UNIT-I

PHP

PHP:

The PHP Hypertext PreProcessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

* PHP is a recursive acronym for "PHP: Hypertext PreProcessor".

* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

Common Uses 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 can 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.

Advantages of PHP:

* PHP runs on various platforms (windows, Linux, Unix, Mac osx, etc..)
* PHP is compatible with almost all servers today (Apache, IIS etc)
* PHP supports a wide range of databases
* PHP is free.
* PHP is easy to learn and it runs effeciently on the server side

Syntax for PHP:

```
<? php
 - - -
 - - -
?>
```

* DECLARING VARIABLES:

In PHP, a variable starts with the '$' sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

* After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

PHP Variables:

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

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

PHP is a Loosely Typed Language:

In the example above, notice that we did not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value. Other languages such as C, C++ and Java, the programmer must declare the name and type of the variable before using it.

PHP Variable Scope:

In PHP, variables can be declared anywhere in the script.

* The scope of the variable is the part of the part of the script where the variable can be referenced / used.

* PHP has three different variable scope.

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?php
$x=5; // global scople
function myTest()
{
    // using x inside this function will not generate an errors
    echo "<p> Variable x inside function is: $x </p>";
}
```

```
myTest();
echo "<p> Variable x outside function is : $x </p>";
?>
```

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function :

```
<?php
function myTest()
{
    $x = 5; // local scope
    echo "<p> Variable x inside function is : $x </p>";
}
myTest();
// using x outside the function will generate an error
echo "<p> Variable x outside function is : $x </p>";
?>
```

* PHP DATA TYPES :

Variables can store data of different types, and different data types can do different things.

* PHP supports the following data types :

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

1) PHP String :

A string is a sequence of characters, like "Hello world!".

* A string can be any text inside quotes. You can use single or double quotes.

Example :

```
<? php
$x = "Hello world!"
$y = 'Hello world!';
    echo $x;
    echo "<br>";
    echo $y;
?>
```

Output :

```
Hello world!
Hello world!
```

2) PHP Integer:

An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.

Rules for integers:

* An integer must have atleast one digit (0-9)
* An integer cannot contain comma or blanks
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based prefix with 0x) or octal (8-based prefix with 0)

Example:

```
<?php
$x = 5985;
var_dump($x);
?>
```

Output:

```
int(5985)
```

3) PHP Float:

A float (floating point number) is a number with decimal point or a number in exponential form. In the following example $x is a float. The PHP var_dump() function returns the datatype and value.

Example:

```
<?php
$x = 10.365;
    var_dump($x);
?>
```

Output

```
float(10.365)
```

4) PHP Boolean:

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

5) PHP Array:

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the datatype and value:

Example:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
    var_dump($cars);
?>
```

Output:

```
[0] => string(5) "Volvo"
[1] => string(3) "BMW"
[2] => string(6) "Toyota"
```

6) PHP object:

An object is a data type which stores data and information on how to process that data.

* In PHP, an object must be explicitly declared.

* First we must declare class of object. For this, we use a class keyword. A class is a structure that can obtain properties and methods.

Example :

```
<?php
class car
{
    function car()
    {
        $this -> model = "VW";
    }
}
// create an object
$x = new car();
// show object properties
echo $x -> model;
?>
```

Output:

VW

7) PHP NULL Value :

NULL is a special data type which can have only one value : NULL.

* A variable of datatype NULL is a variable that has no value assigned to it.

Note: If a variable is created without a value, it is automatically assigned a value of NULL.

* Variables can be emptied by setting a value to NULL

Example :

```
<? php
$x = "Hello world!";
$x = null
var_dump ($x);
?>
```

Output :

NULL

8) PHP Resource :

The special resource type is not an actual data type. It is a storing of the reference and resources external to PHP.

* A common example of using the resource datatype is a database call.

* PHP Array

An array is a special variable, which can hold more than one value at a time.

* There are three different kinds of arrays and each array value is accessed using an ID which is called array index.

Numeric array:

An array with a numeric index. Values are stored and accessed in linear fashion.

Associative array:

An array with strings as index. This stores element values in association with key values in association rather than in a strict linear index order.

Multidimensional array:

An array containing one or more arrays and values are accessed using multiple indices.

Numeric Array:

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example:

Following is the example showing how to create and access numeric arrays.

Here we have used array() function to create array. This function is explained in function reference.

```
<? php
  $numbers = array(1, 2, 3, 4, 5);
  foreach ( $numbers as $value)
  {
      echo " Value is $value <br/>";
  }
?>
```

This will produce the following result –

```
Value is 1
value is 2
value is 3
value is 4
value is 5
```

Associative Arrays:

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between keys and values.

* To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE: Don't keep associative array inside double quote while printing otherwise it would not return any value.

```
<? php
$salaries = array ("Rohit" => 2000, " peter" => 1000,
                   " kelvin" => 500);
    echo "salary of Rohit is". $salaries ['Rohit'].
                   "<br/>";
    echo "salary of peter is". $salaries ['peter'].
                   "<br/>";
    echo "Salary of kelvin is". $salaries ['kelvin'].
                   "<br/>";
?>
```

This will produce the following result –

```
Salary of Rohit is 2000
salary of peter is 1000
salary of kelvin is 500
```

[Therefore from above example we came to know that
Salary of Rohit is high
salary of peter is medium
salary of kelvin is low.]

Multidimensional Arrays:

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example :

In this example we create a two dimensional array to store marks of three students in three subjects — This example is an associative array, you can create array in the same fashion.

```
<?php
   $marks = array (" Rohit" => array ("physics" => 35,
                    "maths" => 30, "chemistry" => 39 ),
                     " peter" => array ("physics" => 30,
                    "maths" => 32 , "chemistry" => 29),
                     "kelvin" => array ("physics" => 31,
                    "maths" => 22 , "chemistry" => 39));

   /* Accessing multi-dimensional array values*/
   echo "Marks for Rohit in physics: ";
   echo $marks ['Rohit'] ['physics'] . "<br/>";

   echo "Marks for peter in maths :";
   echo $marks ['peter'] ['maths'] . "<br/>";

   echo "Marks for kelvin in chemistry :";
   echo $marks ['kelvin'] ['chemistry'] . "<br/>";

?>
```

This will produce the following result -

```
Marks for Rohit in physics : 35
Marks for peter in maths : 32
Marks for kelvin in chemistry : 39
```

* STRINGS

A string is a sequence of characters, like "Hello World!".

Get the Length of a String:

The PHP strlength i.e., strlen() function returns the length of a string.

* The example below returns the length of the string "Hello world!";

```
<?php
    echo strlen("Hello world!"); // outputs 12
?>
```

Count the Number of Words in a string:

The PHP str_word_count() function counts the number of words in a string:

```
<?php
    echo str_word_count("Hello world!"); // outputs 2
?>
```

The output of the code above will be: 2.

String concatenation operator:

To concatenate two string variables together, we use dot (.) operator -

```
<?php
$string1 = "Hello world";
$string2 = "1234";
echo $string1." ".$string2;
?>
```

This will produce the following result -

Hello World 1234

Reverse a String:

The PHP strrev() function reverses a string:

```
<?php
   echo strrev("Hello world"); //outputs dlrowolleH
?>
```

Using the strpos() function

The strpos() function is used to search for a string or character within a string.

* If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

* Let's see if we can find the string "world" in our string -

```
<?php
   echo strpos("Hello World!", "World");
?>
```

This will produce the following result — 6

In PHP we can assign values into the string variables in 3 ways.

→ using single quotation
→ using double quotation
→ heradoc style

* In PHP, we have approximately 100 functions.
* we introduce each function but we have to implement some of the functions of a string.

1) determining string length
2) comparing two strings
3) manipulating string case
4) alternatives for regular expression functions
5) converting string to and from HTML.
6) padding and stripping a string
7) counting characters and words.

1) determining string length:
here we are using strlen() function and this function returns the length of the string.

Eg: int strlen(string str)

2) Comparing two strings:
php provides four functions for performing this task.

a) strcmp()
b) strcasecmp()
c) strtok() strspn()
d) strcmp strcspn()

a) strcmp() :

the strcmp() function performs a binary numbers and compares two strings.

Syntax: (case-sensitive)

int strcmp(string str1, string str2)

→ 0, if str1 and str2 are equal (s1==s2)

→ -1, if str1 is less than str2 (s1<s2)

→ 1, if str2 is less than str1 (s2<s1)

Example:

```
<?php
  $pwd = "abitcse";
  $pwd2 = "abitcse2";
  if (strcmp($pwd, $pwd2) != 0)
  {
     echo "pwd do not match";
  }
  else
  {
     echo "pwd match";
  }
?>
```

b) stricasecmp():

The stricasecmp() function operates exactly like strcmp() but it is case-insensitive.

Syntax:

int stricasecmp(string str1, string str2)

Example:

```
<?php
  $gmail1 = "abit@gmail.com";
  $gmail2 = "ABIT@gmail.com";
  if (!stricasecmp ($gmail1, gmail2)
      echo "the gmail addresses are identical";
?>
```

c) strspn():

calculating the similarity between two strings.

syntax:

int strspn(string str1, string str2[, int start, int length)

Example:

```
<?php
  $pwd = "abc123";
  if (strspn ($pwd, "1234567890") == strlen($pwd))
      echo "the pwd cannot consist solely of
            numbers!";
?>
```

d) strcspn():

calculating difference between two strings.

Syntax:

```
int strcspn(string str1, string str2, int start,
                                    int length)
```

Example:

```
<?php
  $pwd = "abc123";
if (strcspn ($pwd, "1234567890")==0)
{
   echo "pwd can't consist solely of numbers!";
}
?>
```

3) manipulating string case:

In this we have mainly four functions

a) strtolower()

b) strtoupper()

c) ucfirst()

d) ucwords()

a) strtolower():

converting a string to lower case.

Syntax:

```
strtolower (string)
```

Output:

bangaram

Example:

```
<?php
$name = "Bangaram";
echo str_tolower ($name);
?>
```

b) strtoupper( ):

converting a string to uppercase

Syntax:

strtoupper (string)

Example:

```
<?php
  $name = " John";
  echo strtoupper ($name);
?>
```

Output:

JOHN

c) ucfirst( ):

converts the first character of a string to uppercase

Syntax:

ucfirst (string)

Example:

```
<?php
  $name = "gopi";
  echo ucfirst (string
  echo strucfirst ($name);
?>
```

Output:

Gopi

d) ucwords( ):

converts the first character of each word in a string to uppercase.

Syntax:

ucwords (string)

Example:

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

output:

Hello World!

4) Alternatives for regular expression function:

In this we have to describe different types of functions. They are:

a) strtok()
b) explode()
c) implode()
d) strops()
e) strpos()
f) str_replace()
g) stristr()
h) substr()
i) substr_count()
j) substr_replace()

a) strtok():

this function parses the string based on a predefined list of characters.

Syntax:

string strtok(string str, string tokens)

Example:

```
<?php
$info = "abit: abit@gmail.com\siddhartam, tdp";
$tokens = ":\,";
$tokenized = strtok($info, $tokens);
while ($tokenized)
{
    echo "elements= $tokenized <br>";
```

```
    $tokenized = strtok($tokens);
}
?>
```

b) explode();

this function divides the string str into an array of substrings, in this we have to mainly concentrate on areas: sizeof() and strip_tags() to determine the total no. of words.

Example:

```
<?php
$summary = <<< summery
php is a server side scripting language
summary;
$words = size of (explode (' ', strip_tags ($summary)));
echo "total words in summary : $words";
?>
```

c) Implode()

we concentrate array elements to form a single deffined delimited string using the implode() function.

Example:

```
<?php
$cities = array("kdp", "anlpur", "tirupathi");
echo implode("\", $cities);
?>
```

d) strops():

In this function finds the position of the first case-sensitive occurance of a substring in a string.

e) strrpos():

in this function finds the last occurance of a string returning its numerical position.

f) str_replace():

this function case sensitively replaces all instance of a string with another.

Example:

```
<?php
  $gmail = "abit@gmail.com";
  $gmail = str_replace("@", "(is)", $gmail);
     echo "college mail is $gmail";
?>
```

g) strstr:

this function returns the remainder of a string begining with the first occurance of a predefined string.

Example:

```
<?php
  $gmail = "abit@gmail.com";
  echo ltrim (strstr ($gmail, "@"), "@");
?>
```

h) substr():

this function returns the part of a string located between a predefined string offset and length positions.

Eg:
```
<?php
$car = "1994ford";
echo substr($car5);
?>
```

Eg 2:
```
<?php
$car = "1994 ford";
echo substr($car, 0, 4);
?>
```

i) substr_count():

this function returns the no. of times one string excuss another.

Example:
```
<?php
$intu = array("php", "xampp");
$talk = <<< talk
```
PHP is a scripting language and php is server side programming language. xAMPP is a web server

Talk:
```
foreach($info as $it)
{
    echo "the word $it appears ". substr_count
                    ($talk, $it). "time (s) <br/>";
}
?>
```

j) substr_replace():

replace the portion of a string with another string.

Example:

```
<?php
$name = "fbit college";
echo substr_replace ($name, "engg", 0,4);
?>
```

5) Converting string to HTML form :

In this we are using different types of converting function . they are

* Converting newline characters to HTML break tags.

Example :

```
<?php
$info = "aaaaaaaaaaaaaaa
         Bbbbbbbbbbbbbb
         ccccccccccccc
         ddddddddddddddd";
echo nl2br($into);
?>
```

Here we do not use
 statement

* Using special HTML characters for other purpose.

In this we use htmlspecialchars() function .

- & — &
- " — "
- ' —
- < — <
- > — >

Example :

```
<?php
$input = "<php is "scripting" language >";
echo htmlspecialchars($input);
?>
```

6) padding and stripping a string:

php provides no. of functions. They are

a) ltrim()
b) rtrim()
c) trim()
d) str_pad()

a) ltrim():

this function removes various characters from the begining of a string including white space, horizontal tab (\t), newline (\n), carriage return (\v), null (\0).

String ltrim (string str, char

Syntax:

ltrim (string, charlist)

b) rtrim():

this function removes various characters from the end of the string and except designed characters.

String rtrim (string str [, string charlist])

c) trim():

both ltrim and rtrim.

d) str_pad():

this function pads a string with a specified number of characters

Example:
```
<?php
    Echo str_pad ("salad", 10). "is good";
?>
```

Output:

Salad is good.

7) Counting characters and words:

its mainly used for to determine the total number of characters or words in a given string. Php provides two functions.
there are count_chars() and str_word_count.

* PHP Operators :

Operators are used to perform operations on variables and values.

* PHP language supports following type of operators

a) Arithmetic operators
b) Assignment operators
c) Comparision operators
d) Increment / Decrement Operators
e) Logical Operators
f) Array operators

a) Arithmetic Operators:

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

Operator	Name	Example	Result
+	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 and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y^th power.

b) Assignment Operators:

The PHP assignment operators are used with numeric values to write a value to a variable.

* The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression to the right.

Assignment	Same as..	Description
x=y	x=y	The left operand gets set to the value of the expression on the right.
x+=y	x=x+y	Addition
x-=y	x=x-y	subtraction
x*=y	x=x*y	Multiplication
x/=y	x=x/y	Division
x%=y	x=x%y	Modulus

c) Comparision Operators:

The PHP comparision operators used to compare two values (number or string).

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 same type.
!=	not equal	$x!=$y	returns true if $x is not equal to $y

Operator	Name	Example	Result
<>	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 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
>=	greaterthan or equal to	$x >= $y	returns true if $x is greater than or equal to $y
<=	less than or equal to	$x <= $y	returns true if $x is less than or equal to $y.

d) Increment / Decrement Operators:

The PHP increment operators are used to increment a variables value

* The PHP decrement operators are used to decrement a variables value

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one.
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one.

e) Logical Operators:

The PHP logical operators are used to combine conditional statements

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

f) Array Operators:

PHP array operators are used to compare arrays

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	returns true if $x and $y have 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

* CONTROL STRUCTURES

php supports different types of statements like

1) if statement
2) else statement
3) switch statement
4) while statement
5) do...while statement
6) for statement
7) for each statement
8) break and goto statement.
9) continue statement

1) if statement :

Use the if statement to execute some code only if the specified condition is true.

Syntax :

```
if (expression)
{
    statement
}
```

Example :

```
<?php
   $secretnumber = 143;
   if ($_POST['guess'] == $secretnumber)
   {
      echo "<p> Congratulation! </p>";
   }
?>
```

2) else statement:

if the condition is true then if statements are executed otherwise else statements will be executed.

Example:

```
<?php
$secretnumber = 143;
if ($_POST['guess'] == $secretnumber)
{
    echo "<p> Congratulation!</p>";
}
else
{
    echo "<p> Sorry!</p>";
}
?>
```

3) switch statement:

Use the switch statement to select one of many blocks of code to be executed.

* switch statement can compare "=" operations only.

Example:

```
<?php
    $x = 1;
    switch ($x)
    {
        case 1: echo "number1";
                break;
```

```
        case 2: echo "number2";
                break;
        case 3: echo "number3";
                break;
        Default:
                echo "no number b/w 1 and 3";
    }
?>
```

4) while Statement :

while loop checks the condition then only executes the statements if condition is true.

Syntax :

```
while (expression)
{
    statements
}
```

Example :

```
<?php
    $count=1;
    while ($count<5)
    {
        printf("%d squared = %d <br>", $count,
                                    pow($count,2));
        $count++;
    }
?>
```

Output :

1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16

5) do... while statement :

It will execute the statement atleast once even if condition is false (or) true.

Syntax :

```
do
{
    statements
}while (expression);
```

Example :

```
<?php
   $count=11;
   do
   {
      printf ("%d squared = %d <br/>", $count,
                                    pow($count,2));
   }
   while ($count<10);
?>
```

6) for statement :

By using this loop we can run number of iteration

Syntax :

```
for (exp1; exp2; exp3)
{
   statements;
}
```

There are few rules to keep in mind when using php's for loops

- The first expression exp1, is evaluated by default at the first iteration of the loop.
- The second expression exp2, is evaluated at the begining of each iteration. This expression determines whether loop will continue.
- The third expression exp3, is evaluated at conclusion of even loop.

Example:

```
<?php
  for ($kilometers =1; $ kilometers <=3; $kilometers++)
  {
     printf("% kilometers = %.4f miles <br/>",
                    $kilometers, $kilometers * 0.62140);
  }
?>
```

Output:

1 kilometers = 0.6214 miles
2 kilometers = 1.2428 miles
3 kilometers = 1.8642 miles.

7) for each statement:

loops through a block of code for each element in an array.

* The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax:

```
foreach ($array as $value)
{
    code to be executed;
}
```

Example:

```
<?php
$colors = array ("red", "blue", "white", "black");
foreach ($colors as $value)
{
    echo "$value <br>";
}
?>
```

Output:

red
blue
white
black

8) break and goto statements:

break statement: break statement is end execution of a do while, for, foreach, switch, while block.

goto statement: In php goto statement, "BREAK" features was extend to support labels. This means we can suddenly jump to a specific location outside of a looping or conditional construct.

9) Continue statement:

continue statement execute the current loop iteration to the end.

* PHP Functions:

A function is a block of statements that can be used repeatedly in a program. A function will not execute immediately when a page loads. A function will be executed by a call to the function.

Create a User Defined Function in PHP:

A user defined function declaration starts with a word "function":

Syntax:

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

Note: A function name can start with a letter or underscore (not a number)

* Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the begining of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello World!". To call the function, just write its name:

Example :

```
<?php
  function writeMsg()
  {
      echo "Hello World!";
  }
  writeMsg(); // call the function
?>
```

PHP Function Arguments :

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just seperate them with a comma.

* The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name.

Example :

```
<?php
function familyName($fname)
{
    echo "$fname Refsnes.<br>";
}
```

```
familyName ("Jani");
familyName ("Hege");
familyName ("Stale");
familyName ("kai Jim");
familyName ("Borge");
?>
```

PHP Default Argument Value :

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument.

Example :

```
<?php
    function setHeight ($minheight = 50)
    {
        echo "The height is : $minheight <br>";
    }
    setHeight (350);
    setHeight (); // will use the default value of 50
    setHeight (135);
    setHeight (80);
?>
```

Output :

```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

PHP Functions – Returning Values

To let a function return a value, use the return statement.

Example:

```
<? php
   function sum($x,$y)
   {
       $z = $x + $y;
       return $z;
   }
   echo "5+10 = " . sum (5,10). "<br>";
   echo "7+13 = " . sum (7,13). "<br>";
   echo "2+4 = " . sum (2,4). "<br>";
?>
```

Output:

5+10 = 15
7+13 = 20
2+4 = 6

Array Functions:

* Count: it returns total no. of elements

Example :

```
<?php
    $arr = array(10,20,30);
    echo count($arr);
?>
```

Output : 3

* Sort : it returns the elements of an array in ascending order.

Example :

```
<?php
    $arr = array(60,20,30)
    sort($arr);
    print_r($arr);
?>
```

output : 20, 30, 60

* rsort : it returns the elements of an array in descending order.

Example :

```
<?php
    $arr = array(101,104,102);
    rsort_r($arr);
    print_r($arr);
?>
```

output : 104, 102, 101

* asort: it returns the original keys with ascending order

Example:
```
<?php
$arr = array(104 => 40, 101 => 20, 108 => 50);
asort ($arr);
print_r($arr);
?>
```

Output: 101 => 20, 104 = 40, 108 = 50

* ksort: it returns the array in ascending order with based on the "keys".

Example:
```
<?php
$arr = array (104 => 40, 101 => 20, 108 => 50)
ksort($arr);
print_r ($arr);
?>
```

Output: [101] = 20, [104] = 40, [108] = 50

* array_push(): this function adds an element into the end of the array and returns the total no. of elements in that array.

Example:
```
<?php
$arr = array (10,20,30)
echo array_push ($arr, 40);
print_r ($arr);
?>
```

output:
```
10
20
30
40
```

* array_pop() : remove the last element and return the value to that element.

Example :
```
<?php
$arr = array(10,20,30);
echo array_pop($arr);
print_r($arr);
?>
```

* array_change_key_case() : it converts all keys of an array into lower case.

Example :
```
<?php
$arr = array('ABc' => 10,20,30);
print_r(array_change_key_case($arr));
?>
```

* array_chunk() : splits an array into chunck of an array.

Example :
```
<?php
$arr = array(10,20,30,40,50,60);
print_r(array_chunk($arr,2));
?>
```

* array_keys() : it returns new array with keys as value of another array.

Example :
```
<?php
$arr = array('abc' => 10,20,30,40);
print_r(array_keys($arr));
?>
```

* array_values() : return array with the values of an array.

Example :
```
<?php
$arr = array('ABC' = 10,20,30,40,50,60)
print_r (array_values($arr));
?>
```

* array_flip() : exchanges all keys with their associated values in array.

Example :
```
<?php
$arr = array('ABC' -> 10,20,30,40);
// $arr = array (10, 200,400);
printf_r (array_flip ($arr));
?>
```

* array_merge() : merges one or more arrays into one array

Example :
```
<?php
$arr = array ('ABC' => 10,20,30,40);
$arr1 = array (100,200, 300);
printf_r (array_merge ($arr, $arr1));
?>
```

* Shuffle() : shuffle the elements of an array

Example :
```
<?php
$arr = array('ABC' => 10,20,30,40);
Shuffle ($arr);
print_r ($arr);
?>
```

* READING DATA FROM WEB FORM

$_REQUEST :

It is used to collect data after submitting an HTML form

* when user submits the data by clicking on "submit", the form data is sent to the file specified in the action attribute of the <form> tag.

* In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field.

Example :

```
<html>
<body>
<form method = "post" action = "<?php
            echo $_SERVER ['PHP_SELF']; ?>">
Name ; <input type = "text" name = "fname">
<input type = "submit">
</form>
<?php
if ($_SERVER ["REQUEST_METHOD"] == "POST")
{
   $name = htmlspecialchars ($_REQUEST ['fname']);
```

```
    if (empty ($name))
    {
        echo "Name is empty";
    }
    else
    {
        echo $name;
    }
}
?>
</body>
</html>
```

$_POST :

It is widely used to collect form data after submitting an HTML form with method = "POST". $_POST is also widely used to pass variables

$_GET :

This can also be used to collect form data after submiting an HTML form with method = "get".

* $_GET can also collect data sent in the URL.

* HANDLING FILE UPLOADS:

It can be used to upload the files.

a) name
b) type
c) tmp_name
d) error
e) size

Example:

Uploading file to server.

```
<html>
<body>
<form action = "upload.php" method = "post">
<input type = "file" name = "file1">
<input type = "submit" name = "submit" value =
                                    "upload">
</form>
</body>
</html>

<?php
$file_name = $_FILES ['file1'] ['name'];
$file_type = $_FILES ['file1'] ['type'];
$file_path = $_FILES ['file1'] ['tmp_name'];
$file_size = $_FILES ['file1'] ['size'];
$file_error = $_FILES ['file1'] ['error'];
$file_loc = "upload /". $file_name ;
```

?>

Note :

move_uploaded_file

↓

This is the function which uploads your file from client to server

* CONNECTING TO DATABASE

With PHP, you can connect to and manipulate databases.

* MySQL is the most popular database system used with PHP

What is MySQL ?

* MySQL is a database system that runs on a server.
* MySQL is very fast, reliable and easy to use.
* MySQL compiles on a number of platforms
* MySQL is developed, distributed and supported by oracle corporation

PHP connecting to MySQL

1) mysqli
2) PDO (PHP Data Objects)

Establishing a connection

Object-oriented:

```
<?php
$conn = new mysqli("localhost", "userid", "password");
if ($conn -> connect_error)
{
    die ("connection failed :". $conn -> connect_error);
}
else
{
    echo "connection is established";
}
?>
```

Procedure - Oriented :

```
<?php
$conn = mysqli_connect ("localhost", "userid",
                        "password");
if (!$conn)
{
   die ("connection failed", mysqli_connect_error());
}
else
{
    echo " ";
}
?>
```

Closing the connection :

Object - oriented

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

Procedure - oriented

```
mysqli_close ($conn);
```

Execution of Query:

Object Oriented

```
<?php
$sql = "select * from emp";
if ($conn -> query ($sql) == TRUE)
{
    echo "successful";
}
```

```
else
{
   echo "error: " . $conn -> error ;
}
?>
```

Procedure - oriented :

```
<?php
$sql = "     ";
if (mysqli_query ($conn , $sql)
{
    echo " successful";
}
else
 {
    echo " error : " mysqli_error ($conn);
}
?>
```

* EXECUTING SIMPLE QUERIES

Database Queries:

A Query is a question or a request

Create a MySQL Table.

The "create table" statement is used to create a table in MySQL

Example: create table student (sid int(10), sname varchar(20) ....);

Insert Data into MySQL

After a database and a table have been created, we can start adding data in them.

* Here are some syntax rules to follow

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted.
- Numeric values must not be quoted.
- The word NULL must not be quoted.

The "insert into" statement is used to add new records to a MySQL table.

Example:

insert into student values (value1, value2, ...);

Creation of database:

Syntax:

```
create database databasename;
```

Example:

```
create database cse;
```

Select Data From a MySQL Database:

The "select" statement is used to select data from one or more tables:

```
Select column_name(s) from table_name
```

or we can use the * character to select ALL columns from a table.

```
select * from table_name
```

Delete Data From MySQL

The "Delete" statement is used to delete records from a table:

```
delete from table_name where some_column =
                                Some_value
```

Update Data in MySQL

The "update" statement is used to update existing records in a table

```
update table_name set column1 = value1, column2 =
        value2 where some_column=
                                some_value
```

* HANDLING RESULTS.

```
<?php
  $servername = "localhost";
  $username = "username";
  $password = "password";
  $dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
                   $password, $dbname);
// Check connection
if ($conn->connect_error)
{
    die("Connection failed: ". $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM
                                  MyGuests";
$result = $conn->query($sql);

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

* HANDLING SESSIONS AND COOKIES :

$_COOKIE :

* A cookie is often used to identify a user.
* cookies are stored at client side.
* Cookie takes less space to store data.
* It consists of name and a textual value.
* In every http communication between browser and server, a header is included.
* A header stores information about this message (user ids or names)

Create cookies with PHP.

A cookie is created with the setcookie() function.

Syntax :

```
setcookie(name, value, expire, path, domain,
                                    secure, httponly);
```

Only the name parameter is required. All other parameters are optional.

PHP Create / Retrieve a Cookie :

The following example creates a cookie named "user" with the value "John".

* The cookie will expire after 30 days (86400 x 30).

* The '/' means that the cookie is available in entire website.

* We then retrieve the value of the cookie "user" (using the global variable $_COOKIE)

* we also use isset() function to find out if the cookie is set.

Example :

```
<?php
$cookie_name = "user";
$cookie_value = "John";
setcookie ($cookie_name, $cookie_value,
                    time() + (86400 * 30), "/");
 ?>
<html>
<body>
<?php
if (!isset ($_COOKIE [$cookie_name]))
{
    echo ....
}
else
{
   echo .....
 }
$_COOKIE [$cookie_name];
?>
</body>
</html>
```

$_SESSION:

A session is a way to store information (in variables) to be used across multiple pages.

* Unlike a cookie, the information is not stored on the users computer.

Start a PHP session:

A session is started with the session_start() function

Example:

"demo-session.php"

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["favcolor"] = "blue";
$_SESSION["favanimal"] = "dog";
echo "session variables are set.";
?>
</body>
</html>
```

Note:

The session_start() function must be the very first thing in your document. Before any html tags.

Destroy a PHP session:

To remove all global session variables and destroy the session, use session_unset() and session_destroy()

Example:

```
<?php
session_start();
?>
<html>
<body>
<?php
    session_unset();
    session_destroy();
?>
</body>
</html>
```

Chapter - 2

FILE HANDLING IN PHP

File Handling :

File Handling is an important part of web application.

Operations on the file :

1) open
2) read
3) write
4) close.

1) Open a file :

A better method to open files is within the fopen() function. This function takes two parameters

* The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

* The file may be opened in one of the following modes.

modes	description
r	open a file for read only
w	open a file for write only erases the contents of the file and file pointer starts at beginning of the file
a	open a file for read only The existing data in the file is preserved. File pointer starts at the end of the file
x	creates a new file for write only. returns FALSE and an error if file already exists
r+	open a file for read/write.
w+	open a file for read/write
a+	open a file for read/write
x+	creates a new file for read/write

Example :
```
<?php
$myfile = fopen("abc.txt", "w");
?>
```

2) read a file :

The fread() function reads from an open file.

* The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum no. of bytes to read.

3) writing a file:

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.

Example:

```
<?php
$myfile = fopen("abc.txt", "w");
$txt = "Good morning\n";
fwrite ($myfile, $txt);
fclose ($myfile);
```

4) Close a file:

The fclose() function is used to close an open file.

Example:

```
<?php
$myfile = fopen("abc.txt", "r");
fclose ($myfile);
?>
```

Note:

* The fgets() function is used to read a single line from a file.
* The feof() function checks if the "end-of-file" (EOF) has been reached.

* The fgetc() function is used to read a single character from a file.

Example:

```
<?php
$myfile = fopen("abc.txt", "r");
while (!feof($myfile))
{
    $txt = fgets($myfile);
    echo $txt . '<BR>';
}
fclose($myfile);
?>
```

* Listing Directories :

Function	Description
chdir()	changes the current directory
chroot()	changes the root directory
closedir()	close a directory handle
dir()	returns an instance of the directory class
getcwd()	returns the current working directory
opendir()	opens a directory handle
readdir()	returns an entry from a directory handle
scandir()	returns an array of files and directories of a specified directory.