Unit 2 : PHP Basics

LECTURER: NANDAN PANDYA

COURSE: BACHELOR OF COMPUTER APPLICATIONS (BCA)

SEMESTER: 2ND

NAME OF THE SUBJECT: WEB PROGRAMMING

SUBJECT CODE: CS-08

COLLEGE: KAMANI SCIENCE COLLEGE, AMRELI

Topics

- Introduction to PHP
- PHP configuration in IIS & Apache Web server (practical)
- Understanding of PHP.INI file and Understanding of PHP .htaccess file
- PHP Variable, Static & global variable
- GET & POST method
- PHP Operator and Array
- Conditional Structure & Looping Structure
- User Defined Functions
- Variable Functions, String Functions, Math Functions, Date Functions, Array Functions, File handling functions, miscellaneous functions.

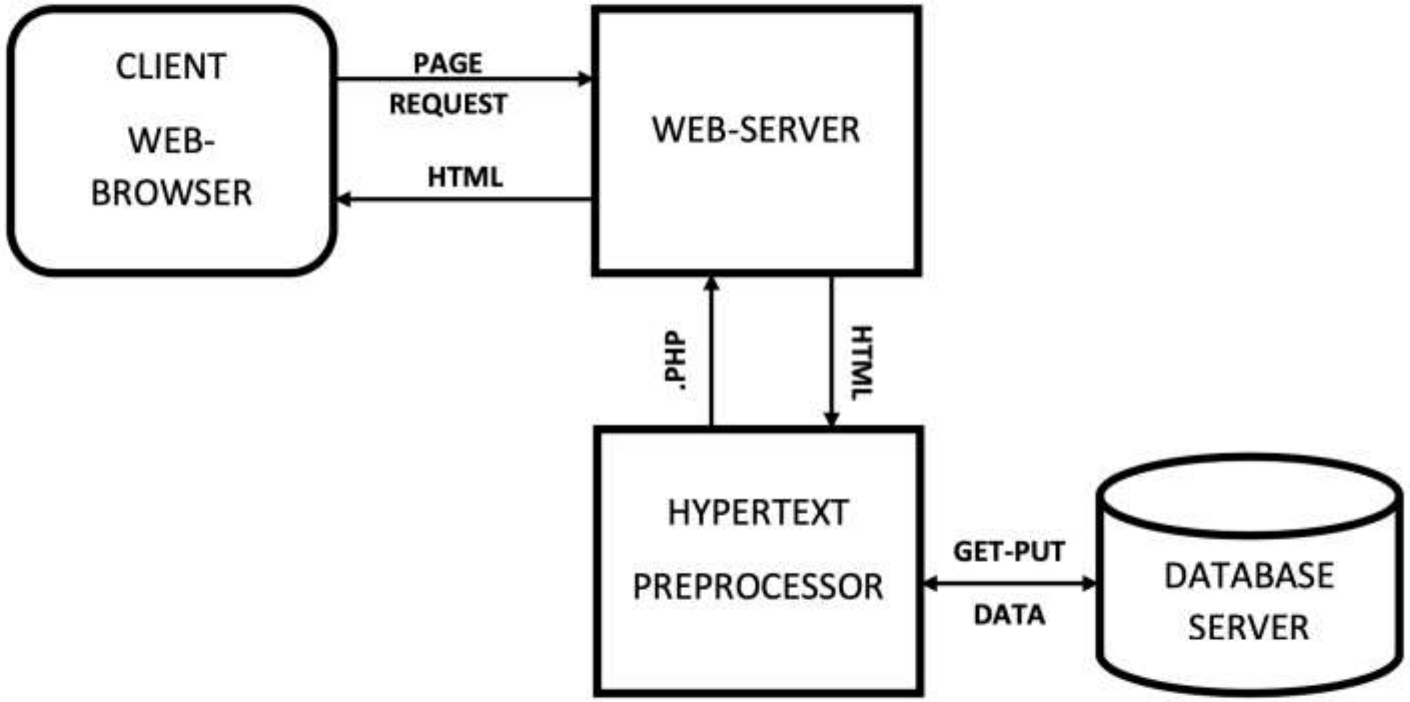
Introduction to PHP

- PHP started out as a small open source project that evolved as more and more people found out how useful it was. **Rasmus Lerdorf** unleashed the first version of PHP way back in 1994.
- 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.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

History of PHP

- PHP was conceived sometime in the fall of **1994 by Rasmus Lerdorf**. Early non-released versions were used on his home page to keep track of who was looking at his online resume.
- The first version used by others was available sometime in early 1995 and was known as the **Personal Home Page Tools**.
- It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff.
- The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data.
- He combined the Personal Home Page tools scripts with the Form Interpreter and added mSQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

How PHP works



Prepared by: Prof. Nandan Pandya
Kamani Science College, Amreli

PHP.INI file Configuration

- The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality. The php.ini file is read each time PHP is initialized.
- The configuration file is well commented and thorough.
- Keys are case sensitive, keyword values are not; whitespace, and lines beginning with semicolons are ignored.
- Booleans can be represented by 1/0, Yes/No, On/Off, or True/False.

PHP .htaccess

- .htaccess is a configuration file for use on web servers running on the web apache server software. when a .htaccess file is placed in a directory which in turn loaded via the Apache web server, then the .htaccess file detected and executed by the Apache server software.
- .htaccess files can be utilized to modify the setup of the Apache server software to empower additional functionality and fetures that the apache web server softwatre brings to the table. We can use the .htaccess file for alteration various configuration in apache web server software.

PHP File structure/syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**
- The default file extension for PHP files is "**.php**".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive but all variable names are case-sensitive!

```
<html>
<body>

<h1>My first PHP page</h1>

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

</body>
</html>
```

Output Statements

- With PHP, there are two basic ways to get output: **echo** and **print**.
- 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.
- The echo statement can be used with or without parentheses: echo or echo().
- The print statement can be used with or without parentheses: print or print().

Example: echo Statement

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
</body>
</html>
```

Example: print statement

```
<html>
<body>
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
</body>
</html>
```

print v/s echo

echo	print
It can accept multiple expressions.	It cannot accept multiple expressions.
It is faster than print.	It is slower than echo.
It is a statement used to display the output and can be used with the parentheses echo or without the parentheses echo.	It is also a statement which is used to display the output and used with the parentheses print() or without the parentheses print.
It doesn't return any value.	It always returns the value 1.

Ways to use PHP

- **Standard PHP tags**

```
<?php
Echo "Hello World";
?>
```

- **Script tags**

```
<script language="php">
        echo "Hello World";
</script>
```

- **Short tags**

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

Comments in PHP

- A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.
- Comments can be used to:
  - Let others understand your code
  - Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

```
<?php
    //This is a single-line comment
    #This is also a single-line comment

    /*This is a multiple-lines comment block
    that spans over multiple lines*/
?>
```

PHP Variable and Variable Rules

- In PHP, a variable starts with the **$** sign
- 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 the $ 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)
- **Remember that PHP variable names are case-sensitive!**

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

PHP: loosely typed language

- PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
- In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
- You will learn more about strict and non-strict requirements, and data type declarations in the PHP Functions chapter.

PHP Variables Scope

- variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
- **Local**: A variable declared within a function has a local scope and can only be accessed within that function.
- **Global**: A variable declared outside a function has a global scope and can only be accessed outside a function. The global keyword is used to access a global variable from within a function.
- **Static**: Normally, when a function is completed/executed, all of its variables are deleted. but, sometimes we want a local variable not to be deleted. We need it for a further job.

PHP Operators

- Operators are used to perform operations on variables and values.
- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Conditional assignment operators
- Bitwise Operators

PHP Arithmetic Operators

- The PHP arithmetic operators are used with numeric values to perform common arithmetical 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

PHP 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 on 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

PHP Comparison Operators

- The PHP comparison operators are 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 the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $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
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's 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

PHP 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

PHP String Operators

- PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Conditional Assignment Operators

- The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = expr1 ? expr 2 : expr3	Returns the value of $x. The value of $x is expr2 if expr1 = TRUE. The value of $x is expr3 if expr1 = FALSE
??	Null coalescing	$x = expr1 ?? exp r2	Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2. Introduced in PHP 7

Flow control statements

- Conditional statements are used to perform different actions based on different conditions. In PHP we have the following conditional statements.
- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - specifies a new condition to test, if the first condition is false
- **switch statement** - selects one of many blocks of code to be executed

The if Statement

- The if statement executes some code if one condition is true.
- if (*condition*) {
  code to be executed if condition is true;
  }

The if...else Statement

- The if...else statement executes some code if a condition is true and another code if that condition is false.

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

The if...elseif...else Statement

- The if...elseif...else statement executes different codes for more than two conditions.
- if (*condition*) {
  code to be executed if this condition is true;
  } elseif (*condition*) {
  code to be executed if first condition is false and this condition is true;
  } else {
  code to be executed if all conditions are false;
  }

The switch Statement

- The switch statement is used to perform different actions based on different conditions.
- Use the switch statement to **select one of many blocks of code to be executed**.

Cont..

-
  ```
  switch (n) {
    case label1:
      code to be executed if n=label1;
      break;
    case label2:
      code to be executed if n=label2;
      break;
    case label3:
      code to be executed if n=label3;
      break;
      ...
    default:
      code to be executed if n is different from all labels;
  }
  ```

First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Loops

- when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.
- Loops are used to execute the same block of code again and again, as long as a certain condition is true.
- In PHP, we have the following loop types:
- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

while Loop

- The while loop executes a block of code as long as the specified condition is true.
- while (*condition is true*) {
  code to be executed;
  }

do while Loop

- The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

-
  ```
  do {
    code to be executed;
  } while (condition is true);
  ```

for Loop

- The for loop is used when you know in advance how many times the script should run.
- for (*init counter; test counter; increment counter*) {
  code to be executed for each iteration;
  }
- **Parameters**:
- ***init counter***: Initialize the loop counter value
- ***test counter***: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- ***increment counter***: Increases the loop counter value

foreach Loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- foreach (\$*array* as \$*value*) {
    code to be executed;
  }
- For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

Nested loop

- Using loop inside other loops body is known as nested loops.
- You can nest any loop should any condition arise.

```
<?php
 $n=5;
 for($i=1; $i<=$n; $i++)
 {
   for($j=1; $j<=$i; $j++)
   {
     echo ' * ';
   }
   echo '<br>';
 }
?>
```

Break and Continue

- **Break:** It was used to "jump out" of a switch statement.The break statement can also be used to jump out of a loop.
- **Continue:** The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Break v/s Continue

Break	Continue
Break is used to terminate the execution of the loop.	Continue is not used to terminate the execution of loop.
It breaks all iteration.	it skips particular iteration.
When this statement is executed, control will come out from the loop and execute statement immediate after loop.	When this statement is executed, control will not come out from loop, but moves in next iteration.
Break is used with loop as well as switch case.	Continues is only used in loop not in switch case.

Array

- An array is a special variable, which can hold more than one value at a time.
- An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this**: $cars1 = "Volvo"; $cars2 = "BMW"; $cars3 = "Toyota";**
- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- The solution is to create an array!
- An array can hold many values under a single name, and you can access the values by referring to an index number.

Types of array

- **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 rather than in a strict linear index order.
- **Multidimensional array** – An array containing one or more arrays and values are accessed using multiple indices

Creating 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.
- **Method 1:**
- Syntax: array(values);
- Example: $numbers = array(1, 2, 3, 4, 5);
- **Method 2:**
- Syntax: $variable[index] = value;
- $numbers[0] = “one”;

Creating 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 key 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.

Cont..

- **Method 1:**
- Syntax:
- $variable = array(“key” => “value”);
- Example: $car = array(”Ciaz" => 1500000,”Baleno” => 1200000);
- **Method 2:**
- Syntax: $variable[‘key’]= value;
- $car[‘Ciaz’]= 1500000;

Creating 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.
- In this example we create a two dimensional array to store marks of three students in three subjects.
- **Note:** Below example is an associative array, you can create numeric array in the same fashion.
- Syntax: $variable = array(“key1” => array(value1, value2), “key2” => array(value1, value2));
- Example: $marks = array("student1" => array ("DWPD" => 35, "JAVA" => 30, "CMTS" => 39), "student2" => array ("DWPD" => 30, "JAVA" => 32, "CMTS" => 29));
- echo $marks[student1'][‘DWPD’];

PHP Functions (UDF)

- PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.
- There are two parts which should be clear to you −
- Creating a PHP Function
- Calling a PHP Function
- In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Creating PHP Function

- Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it.
- Syntax: **function** function_name(){}
- Example: **function** writeMessage()

PHP Functions with Parameters/passing with values

- PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.
- Syntax: **function** function_name(variable){}
- Example: **function** addFunction($num1, $num2)

Passing Arguments by Reference

- It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.
- Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.
- Syntax: **function** function_name(&variable){}
- Example: **function** addSix(**&**$num)

PHP Functions returning value

- A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.
- You can return more than one value from a function using **return array(1,2,3,4)**.
- Syntax: **function** function_name(){return value;}
- Example: : **function** hello(){return "Hello world !"; }

Setting Default Values for Function Parameters

- You can set a parameter to have a default value if the function's caller doesn't pass it.
- Syntax: **function** function_name($variable = value){}
- Example: : **function** hello($greetings="Hello world!"){return $greetings; }

Array functions

array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_merge()	Merges one or more arrays into one array
array_pop()	Deletes the last element of an array
array_push()	Inserts one or more elements to the end of an array
array_replace()	Replaces the values of the first array with the values from following arrays
array_search()	Searches an array for a given value and returns the key
array_unique()	Removes duplicate values from an array

Cont..

sort()	Sorts an indexed array in ascending order
rsort()	Sorts an indexed array in descending order
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
krsort()	Sorts an associative array in descending order, according to the key
ksort()	Sorts an associative array in ascending order, according to the key

String Functions

chr()	Returns a character from a specified ASCII value
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
stristr()	Finds the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strrev()	Reverses a string
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters

Cont..

rtrim()	Removes whitespace or other characters from the right side of a string
trim()	Removes whitespace or other characters from both sides of a string
substr()	Returns a part of a string
ltrim()	Removes whitespace or other characters from the left side of a string
str_replace()	Replaces some characters in a string (case- sensitive).

Math Functions

abs()	Returns the absolute (positive) value of a number
floor()	Rounds a number down to the nearest integer
max()	Returns the highest value in an array, or the highest value of several specified values
min()	Returns the lowest value in an array, or the lowest value of several specified values
pow()	Returns x raised to the power of y
rand()	Generates a random integer
round()	Rounds a floating-point number
sqrt()	Returns the square root of a number
pi()	Returns the value of PI
fmod()	Returns the remainder of x/y

Date Functions

checkdate()	Validates a Gregorian date
date()	Formats a local date and time
getdate()	Returns date/time information of a timestamp or the current local date/time
time()	Returns the current time as a Unix timestamp
strftime()	Formats a local time and/or date according to locale settings
localtime()	Returns the local time
gmdate()	Formats a GMT/UTC date and time

File Functions

mkdir()	Creates a directory
rename()	Renames a file or directory
rmdir()	Removes an empty directory
fopen()	Opens a file or URL
fclose()	Closes an open file
copy()	Copies a file
basename()	Returns the filename component of a path
fwrite()	Writes to an open file

Miscellaneous Functions

define()	Defines a constant
exit()/die()	Prints a message and exits the current script. (die() does not print anything)
Require()	The Require() function is also used to put data of one PHP file to another PHP file. If there are any errors then the require() function produces a warning and a fatal error and stops the execution of the script i.e. the script will continue to execute.
Include()	The Include() function is used to put data of one PHP file into another PHP file. If errors occur then the include() function produces a warning but does not stop the execution of the script i.e. the script will continue to execute.
header()	It is used to redirect a from one web page to another web page in PHP.
Isset()	The isset() function checks whether a variable is set, which means that it has to be declared and is not NULL.

Questions

- Constant can be defined using___________________()
- PHP stands for ?
- Who is the father of php?
- Which function is used to delete variable ?
- Explain default argument function.
- Explain Associated array and foreach loop.
- Explain variable length argument function.
- Explain substr() with example.
- What is PHP ?
- Explain $ POST and $_GET with example.
- What is UDF ? And explain types of UDF.