
$ice_cream[0]="Chocolate";
$ice_cream[1]="Mango";
$ice_cream[2]="Strawberry";

$text=implode(" , ", $ice_cream);      //Outputs: Chocolate, Mango, Strawberry

?>
```

- The **explode function** breaks a string into an array.
- **explode()** accepts three argument, first one is the delimiter, second one is the strings that needs splitting, and third one is not mandatory.

```
<?php

$ice_cream="Chocolate, Mango, Orange";
$ice_cream=explode(" , ", $text);
print_r($ice_cream);  //Outputs: Array( [0]=>Chocolate  [1]=>Mango  [2]=>Strawberry)

?>
```

Handling Multidimensional Arrays

Aditya	25	23
Srikhar	24	22
Surya	22	21

We are keeping track of student mid exam score

```
<?php
$midScore=array(
        array("Aditya",25,23),
        array("Srikhar",24,22),
        array("Surya",22,21)
        )

echo $midScore[0][0].” “.[0][1].” ”.[0][2].”<br>”;
echo $midScore[1][0].” “.[1][1].” ”.[1][2].”<br>”;
echo $midScore[2][0].” “.[2][1].” ”.[2][2].”<br>”;
```

Moving through Arrays

PHP supports number of functions to move through arrays.

1. **current()** – to get current element
2. **next()** – to get next element
3. **prev()** – to get the previous element
4. **end()** – to get the last element

```
<?php
$ice_cream=array(“chocolate”,”strawberry”,”butterscotch”);
echo “current element: “,current($ice_cream),”<br>”;
echo “next element: “,next($ice_cream),”<br>”;
echo “previous element: “,prev($ice_cream),”<br>”;
echo “Last element : “,end($ice_cream),”<br>”;
?>
```

Strings

The String functions

- **explode() - Breaks a string into an array**
- **implode() - Returns a string from the elements of an array**
- **str_replace() - Replaces characters in a string**
- **strrev() - Reverses a string**
  Example: echo strrev("Hello world!"); // outputs !dlrow olleH

- **strtolower() - Converts a string to lowercase letters**
  Example: strtolower("HELLO WORLD");// outputs hello world
- **strtoupper() - Converts a string to uppercase letters**
  Example: strtoupper("hellow World");// outputs HELLO WORLD
- **strlen() - Returns the length of a string**
  Example: echo strlen("Hello world!"); // outputs 12
- **str_word_count() - counts the number of words in a string**
  Example: echo str_word_count("Hello world!"); // outputs 2

Operators

Operators supported by PHP are:

1. Math operators
2. Assignment operators
3. Increment/Decrement operators
4. String operators
5. Comparison operators
6. Logical operators

PHP's Math Operators

Operator	Name	Example	Description
+	Addition	$x + $y	Sum of x and y
-	Subtraction	$x - $y	Difference of x and y
*	Multiplication	$x * $y	Product of x and y
/	Division	$x / $y	Quotient of x divided by y
%	Modulus	$x % $y	Remainder of x divided by y
**	Exponentiation	$x ** $y	Result of x raised to the power of y

PHP's Assignment Operators

The main **assignment operator is the = operator**, which just assigns a value.
Example: To store the value 99 in a variable $number: **$number = 99;**
By using = operator, we can **make multiple assignments** on the same line:
Example: **$numOne = $numTwo = $numThree = 99;**
Assignms the value 99 to three variable $numOne, $numTwo and $numThree

PHP gives us a set of Combination Assignment Operator

Assignment	Description
x += y	Adding x and y and store the result in x
x -= y	Subtracting y from x and store the result in x
x *= y	Multiplying x and y and store the result in x

Incrementing and Decrementing Operators

Increment operators increments the value of a variable and Decrement operator's decrements the value of a variable.

++$a Pre-increment
$a++ Post-increment
--$a Pre-decrement
$a-- Post-decrement

PHP's string Operators

PHP has two operators that work with Strings:

1. The concatenation Operator (.)
2. The combined concatenation assignment operator (.=)

Example:

```
<?php
echo "Hello"."Wolrd"; //Output: Hello World (Hello and World are concatenated)
$str1="Hello";
$str2="World";
echo $str1.=$str2; //str2 is appended to str1
?>
```

The PHP Comparison operators

We use PHP Comparison Operators to compare two values:

== Is equal to
!= Is not equal to
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to

The PHP Logical operators

We use Logical operators when we want to test more than one condition at a time.
Logical operators supported by PHP are **and, or, not.**

&& and operator [need both values to be true]
|| or operator [need one of our conditions to be true]
! not operator

Using logical operator to combine conditions.

Example: both username and password values needs to be true in our test.

```
if( $username =='user' && $password =='password')
```

Expressions

The **most basic forms of expressions** is "$num = 5", we are assigning '5' into variable $num.

```
$num=5
```

Another example of expression is pre- and post-increment and decrement.

Example:

```
$c = $num++;   /* post-increment, assign original value of $num (5) to $c */
$d = ++$num;   /* pre-increment, assign the incremented value of $num to $d */
```

One more type of expressions are **comparison expressions.**

These expressions evaluate to either FALSE or TRUE.

PHP supports: > (greater than) >= (greater than or equal to) == (equal)
!= (not equal) < (less than) <= (less than or equal to)

Control Structures/ Flow Control

Using the if Statement

if statement is a conditional statement that allows us to make decisions on what code to execute.

The structure looks like**:**

```
if (expression)
  statement
```

Here, expression evaluates to a TRUE or FALSE

If expression is TRUE, The statement that follows is executed; if it is FALSE, statement is not executed
We use conditional and Logical Operators to create expression of if statement.

Example: We want to display some text if the outside temperature is above 28 degree.

```
<?php
$temperature=35;
if($temperature>28)
{
        echo "Its hotter outside";
}
?>
```

Example: To check how many minutes someone has been in the pool- if its more than 30minute, it's time to get out.

```
<html>
<head>
<title> Using the if statement</title>
</head>
<body>
<h1>Using the if Statement</h1>
<?php
$minutes=31;
if($minutes > 30)
{
        echo "Your time is UP!!<br>";
        echo "Please get out of the pool.";
}
?>
</body>
</html>
```

The else statement

The else statement

Often we want to execute a statement if a condition is TRUE,
and we execute different statement if the condition is FALSE.

```
if (expression) {
   code to be executed if condition is true;
} else {
   code to be executed if condition is false;
}
```

Example: We want to print a message if the outside temperature is outside the range 24 degree and 32 degree, and another message if the temperature is inside the range.

```
<html>
<head>
<title> Using the else statement</title>
</head>
<body>
<h1>Using the else Statement</h1>
```

```
<?php
$temperature=44;
if($temperature<24 || $temperature>32)
{
        echo "Better Stay inside today!!";
}
else
{
        echo "Nice weather outside";
}
?>
</body>
</html>
```

The elseif statement

elseif, as its name suggests, is a combination of if and else.
If an if statements condition is false, we can make additional tests with elseif.

```
if (condition1) {
   code to be executed when condition1 is true;
} elseif (condition2) {
   code to be executed when condition2 is true;
} else {
   code to be executed when conditions are false;
}
```

Example:

```
<html>
<head>
<title> Using the elseif statement</title>
</head>
<body>
<h1>Using the elseif Statement</h1>
<?php

$a=40;
$b=30;
if ($a > $b)
{
  echo "$a is bigger than $b";
```

```
} elseif ($a == $b)
{
 echo "$a is equal to $b";
} else {
 echo "$a is smaller than $b";
}

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

If the if statement's conditional expression is false → It will check the first elseif statements expression, → **if its true** elseif code is executed and **if its false** it moves to the next elseif statement and so on.
At he end else statament is executed if the no other code is executed in the if statement up to this point.

The switch statement:

- switch statement lets us replace long if-elseif-else ladder of condition checking with switch statement.
- switch statement compares a value against case statement and execute a code block whose value matches the case value.
- if no case value matches the value, default statament is executed.
- break statementends execution of the switch statement, if we don't write break statement, execution will continue with the code.

Syntax:

```
<?php
switch(value){
        case value1:
                // code block 1
                break;
        case value2:
                // code block 2
                break;

        default:
                // default code block
                break;
}
```

Example:

```
<html>
<head>
<title> Using the switch statement</title>
</head>
<body>
<h1>Using the switch Statement</h1>
<?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!";
}

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

Using for loop

- for loop is a repetition control structure that allows us to execute a block of code a specified number of times.
- When we want to execute same block of code over and over again specified number of times.
- Syntax:

```
for( expression1 ; expression1 ; expression1 )
                statement
```

  - First, for loop executes expression1; then it checks the value of expression2 - if true , loop executes statement once.

- Then it executes expression3 and after that checks the expression2 again-- if true loop execute statement once again.
- Then loop executes expression3 again, and keeps going untill expression2 becomes false.

Example:

```
<html>
<head>
<title>Using the for loop</title>
</head>
<body>
<?php
for ($x = 0; $x <= 10; $x++) {
   echo "The number is: $x <br>";
}
?>
</body>
</html>
```

for loop can handle multiple loop counters by seperating them with comma operator.

```
<?php
for ($i=2, $k=2; $i<6 && $k<6 ; $i++, $k++) {
   echo "$i  *  $k = ",$i *$k,"<br>";
}
?>
```

The two variables $i and $k are initialized with 2.
At the end of each each loop $i and $k will be incremented by one.

We can put one loop inside a another loop, call as nested loop.

```
<?php
for ($row = 1; $row <= 5; $row++)
{
   for ($col = 1; $col <= 5; $col++)
   {
      echo '*';
   }
   echo "\n";
}
?>
```

Outputs:

*****

*****

*****

*****

*****

Using While Loop

- When we want to execute same block of code over and over again as long as the condition is true, we go for while loop.
- Syntax:

```
while (expression)
{
        statement
}
```

  - While expression is true, the loop executes statement. When expression is false, the loop ends.

Example:

```
<html>
<head>
<title>
Using the while loop
</title>
</head>
<body>
<h1> Using the while loop </h1>
<?php
$x=1;

while($x<10)
{
        echo "Now \$x holds : ",$x,"<br>";
        $x++;
}

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

Using do... while loop

- "do while" loop is a slightly modified version of the while loop
- In while loop, the condition is checked at the beginning of each iteration, in do-while condition is checked at the end of each iteration.
    - It means do-while loop always executes its block of code at least once even the condition is evaluated to false.

while vs. do-while

- do-while loop body executes at least once even when condition is false whereas while loop body may not execute at all if condition evaluates to false.
- The condition in the do-while loop is evaluated at the end whereas the condition in the while loop is evaluated at the beginning of each iteration.

Syntax:

```
do
{
        statement
}
  while (condition);
```

- do... while loop keeps executing statement while condition is true- note expression is tested at the end.

Example:

```
<html>
<head>
<title>Using the do...while loop</title>
</head>
<body>
<h1> Using the do...while loop </h1>
<?php
$x=1;
do
{
        echo "Now \$x holds : ",$x,"<br>";
        $x++;
}  while($x<10);
?>
</body>
</html>
```

Using the foreach loop

- The foreach loop is especially designed to work with array.
- foreach loop will loop over all elements of an array.
- for loop and while Loop will continue until some condition fails, the foreach loop will continue until it has gone through every item in the array.

- **There are two syntax:**

    - foreach (array_expression as $value) statement
    - foreach (array_expression as $key => $value) statement

Example:
[DO CHECK ARRAY CONCEPT]

Terminating loop early [break]

- We can stop a loop or switch statement at any time with the break statement.
- break ends the execution of for, foreach, while, do-while or switch statement.
- If we use break inside inner loop, it breaks the execution of inner loop only.

```
<html>
<head>
<title> Using the break statement</title>
</head>

<body>
<h1> Using the break statement</h1>
<?php
for ( $x=0 ; $x < 100 ; $x++)
{
        echo "I 'm going to do this hundred times!<br>";
        if ($x == 10)
                {
                echo "Alright, i'm quitting,<br>";
                break;
                }
}
?>
</body>
</html>
```

Skipping iteration [continue]

Sometimes, we want to skip an iteration of a loop to avoid some kind of problem like division by zero-- we can do that with continue statement.

continue is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and proceed to the beginning of the next iteration.

break ends a loop completely, continue just shortcuts the current iteration and moves on to the next iteration.

```
while ($foo) {   <--------------------┐
   continue;    --- goes back here ---┘
   break;       ----- jumps here ----┐
}                                    |
            <--------------------┘
```

break exits the loop in which we are in, continue starts with the next cycle of the loop immediatly.

```
<html>
<head>
<title>Using the continue statement</title>
</head>
<body>
<?php
$x=1;
for( $x=-2; $x < 3 ; $x++ )
{
        if ($x == 0)
        {
                continue;
        }

        echo "1/$x = ",1/$x,"<br>";
        $x++
}

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

Functions

- Now that you should have learned about variables, loops, and conditional statements it is time to learn about functions.
- When we have a block of code that needs to be run multiple times in our application, we put that block of code into a function.
- A function is a block of code (that performs a task) that can be executed whenever we need it.
- Advantages of using functions is, it reduces the repetition of code within a program.

Creating functions in PHP

- while creating a function its name starts with keyword '**function**', followed by the name of the function we want to create followed by parentheses i.e. ()
- We put our function's code inside { and }

Syntax:

```
function functionName()
{
   code to be executed;
}
```

functionName starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

Example:

```
<?php
function display() {
   echo "Hello world!";
}

display(); // call the function
?>
```

Passing Functions Some Data

- Data can be passed to functions via the argument list/ Parameter list.
- We can pass data to functions so functions can operate on that data.
- We can define a function to accept values through parameters.
- A parameter appears with the parentheses "( )" separated by comma.

Syntax:

```
function functionName([parameter_list...])
{
   [statements];
}
```

Example: If we want to pass the person's name, and our function will concatenate this name onto a greeting string.

```
<html>
<head>
<title> Passing data to functions</title>
</head>
<body>
<?php

function display($name)
{
   echo "Hello $name !!";
}

display("Aditya"); // call the function

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

Passing Arrays to Functions

We can pass arrays to functionsis very simple.

Example:Define a function to find total of student's test scores and displays the total.

```
<html>
<head>
<title> Passing arrays to function </title>
</head>
<body>
<h1> Passing arrays to function </h1>

<?php
$scores=array(90,40,65,98,75);        //First create an array of test score

//Now, Lets define an total function:
function total($array)
{
        $totalmarks=0;
        //we use foreach statement to loop over the array
        foreach($array as $value)
```

```
        {
        $totalmarks + = $value ;
        }
}

echo " The total marks: ",total($scores); //Call the function: total($scores)
?>
</body>
</html>
```

Passing By Reference:

- If we pass a variable to a function, a copy is made of that variable and that copy is passed to the function.
- If we actually want to pass the real variable to function, we can pass by reference with an **ampersand (&).**

Example:

```
<?php
$value=4;
echo "Before the call value hold $value<br>";
squarer($value);
echo "after the call value hold $value <br>";
function squarer(&$number)
{
$number *= $number;
}
?>
```

Using default argument:

- If our function takes two arguments and when we pass one argument we get warning.
- We can fix this by supplying a default argument.
- Default argument will be used automatically when argument is missing.

Example:

```
<?php
display("Hello","ACE");
function display($greeting,$message="there")
{
echo $greeting;
echo $message;
}
?>
```

Returning Data from functions

We can create function that returns data using **return** statement.

Example: Create a function named adder that returns the sum of two numbers.

```
<?php
$num1=10;
$num2=20;
echo "Sum of $num1 + $num2 is :",adder($num1,$num2);
function adder($val1,$val2)
{
$sum=$val1+$val2;
return $sum;
}
?>
```

Introduction to variable scope in PHP

- Scope refers to the visibility of variables means which parts of your program can see or use it -inside a function(local variables) or Outside of all functions(global variables)
- Avoids the conflict between the variables.

Example: two variables named $value, one inside the function and other outside the function are independent of each other.

```
<?php
$value=4;
echo "Value out side the function $value <br>";
Myfunction();
function myfunction()
{
$value=800;
Echo "value inside the function $value<br>";
}
?>
```

Accessing Global Data

- What if we actually want to access $value (data outside function) in inside a function.
- We can access the data inside a function with the global keyword.

Example:

```
<?php
$value=4;
echo "Outside the function value is $value<br>";
```

```
myfunction()
{
global $value;
echo "inside the function value is $value";
}
?>
```

Working with static variables

One issue with function is that variable inside function reset every time we call function.

```
<?php
echo "Now the count is :",count_function(),"<br>";
echo "Now the count is :",count_function(),"<br>";
echo "Now the count is :",count_function(),"<br>";

function count_function()
{
$counter=0;
$counter++;
return $counter;
}
```

- If we want to preserve the variable value between function calls i.e if we want those variable to retain their value
- We do by declaring the $counter variable in counter_function as static means it will preserve its values between function call

```
<?php
    echo "Now the count is :",count_function(),"<br>";
    echo "Now the count is :",count_function(),"<br>";
    echo "Now the count is :",count_function(),"<br>";

    function count_function()
    {
    static $counter=0;
    $counter++;
    return $counter;
    }
```

PHP conditional function

- We can define a function in conditional statement like if statement.
- Conditional function doesn't exist untill the if statement executed. So make sure conditional function is called after it exist

```
<?php
        $create_function=true;

        if($create_function)
        {
        function conditional_function()
        {
        echo "Hellow from conditional function<br>";
        }
        }
        conditional_function();
        ?>
```

Nesting functions

- We can nest functions in PHP.
- We can create a function inside another function.
- Note: we can't call nested function untill the enclosing function has been run.

```
<?php
        outer_function();
        inner_function();

        function outer_function()
        {
        echo "inside outer function<br>";

        function inner_function()
        {
        echo "inside inner function";
        }
        }
?>
```

We call outer_function and then inner_function – in order

Creating include files (.inc)

- PHP allows us to create include files whose content will be inserted into our code.
- Lets define a constant in file premium.inc

```
<?php
        define("premium",176.93);
?>
```

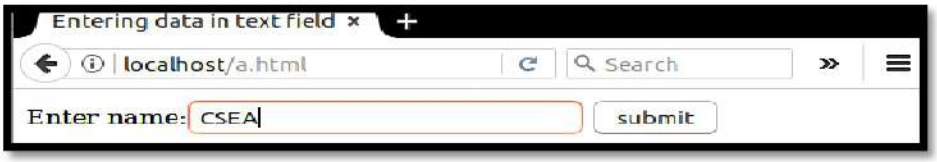
- Now we can include premium.inc in our code using include statement.

```
<?php
        include("premium.inc");
        echo "you pay Rs. ",premium," per month<br>";
?>
```

Reading data from web form controls

Handling text fields

- Text Fields are used for single line of text.
- a.html have a text field and submit button in a form



---------a.html----------

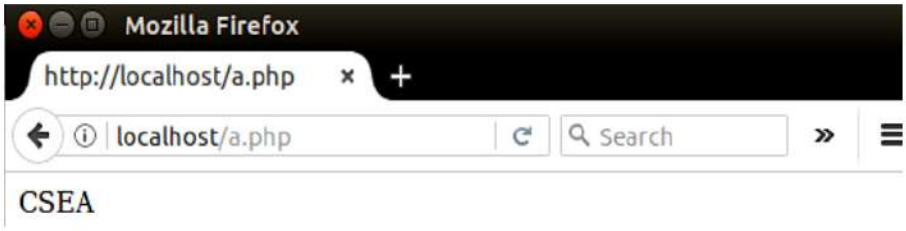
```
<html>
<head>
<title>Entering data in text field</title>
</head>
<body>
<form method="POST" action="a.php">
Enter name:<input type="text" placeholder="Enter name" name="nam"/>
<input type="submit" value="submit"/>
</form>
</body>
</html>
```

Once we enter our name on this web page, name in the text field will be stored in "nam". When we click the submit button, HTML page send its data to PHP a.php.

- PHP script echo the name the user entered into the data text field.
- Data from HTML page will be stored in a variable $_POST['nam']
- We can access the text from html form as $_POST['nam'] in the script

----------a.php----------

```
<html>
<body>
<?php
$name=$_POST['nam'];
echo $name;
?>
</body>
</html>
```



Handling Text Areas

- Text areas are used for multiline text input.
- a.html presents the user with a text area and asks address to enter.

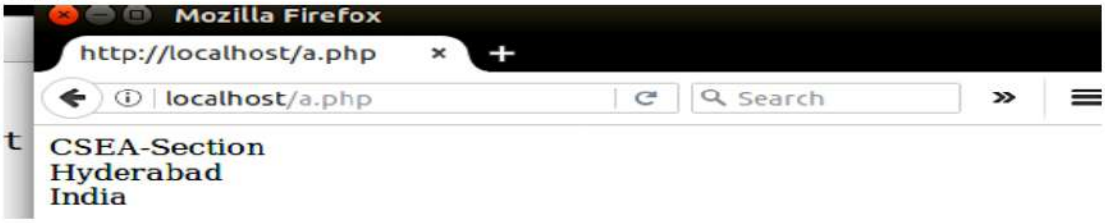
---------a.html----------

```
<html>
<head>
<title>Entering data in text area</title>
</head>
<body>
<form method="POST" action="ace.php">
Address:<textarea rows='3' cols='30' name="add"></textarea>
<input type="submit" value="submit"/>
</form>
</body>
</html>
```

- Data entered by the user in the text area named "add" will be sent to a.php on clicking submit button
- We can access the text from the text area as $_POST['add'] in the below script.
- Note: with the text area, multiline text will be filled with "\n". When we display the text browser will ignore newlines so we use **str_replace()** to replace "\n" with

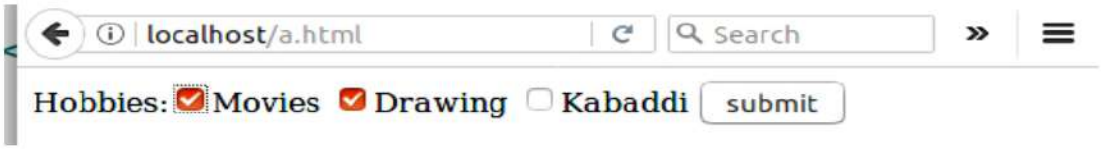
----------a.php----------

```
<html>
<body>
<?php
$address=$_POST['add'];
echo str_replace("\n","<br>",$address);
?>
</body>
</html>
```



Handling checkboxes

- We create check boxes with the <input> element, here we are asking user to select hobbies.
- Check boxes are good if we want the user to select multiple items from a number of choice.
- We can click multiple check boxes.



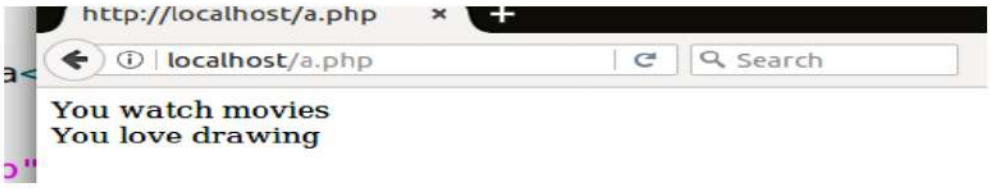
---------a.html----------

```
<html>
<head>
<title>Entering data in check boxes</title>
</head>
<body>
<form method="POST" action="a.php">
Hobbies:<input type="checkbox" name="hobbie1">Movies
<input type="checkbox" name="hobbie2">Drawing
<input type="checkbox" name="hobbie3">Kabaddi
<input type="submit" value="submit"/>
```

```
</form>
</body>
</html>
```

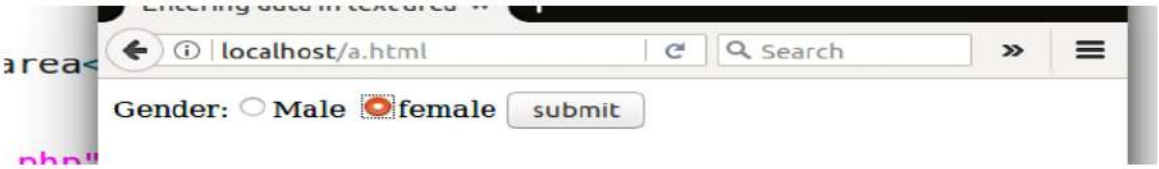
- We read the data from the check boxes, hobbie1 hobbie2 and hobbie3 using **$_POST[‘hobbie1**]...
- We check if checkbox is clicked then we display the data.

```
---------a.php--------------------
<html>
<body>
<?php
if(isset($_POST['hobbie1']))
{
echo "You watch movies<br>";
}
if(isset($_POST['hobbie2']))
{
echo "You love drawing<br>";
}
if(isset($_POST['hobbie3']))
{
echo "You play kabaddi very well<br>";
}
?>
</body>
</html>
```



Handling Radio Buttons

- We use radio button, If we want user to select ONLY ONE item from a number of choice
- Radio button allow user to make one selection at a time.
- Here in an example, asks the user if they are male or female.



---------a.html--------------------

```
<html>
<head>
<title>Entering data in Radio buttons</title>
</head>
<body>
<form method="POST" action="a.php">
Gender:<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="female">female
<input type="submit" value="submit">
</form>
</body>
</html>
```

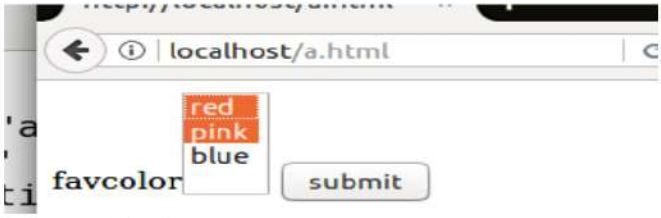
- Data from HTML page will be stored in a variable $_POST[‘gender’]
- We will print the gender of user which radio button was clicked.

---------a.php--------------------

```
<html>
<body>
<?php
if($_POST['gender']=='male')
{
echo "Male<br>";
}
if($_POST['gender'])
{
echo "female";
}
?>
</body>
</html>
```

Handling List Boxes

- Lets say we want user to select their favourite color by creating a list box with <select>.
- To let the user select multiple colors, we add multiple attribute in <select> element.
- As it is multiple selection we give the cntrol the clr[ ] an array not just clr.
- To add colors we put as <option> element in HTML



---------a.html----------

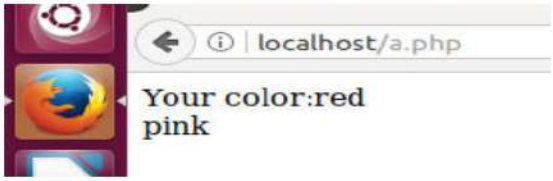
```
<html>
<head>
</head>
<body>
<form method="POST" action="a.php">
favcolor<select name='clr[]' multiple>
<option value="red">red</option>
<option value="pink">pink</option>
<option value="blue">blue</option>
</select>
<input type="submit" value="submit">
</form>
</body>
</html>
```

- **Clr [] is an array, not a single variable so we use foreach loop to display the color selection like this:**

----------a.php----------

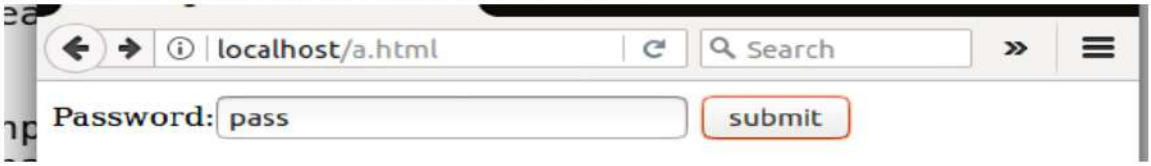
```
<html>
<head>
</head>
<body>
<?php
echo "Your color:";
foreach($_POST['clr'] as $value)
{
echo $value."<br>";
}
?>
```

</body>
</html>



Handling Password

- We usually check password on the server, giving user the access to resources if user enters correct password.
- Lets make an example that asks the user for password.



---------a.html----------

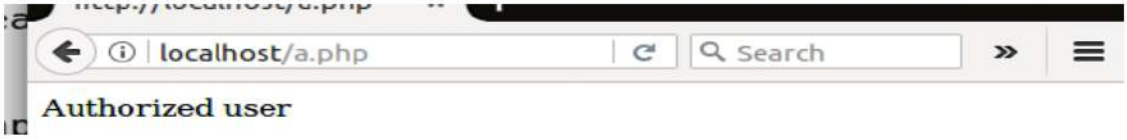
```
<html>
<head>
<title>Entering password</title>
</head>
<body>
<form method="POST" action="a.php">
Password:<input type="passwrd" name="pwd">
<input type="submit" value="submit">
</form>
</body>
</html>
```

- The data from html will be stored in global variable $_POST['pwd'].
- We check $_POST['pwd'] against the password, which is 'pass' and if matches display Authorized user to the user.

----------a.php----------

```
<html>
<body>
<?php
if($_POST['pwd']=='pass')
{
echo "Authorized user <br>";
}
else
{
```

```
echo "Invalid user";
}
?>
</body>
</html>
```



Handling File upload

- In syllabus we have topic 'Handling File Uploads' related to uploading file from web page.
- We set the <form> element an additinal attribute enctype(encoding type) to 'multipart/form-data.
- We use file upload control **<input type="file">**

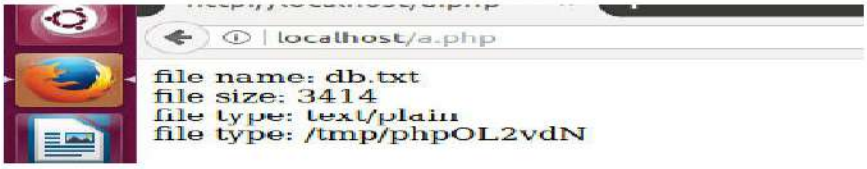
---------a.html----------

```
<html>
<head>
<title>Entering data with file upload</title>
</head>
<body>
<form method="POST" action="a.php" enctype="multipart/form-data">
<input type="file" name="userfile" />
<input type="submit" value="upload">
</form>
</body>
</html>
```

We use $_FILES array in PHP to handle uploaded files.

$_FILES have elements:

1) $_FILES['userfile']['name'] Gives name of file
2) $_FILES['userfile']['size'] Gives the size of uploaded file
3) $_FILES['userfile']['type'] Gives the type of file
4) $_FILES['userfile']['tmp_name'] Temporary file name stored on the server



```
----------a.php--------------------
<html>
<body>
<?php
//print_r($_FILES['userfile']);
echo "file name: ".$_FILES['userfile']['name'];
echo "<br>file size: ".$_FILES['userfile']['size'];
echo "<br>file type: ".$_FILES['userfile']['type'];
echo "<br>file type: ".$_FILES['userfile']['tmp_name'];
?>
</body>
</html>
```

- We open a file using fopen function, Opening a file gives access to the data in that file.
- The name of the file that we open is $_FILES['userfile']['tmp_name'] with read mode.
- Now we have opened the next we read the data from file using fgets method.
- fgets() method reads a single lines and stores in $text variable and displays on the browser.
- While loops over the file's content till end of file.

```
----------a.php--------------------
<html>
<body>
<?php
$file=fopen($_FILES['userfile']['tmp_name'],'r');
while(!feof($file))
{
$text=fgets($file);
echo $text."<br>";
}
?>
</body>
</html>
```

Connecting to database (MySQL as reference), executing simple queries, handling results

Connecting to MySQLDatabase Server

- In order to store or access the data from database, first we need to connect to the database server.
- In PHP we can easily connect to database server by using **mysqli_connect()** function.

  $connection = mysqli_connect('localhost','root','admin');

  mysqli_connect() will return an object if connection is succesful, otherwise FALSE.

Closing the Database Connection

- Once we have done all our queries, we can Close the opened database connection with **mysqli_close() function**

  mysqli_close($connection);

Creating a MySQL Database in PHP

- Once we have established the connection, we create the database using the SQL CREATE DATABASE statement.
- Syntax:

  CREATE DATABASE database_name
- So,first we make a SQL query using the **CREATE DATABASE** statement:
  Example: Query to create a new sale database-

  $query="CREATE DATABASE sale";
- Next,we execute this SQL query using PHP **mysqli_query()** to create table-

  $result=mysqli_query($connection , $query);

Connecting to the database

- Once we are created a Database on Database Server, Next step is to select the Database that we want to be used when performing queries.
- **mysqli_select_db()** function is used to selects the database
- **Syntax**

  mysqli_select_db($connection,'dbname');

  mysqli_select_db() function takes 2 parameter, First the connection object & Second the name of the database.

  mysqli_select_db() function returns TRUE on success or FALSE on failure.

Creating a New Table

- Database contains Tables.
- After creating a database, it's time to create some tables inside the database that will actually hold the data.
- A table organizes the information into rows and columns.
- We create a table using the **SQL CREATE** statement.

- So,First we make a SQL query using the CREATE TABLE statement:
  SQL syntax to create a table-
  CREATE TABLE table_name (column_name column_type);
- Example: Query to create a new fruit table with name and number fields-
  $query="CREATE TABLE fruit(name VARCHAR(20),number VARCHAR(20))";
- Next,we will execute this SQL query by using the PHP **mysqli_query()** function to create our table-
  $result=mysqli_query($connection , $query);

Inserting New Data into a Table

- After creating a database and tables, Next we insert records into a table.
- To insert a new Record into a table, we use SQL **INSERT INTO** statement.
  Syntax
  INSERT INTO table_name (column1, column2,...) VALUES (value1, value2,...);
  Table_name specify a table to which you want to insert data after the INSERT INTO clause
  list of columns separated by comma
  list of values separated by comma after the VALUES clause
- Example: SQL Query to insert a new record for Apples into the fruit table.
  $query="INSERT INTO fruit(name, number) VALUES ('Apples','200')";
- Lets execute the above SQL query using the PHP **mysqli_query()** function.
  $result=mysqli_query($connection , $query);

Next, The goal is to **READ and DISPLAY the database table named FRUIT**

Reading the Table

- The SELECT statement is used to select records from a table.
- The general form of the SELECT statement is:
  SELECT DISTINCT column_names FROM table_name
  WHERE condition ORDER BY how_to_sort
- Let's make a SQL query using the SELECT statement-
  Example: SQL Query for getting all records from the fruit table:
  $query="SELECT * FROM fruits";
- Next, we execute this SQL query using the PHP **mysqli_query()** function to retrieve the table data.
  $result=mysqli_query($connection , $query);
- Data returned by the mysqli_query() function is stored in the $result variable

Displaying the Table Data

- After reading the data from the table, we'll display the data from fruit table in an HTML table.
- We have two fileds in the fruit table are 'name' and 'number', So we create a HTML table with table header Name and NUmber.

```
echo "<table border='1'>";
echo "<tr>";
echo "<th>Name</th><th>Number</th>";
echo "</tr>";
```

- Next, To extract the records from the table, we use the **mysql_fetch_array()** function.

```
mysql_fetch_array('$result');
```

- SO, we use **mysql_fetch_array** to fetch rows from the fruit table and we loop over these rows with the while loop.
- To access the name filed in that row we write **$row['name'],** and for number we write as **$row['number'].**

```
while($row = mysql_fetch_array('$result'))
{
echo "<tr>";
echo "<td>",$row['name'],"</td><td>",$row['number'],"</td";
echo "</tr>";
}
```

- Each time mysqli_fetch_array() is invoked, it returns the next row from the result set.

Updating Database

- Lets say someone buys your products online, So we need to update our database.
  Example: Someone purchased 5 apple, means we should update our apple stock from 200 to 195.
- We use an **SQL UPDATE** statement to change or modify the existing records in a table.
- SQL UPDATE statement is used with the WHERE clause, to change only those records that matches specific condition.
- The basic syntax of the UPDATE statement:
  UPDATE table_name SET column=value WHERE column_name=some_value
- Let's make a SQL query using the UPDATE statement and WHERE clause-
  $query="UPDATE fruit SET number=195 WHERE name='Apples'";
- Now we execute this query with mysqli_query() function to update the tables records.

```
$result=mysql_query($connection , $query);
```

Deleting Records

- Now lets say the season of Apple is gone, So we need to remove from being offered on our website.
- **How do we delete a record?**
  We use SQL DELETE statement to delete records from a table.

- SQL DELETE statement is used with the WHERE clause to delete only those records that matches specific condition.
- The basic syntax of the DELETE statement:
  DELETE FROM table_name WHERE column_name=some_value
- Let's make a SQL query using the DELETE statement and WHERE clause-
  Example: SQL Query to remove the Apples record from the fruit table.
  $query = " DELETE FROM fruit WHERE name='Apples'";
- Next, we execute this query with the PHP **mysqli_query()** function to delete the 'Apples' records.
  $result=mysql_query($connection , $query);
  This will delete the 'Apples' Records.

Handling Sessions and cookies

Difference between session and cookie

1. Cookie is a small amount of information stored at client side.
   Session data are stored at server side.
2. cookie can store LIMITED Amount of data (max 4KB)
   Session-Amount of data to be stored is NOT LIMITED
3. cookies can store only STRING type of data.
   session store data as OBJECTS
4. cookie data is less secure
   session data is more secure

5. Cookie data is available up to expiration date.
   Session data is available for the browser run.
   We lose the session information when we close the browser.

Setting a cookie

- We set cookies on the user's machine with the PHP setcookie() function
  setcookie(name, value, expire);
  Here , Name of the cookie, Value content that we actually want to store, Expiry a expiration time in seconds after which cookie will automatically expire/become inaccessible

```
<html>
<head>
<title> Setting a cookie</title>
</head>

<body>
```

```
<?php
setcookie("name","ACE Engineering college",time()+120);
echo "Done!!";
?>
</body>
</html>
```

Reading a cookie

- Once cookie is set, when we load a page the cookie is sent to the server from the user's machine. We can access the cookie on the next page load with the $_COOKIE[] array.

```
<html>
<head>
<title> Reading a cookie</title>
</head>

<body>
<?php
$college=$_COOKIE['name'];
echo $college;
?>
</body>
</html>
```

Deleting cookies

- We can set the expiration time of a cookie to some time in the past and that will make the browser delete cookie.

```
<html>
<head>
<title> Deleting a cookie</title>
</head>

<body>
<?php
setcookie("name","ACE Engineering college",time()-120);
echo "Deleted!!";
?>
</body>
</html>
```

Storing data in Sessions

- The way of storing data on the server to be used across multiple pages is to use SESSION.
- SESSION store information about the user and browser remembers the information to all pages.
- To work with session, we start session by calling **session_start()**
- To store data in the session we use **$_SESSION[] array**

```
<html>
<head>
<title> Storing data in sessions</title>
</head>

<body>
<?php
session_start();
$_SESSION['user']='Sachin';
$_SESSION['purchase']='12000';
?>
</body>
</html>
```

Retrieving data from sessions

```
<html>
<head>
<title> Retrieving data from sessions</title>
</head>

<body>
<?php
session_start();
echo "thank you, ".$_SESSION['userID']." You have purchased
".$_SESSION['purchase']." worth";
?>
</body>
</html>
```

Writing a Hit counter using Sessions

- Example of showing how session preserve data between page accesses.
- How many times the user has been to a web page.
- We store data in the session key "count" set to 0.
- When the user reloads the page, we increment the value stored in the session.

```
<html>
<head>
<title> A Session hit counter</title>
</head>

<body>
<?php
session_start();

if(isset($_SESSION['count']))
{
$_SESSION['count']++;
}
else
{
$_SESSION['count']=0;
}
echo "Page views:".$_SESSION['count'];
?>
</body>
</html>
```

File Handling in PHP

PHP provides inbuilt functions to read the contents of a file and writes contents in a file.

If we want to read and write a file first we have to open a file by using "Mode".

Opening Files Using fopen

fopen() function opens a file or URL.

Syntax:

fopen ($filename ,$mode)

fopen() takes two parameters: First, the file to open, and Second, the mode - how we would like it opened.

The possible modes are:

- **r (read)** : This mode is used to read the contents of a file.
- **w (write)** : This mode is used to write the contents in a file. It erases the existing content of a file and locates the file pointer at the begining of a file. If file is not available it will create a new file.
- **r+ :**Opens the file for reading and writing. Places the file pointer at the beginning of the file.
- **a (append)** : This mode is used to append the new content with existing content of a file.
- **a+ :**Opens the file for reading and writing only and it places the file pointer at the end of the file.

Closing a file

When we have done with a file, we should close the file using fclose() function.
Closing a file frees up the resources connected with that file.

Syntax

fclose(file_name)

Returns TRUE if the file closed successfully,and FALSE otherwise.

Reading text from a file

1. **fgetc**() function is used to read a single character from the file.
2. **fgets()** function is used to read single line from file.
3. **file_get_contents**() function used to read entire file into a string.

1) **Reading text from a file using fegts()**

- **fgets()** function is used for reading single line from file.
- **fgets()** function returns a line from an open file.

Syntax:

fgets($handle, $length)

This function takes two arguments: $handle must be a file successfully opened by fopen()
$length is Optional parameter, specifies number of bytes to read from file.

Example:

```
<html>
<head>
<title> Reading From a File </title>
</head>

<body>
<h1> Reading from a file</h1>
<?php
$file=fopen("file.txt","r");
while(!feof($file))
```

```
{
$text=fgets($file);
echo $text,"<br>";
}
?>
</body>
</html>
```

2) Reading from a file character by character with fgetc

- fgetc() function is used to get a single character from the file.
- fgetc()function returns a string containing a single character read from the file.

```
<html>
<head>
<title> Reading characters from a File </title>
</head>

<body>
<h1> Reading from a file</h1>
<?php
$file=fopen("file.txt","r");
while($char=fgetc($file))
{
if($char=='\n')
{ $char="<br>"; }
echo $char;
}
?>
</body>
</html>
```

3) Reading a whole file at once file_get_contents

- file_get_contents — Reads entire file into a string
- If we want to load the full-content of a file to a PHP variable, we use **file_get_contents**() function.
- *file_get_contents()* takes one parameter, the filename to open.

Syntax:

file_get_contents($file_name,include_path ,$context, $start,$limit)

$file_name – name or path of the file
include_path-This is a flag and should be set to true if we want to Search for the file in the include_path
$context-This is an optional parameter. If we don't need to use, wecan skip this parameter by NULL
$start and *$limit* offsets are used to get some portion of file content, instead of getting entire file source.
$start –used to set the starting offset of a file to read
$limit – This offset is used to set end limit

```
<html>
<head>
<title> Reading whole file at once </title>
</head>

<body>
<h1> Reading whole file at once</h1>
<?php
$text=file_get_contents("file.txt");
$fixed_text=str_replace("\n","<br>",$text);
echo $fixed_text;
}
?>
</body>
</html>
```

Reading a file into an Array with file

How we convert text file data into array?

file

- file — Reads entire file into an array
- Syntax

  file **($filename , include_path ,$context)**

```
<html>
<head>
<title> Reading a file into an array</title>
</head>
<body>
<h1> Reading a fileinto an array</h1>
<?php
$data=file('file.txt');
foreach($data as $number => $line)
{
echo "Line $number: ",$line,"<br>";
}
?>
</body>
</html>
```

Checking if a File Exists with file_exists

The act of checking whether a file exists is one of the most basic file-related tasks.

How to check if a file exists?

PHP provides functions that allows us to check if a file exists, using the *file_exist()* function.

file_exists — Checks whether a file or directory exists

Syntax:

file_exists ($filename)

Returns **TRUE** if the file exists; **FALSE** otherwise.

Example:

```
<?php
$filename = 'my.text';

if (file_exists($filename)) {
   echo "The file $filename exists";
} else {
   echo "The file $filename does not exist";
}
?>
```

Getting Filesize with filesize

- filesize() function returns the size of the file in bytes.
- Syntax
  filesize(filename)

```
<?php
echo filesize("test.txt");
?>
```

Reading binary files with fread

- fread() function reads uoto length bytes.
- Syntax:
  fread(handle, length)
  handle is reference to an open binary file and legth in bytes we have to read

Example:

First we open file.txt which is a binary file.

```
<?php
$file=fopen("file.txt","rb");
$text=fread($file,filesize("file.txt"));
echo $text;
fclose($file);
?>
```

Example:To read the binary data from the file using fread() & we read FOUR bytes And we use unpack function:

```
<?php
$file=fopen("file.dat","rb");
$data=fread($file,4);
$array=unpack("Ldata",$data);
$data=$array["data"];
echo "Read this value from file:",$data;
?>
```

Parsing Files with fscanf

The fscanf() function parses the input from an open file according to the specified format.

Syntax

fscanf(file,format)

Example: We read a file, lets say student.txt, where the student name is seperated by tabs:

srikhar reddy
surya reddy
pravalika M

The format in this case is "%s\t%s\n".

```
<html>
<head>
</head>
<body>
<?php
//Open the file
$handle=fopen=("student.txt","r");
while($name=fscanf($handle,"%s\t%s\n"))
{
list($firstname,$lastname)=$name;
echo $firstname," ",$lastname,"<br>";
}
fclose($handle);
?>
</body>
</html>
```

Parsing ini Files with parse_ini_file

- The parse_ini_file() function parses a configuration/initialization (ini) file and returns the settings in it in an array.
- **Syntax**
  parse_ini_file(file,process_sections)
  Above function loads the ini filename, and returs the setting in an associative array.

Example:Heres sample.ini file

```
[first_section]
first_color=red
second_color=white
third_color=blue

[second section]
file="/usr/local/code.data"
URL="http://www.php.net"
```

Now we can recover the values in the .ini files using the $array array.
Example: We recover the value key first_color (which is red) like $array["first_color"]

```
<html>
<head>
</head>
<body>
<?php
$array=parse_ini_file("sample.ini");

foreach($array as $key => $value)
{
echo "$key => $value <br>";
}
?>
</body>
</html>
```

Copying Files with copy

- We can copy files with the copy() function.
- Syntax
  copy (source, destination);

here, copy() function takes two parameters - First, the filename that we wish to copy from, and the filename you wish to copy to.

Example: Lets make a copy of file.txt, and report success if it worked.

```
<?php
if(copy('file.txt','copy.txt')){
        echo "File copied SUCCESSFULLY!.";
   } else {
      echo "File can not be copied.";
   }
?>
```

Deleting Files with unlink

- We can delete a files in PHP by using unlink() function.
- We simply pass the filename to unlink().
- Syntax
  unlink("filename")
  filename- name of the file to delete

Example:

```
<?php
   if (unlink("copy.txt")) {
      echo "File Deleted SUCCESSFULLY.";
   } else {
      echo "File can not be deleted.";
   }
?>
```

Writing a File All at Once with file_put_contents

- There's a shortcut to write text to a file using file_put_contents() function.
- file_put_contents() function writes string to a file.
- Syntax:
  file_put_contents(filename, data)
  - filename is the name of file we want to write
  - data is the string text to write

Example:

```
<?php
$data="Here\nis\nthe\nData.";
file_put_contents("file.txt',$data);
?>
```

Write to a File with fwrite

- If we want to write a string to a file we use fwrite() function.
- Syntax:
  fwrite(handle,string)
  We pass handle (reference to an opened file) and String to write.

Example: To write text to a data.txt file. We start by opening the file for writing:

```
<?php
$file=fopen("data.txt","w");
$string="Here\nis\nthe\nData.";
#fwrite($file,$string);
#we can check whether write operation failed or successful
if(fwrite($file,$string)==FALSE)
{
        echo "Writing to file FAILED";
}
else
{
        echo "SUCCESSFUL";
}
fclose($file);
?>
```

Appending to a File with fwrite

- We can also **append data to file using fwrite()** function.
- Syntax:
  fwrite(handle,string)
  We pass handle (reference to an opened file) and String to append to the file.

Example: To write text to a data.txt file. We start by opening the file for writing:

```
<?php
$file=fopen("data.txt","a");
$string="Here\nis\nthe\nData.";
#fwrite($file,$string);
#we can check whether append write operation failed or successful
if(fwrite($file,$string)==FALSE)
{
        echo "Writing to file FAILED";
}
else
```

```
{
        echo "SUCCESSFUL";
}
fclose($file);
?>
```

Reading & Writing Binary Files

So far we have seen text files, But its simple to handle binary files using fwrite() &fread() functions.

A little work we do is to pack binary data into strings using pack function, & unpack binary data using unpack function.

Example: To pack the data into Long integer format-

```
pack("L",$number);
```

L-Unsigned Long

Example:To write data into file:

```
<?php
$number=512;
$file=fopen("data.dat","wb");
if (fwrite($file,pack("L",$number))==FALSE)
{
        echo "Can't write, FAILED!!";
}
else
{
        echo "SUCCESSFULLY written into FILE!!";
}
fclose($file);
?>
```

Reading binary files with fread

- fread() function reads uoto length bytes.
- Syntax:

  fread(handle, length)

  handle is reference to an open binary file and legth in bytes we have to read

Example:

First we open file.txt which is a binary file.

```
<?php
$file=fopen("file.txt","rb");
$text=fread($file,filesize("file.txt"));
```

```
echo $text;
fclose($file);
?>
```

Example:To read the binary data from the file using fread() & we read FOUR bytes And we use unpack function:

```
<?php
$file=fopen("file.dat","rb");
$data=fread($file,4);
$array=unpack("Ldata",$data);
$data=$array["data"];
echo "Read this value from file:",$data;
?>
```