UNIT - 1

Introduction To PHP

1. Define PHP?

PHP Hypertext Pre-processor is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is created in 1994 by Rasmus Lerdorf.

Advantages of PHP -

- It is open source.
- Widely used in all over the world
- Free to download
- It is executed on the server
- To execute php code no need compiler.

2. Define MySQL?

MySQL is a Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). It is also free and open source.

3. Explain PHP Syntax?

PHP Syntax:

```
<?php
        //statements;
?>
```

Explaination -

- PHP code is start with <?
- Every PHP statements end with a semicolon (;)
- PHP code save with .php extension.
- PHP contain some HTML tag and PHP code.
- You can place PHP code any where in your document.

4. Write hello world program using PHP?

```
<!DOCTYPE html>
<html>
        <body>
                <h1>My first PHP page</h1>
                <?php
                        echo "Hello World!";
                ?>
        </body>
</html>
```

5. Define Variables in PHP ? Give example

Variables in PHP are identifiers prefixed with a dollar sign ($). A variable may hold a value of any type.

Example : $name, $Age , $_debugging, $MAXIMUM_IMPACT

6. Define Variable Variable in PHP? Give example

You can reference the value of a variable whose name is stored in another variable.

For example:

```
$foo = 'bar';

$$foo = 'baz';
```

7. Explain variable Scope In PHP?

PHP has three different variable scopes:

1. local -

A variable declared in a function is local to that function. That is, it is visible only to code in that function (including nested function definitions); it is not accessible outside the function.

2. global -

- Variables declared outside a function are global.
- That is, they can be accessed from any part of the program.
- However, by default, they are not available inside functions. To allow a function to access a global variable, you can use **the global keyword** inside the function to declare the variable within the function

3. static -

A static variable retains its value between calls to a function but is visible only within that function. You declare a variable static with the static keyword.

Each time the function is called, that variable will still have the information it contained from the last time the function was called.

Example on static variable

```
<html>

<body>

<?php

        function myFunction() { // Function definition

                static $x=45;

                echo $x;

                echo "<br/>";

                $x++;

        }

        myFunction();

        myFunction();

        myFunction();

        myFunction();

        myFunction();

?>

<body>

<html>
```

OUTPUT of the above given Example is as follows: 45 46 47 48 49

8. List and explain Data types in PHP?

PHP provides eight types of values, or data types.

- Four are scalar (single-value) types: integers, floating-point numbers, strings, and Booleans.
- Two are compound (collection) types: arrays and objects.

- The remaining two are special types: resource and NULL.

1. Integer -

Integers are whole numbers, such as 1, 12, and 256

Example:

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

Output: int(5985)

2. Floating-Point Numbers -

Floating-point numbers (often referred to as real numbers) represent numeric values with decimal digits.

Ex - 6.73, 13.144

//Write example

3. Strings -

A string is a sequence of characters of arbitrary length. String literals are delimited by either single or double quotes:

Ex: ‘BCA‘ “College“

//Write example

4. Booleans -

A Boolean value represents a "truth value "it says whether something is true or not.

//Write example

5. Arrays -

- An array stores multiple values in one single variable.
- An array holds a group of values, which you can identify by position (a number, with zero being the first position) or some identifying name (a string), called an associative:

//Write example

5. 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 a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

For example -

```
<html>

        <body>

                <?php

                class Car {

                        function Car() {

                        $this->model = "VW";

                        }

                }

                // create an object

                $herbie = new Car();
```

```
                    // show object properties

                    echo $herbie->model;

                    ?>

            </body>

</html>
```

6. NULL Value -

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.

//Write example

7. Resource -

- The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.
- A common example of using the resource data type is a database call.
- Their main benefit is that they take care of memory management by themselves. When the last reference to a resource value goes away, the extension that created the resource is called to free any memory, close any connection, etc

Example:

```
<?php

        $fp=fopen("test.txt","w");

        var_dump($fp);

?>
```

Output -

```
resource(5) of type (stream)
```

9. Explain Setting and Checking Variable Data Types?

Unlike other programming languages, where a variable's data type must be explicitly defined by the programmer, PHP automatically determines a variable's data type from the content it holds.

And if the variable's content changes over the duration of a script, the language will automatically set the variable to the appropriate new data type.

Functions for setting and checking variable data types -

- **empty() -** Checks whether a variable is empty
- **is_array() -** Checks whether a variable is an array.
- **is_bool() -** Checks whether a variable is boolean
- **is_float() -** Checks whether a variable is float
- **is_int() -** Checs whether a variable is of type integer.

10. Define Constants? Write its syntax

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Syntax -

```
define(name, value, case-insensitive)
```

Parameters:

- **name:** Specifies the name of the constant
- **value:** Specifies the value of the constant
- **case-insensitive:** Specifies whether the constant name should be case-insensitive. Default is false

Example:

```
<?php

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

echo greeting;
```

```
?>
```

11. Define Operators ? List and explain the types of operators in PHP

Operators are used to perform operations on variables and values.

- **Arithmetic operators -** +, -, *, /, %, **(Exponentiation)
- **Assignment operators -** =, +=, -=, *=, /=, %=.
- **Comparison operators -** ==, >, <, >=, <=, !=, ===(identical), !==(Not identical)
- **Increment/Decrement operators -** ++, -- Pre-increment and Post increment ++, --
- **Logical operators -** and, or, xor, --------- &&, ||, !
- **String operators -**

  . Operator -

  It is used for Concatination.

  Example - $txt1.$txt2

  .= Operator -

  It used for concatination and assignment.

  Example - $txt1 .= $txt2
- **Array operators -**

  The PHP array operators are used to compare arrays.

  Example of array operator

```
<html>
        <body>
                <?php
                        $x = array("a" => "red", "b" => "green");
                        $y = array("c" => "blue", "d" => "yellow");
```

```
                                        print_r($x + $y); // union of $x and $y
                                        var_dump($x == $y);
                                        var_dump($x != $y);
                                ?>
                        </body>
                </html>
```

12. Write a PHP Program to add two numbers ?

```
<html>
        <body>
        <form method="post">
                Enter First Number:
                <input type="number" name="number1" /><br><br>
                Enter Second Number:
                <input type="number" name="number2" /><br><br>
                <input type="submit" name="submit" value="Add">
        </form>
        <?php
                if(isset($_POST['submit'])) {
                        $number1 = $_POST['number1'];
                        $number2 = $_POST['number2'];
                        $sum = $number1+$number2;
                       echo "The sum of $number1 and $number2 is: ".$sum;
                }
        ?>
        </body>
</html>
```

UNIT - 2

Conditionals And Looping's

1. Explain PHP Simple Conditional Statements?

LIKE MOST PROGRAMMING LANGUAGES, PHP ALSO ALLOWS YOU TO WRITE CODE THAT PERFORM DIFFERENT ACTIONS BASED ON THE RESULTS OF A LOGICAL OR COMPARATIVE TEST CONDITIONS AT RUN TIME.

Simple Conditional Statement -

- **The If statement**

  PHP if statement allows conditional execution of code. It is executed if condition is true. If statement is used to execute the block of code exist inside the if statement only if the specified condition is true.

  Syntax -

```
if(Condition){

        //statements

}
```

//Write flowchart and example

- **The If...Else statement**

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

  Syntax -

```
if(condition){

        //Statements

} else{

        //Statements

}
```

//Write flowchart and example

- **The Nested...If statement**

  The nested if statement contains the If bloc inside another if block. The inner if statement executes only when specified condition in outer if statement is true.

  Syntax -

```
If(condition){

        //Statements

        if(condition){

                //Nested if statements

        }

}
```

//Write flowchart and example

2. List and explain more complex conditional statements?

The if-else statement lets you define actions for two eventualities: A true condition and a false conditon. In Reality, however your program may have more than just these two simple outcomes to content with.

For these situation, PHP offers two constructs that alow the programmer to account for multiple possibilities.

- **The If-elseif-else statements -**

  The if-elseif-else statement lets you chain together multiple if-else statements. Thus allowing the programmer to define actions for than just two possible outcomes.

  Syntax -

  ```
  if(condition) //Statements

  if else(condition) //Statements

  else //Statements
  ```

  //Write example

- **Switch-case statements -**

  PHP Switch statement is used to execute one statement from multiple condition. It works like PHP if-else-if statements.

  Syntax -

  ```
  switch(Expression){

          case value:

                  // Code

                  break;

          default:

                  //Code to be executed if no condition matches

  }
  ```

//Write example and flowchart

3. Explain Loops in PHP?

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:

1) WHILE -

LOOPS THROUGH A BLOCK OF CODE AS LONG AS THE SPECIFIED CONDITION IS TRUE

SYNTAX

```
WHILE (CONDITION IS TRUE) {

CODE TO BE EXECUTED;

}
```

2) DO...WHILE -

LOOPS THROUGH A BLOCK OF CODE ONCE, AND THEN REPEATS THE LOOP AS LONG AS THE SPECIFIED CONDITION IS TRUE.

SYNTAX

```
DO {

CODE TO BE EXECUTED;

} WHILE (CONDITION IS TRUE)
```

3) FOR -

LOOPS THROUGH A BLOCK OF CODE A SPECIFIED NUMBER OF TIMES.

SYNTAX

FOR (INIT COUNTER; TEST COUNTER; INCREMENT COUNTER) {

CODE TO BE EXECUTED FOR EACH ITERATION;

}

4) FOREACH -

LOOPS THROUGH A BLOCK OF CODE FOR EACH ELEMENT IN AN ARRAY

SYNTAX

FOREACH ($ARRAY AS $VALUE) {

CODE TO BE EXECUTED;

}

EXAMPLE -

<?PHP

$COLORS = ARRAY("RED", "GREEN", "BLUE", "YELLOW");

FOREACH ($COLORS AS $VALUE) { ECHO "$VALUE <BR>";

}

?>

4. List and explain PHP String functions?

1) PHP STRTOLOWER() FUNCTION -

THE STRTOLOWER() FUNCTION RETURNS STRING IN LOWERCASE LETTER.

SYNTAX

STRING STRTOLOWER (STRING $STRING)

EXAMPLE

<?PHP

$STR="MY NAME IS KHAN";

```
        $STR=STRTOLOWER($STR); ECHO $STR;

?>
```

2) PHP STRTOUPPER() FUNCTION -

THE STRTOUPPER() FUNCTION RETURNS STRING IN UPPERCASE LETTER.

SYNTAX

STRING STRTOUPPER (STRING $STRING)

EXAMPLE

```
<?PHP

        $STR="MY NAME IS KHAN";

        $STR=STRTOUPPER($STR); ECHO $STR;

?>
```

3) PHP UCFIRST() FUNCTION -

THE UCFIRST() FUNCTION RETURNS STRING CONVERTING FIRST CHARACTER INTO UPPERCASE. IT DOESN'T CHANGE THE CASE OF OTHER CHARACTERS.

SYNTAX

STRING UCFIRST (STRING $STR)

EXAMPLE

```
<?PHP

        $STR="MY NAME IS KHAN";

        $STR=UCFIRST($STR); ECHO $STR;

?>
```

4) PHP UCWORDS() FUNCTION -

THE UCWORDS() FUNCTION RETURNS STRING CONVERTING FIRST CHARACTER OF EACH WORD INTO UPPERCASE.

SYNTAX

STRING UCWORDS (STRING $STR)

EXAMPLE

```
<?PHP
        $STR="MY NAME IS SONOO JAISWAL";
        $STR=UCWORDS($STR); ECHO $STR;
?>
```

5) PHP STRREV() FUNCTION -

THE STRREV() FUNCTION RETURNS REVERSED STRING.

SYNTAX

STRING STRREV (STRING $STRING)

EXAMPLE

```
<?PHP
        $STR="MY NAME IS SONOO JAISWAL";
        $STR=STRREV($STR); ECHO $STR;
?>
```

6) PHP STRLEN() FUNCTION -

THE STRLEN() FUNCTION RETURNS LENGTH OF THE STRING.

SYNTAX

STRLEN (STRING $STRING)

EXAMPLE

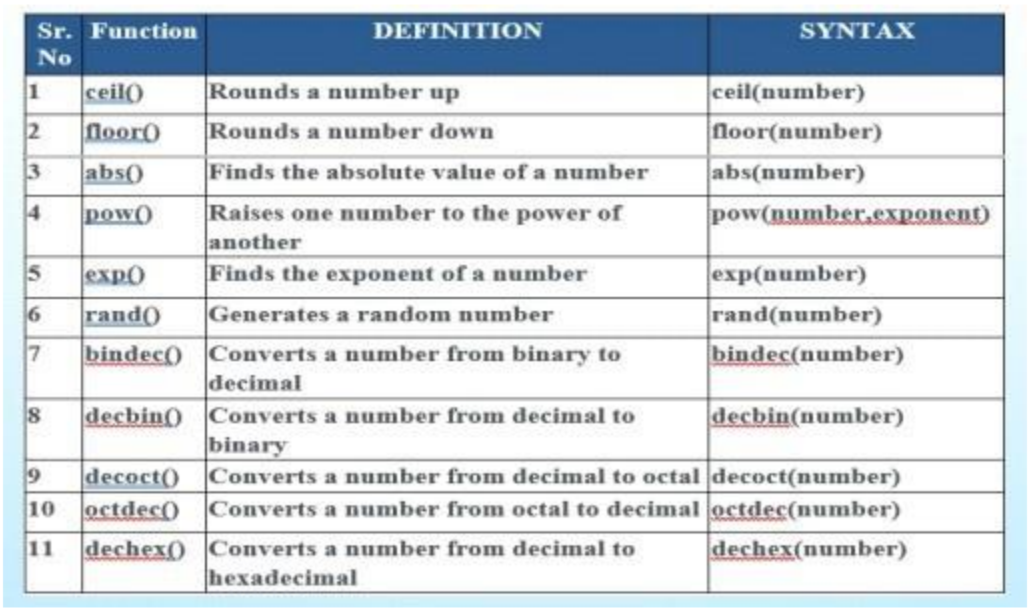
```
<?PHP

        $STR="MY NAME IS SONOO JAISWAL";

        $STR=STRLEN($STR); ECHO $STR;

?>
```

5. List PHP Numeric functions?



Sr. No	Function	DEFINITION	SYNTAX
1	ceil()	Rounds a number up	ceil(number)
2	floor()	Rounds a number down	floor(number)
3	abs()	Finds the absolute value of a number	abs(number)
4	pow()	Raises one number to the power of another	pow(number,exponent)
5	exp()	Finds the exponent of a number	exp(number)
6	rand()	Generates a random number	rand(number)
7	bindec()	Converts a number from binary to decimal	bindec(number)
8	decbin()	Converts a number from decimal to binary	decbin(number)
9	decoct()	Converts a number from decimal to octal	decoct(number)
10	octdec()	Converts a number from octal to decimal	octdec(number)
11	dechex()	Converts a number from decimal to hexadecimal	dechex(number)

//For each function write example and explain

UNIT - 3

Working With Arrays

1. Define Arrays with syntax?

Array variables are “special,” because they can hold more than one value at a time. This makes them particularly useful for storing related values.

Syntax -

```
<?php

        $variable_name = array(element1, element2, element3, .... );

?>
```

2. Explain count() function?

The count() function is used to return the length (the number of elements) of an array

Example

```
<?php

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

        echo count($cars);

?>
```

3. List and Explain Types of arrays in PHP?

1. PHP Indexed Arrays:

Indexed arrays are Arrays with a numeric index. There are two ways to create indexed arrays:

1. The index can be assigned automatically (index always starts at 0), like this:

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

2. Or the index can be assigned manually:

```
        $cars[0] = "Volvo";

        $cars[1] = "BMW";

        $cars[2] = "Toyota";
```

Example -

```
<html>
        <body>
        <?php
                $cars = array("Volvo", "BMW", "Toyota");
                echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
        ?>
        </body>
</html>
```

2. PHP Associative Arrays:

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
        $age['Peter'] = "35";

        $age['Ben'] = "37";

        $age['Joe'] = "43";
```

The named keys can then be used in a script:

Example

```
<?php
```

```
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

        echo "Peter is " . $age['Peter'] . " years old.";

?>
```

3. PHP - Multidimensional Arrays:

A multidimensional array is an array containing one or more arrays. For a two-dimensional array you need two indices to select an element.

For a three-dimensional array you need three indices to select an element.

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array (

array("Volvo",22,18),

array("BMW",15,13),

array("Saab",5,2),

array("Land Rover",17,15)

);
```

Example -

```
<html>

        <body>

        <?php

                $cars = array (

                        array("Volvo",22,18),

                        array("BMW",15,13),

                        array("Saab",5,2),

                        array("Land Rover",17,15)
                );
                echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
                echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
                echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
                echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
        ?>
        </body>
</html>
```

4. Explain how to process arrays with loops and iterations with different types of arrays?

1. Processing Arrays with Loops and Iterations an Indexed Array -

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

```
<html>
        <body>
                <?php
                        $cars = array("Volvo", "BMW", "Toyota");
                        $arrlength = count($cars);
                        for($x = 0; $x < $arrlength; $x++) {
                        echo $cars[$x];
                        echo "<br>";
                }
                ?>
        </body>
</html>
```

2. Processing Arrays with Loops and Iterations an Associative Array -

```
<?php
// define array
$cities = array(
        "United Kingdom" => "London",
        "United States" => "Washington",
        "France" => "Paris",
        "India" => "Delhi"
);
// iterate over array
// print each value
foreach ($cities as $key => $value) {
        echo "$value is in $key. \r\n";
}
?>
```

3. Processing Arrays with Loops and Iterations an Multidimensional Array

We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):

Example -

```
<html>
        <body>
        <?php
                $cars = array (
                        array("Volvo",22,18),
                        array("BMW",15,13),
                        array("Saab",5,2),
                        array("Land Rover",17,15)
                );
                for ($row = 0; $row < 4; $row++)
                {
                        echo "<p><b>Row number $row</b></p>";
                        echo "<ul>";
                        for ($col = 0; $col < 3; $col++)
                        {
                                echo $cars[$row][$col];
                        }
                }
        ?>
        </body>
</html>
```

5. Explain Array Iterator with example?

Alternatively, you may prefer to use an ArrayIterator which provides a ready-made, extensible tool to loop over array elements.

Here’s a simple example:

```
<?php
        $cities = array(
                "United Kingdom" => "London",
                "United States" => "Washington",
                "France" => "Paris",
                "India" => "Delhi"
        );
        $iterator = new ArrayIterator($cities); // create an ArrayIterator object
        $iterator->rewind(); // rewind to beginning of array
        while($iterator->valid())
        {
                // iterate over array and print each value
                print $iterator->current() . " is in " . $iterator->key() . ". \r\n";
                $iterator->next();
        }
?>
```

6. Explain how to use arrays with forms?

Arrays are particularly potent when used in combination with form elements that support more than one value, such as multiple-selection list boxes or grouped checkboxes.

Example -

```
<form method="post" action="array-form.php">
        Select your favourite artists: <br />
        <select name="artists[]" multiple="true">
                <option value="Britney Spears">Britney Spears</option>
```

```
                    <option value="Aerosmith">Aerosmith</option>

                    <option value="Black-Eyed Peas">Black-Eyed Peas</option>

                    <option value="Diana Ross">Diana Ross</option>

                    <option value="Foo Fighters">Foo Fighters</option>

          </select>

          <p>

          <input type="submit" name="submit" value="Submit" />

 </form>

<?php

          foreach($_POST['artists'] as $a) {

                    echo $a.", "

          }

?>
```

Notice the 'name' attribute of the <select> element, which is named artists[].

This tells PHP that, when the form is submitted, all the selected values from the list should be converted into elements of an array. The name of the array will be $_POST['artists'], and it will look something like this:

```
Array (

          [0] => Britney Spears

          [1] => Black-Eyed Peas

          [2] => Diana Ross

)
```

7. Explain Some array functions with examples?

1. explode() -

This function, which accepts two arguments—the separator and the source string—and returns an array.

Here's an example:

```
<?php
	// define string
	$str = 'tinker,tailor,soldier,spy';
	// convert string to array
	// output: ('tinker', 'tailor', 'soldier, 'spy')
	$arr = explode(',', $str);
	print_r($arr);
?>
```

2. implode() -

It's also possible to reverse the process, joining the elements of an array into a single string using user-supplied "glue." The PHP function for this is named implode().

Example -

```
<?php
	// define array
	$arr = array('one', 'two', 'three', 'four');
	// convert array to string
	// output: 'one and two and three and four'
	$str = implode(' and ', $arr);
	print_r($str);
?>
```

3. range() -

This function accepts two end points and returns an array containing all the numbers between those end points.

Example -

```
<?php

// define array

$arr = range(1,1000);

print_r($arr);

?>
```

4. min() and max() -

The min() function returns the minimum number in the array and the max() function returns the maximum number in the array. Note that these functions only accepts array of numbers.

```
<?php

        // define array

        $arr = array(7, 36, 5, 48, 28, 90, 91, 3, 67, 42);

        // get min and max

        // output: 'Minimum is 3 and maximum is 91'

        echo 'Minimum is ' . min($arr) . ' and maximum is ' . max($arr);

?>
```

5. sort() -

This function returns the sorted array.

//write example

8. Write php array functions with their syntax and meaning?

1. array_slice() -

PHP allows you to slice an array into smaller parts with the array_slice() function.

Syntax -

array_slice(array, index position, number of elements);

2. array_shift() -

This function removes the first element of an array.

Syntax -

array_shift(array_name);

3. array_pop() -

This function removes an element from the end of an array.

Syntax -

array_pop(array_name);

4. array_push() -

This function adds an element at the end of an array.

Syntax -

array_push(array_name, "element");

5. array_unshift() -

This functions adds and element at the start of an array.

Syntax -

array_unshift(array_name, "element");

6. shuffle() -

PHP's shuffle() function re-arranges the elements of an array in random order.

Syntax -

shuffle(array_name);

7. array_reverse() -

This function reverses the order of an array's elements.

Syntax -

array_reverse(array_name);

8. in_array() -

The in_array() function looks through an array for a specified value and returns true if found.

Syntax -

in_array('value to be searched', array_name);

9. array_merge() -

PHP lets you merge one array into another with its array_merge() function, which accepts one or more array variables

Syntax -

array_merge(array_1, array_2,.....);

9. List date and time functions?

1. checkdate() -

Checks a date for validity.

Example -

<?php

```
if(checkdate(2, 30, 2008))
{
        echo "Date is valid";
} else{
        echo "Date is invalid";
}
```

2. strtotime() -

Creates timestamps from english language descriptions.

3. gmdate() -

Expresses a timestamp in GMT.

10. Explain date() function in PHP?

The date() function formats a local date and time, and returns the formatted date string.

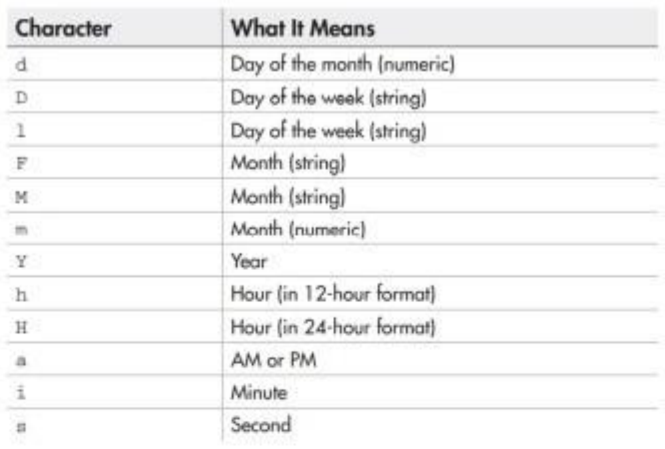
Syntax -

date(format, timestamp)

Example -

```
<?php
        // Prints the day
        echo date("l") . "<br>";
?>
```

Formatting codes for the date() function -



Character	What It Means
d	Day of the month (numeric)
D	Day of the week (string)
l	Day of the week (string)
F	Month (string)
M	Month (string)
m	Month (numeric)
Y	Year
h	Hour (in 12-hour format)
H	Hour (in 24-hour format)
a	AM or PM
i	Minute
s	Second

UNIT - 4

Using Functions And Classes

1. Define functions?

In PHP, a function is a block of code that can be called by a PHP script to perform a task.

2. Explain the main components of functions?

There are three components to every function:

- Arguments, which serve as inputs to the function

- Return values, which are the outputs returned by the function
- The function body, which contains the processing code to turn inputs into outputs

Example :

```
function myMessage() {
 echo "Hello world!";
}
```

3. Explain setting default argument values?

User can assign default values to any or all of these arguments; these default values are used in the event the function invocation is missing some arguments.

Here's an example, which generates an e-mail address from supplied username and domain arguments; if the domain argument is missing, a default domain is automatically assigned.

```
<?php
	// generate e-mail address from supplied values
	function buildAddress($username, $domain = 'mydomain.info') {
	 return $username . '@' . $domain;
	}
	// function invocation
	// without optional argument
	echo 'My e-mail address is ' . buildAddress('john');
	// function invocation
	// with optional argument
	echo 'My e-mail address is ' . buildAddress('jane', 'cooldomain.net');
?>
```

4. Explain dynamic argument lists?

A PHP function definition normally has a fixed argument list, where the number of arguments is known in advance.

However, PHP also lets you define functions with so-called variable-length argument lists, where the number of arguments can change with each invocation of the function.

```
<?php

        // calculate average of supplied values

        function calcAverage() {

                $args = func_get_args();

                $count = func_num_args();

                $sum = array_sum($args);

                $avg = $sum / $count;

                return $avg;

        }

        // function invocation

        // with 3 arguments

        echo calcAverage(3,6,9);

        // function invocation

        // with 8 arguments

        echo calcAverage(100,200,100,300,50,150,250,50);

?>
```

5. Define Recursive function with example?

A recursive function is a function that calls itself repeatedly until a condition is satisfied.

Example -

```
<?php
        function display($number) {
                if($number<=5){
                echo "$number <br/>";
                display($number+1);
                }
        }
        display(1);
?>
```

6. Define Class? Write syntax and example

Class is a blueprint to create objects.

Syntax -

```
class class_name{
        //code
}
```

Example -

```
<?php
        // class definition
        class Automobile {
                //    properties
                public   $color;
                public $model;
                // methods
                public function accelerate() {
```

```
                echo 'Accelerating...';

                }

                public function brake() {

                        echo 'Slowing down...';

                }

                public function turn() {

                        echo 'Turning...';

                }

        }



        // instantiate object

        $car = new Automobile;

        // set object properties

        $car->color = 'red';

        $car->model= 'Ford';

        // invoke object methods

        $car->accelerate();

        $car->turn();

?>
```

7. Define constructor and destructors?

Constructor -

It is a special type of method that is created whenever a new instance of a class is created.

Syntax -

```
__construct() {
        //Statements
}
```

Destructor -

In PHP, a destructor is a method that's automatically called when an object is no longer needed or referenced, or when the script is stopped or exited.

Syntax -

```
__destruct() {
        //Statements
}
```

8. Define polymorphism?

It is defined as the ability of objects of different classes to respond differently based on the same message

9. Define method overloading or function overloading?

Function overloading or method overloading is a feature that permits making creating several methods with a similar name that works differently from one another in the type of the input parameters it accepts as arguments

```
class SampleClass {
   function __call($function_name, $arguments)
   {
      $count = count($arguments);
      // Check function name
      if ($function_name == 'add') {
         if ($count == 2) {
                return array_sum($arguments);

            } else if ($count == 3) {

                return array_sum($arguments) > 10 ? 10 : array_sum($arguments);

            }

        }

    }

}
```

10. What is __call() method?

This is a magic method that PHP calls when it tries to execute a method of a class and it doesn't find it. This magic keyword takes in two arguments: a function name and other arguments to be passed into the function.

Syntax -

```
function __call(string $function_name, array $arguments) {}
```

11. Define abstraction?

Abstraction in object-oriented programming (OOP) refers to the concept of hiding the complex implementation details of an object and exposing only the essential features or functionalities.

12. Define Encapsulation?

Encapsulation refers to the binding data and methods into single unit.

13. Explain inheritance?

Inheritance is the ability to acquire properties and behaviour of parent class into child class.

Example -

```
<?php

        class Animal {
              public $name;

              public function __construct($name) {

                     $this->name = $name;

              }

              public function eat() {

                      return $this->name . ' is eating.';

              }

       }

       class Dog extends Animal {

              public function bark() {

                     return $this->name . ' is barking.';

              }

       }


       $obj = new Dog("Jerman Sheffard");

       $obj.bark();

?>
```

UNIT - 5

Introduction To Laravel Framework

1. Define Laravel?

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application.

Advantages -

- The web application becomes more scalable, owing to the Laravel framework.
- Considerable time is saved in designing the web application, since Laravel reuses the components from other framework in developing web application.
- It includes namespaces and interfaces, thus helps to organize and manage resources.

2. Define Composer?

Composer is a tool which includes all the dependencies and libraries.

3. Define Artisan?

Command line interface used in Laravel is called **Artisan**. It includes a set of commands which assists in building a web application.

4. List features of Laravel?

Modularity

Laravel provides 20 built in libraries and modules which helps in enhancement of the application.

Testability

Laravel includes features and helpers which helps in testing through various test cases.

Routing

Laravel provides a flexible approach to the user to define routes in the web application.

Configuration Management

A web application designed in Laravel will be running on different environments, which means that there will be a constant change in its configuration.

Query Builder and ORM

Laravel incorporates a query builder which helps in querying databases using various simple chain methods

Schema Builder

Schema Builder maintains the database definitions and schema in PHP code.

Authentication

Laravel eases designing authentication as it includes features such as **register, forgot password** and **send password reminders.**

E-mail

Laravel includes a **mail** class which helps in sending mail with rich content and attachments from the web application.

5. Write steps to install laravel?

Step 1 – Visit the following URL and download composer to install it on your system.

https://getcomposer.org/download/

Step 2 – After the Composer is installed, check the installation by typing the Composer command in the command prompt

Step 3 – Create a new directory anywhere in your system for your new Laravel project. After that, move to path where you have created the new directory and type the following command there to install Laravel.

composer create-project laravel/laravel –-prefer-dist

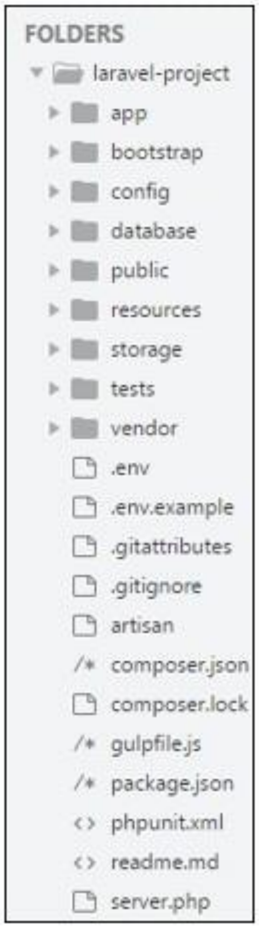
Step 4 – The above command will install Laravel in the current directory. Start the Laravel service by executing the following command.

php artisan serve

Step 5 – After executing the above command,

Step 6 – Copy the URL underlined in gray in the above screenshot and open that URL in the browser. If you see the following screen, it implies Laravel has been installed successfully.

6. Explain Laravel Application Structure?



1. App

It is the application folder and includes the entire source code of the project. . The app folder comprises various sub folders as explained below −

Console

Console includes the artisan commands necessary for Laravel. It includes a directory named **Commands**, where all the commands are declared with the appropriate signature. The file **Kernal.php** calls the commands declared in **Inspire.php**

Events

This folder includes all the events for the project.

Exceptions

This folder contains all the methods needed to handle exceptions.

Http

The **Http** folder has sub-folders for controllers, middleware and application requests.

2. Bootstrap

This folder encloses all the application bootstrap scripts. It contains a sub-folder namely **cache**, which includes all the files associated for caching a web application.

3. Config

The **config** folder includes various configurations and associated parameters required for the smooth functioning of a Laravel application.

4. Database

As the name suggests, this directory includes various parameters for database functionalities. It includes three sub-directories as given below –

- **Seeds** – This contains the classes used for unit testing database.
- **Migrations** – This folder helps in queries for migrating the database used in the web application.
- **Factories** – This folder is used to generate large number of data records.

5. Public

It is the root folder which helps in initializing the Laravel application. It includes the following files and folders –

- **.htaccess** – This file gives the server configuration.
- **javascript and css** – These files are considered as assets.
- **index.php** – This file is required for the initialization of a web application.

6. Resources

Resources directory contains the files which enhances your web application.

The sub-folders included in this directory and their purpose is explained below –

- **assets** – The assets folder include files such as LESS and SCSS, that are required for styling the web application.
- **lang** – This folder includes configuration for localization or internalization.
- **views** – Views are the HTML files or templates which interact with end users and play a primary role in MVC architecture.

7. Storage

This is the folder that stores all the logs and necessary files which are needed frequently when a Laravel project is running. The sub-folders included in this directory and their purpose is given below –

- **app** – This folder contains the files that are called in succession.
- **framework** – It contains sessions, cache and views which are called frequently.
- **Logs** – All exceptions and error logs are tracked in this sub folder.

8. Tests

All the unit test cases are included in this directory. The naming convention for naming test case classes is **camel_case** and follows the convention as per the functionality of the class.

9. Vendor

Laravel is completely based on Composer dependencies, for example to install Laravel setup or to include third party libraries, etc.

7. Define Environment variables?

Environment variables are those which provide a list of web services to your web application.

8. What is maintainance mode? How to enable it?

Sometimes you may need to update some configuration values or perform maintenance on your website.

How to enable it?

The following command is used to enable maintainance mode -

php artisan down for enabling

and

php artisan up for disabling

9. Define Routing?

In PHP, routing is a mechanism that determines which controllers and actions to execute based on a requested URL.

10. List and explain categories of routing in Laravel?

1. Basic Routing

All the application routes are registered within the **app/routes.php** file. This file tells Laravel for the URIs it should respond to and the associated controller will give it a particular call.

```
<?php
Route::get('/', function () {
   return view('welcome');
});
```

2. Route Parameters

Sometimes in the web application, you may need to capture the parameters passed with the URL. For this, you should modify the code in **routes.php** file.

Example -

```
Route::get('ID/{id}',function($id) {
   echo 'ID: '.$id;
});
```

3. Named Routes

Named routes allow a convenient way of creating routes. The chaining of routes can be specified using name method onto the route definition.

The following code shows an example for creating named routes with controller –

Route::get('user/profile', 'UserController@showProfile')->name('profile');