INDEX

INTRODUCTION

SIDES OF WEB DEVELOPMENT:-

The client-side environment (Front-end):- is the web browser. Here scripting languages processing take place on end user's computer. The source code transferred to the client's computer over the internet running directly on the web browser. It is important that client-side scripting language is enabled on web browser.

The server-side environment (Back-end):- is the web server. Any request by a user is fulfilled by this script running directly on web server, by generating dynamic HTML pages. This content sent to the client's browser. This environment enables creation of interactive websites that interface with database and data stores on servers.

In client-side scripting, scripts executed by viewing the web browsers, generally in JavaScript. But in server-side scripting, responses are customized based on user requirements, queries and access rights. This gives the server-side, programming leverage over the client-side. Examples are PHP, python, ruby, etc.

HISTORY:-

It was started as a small open-source project, which became more and more profound as people started comprehending its utility. Its first version was released in 1994 by Rasmus Lardof.

- PHP recursively corresponds to "Hypertext Preprocessor". At the time of its development, it was famous as "Personal Home Pages". Hence, the name PHP.
- It is an HTML embedded server-side programming language. It manages dynamic content, session tracking, databases and can even build an entire e-commerce site.
- It can be integrated with many databases, such as MySQL, PostgreSQL, Oracle, SyBase, Informix and Microsoft's SQL Server.
- It is very fast executing language, especially when compiled using Apache module on UNIX. The MySQL database, once started, can run many complex and large queries at a record time.
- It also supports many major protocols, such as POP3, LDAP and IMAP. PHP has also supported Java and object architectures, such as COM and COBRA, making it an n-tier development language.
- PHP practically owns the web. It empowers more than 80% of the server-side programming on the web. Websites like facebook, Wikipedia and tumblr are powered by PHP.
- Syntax of PHP is like C language.

IMPORTANCE:-

It is server-side programming language, solely created for web development, unlike Java and Python. This makes it running on servers relatively easy. It became more powerful and secure with each newer version. It supports well-structured frameworks, such as Laravel and Symfony, which have enabled development of elegant and high-quality applications.

Dynamic Webpage Creation: - today, PHP is used by most of the websites developers to enhance the features and appearance of their websites. It enables easy creation of dynamic web pages.

Support for OOPs: - it has enabled many object-oriented programming (OOP) features in latest versions; it was not designed as a OOPs language. Some OOP concepts are supported by PHP.

- **Constructors: -** it allows the developers to include constructors in classes. These constructors help in creating of new instance of class.
- **Destructors: -** the constructors occupy some amount of memory for the new instances. To free this storage space after their execution, destructors are called. It is done when no reference of object is left behind.
- **Access modifiers: -** there are three types of access modifiers available as public, private and protected. Private used define variables and methods that only visible in class. Public define variables and methods for global visibility; it can be accessed by any class or object outside the scope of class in which they defined. Protected used when the visibility is enabled only to some specific classes.
- **Interfaces: -** are like classes which are defined by interface keyword. Their functionality is like that of classes, except that all methods declared in an interface are public. Interface cannot include member variables.

MySQL Extension: - MySQL database has progressed towards being an integral partner of PHP in all these years of its evolution. Every PHP developer uses MySQL as their choice for websites integrated with databases.

SQLite Embedded database: - Though MySQL has been accepted as the database for PHP applications, PHP has embedded SQLite database, which advisable to use in case of smaller applications. SQLite supports common database features as Transactions, Sub Queries and Triggers.

Better Error and Exception Handling: - It was introduced in PHP 5. Through an efficient exception mechanism, it was possible to skip the checking process of the return value of every function. Additionally, we could separate our program logic from that of the logic of exception handling. In latest version PHP 7, the range of exceptions was increased, covering many fatal errors, which were converted to exceptions. Also exception hierarchy was considerably improved in comparison to PHP 5.

USES OF PHP:-

The most important function of PHP is that of performing system functions. These include all the functions involved in manipulating a file on system, such as creating, opening, closing, reading and writing. Also PHP perform some these functions:-

- PHP handles forms by gathering data from files and saving it back in the files. Through e-mails that are running on PHP, the user can send and receive data.
- It helps us add, delete and modify elements in our database.
- It helps in accession of cookies and setting these cookie variables.

- It can also restrict specific or all users to only a certain part of a website.
- It is widely used for encrypting data on cloud.

CHARACTERISTICS: -

Simplicity: - based on the job requirement, a PHP script can either be of one or of a thousand of lines. There is no library inclusion requirement or special directory compilation to begin coding. The PHP engine begins the execution from the first escape sequence () itself. If the code is syntactically correct, it will be displayed exactly as desired.

Efficiency: - in a multiuser environment like the World Wide Web, efficiency is important. By including OOPs features in programming, PHP has introduced some resource allocation mechanisms and session management features. Also, reference counting has enabled elimination of unnecessary memory allocation in PHP programs.

Security: - there are many flexible and efficient sets of security safeguards available for the developers and administrators. They are divided into two frames:-

- **System-level: -** when PHP is properly configured, it can provide an ample number of security mechanisms that administrators can manipulate. This provides the maximum amount of freedom and security. PHP has a safe mode feature of running, which can prevent exploitation of its implementation in many important ways. Server performance is also improved by providing a check on the maximum execution time and usage.
- **Application-level: -** PHP's predefined function set supports many data encryption options. PHP is also compatible with third party applications that provide security to e-commerce websites. Also PHP code is not visible through the browser because it is parsed before uploaded to the requesting user. This advantage of its server-side architecture prevents the loss of creative scripts to mundane users (knowledgeable enough to "view source").

Flexibility: - PHP is known for its flexibility in developing frameworks and websites. It supports different types of frameworks, such as Laravel and Symfony, e-commerce websites, secure transactions, applications which are diverse and complex, etc.

Familiarity: - programmers that work on different languages can easily get accustomed to PHP. Most of the language's construct is borrowed from C and Perl, and in many cases, PHP code is indistinguishable from that of Pascal. This minimizes the learning curve significantly.

FEATURES:-

PHP 7 is a revolution in the way of delivery of applications, from mobiles and websites to enterprises and the cloud. This is the most significant development in the language since the release of PHP 5 in 2004. PHP 7 brings major performance improvements, reduced memory consumption and a host of brand new features, making an application soar.

Improved performance: - PHP 7 is twice as fast as PHP 5 because the codes of PHP are also merged in PHP 7.

Reduced memory consumption: - the optimized features of PHP 7 make it consume fewer resources.
Scalar data type declarations: - variables and return types of methods can be enforced.
64-bit consistent support: - 64-bit architecture computer systems are now consistently and efficiently supported by PHP 7.
Exception hierarchy improved: - the hierarchy of ways in which the exception handling will be performed in the programs has been highly improved.
Fatal errors converted to exceptions: - the range of exceptions has drastically increased, covering many fatal errors as exceptions.
Secure random number generators: - Many APIs have been added, which have enabled secure generation of random numbers.

PHP 7 uses the latest Zend Engine 3.0, which improves an application's performance up to twice its original and allows 50% better memory consumption than PHP 5.6. It enables to serve more consistent users, without applying any additional hardware support. PHP 7 is tailored and refactored according to the needs of today's technology workloads and requirements.

PHP 7

SERVER-SIDE PROGRAMMING:-

PHP is not a static web development site like HTML; it creates dynamic pages, whose content change depending on various factors. This dynamism is enabled by an intelligent system, which is PHP server. It is responsible for taking user input, such as which page the user wants to access, initializing input parameters and giving an output that a browser can understand. All normal web browsers are not intelligent enough to understand and perform this processing and thus, we need servers and databases to run PHP codes. The property of PHP being a server-side development language provides a lot of security to the user data.

In order to get PHP to function on your system, you need three vital components to be installed on your computer system, is a web server, a database and a PHP parser.

Web server: - PHP will work virtually with all the server software. The most used web server is Apache, which is freely available. Through there are other servers, such as Microsoft's Internet Information Server (IIS), Apache is more popular and frequently used.

Database: - PHP works with all the database software, such as Oracle, PostgreSQL and Sybase, MySQL database. It is the most common and frequently used database software and has been integrated with PHP for a long time now.

PHP parser: - to compile and process PHP scripts, we will need a PHP parser. Parsers are software that converts an instruction into smaller executable units. This parser generates HTML outputs, which could be sent to Web Browser.

You can install all the components individually, but downloading one package called XAMPP. XAMPP is a free and open-source cross-platform web server developed by Apache and its friends like MariaDB database, and interpreters, to compile and process programming languages, such as PHP and Perl. Apache is the world's most used web server software and MariaDB is a community fork of MySQL relational database management system, which is freely available. Interpreters are parsers that can execute and compile PHP codes.

DOWNLOADING XAMPP:-

XAMPP is a simple, free and open-source cross-platform web server, which means that it works well on windows, Linux and Mac. It was developed by Apache Friends. We can create and manipulate database using XAMPP. Download and Install XAMPP by instructions. Then open control panel of XAMPP then start Apache and MySQL to stop these modules click on stop.

STARTING:-

XAMPP supports PHP and Perl. Perl is a text processing language, and stands for Practical Extraction and Report Language. It was initially designed for text manipulation, but it is a multi-purpose language.

Check version of PHP downloaded in XAMPP as:-

1. Type 'localhost' in the URL bar of your browser.

2. This will open dashboard screen if setup is right. For this must start Apache and MySQL in your XAMPP control panel.
3. Go to the PHPInfo option present at the top of the webpage. This will open new webpage. On this page, the version of downloaded PHP will be present at the top.

Run the first PHP program on server to test XAMPP functioning.

1. Go to C: → XAMPP → htdocs
2. Create folder in htdocs and name it.
3. Write PHP code on text editor and save with name and extension as .php

```
<!DOCTYPE html>
<html>
  <head>
    <title> hello </title>
    <style>

    </style>
  </head>
  <body>
    <?php
      echo "hello world";
    ?>
  </body>
</html>
```

4. Go to web browser and type http://localhost/folder_name/file_name.php in URL bar then output will appear.

PHP SYNTAX:-

Syntax of any language is very significant because it help understand the code and readable syntax also enables understanding program. PHP can be written in any Text editor. It just need to store with .php extension.

PHP in HTML code:-

```
<?php
    Code is here
?>
```

Comments: - are included in a code to increase its readability. These comments help to understand code.

Single-line comments:-

1. # comment here (comment with hash sign)
2. // comment here also (comment with double forward slash)

Multi-line comments:-

```
/*
Multiline comments
```

Here can do
*/

Insensitive of whitespace: - in PHP it doesn't matter how many space is between variables, assignment operators, and identifiers, constant or in any code.

Case sensitive: - PHP is case sensitive. The names of variables if written in different case will indicate to PHP engines that they are different variables.

Semicolon terminates each line: - semicolon follows each written instruction. An instruction is any valid sequence of PHP syntax.

Tokens: - are the smallest indivisible building blocks of program.
Example: - numbers, variables, strings, constants, etc.

Blocks {}: - A sequence of statements and instructions can be combined in one block corresponding to specific functions, conditional statements, etc.

VARIABLES: - are used to store data in the middle of program. Variables can store numbers, strings, Boolean values, etc.

- Variables are denoted by putting a dollar sign ($) before their name. Example: - '$name'.
- The value present in any variable is the value of its most recent assign value. The value of variable might change during the course of the program, but only last update is present in that variable.
- Assignment of any number, string or any type value is done by using the '=' operator. While doing assignment, the variable is written on the left side. While the value to which it is assigned is present on right side.
- There is no need to declare variables before assignment. But even if you do, no error will occur. The variables, which are declared before they are assigned, contain default values.
- A variable in PHP does not have intrinsic types. It does not know in advance, what type of value it will store.
- PHP automatically converts types without any explicit conversions.

DATA TYPES: - is a classification of data that tells the compiler or interpreter how the programmer should use the data. A different programming language supports different data types. The data type defines which operations can safely be performed to create, transform and use the variable in another computation.

Integers: - are whole numbers without any decimal point. Integers can be positive or negative. They are assigned to variables or are used directly in expressions.
Integer's representations:-

- Decimal (base 10)
- Octal (base 8)
- Hexadecimal (base 16)

The default format is decimal, while the octal format is specified by leading 0. Hexadecimals can be written by adding a leading 0x. The largest integer corresponds to (2^31)-1 or 2,147,483,647 while the smallest is the negative (2^31)-1 or -2,147,483,647.

Doubles/floats: - are the numbers containing the decimal points. When giving an output, floats print the minimum number of digits required after the decimal point. 5.29999 + 1.20001 = 6.5.

Booleans: - variable can hold any one of the two values i.e. 'TRUE' or 'FALSE'. In PHP, Booleans are present as constants predefined as 'TRUE' and 'FALSE'.

Truth values point:-

- The truth value of a number is always 'true', until it is not 'zero'. Truth value of zero is 'false'.
- The truth value of empty string is 'false', if it is not empty value is 'true'.
- The truth value of null is 'false'.
- The truth value of array and objects are 'false' if they are empty, otherwise 'true'.

NULL: - is one of the special types in PHP that has only one value. It is also very simple to use. NULL values can be assigned to any variable as $var = NULL;

Strings: - strings are a sequence of characters. They can be sentences or paragraphs. It includes all type of characters, i.e. numbers, letters and special characters.

Example:-

$a="A string in double quotes";
$b='A string in single quotes';
Sc=""; //empty string

Strings written in single quotes are treated literally, when it written in double quotes interpret special characters and assign values to variables.

Example:-

$num=67;
Echo 'number is $num.';
Output: - number is $num.
Echo "number is $num.";
Output: - number is 67.

Var_dump function: - this function dumps information about one or more variables. The information holds type and value of the variables.

Ex: -

$good=false;
echo var_dump($good);

Escape sequences:-

Escape sequence	replacement	Escape sequence	replacement
\n	Newline character	\r	Carriage-return
\t	Tab character	\$	Dollar sign

\”	Double quote	\\	Backslash
\v	Vertical tab	\e	Escape
\f	Form feed	\’	Single quote

Note: - \’ use when needs single quote in literal string **example: -** ‘my name is \’ram\’ ’.

Scope of variables: - is defined as visibility of the variable or the extent of the program code within the variable can be accessed or declared. It is easy to manage code with variable scope, particularly local scope.

Types of scope:-

- Local variables
- Global variables
- Function parameters
- Static variables

Local variables are those which are declared inside a function. This means that they are accessible only by that particular function.

Global variables can be accessed from anywhere in the program. Global variables must be declares outside of function.

Function parameters are declared inside the function parentheses, just after the function name. They are very much like typical variables.

Static variables hold the values in them even after the function they were declared in is destroyed. These variables are used when the function is called upon again.

Naming a variable: -

There are some rules to naming the variable.

- The name must begin with a letter or an underscore.
- Variable name can contain letters, numbers or underscore, but not special characters as +,-, (,),*, $, %, &, etc.

CONSTANTS: -

A constant is an identifier for a simple value, which remain constant for entire execution of code. It is case sensitive. Once it defined then it never changed or undefined throughout the code. Constants are defined by using the define() function. To retrieve a value of a constant, use their name directly.

Example:-

```
define("num",45);
echo "number ".num;
echo"<br>";
echo num;
echo"<br>";
echo constant("num");
```

Differences between constants and variables:-

Constant	Variables
Constants do not require ‘$’ sign.	The ‘$’ sign is used to denote a variable.

Constants need 'define()' function to be defined.	Variables can be defined by simple assignment
Constants can be defined anywhere without keeping in mind the scoping rules.	Variables are bound by scope definitions in a code.
Constants cannot be redefined, once they are set.	Variables can be modified throughout the program.

Predefined Constants:-

__LINE__ :- corresponds to the file's current line number.
Example: - echo __LINE__;

__FILE__ :- contains the absolute path and filename of file.
Example: - echo __FILE __;

__FUNCTION__ :- returns the function name in the way in which it was declared.
Example: - echo __FUNCTION __; //should be write in function.

__CLASS __ :- returns the class name in the way in which it was declared.
Example: - echo __CLASS __; //should be write in class's function.

__METHOD __ :- returns the class name in the way in which it was declared.
Example: - echo __METHOD __; //should be write in class's method.

OPERATORS:- are symbols that signify any operation. These symbols are used to perform operations on different variables and values.

- Arithmetic operators
- Comparison operators
- Assignment operators
- Logical operators
- Bitwise operators
- Conditional operators
- Array operators

Arithmetic operators: - are used to perform operations on variables.

Operator	Name	Description	Example
+	Addition	Add two operators	$z=$x +$y
-	Subtraction	Subtract one operand from the other	$z=$x-$y
*	Multiplication	Multiply two operands.	$z=$x*$y
/	Division	Divide one operand with other	$z=$x/$y
%	Modulo	Return the remainder when two operands are divided	$z=$x%$y
++	Increment	Increase value of an operand by one	$x++
--	Decrement	Decrease the value of an operand by one	$y--
**	Exponentiation	Raise one operand to another for calculate the exponential	$z=$y**$x

Comparison operators: - used to compare two values and perform rest of code on it.

Operator	Name	Description	Example

==	Equal	Return true if values are equal	$z=$x==$y
===	Identical	Return true if values and data types are equal	$z=$x===$y
!= or <>	Not equal	Return true if values are not equal	$z=$x != $y
!==	Not identical	Return true if values or datatype not equal	$z=$x !== $y
<	Less than	Return true if value of left is smaller than right	$z=$x < $y
>	Greater than	Return true if value of left is greater than right	$z=$x > $y
<=	Less than or equal	Return true if value of left is less or equal to right	$z=$x<=$y
>=	Greater than or equal	Return true if value of left is greater or equal to right	$z=$x<=$y
<=>	Spaceship	Return -1 if left is less than right, 0 if both equal and 1 if left value is greater.	$z=$x<=>$y

Assignment operators: - assign values to variables. They are used to fill a variable with any new value. The values that can be assigned are integers, strings, another variable, objects, arrays, etc. the most universal assignment operator is the equals sign "=".

Operator	Name	Description	Example
=	Simple assignment	Assigns right value to left variable	$x=$y
+=	Add and assignment	Add left and right value and assign to left variable	$x+=$y
-=	Subtract and assignment	Subtract left and right value and assign to left variable	$x-=$y
*=	Multiply and assignment	Multiply left and right value and assign to left variable	$x*=$y
/=	Divide and assignment	Divide left and right value and assign to left variable	$x/=$y
%=	Modulus and assignment	Modulus left and right value and assign to left variable	$x%=$y

Logical operators: - are the ones that help in forming decisions after evaluating certain conditions. Each operand in a logical expression is considered as a condition, which has to be evaluated using logical operators.

Operator	Name	Description	Example
and &&	AND	If both operands are true then condition is true	$x and $y $x && $y
or ||	OR	If any one of operands are true then condition is true	$x or $y $x || $y
!	NOT	Reverse the logical state of any operand	!($x)
Xor	XOR	If one of operands is true and not both	$x xor $y

Bitwise operators:- evaluate and manipulate only specific bits of an integer value. They work on bits of an operand and thus, can perform micro functions to a variable.

Operator	Name	Description	Example
&	AND	Set bits in both operands are set.	$x & $y
|	OR	Set bits in either of the two operands are set.	$x | $y
^	XOR	Set bits in any one operand and not both, are set	$x ^ $y
~	NOT	Set bits in operand are not set and not set are get set	~$x

<<	SHIFT LEFT	Bits of left operand shift left by numbers of times in right operand	$x << $y
>>	SHIFT RIGHT	Bits of left operand shift right by numbers of times in right operand	$ >> $y

Conditional operators: - are the ones which check the truth value of a given expression and assign a desired value to a variable.
Operator: - ? : **Name: -** conditional operator
Description: - if the condition before '?' is true then value evaluate before ':' sign. if it is false then value evaluate after ':'. **Example: -** $z= ($x > $y)? $x: $y

Array operators: - in PHP operations on arrays can also be performed directly, without using any loops or conditional statements.

Operator	Name	Description	Example
+	UNION	Takes the union of the two arrays	$x + $y
==	EQUALITY	Return true if both array have same key-value pair	$x == $y
===	IDENTITY	Return true if both array have same key-value pair at same position with same type	$x === $y
!= <>	INEQUALITY	Return true if the arrays are not equal	$x !=$y
!==	NON-IDENTITY	Return true if the arrays are not identical	$x !== $y

Precedence of operators: - in an expression, there might be more than one operator acting on many operands. In that case, it is important to know which operator will be evaluated before, and which will be evaluated after. Because of this, the precedence of operators must be known, as this affects the evaluation of expressions. The operators having higher precedence are evaluated before the ones having lower precedence.

Classification	Operator	Associativity
Unary	!,++,--	Right to left
Multiplicative	*,/,%	Left to right
Additive	+,-	Left to right
Relational	<,<=,>,>=	Left to right
Equality	==,!=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	? :	Right to left
Assignment	=,+=,-=,*=,/=,%=	Right to left

ESSENTIALS

EXPRESSIONS:- simply, can be defined as any statement that holds some value in program code. Constants and variables are the most basic forms of an expression. When assign value to variables it make an expressions. **Example: -** $a=10; $b=$a+90;

Values in PHP are can be integers, floats, strings and Booleans. These all are scalar type values and cannot be broken down into smaller parts. The two non-scalar types of values are arrays and objects. A function can return each of these values, and a variable can hold any of the types of value. Assign value to multiple variables as $a=$b=$c=10.

- **Pre-increment and pre-decrement: -**
  When pre-increment or pre-decrement operator used, it first increment or decrement value of variable and then reads it.
- **Post-increment and post-decrement: -**
  When post-increment or post-decrement operator used, it first reads value of variable and then increment or decrement it.

STATEMENTS: - scripts of PHP are a collection of different kinds of statements written in a logical order. These can be assignment statements, function calls, loops, conditional or empty statements. The end of it marks by semicolon (;). Also we can encapsulate many statements together by grouping them using the curly braces ({}). A group of statements present in block (curly braces) is a statement itself. These statements, when executed in order, output the desired result in browser.

Types of statements:-

- Conditional statements
- Loops
- Jump statements

Conditional statements: - are the one which are used to take decision in program. Hence they are called decision-making statements. There are three types of conditional statements as:-

- If-else statement
- Else-if statement
- Switch statement

If-else statement: - are used when we want to execute a fraction of code if a condition is true and another faction if condition is false.

Syntax:-

If (condition)

    Execute when condition is true.

Else

    Execute when condition is false.

Example:-

```
$x=15;
if($x>10)
   echo "x is greater than 10<br>";
else
   echo "x is smaller than 10<br>";
```

Alternative syntax:-

```
If (condition):
Statements when condition is true.
Statements
Else:
Statements when condition is false.
Statements
Endif;
```

Else-if statement: - is use when multiple conditions need to check. Elseif block checks one condition only when previous block condition is false.

Syntax:-

```
If (condition_1)
        Execute when condition is true.
Elseif (condition_2)
        Execute when condition is true.
Else
        Execute when condition is false.
```

Example:-

```
$x=5;
if($x>20)
{
   echo "x is greater than 20<br>";
}
elseif($x>10)
{
   echo "x is greater than 10<br>";
}
else
   echo "x is smaller than 10<br>";
```

Nested if-else and else-if: - these statement can be nested includ many if-else and else-if statements.

Example:-

```
$x=3; $y=8; $z=5;
if($x>$y)
   if($x>$z)
      echo "x is greater<br>";
   else
      echo "z is greater<br>";
```

```
else
   if($y>$z)
      echo "y is greater<br>";
   else
      echo "z is greater<br>";
```

Switch statement: - was introduced to reduce the redundancies of if-else and else-if statements.
Syntax:-

```
Switch (expression)
{
        Case label_1: execute when expression = label_1;
        Break;
        Case label_2: execute when expression = label_2;
        Break;
        Case label_3: execute when expression = label_3;
        Break;
        Default: execute when not match any case;
        Break;
}
```

Example:-

```
$x=1;
switch($x)
{
   case 1:echo "rank 1";
      break;
   case 2:echo "rank 2";
      break;
   case 3:echo "rank 3";
      break;
   default: echo "no rank";
      break;
}
```

Loops: - are used to execute block of code for specified number times. This can be based on a condition.
Types of loops:-

- For statement
- While statement
- Do-while statement
- Foreach statement

For loop: - is most frequently used in program. It is used when we know exactly number of times the block of code should be executed.
Syntax:-

```
For (initializing; condition; increment/decrement)
{
        //Statements
```

}
Example:-

```
for($x=1;$x<11;$x++)
{
   echo "$x good day<br>";
}
```

While loop: - is entry controlled loop, it check condition before starting loop. Until the condition is true block of while loop is executes.

Syntax:-

```
While (condition)
{
        //Statements
}
```

Example:-

```
$x=1;
while($x<11)
{
   echo "good day<br>";
   $x++;
}
```

Do-while loop: - is an exit controlled loop. It checks condition at end of loop block. These loop block at least executed one time.

Syntax:-

```
Do {
        //statements
} while (condition);
```

Example:-

```
$x=1;
do{
   echo "good day<br>";
   $x++;
}while($x<11);
```

Foreach loop: - is used to loop trough an array. In every pass, the value of the current element of the array is assigned in declared variable in loop's parenthesis.

Syntax:-

```
Foreach (array as $value)
{
        Code for execute;
}
Foreach (array as $key=>$value)
{
        Code for execute;
}
```

Example1:-

```
$arr=array(4,56,43,23,57);
foreach($arr as $value)
{
   echo "$value <br>";
}
unset($value);  //for undefined value
```

Example2:-

```
$arr=array(4,56,43,23,57);
foreach($arr as $key => $value)
{
   echo "$key => $value <br>";
}
unset($value);  //for undefined value
```

Jump statements: - control structure of program also determined by using these statements.
Types of jump statement:-

- Break
- Continue
- Return
- Goto

The break statement: - is used for end the executions of any given loop or switch structure. When break encountered the control of program shifts after loop or switch statement, that's why it called break statement.
Example:-

```
for($i=0;$i<10;$i++)
{
   if($i==5)
      break;
   echo "$i for loop<br>";
}
```

The continue statement: - used to skip current iteration of loop, whenever loop encountered continue statement it skip remaining code in current iteration.
Example:-

```
for($i=0;$i<10;$i++)
{
   if($i==5)
      continue;
   echo "$i for loop<br>";
}
```

The return statement: - take back control from calling function, and return value from that function.

The goto statement: - helps in jumping from one position to another position. This done by label which point to position where program control has to jump.

Example:-

```
for($i=0;$i<10;$i++)
{
   if($i==5)
      goto flag;
   echo "$i for loop<br>";
}
flag:
echo "good day<br>";
```

ECHO VERSUS PRINT IN PHP: - in PHP 'echo' and 'print' are two language constructs. Both of them can be used to output a string containing argument in PHP.

Difference between 'echo' and 'print': -

Echo	Print
It does not return any value	It returns 1 on every execution
Syntax void echo (string $var1);	**Syntax** int print (string $var);
It is faster than print	It is slower than echo
It can take multiple parameters	It can take only one parameter
It is just language construct	It is also language construct but behave like function by returning 1

ARRAYS: - in PHP are like maps that associate a value to key. They have been optimized in very version because of their multiple uses. PHP also have multidimensional arrays to be created and manipulated. Arrays can be defined as data structures that can store similar or different types of multiple values in a single variable. Multiple variables can also store in array.

Example:-

```
$city="nashik";
$marks=98;
$student = array("ram",14,$city,$marks);
foreach($student as $value)
   echo "$value<br>";
```

Types of array:-

- Numeric Array
- Associative Array
- Multidimensional Array

Numeric arrays: - can store different types of values their indices are only numeric. Indices are the keys that are used to access each value in an array. the default starting value of these indices are zero.

Example1:-

```
$marks = array(89,87,82,78,92);
foreach($marks as $key=>$value)
```

```
    echo "$key $value<br>";
```

example2:-

```
$num[0]=45;
$num[1]=56;
$num[2]=67;
$num[3]=89;
$num[4]=48;
foreach($num as $key=>$value)
  echo "$key $value<br>";
```

Associative arrays: - have similar functionality as numeric arrays. The index of these arrays is string, which forms a string association between the array value and its key.

Example1:-

```
$marks = array("aman"=>98,"sagar"=>56,"shyam"=>67,"ram"=>88);
foreach($marks as $key=>$value)
  echo "$key $value<br>";
```

Example2:-

```
$marks["aman"]=98;
$marks["sagar"]=56;
$marks["shyam"]=67;
$marks["ram"]=88;
foreach($marks as $key=>$value)
  echo "$key $value<br>";
echo "marks of aman is ".$marks["aman”]."<br>";
echo "marks of ram is $marks[ram]<br>";
```

Multidimensional arrays: -

Arrays contain other arrays as values are called multidimensional arrays. Elements in these arrays can be another array. Values in multidimensional arrays are accessed using multiple indices.

Example:-

```
$marks=array("aman"=>array("s1"=>45,"s2"=>58,"s3"=>78),
        "shyam"=>array("s1"=>56,"s2"=>87,"s3"=>67),
        "ram"=>array("s1"=>68,"s2"=>97,"s3"=>89));
echo "marks of aman in s3 is ".$marks['aman']['s3']."<br>";
echo "marks of ram in s2 is ".$marks['ram']['s2']."<br>";
foreach($marks as $key=>$value)
{
  echo $key;
  foreach($marks[$key] as $k=>$val)
    echo " $k=>$val ";
  echo "<br>";
}
```

FUNCTIONS: - is collection of statements or instructions, which performs specific task in a program structure. A function takes one or more parameters and returns value after processing the input. It can be called repeatedly, as many times as it is required, by using name.

Types of function:-

- Built-in functions
- User-defined functions

Built-in functions: - there are many functions and language constructs that are standard in PHP. These functions have been defined in the PHP library at the time of construction. These function names are reserved in PHP. These functions can be used directly in the code. Ex: - phpinfo ();

User defined functions: - are those, which are defined by a programmer and are not present in the language library beforehand. They must be defined in the program. Naming procedure of these functions is same as that in the case of naming variables. The only difference that '$' sign is not used when naming functions. Also function has parentheses '()', which might contain the values that are passed to it for processing. Define function as:-

```
<?php
        Function <function_name>(<arg1>,<arg2>,<arg3>,.....)
        {       #function code here
                Return <value>;
        }
?>
```

The 'function' keyword is used to define a function and the passed arguments are present int the parentheses '()'. A function can have any kind of valid PHP code, even new functions and classes can be defined in it. All functions in PHP carry a global scope. These means they can be called outside the function, even when their definition exists inside the function.

Example:-creating and calling function

```
function good()
{
   echo "good day to all";
}
good();
```

Example:-passing parameters with default value.

```
function add($a=0,$b=0)
{
   $c=$a+$b;
   echo "addition is $c";
}
add(4,5);
```

Example:-passing parameters by value

```
function add($a,$b)
{
```

```
    $c=$a+$b;
    echo "addition is $c";
}
add(4,5);
```

Example:-passing parameters by reference use '&' ampersand operator before parameter in function.

```
function swap(&$a,&$b)
{
    $c=$a;
    $a=$b;
    $b=$c;
}
$x=4;
$y=5;
echo "before swapping X=$x Y=$y<br>";
swap($x,$y);
echo "after swapping X=$x Y=$y<br>";
```

Return from the functions: - in the end, function in PHP can return any object back to the calling function, using 'return' keyword. It is possible to return more than one value from function using 'return array(val1, val2, val3)'.

Example:-

```
function mul($a,$b)
{
    $c=$a+$b;
    return $c;
}
$x=5;$y=6;
$z=mul($x,$y);
echo "addition of $x and $y is $z";
```

Calling function dynamically: -

Assign functions to variables and those variables can be treated as function.

Example:-

```
function add($a, $b)
{
    $c=$a+$b;
    return $c;
}
$z="add";   //assign function to variable
$a=$z(5,6); //calling function from variable
echo $a;
```

WEB CONCEPTS OF PHP: - PHP can provide dynamic content to the web, according to the type of browser, user input and randomly generated numbers. It can generate Images randomly,

makes html forms more interactive to the users, etc, and all these can be achieved with few blocks of codes.

Displaying images randomly:-

The 'rand()' function in PHP is used to generate numbers randomly. Before PHP7 srand() function needs to seed rand() function which enabled it to generate non-repetitive numbers.

Example: - random function

```
$z=rand();
echo $z."<br>";

$a=rand(1,10);  //number between range
echo $a."<br>";
```

Example: - image display

```
$number = rand(1,5);
echo "randomly generated image : <br>";
switch($number)
{
   case 1: $image ="https://wallpaper.dog/large/17147785.jpg";
      break;
   case 2: $image ="https://w.wallhaven.cc/full/wq/wallhaven-wqom86.jpg";
      break;
   case 3: $image ="https://4kwallpapers.com/images/wallpapers/dragons-dogma-2--
13557.jpeg";
      break;
   case 4: $image ="https://wallpapers.com/images/featured/roblox-4k-
in10cj1sl84hm3v4.jpg";
      break;
   case 5: $image ="https://w0.peakpx.com/wallpaper/647/250/HD-wallpaper-forza-
motorsport-7-2017-ferrari-fxx-k-ferrari-458-italia-poster-new-game.jpg";
      break;
}
echo "<img src=$image width='800px' height='500px'>";
```

HTML FORMS IN PHP:-

HTML forms are created to enable users to fill their relevant details in any webpage. HTML forms and PHP complement each other, as whatever details are submitted in these forms, they are automatically received by PHP scripts.

Example:-

Html file with .html:-

```
<!DOCTYPE html>
<html>
  <head>
    <title> html form </title>
  </head>
  <body>
```

```
    <form action="test.php" method="post">
      <p>Name: <input type="text" name="name"></p>
      <p>Age: <input type="text" name="age"></p>
      <p>Write a tagline: <input type="text" name="tagline"></p>
      <p><input type="submit"></p>
    </form>
  </body>
</html>
```

Php file with .php:-

```
hi <?php echo htmlspecialchars($_POST['name']) ?>.<br>
you are <?php echo (int)$_POST['age']?> year old.<br>
you belive in : <?php echo htmlspecialchars($_POST['tagline']) ?>.
```

- **htmlspecialchars() :-** keeps any html special characters present in the input , properly encoded.
- **(int) :-** convert data into integer explicitly, because when it entered, it is string type.
- **$_POST:-** this variable is used to post the data to php script.
- **$_POST['key']:-** the key of variable post is same as name in input tag.

GET AND POST METHODS:-

When users enter any information in the browser, that information is sent to the server, using specific methods. The server can process add, delete or store that information in the database. Therefore, to send this information to server, two methods are used:-

- GET
- POST

Before sending the information to the server, the browser encodes it using URL encoding. This means that the form of the message is changed. In URL scheme, the name-value pairs are concatenated with equals sign and different pairs are joined by the ampersand sign. Any spaces are replaced by '+' sign and the non-alphanumeric characters are replaced by hexadecimal values.

GET method: - this method sends the encoded information appended with the requested page. The "?" sign is used to separate the page with the encoded information.
While working with GET method remembers this:-

- It produces a long string, which you can find in your server logs in the location: box of the browser.
- This method can send only up to 1024 characters to the server.
- GET method should not be used to send passwords or sensitive information.
- Binary data like images and words cannot be sent using this method.
- QUERY_STRING environment variable is used to access data sent by GET method.
- PHP also provides the $_GET array (associative) to access the information sent using GET method.

Example:-
Html file:-

```
<!DOCTYPE html>
```

```
<html>
   <head>
      <title> get method </title>
   </head>
   <body>
         <form action="13get.php" method="get">
            Name : <input type="text" name="name"><br>
            Marks out of 100 : <input type="text" name="marks"><br>
            <input value="Enter" type="submit">
         </form>
   </body>
</html>
```

Php files:-

```
<?php
if(isset($_GET['name']) || isset($_GET['marks']))
{
   echo "welcome ".$_GET['name']."<br>";
   echo "Your marks are ".(int)$_GET['marks']."<br>";

   if((int)$_GET['marks']>=90)
      echo "<h2>Excellent</h2>";
   elseif((int)$_GET['marks']>=80)
      echo "<h2>Awesome</h2>";
   elseif((int)$_GET['marks']>=70)
      echo "<h2>Good Job</h2>";
   elseif((int)$_GET['marks']>=60)
      echo "<h2>Good but you can do better</h2>";
   elseif((int)$_GET['marks']>=50)
      echo "<h2>Better luck next time</h2>";
   else
      echo "<h2>POOR</h2>";
   exit();
}
?>
```

POST method: - is used to transfer information to the server using HTTP headers. The encoding is the same as that in the case of GET method, and the encoded string is put in a header called QUERY_STRING. The features of POST method are:-

- The POST can have any sized string to be sent to the server, unlike GET method.
- The POST method can send both ASCII and binary data.
- Since the POST method data is sent through HTTP header, you can make it secure by protecting the header.
- The $_POST array is used to access information send by POST method.

Example:-
Html code:-

```
<!DOCTYPE html>
<html>
   <head>
      <title> post method </title>
   </head>
   <body>
      <center>
         <form action="14post.php" method="post">
            Enter name : <input type="text" name="name">
              
            Favourite Color (in caps): <input type="text" name="color">
            <br><br>
            <input value="Enter" type="submit">
         </form>
      </center>
   </body>
</html>
```

Php code:-

```
<?php
if(isset($_POST["name"]) || isset($_POST["color"])){
   /*
   if()

   */
   echo "welcome ".$_POST['name']."<br>";
   echo "Favourite color is ".$_POST['color']."<br>";

   if($_POST['color']=='RED')
      echo "<h2>Fiery person</h2>";
   elseif($_POST['color']=='BLUE')
      echo "<h2>Stable person</h2>";
   elseif($_POST['color']=='GREEN')
      echo "<h2>Energetic person</h2>";
   elseif($_POST['color']=='YELLOW')
      echo "<h2>Hopful person</h2>";
   elseif($_POST['color']=='BLACK')
      echo "<h2>Mysterious person</h2>";
   elseif($_POST['color']=='WHITE')
      echo "<h2>Perfect person</h2>";
   else
      echo "<h2>Unusal Person</h2>";
   exit();
}
?>
```

- **isset():-** method return true when passed value is not null or empty.

This method has many advantages, because most of work is done with this method. These web concepts make the programmer closer to web and help to understand the number of aspects of a webpage that are created.

OOPS CONCEPTS

BASIC CONCEPTS:-

Class: - this is user defined data type, can include member variables and functions in its definition. It treated as template for making instances of the same kind of objects.

Object: - is an individual instance of a class. Many objects of one class can be defined, and they might hold similar type of data but with different values, Objects are also referred to as instances.

Member variable: - are created inside the class and are invisible outside the class. They can be accessed only by the objects of the class outside. They are also available to other member functions of the class in which these member variables are defined.

Member functions: - these functions are created inside the class and are invisible outside the class. They can only be accessed by the objects of the class outside. They are available to other member functions present in the class in which they are defined.

Inheritance: - is a substantial feature of OOPs in which one class can access the member variables and member functions of other defined classes.

Parent class: - in inheritance, the class that one or more classes inherit, is called parent class. it is also known as base class or super class.

Child class: - is the one which inherits from one or more base classes.

Polymorphism: - it is typical object-oriented feature in which different purposes of programming are fulfilled using a single function. The name of the function will remain same, but it might take different sets of arguments each time it is called to perform different tasks.

Overloading: - another feature of OOPs, where one or more than one operator has different usage, depending on the kind of arguments that are present with it. Function overloading is a famous concept in which different functions are overloaded similarly.

Encapsulation: - refers to the binding of data variables and functions under a single name, which could be an object of class. It is a concept of encapsulating data and member functions together, forming an object.

Abstraction: - when the data is represented in a way that hides the implementation details only to present the part which is necessary to the user, it is called data abstraction.

Constructors: - they are a special type of function, which are automatically called when an object of a class is formed. They can be default or user-defined.

Destructors: - they are a special type of function, which are automatically called when an object is deleted or is out of scope.

DEFINING CLASSES:-

A class can be defined by using the keyword "class". This keyword should be followed by the class name. A class name conventionally begins with capital letter, but PHP allows it to start with any letter or underscores. It can be followed by another set of letters, numbers or underscores. The constants and variables present in class are called properties, while function called methods.

Example:-

```
class good
{
   public $good="good day";
   public function show()
   {
      echo $this->good;
   }
}
```

- The 'class' keyword is used to define a class. It is followed by the class name. After it there is opening curly bracket '{', which marks the beginning of the scope of the class.
- The keyword 'public' is an access modifier, which sets the visibility of member variables and functions. The members that are declared as public can be accessed everywhere.
- The next line shows the declaration of member function, which again has public access. Function keyword is used to define methods in classes, function name and parentheses follows function keyword. After it curly bracket '{'which marks the opening of method scope.
- The next line uses '$this' pseudo-variable, which is made available when an object calls a method. '$this' is treated as a reference to the object that is calls function. '$this' variable assigns the value of the 'quote' variable to the referenced object class.

OBJECTS:-

Member variables and functions of a class can only be accessed by implementing objects of class. Once class is created, it can be used to create many objects. These objects can be used in a way that delivers the best features of object oriented programming. Objects are declared using the 'new' keyword. This keyword directs PHP engine to create space for new object.

Example:-

```
$g= new good();
echo $g->show();
```

The object is created using the keyword 'new'. The class name is used with parentheses, when new object is being created, is the default constructor of the class that is called whenever an object is created. After the object created, with the help of it access the method of class. The '->' symbol is used to access data members of the class using objects.

CONSTRUCTORS:- are special type of functions, which are called whenever an object is created. Constructors can be default or user-defined. When constructors are user-defined, they can be used for initializing many data members. PHP has a special function called the '__construct()'. A constructor can have as many parameters.

Example:-

```
class biodata
{
   public $name;
   public $age;
   public function __construct($n,$a)
   {
      $this->name=$n;
      $this->age=$a;
   }
}
$bio1 =new biodata("ram","14");
echo $bio1->name;
echo "<br>";
echo $bio1->age;
```

Destructors: - when a constructor is created, they occupy some sort of memory in the server. Therefore, for free that memory when object reach out of the scope, destructor is used. Define destructor using '__destruct()' function.

INHERITANCE: - is an established programming principle that is used by PHP in its object model. This principle affects the way objects and classes relate to each other. Classes in PHP can optionally inherit from other classes called parent classes. When a class inherits from any other class, it inherits its properties and methods according to the parent class member's visibility via objects. Inheritance can be enabled using the 'extends' keyword.

```
Class child_class extends parent_class
{
        #class definition
}
```

The child class (subclass or derived class) as these properties:-

- It gets access to all the member variables of the parent class.
- It gets access to all the member functions of the parent class, which by default, work in the same way as they work in the parent class.

Example:-

```
class biodata
{
   public $name="some";
   public $age="45";
}
class resume extends biodata
{
   public $marks="god";
   public $qualification="BBA";
   public function show()
   {
      echo $this->name."<br>";
      echo $this->age."<br>";
      echo $this->marks."<br>";
      echo $this->qualification."<br>";
   }
}
$emp =new resume();
$emp->show();
```

The inherited class has access to the variables of parent class as well as to its own new methods.

Visibility of data members: - data members in PHP are defined by the type of visibility they have in a class and program. Public, private and protected are types of visibility in PHP. The members declared as public can be accessed everywhere in the program code using object. The member declared as private, can be accessed only by the member functions present inside the class. The members declared as protected are available to be accessed only by the class which defines them. The properties of a class (member variables) must be declared as public, private or protected. If they are declared using the keyword 'var', they will be defined as public.

Public members: - all members of class are public by default. They can access from:-

- Outside the class in which they are declared.
- Inside the class in which they are declared.
- Inside the class which extends the class in which they are declared.

Private members: - to limit the accessibility of a member of class in which it is declared, we use the private accessibility. The private members of a class cannot be accessed from outside the class or from the class which inherits it. A class member can be made private using the keyword 'private'. The methods of a class can be access private members of that class. Thus, when we call the member functions using objects of that class, we can indirectly call the private members.

Protected members: - a protected method and property can be accessed in the class in which it is declared, as well as in all those classes. The 'protected' keyword is used to make any property or method protected.

INTERFACES: - provide common named functions to the classes that implement them. Different implementers (classes that implement the interface) might implement these interfaces

according to their requirements. They specify the methods that a class should implement. Interfaces defined with keyword 'interface' to create them. Also methods in interfaces do not have their contents defined. All methods in interfaces are public. When class implements an interface, the 'implement' keyword is used. More than one interface can be implemented by a class, separated by a comma.

Example:-

```
interface student{
   function setname($n);
   function getname();
   function setmarks($m);
   function getmarks();
}
class student_new implements student{
   private $n;
   private $m;
   function setname($name){
      $this->n=$name;
   }
   function getname(){
      echo $this->n;
   }
   function setmarks($marks){
      $this->m=$marks;
   }
   function getmarks(){
      echo $this->m;
   }
}
$s1 = new student_new();
$s1->setname("good");
$s1->getname();
echo "<br>";
$s1->setmarks("56");
$s1->getmarks();
```

CONSTANT: - are immutable. Once they are defined, they cannot be changed. The default visibility of constant is public. Interfaces can also define constants. The value of class constants is defined only once while defining the class, ant not with every instance of the class.

Example:-

```
class circleA
{
const PI=3.14159;
public $radius;
public $circle;
function area($rad)
```

```
        {
            $this->radius=$rad;
            $circle=self::PI*($this->radius)*($this->radius);
            return $circle;
        }
    }
    $cA= new circleA();
    echo "value of PI = ".$cA::PI."<br>";
    echo "area of circle = ".$cA->area(4)."<br>";
```

The constant is declared by using const keyword in the class. Also to access the declared constant, use the self keyword with this (::) symbol. Self keyword is used to refer to current class, '$this' keyword, refers to the current object. The '$this' keyword used for non-static members while 'self' is used for static members.

ABSTRACT CLASSES: - there are classes that cannot be used in the object formation, but can be inherited. Abstract classes are defined using the 'abstract' keyword. Any class that contains an abstract method must also be abstract. Abstract methods only contain the method signature, and not its implementation. The methods defined as abstract in an abstract class must be defined in the class that inherits it with the same access visibility. The signature of an abstract method, when it is declared and implemented, must be same.

Example:-

```
abstract class data{
    const br="<br>";
    protected $name;
    protected $age;
    abstract protected function setdata($n,$a);

    public function show(){
        echo "Name:- ",$this->name,self::br;
        echo "Age:- ",$this->age,self::br;
    }
}

class student extends data
{
    function setdata($name,$age)
    {
        $this->name=$name;
        $this->age=$age;
    }
}

$s1= new student();
$s1->setdata("ram",23);
```

```
$s1->show();
```

Abstract class inherited by other classes. The abstract methods are also defined in the inherited class. Abstract class can have normal methods without the 'abstract' keyword as well.

STATIC KEYWORD: - the members declared as static are accessible from outside the class without the use of any instance. A static member function can be accessed with a class object, but not a static member variable.

STATIC METHODS: - can be called without using the class objects. The methods declared as static do not allow the use of '$this' keyword inside their function definition. Though it is not required, the object of the class in which a static method is declared, can call static function.

Example:-

```
class storage{
public static function show($var)
{
   echo $var."<br>";
}
}
storage::show(67);
$store= 'storage';
$store::show(87);
$q= new storage();
$q->show(45);
```

The static function is accessed in many ways. First access is with the use of (::) operator, second access is by using the class name holder, and the third access is by using class object.

STATIC PROPERTIES: - the static variables can also be used outside the class without using the class instance. Also, you cannot use static member variables with objects using '->' operator. The value of a variable should not be a reserved word or a keyword in PHP.

Example:-

```
class statevar{
   static $state =45;
   function showstate(){
      echo self::$state." is state";
   }
}
echo statevar::$state."<br>";
$sr='statevar';
echo $sr::$state."<br>";
$st= new statevar();
$st->showstate();
echo $st->state;    //cannot access as non-static property
```

FINAL KEYWORD: - was introduced in PHP 5. It prevents the child classes from overriding a method definition. This is done by prefixing the keyword 'final' to them at the time of definition. Classes can also be defined as final these types of classes cannot be extended any further.

Example:-

```
//      final class finale{   #show final class cannot extend
      class finale{
         final public function show($num)
         {
            echo "this is final show no $num<br>";
         }
      }
      class semifinal extends finale{
//          public function show($num)
//          {
//             echo "this is semfinal show no $num<br>";
//          }    #show cannot override
      }
      $fo = new finale();
      $fo->show(12);
```

OVERLOADING AND OVERRIDING:-

OVERLOADING: it's the process of creating property and method. This can be established in a class to perform different action types. The methods that perform overload are invoked when interacting with methods or properties that have not been declared or are not visible currently. All overloading methods are defined as public. Use __call() magic method for overloading

Example:-

```
      class A{
         public function __call($operation, $vars)
         {
            if($operation == "Sum")
            {
               echo "the calculated sum is : ".$this->_getsum($vars)."<br>";
            }
            elseif($operation == "Multiply")
            {
               echo "the calculated product is : ".$this->_getproduct($vars)."<br>";
            }
            else
            {
               echo "you called operation $operation which we cannot provide right now.<br>";
            }
         }
         private function _getsum($vars)
```

```
    {
        $added=0;
        foreach($vars as $var)
        {
            $added+=$var;
        }
        return $added;
    }
    private function _getproduct($vars)
    {
        $product=1;
        foreach($vars as $var)
        {
            $product*=$var;
        }
        return $product;
    }
}
$o= new A();
$o->Sum(10,50,30);
$o->Multiply(10.75,101);
```

OVERRIDING: It's simple, as it only changes the form of a function of parent class, when declared in the child class, with same name, arguments and data type.

Example:-

```
class A{
    public $var1;
    public $var2;
    function operate($a,$b){
        $this->var1=$a;
        $this->var2=$b;
        $c=$this->var1+$this->var2;
        return $c;
    }
}
$obj1= new A();
echo $obj1->operate(30,20)."<br>";
class B extends A{
    public $val1;
    public $val2;
    public $op;
    function operate($a,$b){
        $this->var1=$a;
        $this->var2=$b;
        $c=$this->var1-$this->var2;
        return $c;
```

```
    }
}
$obj2=new B();
echo $obj2->operate(30,20);
```

ENCAPSULATION: - is a term that present in the object-oriented infrastructure. It means wrapping up or binding related properties and methods in a single program module. Hiding the essential properties of these modules is termed as data abstraction. Encapsulation secures data and information in an object from another object. Creating different objects, storing and manipulating the data variables and functions through them without affecting the properties and methods of other objects, and defining accessibility (or visibility) of data members, are the main focuses of encapsulation.

Example:-

```
class details{
    private $name;
    private $age;
    private $email;
    function __construct($n,int $a,$e)
    {
        $this->name=$n;
        $this->age=$a;
        $this->email=$e;
        $this->show();
    }
    private function show()
    {
        echo "name :- $this->name <br>";
        echo "age :- $this->age <br>";
        echo "email :- $this->email <br>";
    }
}
$d = new details("sample",45,"sam124@gmail.com");
```

POLYMORPHISM: - describes a pattern in OOPs, in which classes have different functionalities, even while sharing a common interface. The code, which is working with the different classes, is not made aware of which class is using it, since all classes use it in a similar way. Polymorphism makes the applications modular and extensile. Instead of using messy and nested conditional statements, use polymorphism, through which objects automatically perform based on our needs.

Example:-

```
interface calculate
{
    function area();
}
class square implements calculate
{
```

```
    private $side;
    function __construct($s)
    {
        $this->side=$s;
    }
    function area()
    {
        $a=$this->side*$this->side;
        echo "area of square is $a <br>";
    }
}
class rectangle implements calculate
{
    private $lenght;
    private $width;
    function __construct($l,$w)
    {
        $this->lenght=$l;
        $this->width=$w;
    }
    function area()
    {
        $a=$this->lenght*$this->width;
        echo "area of rectangle is $a <br>";
    }
}
class circle implements calculate
{
    private $radius;
    function __construct($r)
    {
        $this->radius=$r;
    }
    function area()
    {
        $a=2*3.14*$this->radius*$this->radius;
        echo "area of circle is $a <br>";
    }
}
$a1=new square(5);
$a1->area();
$a2=new rectangle(5,3);
$a2->area();
$a3=new circle(5);
$a3->area();
```

CONNECTING WITH DATABASES

MySQL:-

Database: - is a structured collection of data. It can hold anything, ranging from the vast amount of information of the corporate network to a simple shopping list. To access, process and add information from this stored data, we require a database management system (DBMS).

MySQL: - is one of the most popular open source SQL DBMS, MySQL is developed, supported and distributed by the Oracle Corporation. The 'My' in MySQL is taken from the name of the daughter (My) of its founder Michael Widenius, while SQL stands for 'Sequential Query Language' and now also known as 'Structured Query Language'. SQL is one of the most frequently used languages to access databases. Some features of MySQL DBMS are:-

Relational: - Databases in MySQL are relational. Relational databases are stored in separate tables consisting of rows and columns. The logical model which inbuilt with objects in the relational databases containing views, rows and columns, provides flexibility in programming environments. Relating these tables to one another is also possible using different one-to-one and one-to-many relationships in the database. The database follows a set of rules, which enable quality design and avoid inconsistent, duplicate, out of date or missing data.

Open source: - MySQL is completely free and is available globally. Any interested user can download SQL from the internet and use it free of cost.

Fast, reliable, scalable and easy: - databases store a lot of information, and thus, it is possible for them to become slow with time. MySQL can comfortably perform on a laptop alongside your other applications. MySQL handles large databases efficiently, and is in use successfully in highly demanding environments of production since several years. Adjust MySQL settings to take advantage of your computer's memory, CPU power and I/O capacity.

Highly supportive: - many prominent languages on web support MySQL, PHP being one of them.

OPENING DATABASE CONNECTION:-

To integrate databases in web applications, PHP offers full support to MySQL database, which can be easily, connected using the XAMPP server. There is PDO (PHP Data Object) for handle database operations with queries. PHP provides 'mysqli_connect()' function to open the database's connection. This function takes four parameters, which are of a server, username, password and database.

parameter	Description
Server	It specifies the host name running the database server.
User	It specifies the username of the user accessing the database.
Password	It specifies the password of the user accessing the database.
Database	It specifies the database name that is to be integrated.

1. To go XAMPP control panel and click on the Start button present in front of MySQL. Also click on the Admin option present next to Start.
2. Then, the phpMyAdmin page will open. To create a new database, click on New, which is present below the phpMyAdmin symbol.
3. Then Databases page will open for create database.
4. Create a new database by adding a name and clicking on Create.
5. Database will show with other databases in list. Then give a name to new database table. Specify the number of columns and click on Go.
6. Then fill the details of all columns, as name, type, length, values and other properties in each column of the table. Also can set these columns as unique or primary, by setting the null index but there can only one primary key, also can add table comments by scrolling down the page.
7. After filling all the columns above, scroll down below and click on save. Also can change the table name from 'operation' tab. It is also important to leave the storage engine as InnoDB. Do not change it. Otherwise, table might convert into read only table and you will be unable to insert or modify any values.
8. After saving table, a new page opens, which takes, you to your newly created table.

PHP contains a 'mysqli' class that can be used in place of 'mysqli_connect()' function.

CONNECTING WITH THE DATABASE:-

Use the 'mysqli_connect()' function. This function takes four parameters and returns a link identifier in case of success and a FALSE in case of failure.

Example:-

```
$server="localhost";
$user="root";
$pass="";
$db="phpnotes";
$con=mysqli_connect($server,$user,$pass,$db);   //procedural
echo "connected";
$con = new mysqli($server,$user,$pass,$db);     //object-oriented
echo "connected";
```

The first parameter of the 'mysqli_connect()' function is the server in which the PHP is operating, is 'localhost'. The second argument is username; this is set to be 'root'. If any other username is set than 'root' than access will denied. The third parameter is the password. XAMPP disables all the security features of the server for run websites freely. Therefore the password set as empty string (“”). The fourth and the last parameter is that of the database name on which the connection is required. The die function executed if something goes wrong with the 'mysql_connect()' parameters. Otherwise the success is achieved, which can be display with echo.

CREATING AND DELETING DATABASE USING PHP:-

Selecting a database:-

Example:-

```
$server="localhost";
$user="root";
$pass="";
$db="";
$db=mysqli_connect($server,$user,$pass,$db) or die("cannot connect");   //procedural
echo "connected";
mysqli_select_db($db,"not3");
mysqli_close($db);

$db=new mysqli($server,$user,$pass,$db) or die("connot connect");   //object-oriented
echo "connected";
$db->select_db("1notes");
$db->close();
```

Creating tables: - it is same as creating database. First, a query is created, and then it is sent to the 'mysqli_query()' function.

```
$server="localhost";
$user="root";
$pass="";
$db="phpnotes";
$con=mysqli_connect($server,$user,$pass,$db) or die("cannot connected");    //procedural
echo "connected";
$sql="CREATE TABLE student9 (roll int(10) NOT NULL AUTO_INCREMENT PRIMARY KEY, name
varchar(20) NOT NULL, marks int (4))";
$res= mysqli_query($con,$sql);
if(!$res)
{
   echo mysqli_error($con);
   die("Error occured");
}
else
{
   echo "table created";
}
mysqli_close($con);

$con= new mysqli($server,$user,$pass,$db) or die("cannot connected");   //object-oriented
echo "connected";
$sql="CREATE TABLE student10 (roll int(10) NOT NULL AUTO_INCREMENT PRIMARY KEY, name
varchar(20) NOT NULL, marks int (4))";
$res= $con->query($sql);
if(!$res)
{
   echo $con->error;
   die("Error occured");
```

```
}
else
{
   echo "table created";
}
$con->close();
```

Creating new tables needs to specify what kind of values you will require in the table. This is done through an SQL query, which can be saved in variable ($sql). Then, this query is performed on the database by using the 'mysqli_query()' function. The 'query()' function is like the 'mysqli_query()' function. It takes one parameter, which is the string valued query, to be executed.

Deleting tables: - here must select the database from which the deletion of table has to be carried out.

Example:-

```
$server="localhost";
$user="root";
$pass="";
$db="phpnotes";
$con=mysqli_connect($server,$user,$pass);   //procedural
if(!$con)
{
   die("cannot connect");
}
mysqli_select_db($con,$db);
$sql="DROP TABLE student2";
$delete=mysqli_query($con,$sql);
if(!$delete)
{
   die("cannot delete");
}
else
{
   echo "deleted";
   mysqli_close($con);
}
$con = new mysqli($server,$user,$pass);     //object-oriented
if(!$con)
{
   die("cannot connect");
}
$con->select_db($db);
$sql= "DROP TABLE student3";
$delete=$con->query($sql);
if(!$delete)
{
```

```
    die("cannot delete");
  }
  else
  {
    echo "deleted";
    $con->close();
  }
```

INSERTING VALUES IN THE DATABASE:-

There are two ways to insert values in the MySQL database. One is to insert it directly from the phpMyInfo page, using the 'insert' option present on the page. Another way is by writing PHP code. Since, from MySQL page, we can do practically everything on the database quite easily. Inserting values in MySQL through PHP is like the insertion queries present in SQL.

Example:-

```
  $server="localhost";
  $user="root";
  $pass="";
  $db="phpnotes";
  $con=mysqli_connect($server,$user,$pass,$db) or die("connot connect");
  echo "connected";
  $ins = "INSERT INTO student(rollno,name,marks) VALUES (4,'sam',123)";
  $query=mysqli_query($con,$ins);
  if(!$query)
    echo "problem";
  else
    echo "data inserted";
```

FETCHING VALUES FROM DATABASE:-

The whole point of storing values in databases is to retrieve them conveniently, whenever required. PHP provides easy fetching options to access values from the MySQL database. Access values using 'mysqli_fetch_array()' function. This function gives back a row that might be an associative array, a numeric array, or both. It might also return FALSE in the absence of any rows.

Example:-

```
  $server="localhost";
  $user="root";
  $pass="";
  $db="phpnotes";
  $con= mysqli_connect($server,$user,$pass) or die("cannot connect");
  $sql="SELECT rollno, name, marks FROM student";
//    $sql="SELECT * FROM student";
  mysqli_select_db($con,$db);
  $show=mysqli_query($con,$sql);
```

```
if(!$show)
{
   die("not found");
}
while($row= mysqli_fetch_array($show,MYSQLI_ASSOC))
{
   echo "Roll number : {$row['rollno']} <br>".
   ">>Name : {$row['name']} <br>".
   ">>Marks : {$row['marks']} <br>";
}
mysqli_close($con);
```

Here are using an associative array to output the table data. Here the MYSQLI_ASSOC constant is used from 'mysqli_fetch_array()'. An associative array helps in accessing the values using their name directly instead of the indices.

Example: - mysqli_fetch_assoc() implementation

```
$server="localhost";
$user="root";
$pass="";
$db="phpnotes";
$con= mysqli_connect($server,$user,$pass) or die("cannot connect");
$sql="SELECT rollno, name, marks FROM student";
mysqli_select_db($con,$db);
$show=mysqli_query($con,$sql);
if(!$show)
{
   die("not found");
}
while($row= mysqli_fetch_assoc($show))
{
   echo "Roll number : {$row['rollno']} <br>".
   ">>Name : {$row['name']} <br>".
   ">>Marks : {$row['marks']} <br>";
}
mysqli_close($con);
```

When numeric array is need to fetch the data then pass MYSQLI_NUM constant as second argument to 'mysqli_fetch_array()'. In this case, the array will be returned with numeric index.

Example:-

```
$server="localhost";
$user="root";
$pass="";
$db="phpnotes";
$con= mysqli_connect($server,$user,$pass) or die("cannot connect");
$sql="SELECT rollno, name, marks FROM student";
```

```
mysqli_select_db($con,$db);
$show=mysqli_query($con,$sql);
if(!$show)
{
   die("not found");
}

while($row= mysqli_fetch_array($show,MYSQLI_NUM))
{
   echo "Roll number : {$row[0]} <br>".
   ">>Name : {$row[1]} <br>".
   ">>Marks : {$row[2]} <br>";
}
mysqli_free_result($show);
mysqli_close($con);
```

The accessing indices of the array have been changed to numbers. The 'mysqli_free_result()' is used to free the memory saved in the cursor.

UPDATING THE DATABASE:-

To update values in a table, use the PHP function 'mysqli_query()' and insert the query containing the update tag. When updating records to a table, it is important to locate that record using a conditional clause (WHERE).

Example:-
Html form

```
<!DOCTYPE html>
<html>
   <head>
      <title> update form </title>
   </head>
   <body>
      <?php
      $server="localhost";
      $user="root";
      $pass="";
      $db="phpnotes";
      $con=mysqli_connect($server,$user,$pass) or die("cannot connect");
      mysqli_select_db($con,$db);
      $sqli ="SELECT * FROM student";
      $show = mysqli_query($con,$sqli);
      ?>
      <table>
         <tr>
            <th>Roll No.</th>
            <th>Name</th>
```

```
            <th>Marks</th>
          </tr>
          <?php
          $row=mysqli_fetch_array($show);
          echo "<form action=44update.php method=post name=log>";
            echo "<tr>";
            echo "<td><input type=text name=roll value'".$row[0]."'></td>";
            echo "<td><input type=text name=name value'".$row[1]."'></td>";
            echo "<td><input type=text name=marks value'".$row[2]."'></td>";
            echo "<td><input type=submit value=done></td>";
            echo "</tr>";
          echo "</form>";
          ?>
        </table>
    </body>
</html>
```

$_POST variable is access data entered by user in html form. Then data passed to update.php file, which updates it in the table present in database.

Php update file:-

```
    $server="localhost";
    $user="root";
    $pass="";
    $db="phpnotes";
    $con=mysqli_connect($server,$user,$pass,$db) or die("cannot connect");
    mysqli_select_db($con,$db);
    $sql ="UPDATE student SET name = '".$_POST['name']."', marks = '".$_POST['marks']."' WHERE
rollno = '".$_POST['roll']."' ";
    if(mysqli_query($con,$sql))
      header("refresh:2; url=44updatehtmlform.php");
    echo "updating....";
```

Here roll no is primary key. Because updating table requires the presence of a primary key. Only then php engine understand which query to update.

DELETING RECORDS:-

Delete record of table in the same way update them. The DELETE query and the HTML forms are used to delete the records from the table.

Example:-

Html form

```
<!DOCTYPE html>
<html>
  <head>
    <title> update form </title>
```

```
  </head>
  <body>
    <?php
    $server="localhost";
    $user="root";
    $pass="";
    $db="phpnotes";
    $con=mysqli_connect($server,$user,$pass) or die("cannot connect");
    mysqli_select_db($con,$db);
    $sqli ="SELECT * FROM student";
    $show = mysqli_query($con,$sqli);
    ?>
    <table>
      <?php
      $row=mysqli_fetch_array($show);
      echo "<form action=45delete.php method=post name=log>";
        echo "<tr>";
        echo "<th>Roll No.</th>";
        echo "<td><input type=text name=roll value'".$row['rollno']."'></td>";
        echo "<td><input type=submit value=delete></td>";
        echo "</tr>";
      echo "</form>";
      ?>
    </table>
  </body>
</html>
```

Example:-

```
<!DOCTYPE html>
<html>
  <head>
    <title> connect database </title>
  </head>
  <body>
    <?php
    $server="localhost";
    $user="root";
    $pass="";
    $db="phpnotes";
    $con=mysqli_connect($server,$user,$pass,$db) or die("cannot connect");
    mysqli_select_db($con,$db);
    $sql ="DELETE FROM student WHERE rollno = '".$_POST['roll']."' ";
    if(mysqli_query($con,$sql))
      header("refresh:2; url=45deletedata.php");
    echo "DELETING....";
    ?>
  </body>
```

```
</html>
```

HANDLING DATES AND TIMES

ACCESSING THE TIME STAMP WITH TIME():-

PHP has a built-in function 'time()', which gives all information required about the current date and time of server on which script is being operated. This function does not require any arguments, and it returns the timestamp as an integer. This returned integer denotes the number of seconds that have passed since midnight GMT (Greenwich Mean Time) of January 1, 1970 (UNIX epoch). The number of seconds, which have been elapsed since this time, is known as time stamp. To display current time stamp execute function 'time()'.

Example:-

```
<!DOCTYPE html>
<html>
  <head>
    <title> time stamp </title>
  </head>
  <body>
    <?php
    echo time();
    ?>
  </body>
</html>
```

CONVERSION OF TIMESTAMP:-

The inbuilt function 'time()' is used to calculate the current timestamp. But the value that gets as result; is very difficult to work upon. PHP has other functions that extract useful information from this timestamp, which can be used in the code efficiently. The two functions which are frequently used to perform this extraction are:-

- getdate()
- date()

The getdate() function:- accepts a timestamp (optionally) and returns the information about the date in the form of an associative array. When no time stamp passed it extract information about current timestamp. Array return by getdate() function contains the key elements as shown:-

Key value	Description	Values returned
Second	Represents the seconds numerically.	0 to 59
Minute	Represents the minutes numerically.	0 to 59
Hours	Represents the hours numerically.	0 to 23
Mday	Represents the day of the month numerically.	1 to 31
Wday	Represents the day of the week numerically.	0 (Sunday) to 6 (Saturday)
Mon	Represents the month numerically.	1 to 12
Year	Represents the year numerically in four digits.	e.g. 1999 or 2000
Yday	Represents the day of the year numerically.	0 to 365
Weekday	Represents the day of the week textually (in full).	Sunday to Saturday
Month	Represents the month textually (in full).	January to December
0	Represents the seconds since UNIX Epoch as time().	-2147483648 to 2147483647

Example:-

```
$day=getdate();
foreach($day as $key=>$value)
{
   echo "$key==>$value<br>";
}
```

The date() function:- is used to return a formatted string of date. PHP allows control over the date and time format, while using the 'date' function. This is done by passing a string value to the 'date' function. This function also takes time stamp as optional parameter. If it is not given then it take current time stamp.

Format	Description	Example
a	Return's 'am' or 'pm' (lowercase)	am
A	Return's 'AM' or 'PM' (uppercase)	AM
d	Returns day of the month as a number with leading zeroes	18
D	Returns day of week in three letters	Tue
F	Return the name of the month	July
h	Returns the hour in 12-hour format with leading zeroes	01
H	Returns the hour in 24-hour format with leading zeroes	13
g	Returns the hour in 12-hour format without leading zeroes	1
G	Returns the hour in 24-hour format without leading zeroes	13
i	Returns minutes	20
j	Returns the day of month without leading zeroes	3
l	Return the day of the week	Tuesday
L	Returns 1 if leap year, and 0 if not a leap year	0

m	Returns month of the year with leading zeroes	07
M	Returns the month of the year in three letters	Jul
r	Returns the RFC 2822 formatted date	Tue, 18 Jul 2017 13:37:26 +0200
n	Returns the month of the year without leading zeroes	7
s	Returns seconds of the hour	36
U	Returns the time stamp	1500377846
y	Returns the year in two digits	17
Y	Returns the year in four digits	2017
z	Returns the day of the year in number	198
Z	Returns the time zone offset in seconds from GMT	7200

Example:-

```
$time =date("h:i:s A");
echo $time;
```

PHP DATE AND TIME FUNCTIONS:-

There are more date time functions extract date and time. 'Date' and 'time' functions get timestamp from the server. They can be used to format date and time in a multitude of ways. These functions are affected by the configuration settings in the php.ini file. This file read only once when the server starts.

Example:-

```
echo date_default_timezone_get()."<br>";   //return default timezone
date_default_timezone_set("Asia/Kolkata");  //set default timezone
foreach(getdate() as $key=>$value)
{
   echo "$key ==> $value<br>";
}
```

Date and time configuration parameters in php.ini

Name	Default	Description	Changeable
date.default_latitude	"31.7667"	States the default latitude	PHP_INI_ALL
date.default_longitude	"35.2333"	States the default longitude	PHP_INI_ALL
date.sunrise_zenith	"90.583333"	States the default sunrise zenith	PHP_INI_ALL
date.sunset_zenith	"90.583333"	States the default sunset zenith	PHP_INI_ALL
date.timezone	""	States the default time zone	PHP_INI_ALL

Mode: - PHP_INI_ALL == Entry can be set anywhere

HANDLING ERRORS

HANDLING ERRORS WITH DIE FUNCTION:-

The 'die' function is same as the 'exit' function. When there is possibility of an error in program then use of die function is required. With it user can display custom message when this function executed. When this function executed then it will display message as pass to argument of this function, then terminate remaining execution of script.

Example:-

```
$a=50;
$b=4;
if($b==0)
{
   die("divide by zero");
}
$c=$a/$b;
echo $c;
```

CUSTOMIZED HANDLING OF ERRORS:-

There are customized functions through which we can create our own measures to react to errors. These functions can accept five parameters. These parameters define error levels, error message to be displayed, file in which error occurred, line in which the error occurred.

The syntax: - error_function(level_of_error, message_of_error, file_of_error, line_of_error);

Parameter	**Description**
Level_of_error	A compulsory parameter which specifies the level of the error encountered
Message_of_error	A compulsory parameter which specifies the error message for any user defined error

File_of_error	An optional parameter which specifies the file name where the error occurred
Line_of_error	An optional parameter which specifies the line number where the error occurred

Example:-

```
function error($level,$message,$file,$line)
{
   echo "error level == $level<br>";
   echo "error message == $message<br>";
   echo "error file == $file<br>";
   echo "error line == $line<br>";
   die();
}
set_error_handler("error");
$arr=array(34,5,5,55,4);
echo $arr;
echo "<br>hello world<br>";
```

HANDLING EXCEPTION IN PHP:-

It makes error handling easy and provides a better control on error generated in PHP code. Exception handling changes the normal flow of the program execution when a specified error (exception) is encountered in code.
The method of exception handling is based on three keywords as:-

- Try
- Throw
- Catch

Try: - any code that has any possibility of an error should be present inside the 'try' block. If the exception (specified error) is not encountered in the 'try' block, the code will run as normal, but whenever an exception is triggered, it must be thrown to the 'catch' block.

Throw: - this keyword is a way of triggering exceptions. Teach throw must have at least one 'catch'.

Catch: - this block performs a specified action on exception that is thrown to it. It retrieves the exception, and then creates an object, which contains the information about the exception being created.

Example1:-

```
try{
   $a=8;
   $b=0;
   if($b==0)
   {
      throw new exception("divide by zero");
   }
```

```
   echo $a/$b;
}
catch(exception $err)
{
   echo $err->getMessage()."<br>";
}
```

Example2:-

```
try{
   $c=mysqli_connect("localhost","root","pass");
}
catch(exception $err)
{
   echo "meassage == ".$err->getMessage()."<br>";
   echo "code == ".$err->getCode()."<br>";
   echo "file == ".$err->getFile()."<br>";
   echo "line == ".$err->getLine()."<br>";
   echo "trace == ".$err->getTraceAsString()."<br>";
   echo "string error == ".$err->__toString()."<br>";
}
```

DEBUGGING: -

Debugging is a practice of identifying errors, isolating the source of errors, and then either rectifying the error or finding a method to work around those errors.

Missing Semicolons: - all statements must end with a semicolon. This is because, until and unless a semicolon is not encountered, the reading of a statement does not stop. This means forgetting semicolon will consider by PHP engine as next line is part of first line that continue after it.

Insufficient Equal Signs: - Whenever comparing two values for equality, use two equal signs (==) instead of one, and for comparing two values for identity, use three equal signs (===). It is common mistakes to use one equal sign when checking equality, and two equal signs when checking identity.

Variable Names Misspelled: - Whenever a programmer misspells a variable, PHP considers it as a new variable.

Forgetting Dollar signs: - finding dollar sign ahead of variables is difficult when code is huge, it is important to write dollar with every variable. Variables in PHP start with dollar sign.

Troubling quotes while printing: - it is important to remember when to put single quotes or double quotes in the strings. Especially while using ‘print’ and ‘echo’, remember when variables will be printed and when not.

Missing parentheses: - always check for any missing parentheses in code. They should always be in pairs. Also, do not forget to check the scope of the code.

Array Indices: - always remember that arrays begin with zero (0) and not one (1).