**Introduction to PHP(CH-1)**

**Programming Language:-**

The language which is used to represent the instructions of a program to execute it is called a programming language.

Types:-

1. structural/procedural language:-it is based on structure or procedures.

ex-c

1. partial object oriented language:-which may or may not use the concepts of oops to write a program.

ex-c++, php

1. fully object oriented language:-without the help of oops we cannot write a program.

ex-java,.net

1. object based language:-based on objects.

ex-javascript, vbscript

**Historical Development of Programming Languages:-**

1. PASCAL - 1954
2. COBOL - 1957
3. ALGOL - 1960
4. CPL - 1963
5. BCPL - 1967
6. B - 1970
7. C - 1972
8. C++ - 1980
9. PYTHON – 1980-1990
10. JAVA - 1991
11. .NET - 1991
12. PHP – 1993
13. R – 1993
14. SCALA - 2003

|  |  |
| --- | --- |
| Static Website | Dynamic Website |
| Prebuilt content is same every time the page is loaded. | Content is generated quickly and changes regularly. |
| It uses the HTML code for developing a website. | It uses the server side languages such as PHP,SERVLET, JSP, and ASP.NET etc. for developing a website. |
| It sends exactly the same response for every request. | It may generate different HTML for each of the request. |
| The content is only changed when someone publishes and updates the file (sends it to the web server). | The page contains "server-side" code which allows the server to generate the unique content when the page is loaded. |
| Flexibility is the main advantage of static website. | Content Management System (CMS) is the main advantage of dynamic website. |

**PHP:-**

Stands for Hypertext Pre-processor. is a [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting" \o "Server-side scripting) language designed for [web development](https://en.wikipedia.org/wiki/Web_development" \o "Web development) but also used as a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language" \o "General-purpose programming language). It was originally created by [Rasmus Lerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf" \o "Rasmus Lerdorf) in 1994. PHP code may be embedded into [HTML](https://en.wikipedia.org/wiki/HTML" \o "HTML) code, or it can be used in combination with various [web template systems](https://en.wikipedia.org/wiki/Web_template_system" \o "Web template system), web content management systems, and [web frameworks](https://en.wikipedia.org/wiki/Web_framework" \o "Web framework). PHP code is usually processed by a PHP [interpreter](https://en.wikipedia.org/wiki/Interpreter_(computing)" \o "Interpreter (computing)) implemented as a [module](https://en.wikipedia.org/wiki/Plugin_(computing)" \o "Plugin (computing)) in the web server or as a [Common Gateway Interface](https://en.wikipedia.org/wiki/Common_Gateway_Interface" \o "Common Gateway Interface) (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface" \o "Command-line interface)(CLI) and can be used to implement [standalone](https://en.wikipedia.org/wiki/Computer_software" \o "Computer software) [graphical applications](https://en.wikipedia.org/wiki/Graphical_user_interface" \o "Graphical user interface). The standard PHP interpreter, powered by the [Zend Engine](https://en.wikipedia.org/wiki/Zend_Engine" \o "Zend Engine), is [free software](https://en.wikipedia.org/wiki/Free_software" \o "Free software) released under the [PHP License](https://en.wikipedia.org/wiki/PHP_License" \o "PHP License).

**Versions of PHP:-**

1. PHP1.0-1995
2. PHP2-1997
3. PHP3-1998
4. PHP4-1999
5. PHP5-2004
6. PHP6-2005
7. PHP7-2014-2015

**Features of PHP:-**

1. Simple
2. Interpreted
3. Faster
4. Open source
5. Platform independent
6. Case sensitive
7. Flexible
8. Efficient
9. Familiar
10. Portable

**Advantages of PHP:-**

1. Open source
2. Easy to use
3. Stable
4. Powerful library support
5. Built-in database connection modules
6. Platform independent

**Disadvantages:-**

1. Security
2. Weak type
3. Not suitable for large applications

**What is a PHP File?**

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code.
* PHP code are executed on the server, and the result is returned to the browser as plain HTML
* PHP files have extension ".php"

**What Can PHP Do?**

* PHP can generate dynamic page content
* PHP can create, open, read, write, delete, and close files on the server
* PHP can collect form data
* PHP can send and receive cookies
* PHP can add, delete, modify data in your database
* PHP can be used to control user-access
* PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

**Why PHP?**

* PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* PHP is compatible with almost all servers used today (Apache, IIS, etc.)
* PHP supports a wide range of databases.
* PHP is free. Download.
* PHP is easy to learn and runs efficiently on the server side.

**How to install and execute PHP:-**

For using php we must install “xampp” application s/w. After installing it, it must install inside “c:” drive as a default installed path. Inside that drive a folder called “xampp” will automatically created” and inside “xampp” folder “htdocs” is available. Inside that we have to create our own folder and save all php files inside it. Open xampp control panel and start apache server. After that open any browser and type “localhost/foldername”. Your file is visible on the browser, then we click on the file to run it.

**Simple Program:-**

A PHP script is executed on the server, and the plain HTML result is sent back to the browser. A PHP script can be placed anywhere in the document.A PHP script starts with **<?php** and ends with **?>**

<?php  
//code  
?>

**Write a program to print a message ?**

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

**Elements of PHP(CH-2)**

**Constants:-**

It is a fixed quantity whose value never changed

throughout the program.

**Types:-**

**1.Integer constant:-**

These are numeric values which may be positive or negative like:

89,-234 etc.

**2.Real/float constant:-**

These are decimal numbers which may be positive or negative like:

-2.3,89.86,9/12,-1/3 etc.

**3.Character/String constant:-**

The characters which are enclosed within a pair of double quote(“”) are called string constant like:

“anil”

**4.define():-**

To create a constant, we use define() function.

Syn:- define(name, value);

**Write a program for define() constant ?**

<?php  
define("GREETING", "Welcome to php programming");  
echo constant ”GREETING”;  
?>

**Write a program for define() constant ?**

<?php  
define("GREETING", "Welcome to W3Schools.com!", true);  
echo greeting;  
?>

**Variable:-**

It is a named memory location where the constant values are stored.

**Rules for creating a variable:-**

* A variable starts with the **$** sign, followed by the name of the variable.
* A variable name must start with a letter or the underscore character.
* A variable name cannot start with a number but numbers can be placed after first place.
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_).
* Variable names are case-sensitive ($age and $AGE are two different variables).
* A variable doesn’t allow blank space within it.

**Write a program for variable ?**

<?php

$a=7;

echo “value of a”.$a;

?>

**Php global variables:-**

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

* $GLOBALS
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

**Difference between echo and print:-**

Echo has no return value while print has a return value of 1 so it can be used in expressions. Echo can take multiple parameters while print can take one argument. Echo is marginally faster than print.

**Data-types:-**

Variables can store data of different types, and different data types can do different things.PHP supports the following data types:

**1.String:-** it is the combination of more than one character enclosed within double quote(“”).

**Write a program for string data-type?**

<?php   
$x = "Hello world";  
echo $x;  
?>

**2.Integer**:-

An integer data type is a non-decimal number and its range is in between

-2,147,483,648 to +2,147,483,647.

**Write a program for integer datatype ?**

<html>  
<body>  
<?php    
$x = 5985;  
echo $x;  
?>    
</body>  
</html>

**Note:-**Var\_dump() returns datatype.

**3.Float:**-

Floating point numbers are also called double which accept decimal/fractional nos.

**Write a program for float datatype ?**

<?php   
$x = 10.365;  
echo $x;  
?>

**4.Boolean:-**

It is used to store Boolean values like true or false by 1 or 0 as binary.

**5.Array:-**

Array is a derived data-type in php which is used to store values of similar type/homogeneous data inside a single variable.

**Write a program for array datatype ?**

<?php   
$a=Array(4,78,0,123);

var\_dump($a);

?>

**6.Object:-** an object is an instance/copy of a class which is used to access all class data which are dis-similar type.

**7.NULL:-**

**Write a program for null datatype ?**

<?php  
$x = "Hello world!";  
$x = null;  
var\_dump($x);  
?>

**8.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.

**Operators:-**

These are the special symbols which are used to operate or perform task over the operand or variable.

Types:-

1. **Assignment operator**:- used to assign a value to the variable. it is of two types.
2. **simple assignment(=)**

ex: $a=5;

1. **compound assignment(+=,-=,\*=,/=)**

ex: $sum+=$i;

1. **Special operator**:-
2. **comma operator:-**used to separate constants, variable and expressions.

Ex:-

constant separation $a = array(3,5,8,4);

variable separation var $a,$b;

1. **semicolon:-** used to terminate a line of code.

ex: $a=9;

1. **dot:-** used to separate string with variable or html tags.

Ex: echo “<h1>”.”addition value=”.$c.”</h1>”;

1. **Arithmetic operator**:- used to perform mathematical operation over operands using different types of symbols like +,-,\*,/,%.

**(4)Write a program to add two integer number ?**

<?php

$a=8;

$b=5;

$c=$a+$b;

echo $c;

?>

**(7)Write a program to find the area and circumference of a circle ?**

<?php

$r=6;

$a=3.141\*$r\*$r;

$c=2\*3.141\*$r;

echo “area=”.$a.”<br>”;

echo “perimeter=”.$c;

?>

**(8)Write a program to input temperature in fahrenheit scale and display it in celsius scale ?**

<?php

$f=98;

$c=($f-32)\*5/9;

echo “celcius value=”.$c;

?>

**(9)Write a program to swap two integer number using third variable ?**

<?php

$a=5;

$b=4;

$c=$a;

$a=$b;

$b=$c;

echo “after swapping value of a=”.$a.”<br>”;

echo “after swapping value of b=”.$b;

?>

**(10)Write a program to swap two integer number without using third variable ?**

<?php

$a=5;

$b=4;

$a=$a+$b;

$b=$a-$b;

$a=$a-$b;

Echo “after swapping value of a=”.$a.”<br>”;

Echo “after swapping value of b=”.$b;

?>

1. **Relational/Comparison operator**:- used to compare between two or more operands using different symbols like >,<,>=,<=,==,!=.
2. **Logical/Boolean operator**:- used to perform logical operation like producing true/false value using and, or, not, xor gate.

AND(&&) OR(||) NOT(!) XOR(^)

T T->T T T->T T->F T T->F

T F->F T F->T F->T T F->T

F T->F F T->T F T->T

F F->F F F->F F F->F

1. **Unary operator**:- the operator which is used to perform over a single operand is called unary operator.

ex:- $a++

1. **Binary operator**:- the operator which is use to operate over two operand is called binary operator.

ex:- $a+$b

1. **Ternary/conditional operator**:- the operator which is used to operate over more than two operands are called ternary operator.

syn:- $variable=(condition)? print statement1:print statement2;

**(13)Write a program to find the greater number among two inputted number using ternary operator ?**

<?php

$a=9;

$b=8;

$c=($a>$b)?”a is greater”:”b is greater”;

echo $c;

?>

**(14)Write a program to check for leap year using ternary operator ?**

<?php

$y=2018;

$result=((($y%100!=0)&&($y%4==0))||($y%400==0))?“leap year”:“not”;

echo $result;

?>

1. **Bitwise operator:-**the operator which is used to perform bit operation over the operand or variable as 0 and 1 is called bitwise operator.

typs:

1. biwise and(&)
2. bitwise or(|)
3. bitwise xor(^)
4. bitwise complement(~)
5. bitwise left shift(<<)
6. biwise right shift(>>)

**(16)Write a program for bitwise left-shift operator ?**

<?php

$a=7;

$b=2;

$c=$a<<$b;

echo “value of c=”.$c;

?>

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0

(11110)2=1x24+1x23+1x22+0x21+0x20=16+8+4+0+0=28

**(17)Write a program for bitwise complement operator ?**

<?php

$a=7;

$c=~$a;

echo “value of c=”.$c;

?>

**(18)Write a program for bitwise and operator ?**

<?php

$a=14;

$b=7;

$c=$a&$b;

echo “value of c=”.$c;

?>

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0

(110)2=1x22+1x21+0x20=4+2+0=6

1. **Increment/decrement operator:-**
2. **Increment operator:-** this operator is used to increase the value of the variable by one in two different ways.
3. **pre-increment:-** first increase the value after that update it.

ex- ++$a

1. **post-increment:-** first update the value after increase it.

ex- $a++

**(19)Write a program for increment operator ?**

<?php

$a=5;

$b=++$a;

$c=$a++;

echo $b.”<br>”.$b.”<br>”;

echo $b.”<br>”.$c;

?>

1. **Decrement operator:-** this operator is used to decrease the value of the variable by one in two different ways.
2. **pre-decrement:-** first decrease the value after that update it.

ex- --$a

1. **post-decrement:-** first update the value after decrease it.

ex- $a--

**(20)Write a program for decrement operator ?**

<?php

$a=8;

$b=$a--;

$c=$a--;

echo “value of b=”.$b.”<br>”.”value of c=”.$c;

?>

**Control Structure(CH-3)**

Control structure is the control flow of program execution where we can use different types of statements to process our programs. There are different types of statements available in control structure that are:

1. conditional statements
2. unconditional statements
3. looping statements
4. **Conditional statements:-** These are the types of statements where we use conditions to process the program using some relational operators to produce true or false value.

Types:-

1. **If/simple if:-**

Here we are able to specify only one condition and one print statement. the demerit of this statement is it cannot go to the default part.

syn:- if(condition)

print statement;

**(21)Write a program to check whether an inputted number is even or not using simple if ?**

<?php

$a=6;

if($a%2==0)

echo “a is even”;

?>

1. **If else:-**

We are able to specify only one condition but two print statement means one for true part and another for false part.

syn:- if(condition)

print statement1;

else

print statement2;

**(22)Write a program to input two integer number and find the greater one using if else ?**

<?php

$a=7;

$b=10;

if($a>$b)

echo “a is greater”;

else

echo “b is greater”;

?>

1. **Nested if:-**

Here we are able to specify n no of conditions and n number of print statements. but the problem is that it is a very complex type of statement where we get confuse about the starting and ending of if block because of nested structure.

syn:- if(condition1)

if(condition2)

print statement1;

else

print statement2;

.

.

else

if(conditionn)

print statement-1;

else

print statementn;

**(23)Write a program to input a year and check whether it is leap year or not using nested if ?**

<?php

$y=2018;

if($y%100!=0)

if($y%4==0)

echo “leap year”;

else

echo “not leap year”;

else

if($y%400==0)

echo “leap year”;

else

echo “not leap year”;

getch();

}

**(24)Write a program to input three integer number and find the greatest one using nested if ?**

<?php

$a=89;

$b=24;

$c=90;

if($a>$b)

if($a>$c)

echo “a is greatest”;

else

echo “c is greatest”;

else

if($b>$c)

echo “b is greatest”;

else

echo “c is greatest”;

?>

1. **Ladder else if:-**

It is the type of conditional statement where we can specify n no of conditions and n no of print statements. it is very simple as compare to nested if because here no nesting of condition is there and we can specify one condition and one print statement so on.

syn:- if(condition1)

print statement1;

else if(condition2)

print statement2;

.

.

else

print statement;

**(25)Write a program to input an integer number and check whether it is even or odd and positive or negative using ladder else if ?**

<?php

$n=-6;

if($n%2==0&&$n>0)

echo “number is even and positive”;

else if($n%2!=0&&$n<0)

echo “number is odd and negative”;

else if($n%2==0&&$n<0)

echo “number is even and negative”;

else if($n%2!=0&&$n>0)

echo “number is odd and positive”;

else

echo “special number”;

?>

**(26)write a program to input three integer number and find the greatest one using ladder else if ?**

<?php

$a=8;

$b=5;

$c=2;

if($a>$b&&$a>$c)

echo “a is greatest”;

else if($b>$a&&$b>$c)

echo “b is greatest”;

else

echo “c is greatest”;

?>

**(28)Write a program to find the grade of an emp by finding gross salary where ta=5%, da=7.5% and hra=10% of basic salary and basic salary is given?**

<?php

$bs=34000;

$ta=0.05\*$bs;

$da=0.075\*$bs;

$hra=0.1\*$bs;

$gs=$bs+$ta+$da+$hra;

if($gs>=100000)

echo “A grade employee”;

else if($gs>=75000&&$gs<100000)

echo “B grade employee”;

else if($gs>=50000&&$gs<75000)

echo “C grade employee”;

else if($gs>=20000&&$gs<50000)

echo “D grade employee”;

else

echo “E grade employee”;

?>

**(29)Write a program to find the daily wages of a worker according to the following conditions using ladder else if statement ?**

duty in hours amount in rupees

within first 8 hours 100 rupees

next 4 hours 20 rs/hr

next 4 hours 40 rs/hr

next 4 hours 60 rs/hr

next 4 hours 80 rs/hr

<?php

$hr=17;

If($hr>=1&&$hr<=8)

$amt=100;

else if($hr>=9&&$hr<=12)

$amt=100+($hr-8)\*20;

else if($hr>=13&&$hr<=16)

$amt=180+($hr-12)\*40;

else if($hr>=17&&$hr<=20)

$amt=340+($hr-16)\*60;

else if($hr>=21&&$hr<=24)

$amt=580+($hr-20)\*80;

Echo “amount incurred by the worker=”.$amt;

?>

1. **Unconditional statements:-** These are the types of statements where we don’t use any condition rather we use no. of cases to produce true or false statement. Cases consist of constants or expressions.

Types:-

1. **Switch-case statement:-**

It is the type of unconditional statement where we are able to specify n no of cases. Case blocks are created by the help of keyword case and exited by the help the keyword break. if no case is satisfied inside the switch block then it goes to the default block specified by default keyword.

syn:- switch(expression)

{

case <case-constant 1>:

statement(s);

break;

.

.

case <case-constant n>:

statement(s);

break;

default:

statement(s);

}

**(31)Write a program to display the day name using switch case ?**

<?php

$day=3;

switch($day)

{

case 1:

echo “the day is sunday”;

break;

case 2:

echo “the day is monday”;

break;

case 3:

echo “the day is tuesday”;

break;

case 4:

echo “the day is wednesday”;

break;

case 5:

echo “the day is thursday”;

break;

case 6:

echo “the day is friday”;

break;

case 7:

echo “the day is saturday”;

break;

default:

echo “wrong choice”;

}

?>

**(32)Write a program to input a character and check whether it is vowel or consonant using switch case ?**

<?php

$ch=’i’;

switch($ch)

{

case ‘a’:

case ‘e’:

case ‘i’:

case ‘o’:

case ‘u’:

case ‘A’:

case ‘E’:

case ‘I’:

case ‘O’:

case ‘U’:

echo “it is a vowel”;

break;

default:

echo “it is consonant”;

}

?>

**(33)Write a program to implement a calculator using switch case ?**

<?php

$a=10;

$b=2;

$ch=”+”;

switch($ch)

{

case ‘+’:

$c=a+b;

break;

case ‘-‘:

$c=a-b;

break;

case ‘\*’:

$c=a\*b;

break;

case ‘/’:

$c=a/b;

break;

case ‘%’:

$c=a%b;

break;

default:

echo “wrong choice”;

}

echo “calculated value=”,$c;

?>

1. **Looping statements:-** These are also called iterative or repetitive statements. When we want to execute or print one statement for more than one time with a single specification then we use the concept of looping. For creating a looping statement we need components.
2. **Initialization:-** from which the variable value starts execution or printing.
3. **Condition:-** upto which the variable value should goes.
4. **Incr/decr:-** used to increase or decrease the value of the variable after each execution.

Types:-

1. **While loop:-**

It is otherwise called as top-tested loop or pre-tested loop or entry control loop. in this type of looping statement first the condition is checked after the statements are get executed and printed. if the condition is false then no statement is executed or printed.

syn:- variable value initialization;

while(condition)

{

statement(s);

incr/decr;

}

print statement(s);

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**(37)Write a program to print 1 to 10 ?**

<?php

$i=1;

while($i<=10)

{

echo $i.”<br>”;

$i++;

}

?>

**(38)Write a program to find the output of 12+22+32+....n2 ?**

<?php

$i=1;

$n=10;

$sum=0;

while($i<=$n)

{

$sum=$sum+($i\*$i);

$i++;

}

?>

**(40)Write a program to input a number and check whether it is prime or not ?**

<?php

var num=7,$i=1,$counter=0;

while($i<=$num)

{

if($num%$i==0)

$counter++;

$i++;

}

if($counter==2)

echo “number is prime”);

else

echo “number is not prime”;

?>

**(41)Write a program to find the reverse of a number ?**

<?php

**123 -> 321**

$num=123;

$rev=0;

while($num!=0)

{

$rem=$num%10;

$rev=$rev\*10+$rem;

$num=$num/10;

}

Echo “reverse value=”.$rev;

?>

**(42)Write a program to input a number and check whether it is palindrome or not ?**

<?php

**121-> 121**

$num=121;

$rev=0;

$temp=$num;

while($num!=0)

{

$rem=$num%10;

$rev=$rev\*10+$rem;

$num=$num/10;

}

if($temp==$rev)

echo “number is palindrome”;

else

echo “number is not palindrome”;

?>

**(43)Write a program to convert a decimal number into its binary equivalent ?**

<?php

**5->101**

$num=5;

$sum=0;

$prd=1;

while($num!=0)

{

$rem=$num%2;

$sum=$sum+$rem\*$prd;

$prd=$prd\*10;

$num=$num/2;

}

Echo “binary equivalent=”.$sum;

?>

**(44)Write a program to input a number and check whether it is strong number or not ?**

<?php

$num=145;

**145=1!+4!+5!=1+24+120=145**

$sum=0;

while($num!=0)

{

$fact=1;

$rem=$num%10;

while($rem!=0)

{

$fact=$fact\*$rem;

$rem--;

}

$sum=$sum+$fact;

$num=$num/10;

}

if($sum==$temp)

echo “number is strong”;

else

echo “number is not strong”;

?>

**(45)Write a program to input a number and check whether it is perfect number or not ?**

<?php

6=1+2+3=6

$num=6;

$sum=0;

$i=1;

while($i<$num)

{

if($num%$i==0)

$sum=$sum+$i;

$i++;

}

if($sum==$num)

echo “number is perfect”;

else

echo “number is not perfect”;

?>

1. **Do-while loop:-**

It is otherwise called as bottom-tested loop or post-tested loop or exit control loop. in this type of looping statement first the statements are get executed and printed after that the condition is checked. If the condition is false then no statement is executed or printed.

syn:- variable value initialization;

do

{

statement(s);

incr/decr;

} while(condition);

print statement(s);

**(46)Write a program to print the series like 11, 22, 33,....,99 ?**

<?php

$i=11;

do

{

Echo $i.”<br>”;

$i=$i+11;

}

while($i<=99);

?>

**(47)Write a program to input an integer number and find the sum of its digits ?**

<?php

123=1+2+3=6

$sum=0;

$num=123;

do

{

$rem=$num%10;

$sum=$sum+$rem;

$num=$num/10;

}

while($num!=0);

echo “sum of digits=”.$sum);

?>

**(48)Write a program to input a number and find its factorial value ?**

<?php

5!=5x4x3x2x1=120

$fact=1;

$num=5;

do

{

$fact=$fact\*$num;

$num--;

}

while($num!=0);

echo “factorial value=”$fact;

?>

**(49)Write a program to input a number and check whether it is armstrong or not ?**

<?php

153=13+53+33=1+125+27=153

$sum=0;

$num=153;

$temp=$num;

do

{

$rem=$num%10;

$sum=$sum+($rem\*$rem\*$rem);

$num=$num/10;

}

while($num!=0);

if($temp==$sum)

echo “number is armstrong”;

else

echo “number is not armstrong”;

?>

1. **For loop:-**

It is the simplest type of looping statement because here all the three parts of the loop written in one line; so it reduce the line of codes.

syn:- for(initialization; condition; incr/decr)

**(50)Write a program to print the series like 50,45,40,....5 ?**

<?php

for($i=50;$i>=5;$i=$i-5)

{

echo $i.”<br>”;

}

?>

**(51)Write a program to find the hcf and lcm of two integer number ?**

<?php

$num1=10;

$num2=20;

for($i=1;$i<=$num1,$i<=$num2;$i++)

{

if($num1%$i==0&&$num2%$i==0)

$hcf=$i;

}

$lcm=($num1\*$num2)/$hcf;

Echo “hcf=”.$hcf.”<br>”;

Echo “lcm=”.$lcm;

?>

**(52)Write a program to find n fibonacci numbers starting from 0 ?**

$num=8;

0,0,1,2,3,5,8,13,21,34,...

$prev=0;

$curr=1;

$next=0;

Echo $prev.”<br>”;

Echo $curr.”<br>”;

for($i=1;$i<=$num-2;$i++)

{

$next=$prev+$curr;

Echo $next.”<br>”;

$prev=$curr;

$curr=$next;

?>

**(53)Write a program to display all the even numbers and all the odd numbers in between 1 to 10 ?**

<?php

$even=0;

$odd=1;

for($i=1;$i<=10;$i++)

{

if($i%2==0)

$even=$even+$i;

else

$odd=$odd\*$i;

}

echo “even sum=”.$even.”<br>”;

echo “odd product=”.$odd;

?>

1. **foreach loop:-**

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

Syn:-

foreach ($array  as  $value)

{  
    //code to be executed;  
}

### Ex:-

### <?php  $a=array(33,2,89,1,67);  foreach ($a as $i)

### { echo $i .“<br>"; } ?>

**Nested for:-**

When one or more than one for loops are nested inside another for loop then it is called nested for loop.

**(55)Write a program for pyramid series ?**

<?php

1

1. 2

1 2 3

1 2 3 4

1 2 3 4 5

for($i=1;$i<=5;$i++)

{

for($j=1;$j<=$i;$j++)

echo $j;

echo “<br>”;

}

?>

**(56)Write a program for pyramid series ?**

<?php

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

for($i=5;$i>=1;$i--)

{

for($j=1;$j<=$i;$j++)

echo $j;

echo “<br>”;

}

?>

getch();

}

**(58)Write a program for pyramid series ?**

<?php

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

$k=1;

for($i=1;$i<=5;$i++,$k=$k+2)

{

for($j=$i;$j<=4;$j++)

echo “ “;

for($j=1;$j<=$k;$j++)

echo “\*”;

echo “<br>”;

}

?>

**(59)Write a program to print floyd triangle upto five rows ?**

<?php

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

$k=1;

for($i=1;$i<=5;$i++)

{

for($j=1;$j<=$i;$j++)

{

echo $k.”\t”;

$k++;

}

echo “<br>”;

}

?>

**(60)Write a program to generate pascal triangle upto five rows ?**

<?php

1

1. 1

1 2 1

1 3 3 1

1 4 6 4 1

$coef=1;

for($i=1;$i<=5;$i++)

{

for($space=1;$space<=5-i;$space++)

echo “ “;

for($j=0;$j<=$i;$j++)

{

if($j==0||$i==0)

$coef=1;

else

$coef=$coef\*($i-$j+1)/$j;

echo $coef;

}

echo “<br>”;

}

?>

**Array(CH-4)**

Array is a derived data-type in PHP which is used to store similar type of values or homogeneous values inside a single variable by defining subscript/size. Array elements are accessed by the help of unique index number which is by default starts from zero.

Types:-

1. **Indexed Arrays/Numeric Arrays** - Arrays with a numeric index is called numeric array or indexed array.

Ex1:-

<?php  
$a=array(45, 8,23,1,34);  
echo $a[0].",".$a[1] . “,”.$a[2].”,”.$a[3].”,”.$a[4];  
?>

Ex2:-

<?php  
$a=array("one","two","three");  
$length=count($a); //using count method  
for($i=0;$i< $length;$i++)

{  
echo $a[$i];  
echo "<br>";  
}  
?>

1. **Associative arrays** - Arrays with named keys means every element of array is associated with a key value is called associative array.  
   Ex:-

<?php  
$age = array("Anil"=>"28", "Swarnaa"=>"20", "Ansu"=>"0");  
echo "Anil is " . $age['Anil'] . " years old.".”<br>”;

echo " Swarnaa is " . $age['Swarnaa'] . " years old.".”<br>”;

echo " Ansu is " . $age['Ansu'] . " years old.";  
?>

Ex:-

<?php  
$age = array("Anil"=>"28", "Swarnaa"=>"20", "Ansu"=>"0");  
foreach($age as $i => $value)

{  
echo "Key=".$i.",Value=".$value."<br>";  
}  
?>

**Pre-defined array methods:-**

**1.count():-** used to count the length or number of elements present inside the array.

Ex:-

<?php  
$a=array(4, 6, 2, 22, 11);  
$length=count($a);

echo $length;

?>

**2.sort():-**used to sort an index/numeric array elements in ascending order.

Ex:-

<?php  
$numbers=array(4, 6, 2, 22, 11);  
sort($numbers);

var\_dump($numbers);  
?>

**3.rsort():-** used to sort all index array elements in descending order of alphabets.

Ex:-

<?php  
$cars=array("Volvo", "BMW", "Toyota");  
rsort($cars);

var\_dump($cars);  
?>

**4.asort():-** used to sort all the associative array elements in ascending order according to their values.

Ex:-

<?php  
$age=array("manish"=>"35", "akash"=>"47", "suraj"=>"43");  
asort($age);

var\_dump($age);  
?>

**5.ksort():-** used to sort all the associative array elements in ascending order according to their key indexes.

Ex:-

<?php  
$age=array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
ksort($age);

var\_dump($age);  
?>

**6.arsort():-** used to sort all the associative array elements in descending order according to their values.

Ex:-

<?php  
$age=array("Peter"=>"37", "Ben"=>"35", "Joe"=>"43");  
arsort($age);

var\_dump($age);  
?>

**7.krsort():-** used to sort all associative array elements in descending order according to their key values/indexes.

Ex:-

<?php  
$age= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
krsort($age);

var\_dump($age);  
?>

**8.join():-**

Join array elements to form a string using delimiter or special symbol.

<?php  
$arr = array(1,2,3,4,5);  
echo join("\*",$arr);  
?>

**9.array\_flip():-**

Flip/exchange all keys with their associated values in an array:

<?php  
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");  
$result=array\_flip($a1);  
print\_r($result);  
?>

**10.array\_filter():-**

Filter the values of an array using a call-back function.

<?php  
function abc($a)

{

if($a%2==0)

return TRUE;

else

return FALSE;

}

$a=array(1,2,3,4,5);

print\_r(array\_filter($a, ”abc”));  
?>

**11.array\_intersect():-**

Compare the**values** of two arrays, and return the matches.

<?php  
$a1=array(1,2,3,4);  
$a2=array(3,4,5);  
$result=array\_intersect($a1,$a2);  
print\_r($result);  
?>

**12.array\_merge():-**

Merge/join two arrays into one array:

<?php  
$a1=array(1,2);  
$a2=array(3,4);  
print\_r(array\_merge($a1,$a2));  
?>

**13.array\_pad():-**

Return 5 elements and insert a value of "blue" to the new elements in the array.

<?php  
$a=array("red", "green");  
print\_r(array\_pad($a,5,"blue"));  
?>

**14.array\_pop():-**

Delete the last element from an array:

<?php  
$a=array("red", "green", "blue");  
array\_pop($a);  
print\_r($a);  
?>

**15.array\_product():-**

Calculate and return the product/multiplication of an array.

<?php  
$a=array(5,5);  
echo(array\_product($a));  
?>

**16.array\_push():-**

Used to push/insert new elements into the existing array. Insert "blue" and "yellow" to the end of an array.

<?php  
$a=array("red", "green");  
array\_push($a, "blue", "yellow");  
print\_r($a);  
?>

**17.array\_reverse():-**

Return an array in the reverse order:

<?php  
$a=array(1,4,2,7);  
print\_r(array\_reverse($a));  
?>

**18.array\_search():-**

Search an array for the value "red" and return its key:

<?php  
$a=array("a"=>"red", "b"=>"green", "c"=>"blue");  
if(array\_search("red", $a))

echo “element found”;

else

echo “element not found”;  
?>

**19.array\_shift():-**

Remove the first element from an array and return the value of the removed element.

<?php  
$a=array("a"=>"red","b"=>"green","c"=>"blue");  
echo array\_shift($a).”<br>”;  
print\_r ($a);  
?>

**20.array\_slice():-**

Start slice from the desired position(third) of array element, and return the rest of the elements in the array.

<?php  
$a=array("red", "green", "blue", "pink", "brown");  
print\_r(array\_slice($a,2));  
?>

**21.array\_splice():-**

Remove elements from an array and replace it with new elements.

<?php  
$a1=array("a"=>"red", "b"=>"green", "c"=>"blue", "d"=>"pink");  
$a2=array("a"=>"purple", "b"=>"orange");  
array\_splice($a1,0,2,$a2);  
print\_r($a1);  
?>

**22.array\_sum():-**

Return the summation/addition of all the values in the array.

<?php  
$a=array(5,15,25);  
echo array\_sum($a);  
?>

**23.array\_unique():-**

Remove duplicate values from an array.

<?php  
$a=array("a"=>"red", "b"=>"green", "c"=>"red");  
print\_r(array\_unique($a));  
?>

**24.array\_unshift():-**

Insert the element "blue" to an array as a single value only.

<?php  
$a=array("a"=>"red", "b"=>"green");  
array\_unshift($a, "blue");  
print\_r($a);  
?>

**25.extract():-**

Assign new values to the variable which contain old values previously, for example assign the values "Cat", "Dog" and "Horse" to the variables $a, $b and $c.

<?php  
$a = "Original";  
$my\_array = array("a" => "Cat", "b" => "Dog", "c" => "Horse");  
extract($my\_array);  
echo "\$a = $a; \$b = $b; \$c = $c";  
?>

**26.in\_array():-**

Search for an element present inside the array or not and returns a value.

<?php  
$people = array("Peter", "Joe", "Glenn", "Cleveland");  
if (in\_array("Glenn", $people))  
{  
echo "Match found";  
}  
else  
{  
echo "Match not found";  
}  
?>

**27.implode():-**

Join array elements to get a string.

<?php  
$arr = array('Hello', 'World!', 'Beautiful', 'Day!');  
print\_r($arr).”<br>”;

echo implode("\*", $arr);  
?>

**Php String(CH-5)**

A string is a collection of array of characters. A string may be enclosed within a pair of single quote (‘ ’) or double quote (“ ”). There are different types of pre-defined string methods available in php.

**1.strlen():-**

This function is used to return the length of the string by counting the number of characters present inside it and it always returns a numeric value.

<?php  
echo strlen("Hello world!");  **// output 12**  
?>

**2.str\_word\_count():-**

It counts the number of words present inside the string and returns a numeric value.

<?php  
echo str\_word\_count("Hello world!");  **// output 2**  
?>

**3.strrev():-**

Used to reverse a string.

<?php  
echo strrev("Hello world");  **// outputs dlrow olleH**  
?>

**4.strpos():-**

Used to return the position of a sub-string from a given large string.

<?php  
echo strpos("Hello world!", "world");  **// outputs 6**?>

**5.str\_replace():-**

Used to replace a string from old string with a new string.

<?php  
echo str\_replace("world", "Dolly", "Hello world!");  **// outputs** Hello Dolly!

?>

**6.chop():-**

Remove characters from the right end of a string.

<?php  
$str = "Hello World!";  
echo $str . "<br>";  
echo chop($str,"World!");  
?>

**7. chr():-**

Return characters from different ASCII values.

<?php  
echo chr(65);

?>

**8. ltrim():-**

Remove characters from the left side of a string.

<?php  
$str = "madam”;  
echo ltrim($str,"m");  
?>

**9. rtrim():-**

Remove characters from the right side of a string.

<?php  
$str = "madam";  
echo $str . "<br>";  
echo rtrim($str, "m");  
?>

**10. trim():-**

Remove characters from both ends/sides of a string.

<?php  
$str = "madam";  
echo $str . "<br>";  
echo trim($str,"m");  
?>

**11.wordwrap():-**

Wrap a string into new lines when it reaches a specific length.

<?php  
$str="this is the example of wordwrap function";  
echo wordwrap($str,15,"<br>\n");  
?>

**12.substr():-**

Return a short string from a full string by specifying position.

<?php  
echo substr("Hello world",6 );  
?>

**13.strtr():-**

Replace/translate the old characters "ia" in the string with new characters "eo".

<?php  
echo strtr("Hilla Warld","ia","eo");  
?>

**14.strtoupper():-**

Convert all characters to uppercase:

<?php  
echo strtoupper("Hello WORLD!");  
?>

**15.strtolower():-**

Convert all characters to lowercase.

<?php  
echo strtolower("Hello WORLD.");  
?>

**16.strstr():-**

Find the first occurrence of "world" inside "Hello world!" and return the rest of the string.

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

**17.ord():-**

Return the ASCII value of character.

<?php  
echo ord("a")."<br>";  
echo ord("A");  
?>

**18.explode():-**

Break a string into an array.

<?php  
$str = "Hello world. It is a beautiful day.";  
print\_r (explode(" ", $str));  
?>

**Php forms(CH-6)**

The PHP super-global variable $\_GET and $\_POST are used to collect form-data. Means using these variables we can collect data from html form components like textbox, text-area, radio, checkbox etc and transfer these values from html page to php and store them inside variables for further use.

**GET POST**

1. Less secure. 1. More secure.

2. URL is visible on the address bar. 2. URL is invisible on the address bar.

3. Size for taking input of variable 3. Size for taking input of variable data

data is 2000 bytes. is 4000 bytes.

**Ex:- form.html**

<html>

<head>

<title>abc</title>

</head>  
<body>  
<form action="input.php" method="post">  
Enter Name: <input type="text" name="name">  
Enter E-mail: <input type="text" name="email">  
<input type="submit">  
</form>  
</body>  
</html>

**Ex:- input.php**

<html>  
<body>  
<h1>Welcome !</h1>

<?php

$a=$\_POST["name"];

$b=$\_POST["email"];

echo “Your name is:”.$a.”<br>”;  
echo “Your email address is:”.$b;

?>  
</body>  
</html>

**Form validation:-**

<html>  
<head>

<title>abc</title>  
<style type=”text/css”>  
.error

{

color:red;

}  
</style>  
</head>  
<body>    
<?php

$nameErr=$emailErr=$genderErr=$websiteErr="";  
$name=$email=$gender=$comment=$website="";  
if($\_SERVER["REQUEST\_METHOD"]=="POST")

{  
if(empty($\_POST["name"]))

{  
$nameErr="Name is required";  
}

else

{  
$name=test\_input($\_POST["name"]);  
//check if name only contains letters and whitespace  
if(!preg\_match("/^[a-zA-Z ]\*$/",$name))

{  
$nameErr="Only letters and white space allowed";   
}  
}  
if(empty($\_POST["email"]))

{  
$emailErr="Email is required";  
}

else

{  
$email=test\_input($\_POST["email"]);  
// check if e-mail address is well-formed  
if(!filter\_var($email, FILTER\_VALIDATE\_EMAIL))

{  
$emailErr="Invalid email format";   
}  
}  
if(empty($\_POST["website"]))

{  
$website = "";  
}

else

{  
$website=test\_input($\_POST["website"]);  
// check if URL address syntax is valid  
if(!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i",$website))

{  
$websiteErr="Invalid URL";   
}      
}  
if(empty($\_POST["comment"]))

{  
$comment="";  
}

else

{  
$comment=test\_input($\_POST["comment"]);  
}  
if(empty($\_POST["gender"]))

{  
$genderErr="Gender is required";  
}

else

{  
$gender=test\_input($\_POST["gender"]);  
}  
}  
function test\_input($data)

{  
$data=trim($data);  
$data=stripslashes($data);  
$data=htmlspecialchars($data);  
return $data;  
}  
?>  
<h2>PHP Form Validation Example</h2>  
<p><span class="error">\* required field</span></p>  
<form method="post" action="<?php echohtmlspecialchars($\_SERVER["PHP\_SELF"]);?>">    
Name: <input type="text" name="name">  
<span class="error">\* <?php echo $nameErr;?></span>  
<br><br>  
E-mail: <input type="text" name="email">  
<span class="error">\* <?php echo $emailErr;?></span>  
<br><br>  
Website: <input type="text" name="website">  
<span class="error"><?php echo $websiteErr;?></span>  
<br><br>  
Comment: <textarea name="comment" rows="5" cols="40"></textarea>  
<br><br>  
Gender:  
<input type="radio" name="gender"value="female">Female  
<input type="radio" name="gender" value="male">Male  
<input type="radio" name="gender" value="other">Other  
<span class="error">\* <?php echo $genderErr;?></span>  
<br><br>  
<input type="submit" name="submit" value="Submit">    
</form>  
<?php  
echo "<h2>Your Input:</h2>";  
echo $name;  
echo "<br>";  
echo $email;  
echo "<br>";  
echo $website;  
echo "<br>";  
echo $comment;  
echo "<br>";  
echo $gender;  
?>  
</body>  
</html>

**File Handling(CH-7)**

File is a collection of related data to be stored in a particular area on the disk for a long period of time. There are different types of files available in programming language like text file and binary file. The file which contains text, audio, video, image in human readable format is called text file where as the file whose data are in binary format like 0 and 1 and readable by computer only is called binary file.

The data which are captured by html forms are stored inside a file using some file handling techniques.

**File operations:-**

**i. opening a file:-** for opening a file for reading/writing purpose we use fopen() function which takes two arguments; one is filename and another is filemode. There are different modes available for opening a file in php that are:

**1.” r” mode:-** opens a file for reading.

**2. “w” mode:-** opens a file for writing.

**3. “a” mode:-** opens a file for appending new contents to the end of file.

**4. “x” mode:-** create a file for writing and return false and error if the file is already exist.

**5. “r+” mode:-** open a file for reading and writing and file pointer starts at the beginning of the file.

**6. “w+” mode:-** open a file for reading or writing and erase the contents of the file or creates a new file if it doesn’t exist. File pointer starts at the beginning of the file.

**7. “a+” mode:-** open a file for reading or writing . It saves the existing data and file pointer starts from end of file and creates a new file if the file is not exists.

**8. “x+” mode:-** creates a new file for reading or writing and returns false if file is already exist.

**ii. closing a file:-** after completing all the operations on a file we have to close the file using fclose() method.

**Pre-defined file i/o functions:-**

**1. readfile():-** used to read a file and returns number of bytes.

Syn:- readfile(“filename”);

Ex:-

<?php  
echo readfile("abc.txt");  
?>

**2. fopen():-** used to open a file for reading or writing.

Syn:- filevariablename=fopen(“filename”,”mode”);

Ex:- $p=fopen(“abc.txt”,”r”);

**3. fclose():-** used to close a file after all operations.

Syn:- fclose(filevariable);

Ex:- fclose($p);

**4. fread():-** used to read the contents of the file.

Syn:- fread(filevariable,filesize(“filename”));

Ex:-

<?php  
$myfile=fopen("abc.txt", "r") or die("Unable to open file!");  
echo fread($myfile,filesize("abc.txt"));  
fclose($myfile);  
?>

**5. fgets():-** used to read a single line from the file.

Syn:- fgets(filevaraible);

Ex:-

<?php  
$myfile = fopen("abc.txt", "r") or die("Unable to open file!");  
echo fgets($myfile);  
fclose($myfile);  
?>

**6. feof():-** used to check whether file reached to the end of file or not.

Syn:- feof(filevariable);

<?php  
$myfile=fopen("abc.txt", "r") or die("Unable to open file!");  
while(!feof($myfile))

{  
echo fgets($myfile)."<br>";  
}  
fclose($myfile);  
?>

**7. fgetc():-** used to read a single character from a file.

Syn:- fgetc(filevariable);

<?php  
$myfile=fopen("abc.txt", "r") or die("Unable to open file!");  
while(!feof($myfile))

{  
echo fgetc($myfile);  
}  
fclose($myfile);  
?>

**8. fwrite():-** used to write to a file.

Syn:- fwrite(filevariable, ”text/string”);

<?php  
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");  
$txt = "anil kumar\n";  
fwrite($myfile, $txt);  
$txt = "anil meher\n";  
fwrite($myfile, $txt);  
fclose($myfile);  
?>

**9. fputs():-** used to write a string/line to an opened file.

Syn:- fputs(filevariable, string, length);

Ex:-

<?php  
$myfile = fopen("abc.txt", "w") or die("Unable to open file!");

fputs($myfile,”hello”,5);

?>

**Uploading a File:-**

**Ex:-**

**form.html**

<html>

<head>

<title>abc</title>

</head>  
<body>  
<form action="upload.php"  method="post" enctype="multipart/form-data">  
Select image to upload:  
<input type="file" name="fileToUpload" id="fileToUpload">  
<input type="submit" value="Upload Image" name="submit">  
</form>  
</body>  
</html>

**upload.php**

<?php  
$target\_dir = "uploads/";  
$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);  
$uploadOk = 1;  
$imageFileType=strtolower(pathinfo($target\_file,PATHINFO\_EXTENSION));  
if(isset($\_POST["submit"]))

{  
$check = getimagesize($\_FILES["fileToUpload"]["tmp\_name"]);  
if($check !== false)

{  
echo "File is an image - " . $check["mime"] . ".";  
$uploadOk = 1;  
}

else

{  
echo "File is not an image.";  
$uploadOk = 0;  
}  
}  
?>

**Advance PHP(CH-8)**

**date() method:-**

Used to catch system date and time and display it on the screen.

Ex:-

<html>  
<body>  
<?php  
echo "Today is " . date("d/m/Y") . "<br>";  
echo "Today is " . date("m.d.y") . "<br>";  
echo "Today is " . date("y-m-d") . "<br>";  
echo "Today is " . date("l") . "<br>";

echo "Today is ". date("h:i:sa");

?>  
</body>  
</html>

**include() method:-**

Include statement takes all the text/code/mark-up that exists in the specified file and copies it into the file that uses the include statement. Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

Syn:**-** include ‘filename’;

Ex:**-**

**abc.php**

<?php  
echo "<p>hello students</p>";  
?>

**xyz.php**

<html>  
<body>  
<h1>Welcome to my home page!</h1>  
<p>welcome</p>  
<p>hi</p>  
<?php

include 'abc.php';

?>  
</body>  
</html>

**require() method:-**

Require statement is also used to include a file into the PHP code. This method is just similar to include.

Syn:- require ‘filename’;

Ex:-

**abc.php**

<?php  
echo "<p>hello students</p>";  
?>

**xyz.php**

<html>  
<body>  
<?php

require 'abc.php';

?>

</body>  
</html>

**Difference between include and require:-**

When a file is included with the include statement and PHP cannot find it, the script will continue to execute. If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after require statement returned a fatal error. Use require when the file is required by the application. Use include when the file is not required and application should continue when file is not found.

**Example for include:-**

<html>  
<body>  
<h1>Welcome to my home page!</h1>  
<?php

include 'pqr.php';  
echo "it is displayed.";  
?>  
</body>  
</html>

**Example for require:-**

<html>  
<body>  
<h1>Welcome to my home page!</h1>  
<?php

require 'pqr.php';  
echo "It is displayed.";  
?>  
</body>  
</html>

**Cookies:-**

A cookie is used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, we can both create and retrieve cookie values.

**Cookie operations:-**

1. **Create a cookie:-**

Syn:-

setcookie(name, value, expire, path, domain, secure, http-only);

Ex:-

<?php  
$cookiename = "user";  
$cookievalue = "anil";  
setcookie($cookiename, $cookievalue, time( ) + (86400 \* 30), "/");

?>  
<html>  
<body>  
<?php  
if(!isset($\_COOKIE[$cookiename]))   
echo "Cookie named" . $cookiename . " is not set!";

else   
echo "Cookie " .$cookiename . " is set!<br>";  
echo "Value is: " . $\_COOKIE[$cookiename];  
?>  
</body>  
</html>

**2. modify a cookie:-**

Ex:-

<?php  
$cookie\_name = "user";  
$cookie\_value = "Alex Porter";  
setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/");  
?>  
<html>  
<body>  
<?php  
if(!isset($\_COOKIE[$cookie\_name]))

{  
echo "Cookie named '" . $cookie\_name . "' is not set!";  
}

else

{  
echo "Cookie '" . $cookie\_name . "' is set!<br>";  
echo "Value is: " . $\_COOKIE[$cookie\_name];  
}  
?>  
</body>  
</html>

**3. delete a cookie:-**

Ex:-

<?php  
setcookie("user", "", time() - 3600);  
?>  
<html>  
<body>  
<?php  
echo "Cookie 'user' is deleted.";  
?>  
</body>  
</html>

**4. enabling a cookie:-**

<?php  
setcookie("test\_cookie", "test", time() + 3600, '/');  
?>  
<html>  
<body>  
<?php  
if(count($\_COOKIE) > 0)

{  
echo "Cookies are enabled.";  
}

else

{  
echo "Cookies are disabled.";  
}  
?>  
</body>  
</html>

**Sessions:-**

A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user computer. Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, value etc). By default, session variables last until the user closes the browser. So; Session variables hold information about one single user, and are available to all pages in one application.

**Session operations:-**

**1. start session:-**

<?php  
session\_start();  
?>

<html>  
<body>  
<?php  
$\_SESSION["name"] = "user";  
$\_SESSION["value"] = "anil";  
echo "Session variables are set.";  
?>  
</body>  
</html>

**2. modify a session:-**

<?php  
session\_start();  
?>

<html>  
<body>  
<?php  
$\_SESSION["value"] = "abc";  
print\_r($\_SESSION);  
?>  
</body>  
</html>

**3. destroy session:-**

<?php  
session\_start();  
?>

<html>  
<body>

<?php  
session\_unset();   
session\_destroy();   
?>  
</body>  
</html>

**Exception Handling(CH-9)**

Exceptions are the errors which are occur during program execution process. There are two types of exceptions occur inside the program. That are :

1. **Compile time error:-** the errors which are occurred at the time of compilation of the program are called compile time errors. These are also called as syntax error.

Ex:- statement missing, termination error etc.

1. **Runtime error:-** the errors which are occurred at the time of running the program are called runtime errors. There are two types of runtime exceptions available like:
2. **Synchronous exception:-** the exceptions which are under the control of the programmer/computer are called synchronous exceptions. Ex:- loop overflow, zero divide, number format error.
3. **Asynchronous exception:-** the exceptions which are beyond the control of the programmer/computer are called asynchronous exceptions. Ex:- system crash, hard-disk failure etc.

**Handling an exception:-**

To handle an exception in php we must use three keywords.

1. **Try:-** it is a block of statement where we can generate an exception.
2. **Catch:-** it is also a block of statement where we can display an exceptional message generated by try block by the help of exceptional type argument.
3. **Throw:-** it is a keyword which is used inside try block to throw an exception from try block to catch block.

Syn:- try

{

//statement(s);

throw exception;

}

catch(type argument)

{

// print error message

}

Write a program for zero divide?

<?php

$a=5;

$b=0;

try

{

$c=$a/$b;

echo $c;

if($b==0)

return $b;

}

catch(Exception $e)

{

echo “a number cannot divide with zero”;

}

?>

**OOPs in PHP(CH-10)**

As we discussed php is a partial object oriented programming language. So it supports the concept of oops.

**Features of OOPs:-**

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

1. **Class:-** class is an user defined data-type which describe about a particular item or person or place. In OOP languages it is must to create a class for representing data. Class contains variables for storing data and functions to specify various operations that can be performed on data. Class will not occupy any memory space and hence it is only logical representation of data.
2. **Object:-** object is the instance/copy of the class which describes the characteristics/behaviour of the class. We can create more than one object for a single class.
3. **Inheritance:-** when one class inherits the features of another class then it is called inheritance. The main advantage of using inheritance is code reusuability and security. In java there are two types of inheritance; one is single and another is interface.
4. **Polymorphism:-** greek word poly means many and morphism means form. When a single method with same name process to form different- different operations then it is called polymorphism. There are two types of polymorphism like compile time polymorphism and run time polymorphism.
5. **Data encapsulation:-** wrapping up of data into a single unit is called encapsulation. Encapsulation is hiding the functional data from the object calling it. Integration of data and operations/functions in a class is Encapsulation. So it is also called data hiding.
6. **Data abstraction:-** It is used to display only necessary and essential features of an object to the outside world. Choosing of necessary data by discarding unnecessary data is called data abstraction.
7. **Message passing:-** when one object copy/pass the messages of it to another object then it is called as message passing. Passing can be done either implicitly or explicitly.

**Class:-**

a class is an user-defined datatype which is used to store heterogeneous or dissimilar type of data. It is created by the help of the keyword class. It describes about a particular person or place or thing. A class consist of two things.

1. **Data members:-** these are just like variables or instance variables which are used to hold class data inside the program. The type of data held by the data members are decided by data-type preceded by the data-members.
2. **Member functions:-** these are the user-defined functions which are used to operate over the data-members of the class. Member functions also preceded by a return type. We can use n number of member functions inside a class program.

**Object:-**

Object is an instance/copy of class. It is created by the help of a class keyword to call the member functions by the help of (->) operator to perform operations.

**Access specifiers:-**

These are the access restrictions provided to every class members that may be data members or member functions so that the class data must be secured. There are different types of access specifiers available in c++ programming like:

1. **Private:-** this is the default access specifier used in c++ programming. If any data is declared as private that means its data are access inside that class only.
2. **Public:-** if any class data are declared as public then they must be accessed throughout the program that may be inside different class.
3. **Protected:-** it is specially used incase of inheritance where we use at least two class. If any member is declared as protected then its dada are only shared with that classes who are inherted from this class.

**Syntax for creating a class:-**

class classname

{

var $variables;

function functionname()

{

//body of function;

}

}

**Syntax for creating object:-**

$objectname=new classname;

Calling member functions:-

$objectname=functionname(argument values);

<?php   
class Foo {   
    public $aMemberVar = 'aMemberVar Member Variable';   
    public $aFuncName = 'aMemberFunc';   
      
      
    function aMemberFunc() {   
        print 'Inside `aMemberFunc()`';   
    }   
}   
  
$foo = new Foo;   
?>

Static:-

Declaring class members or methods as static makes them accessible without needing an instantiation of the class. A member declared as static can not be accessed with an instantiated class object (though a static method can).

<?php

class hello

{

public static function abc()

{

return ‘hello’;

}

}

Hello::abc();

?>

<?php

class Foo {

public static $my\_static = 'foo';

public function staticValue() {

return self::$my\_static;

}

}

print Foo::$my\_static . "\n";

$foo = new Foo();

print $foo->staticValue() . "\n";

?>

Final :-

PHP 5 introduces the final keyword, which prevents child classes from overriding a method by prefixing the definition with final. If the class itself is being defined final then it cannot be extended.

Abstract:-

An abstract class is one that cannot be instantiated, only inherited. You declare an abstract class with the keyword **abstract**, like this −

When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child; additionally, these methods must be defined with the same visibility. that function definitions inside an abstract class must also be preceded by the keyword abstract. It is not legal to have abstract function definitions inside a non-abstract class.

Interface:-

Interfaces are defined to provide a common function names to the implementers. Different implementors can implement those interfaces according to their requirements. You can say, interfaces are skeletons which are implemented by developers.

Constructor:-

Constructor Functions are special type of functions which are called automatically whenever an object is created. So we take full advantage of this behaviour, by initializing many things through constructor functions.

PHP provides a special function called **\_\_construct()** to define a constructor. You can pass as many as arguments you like into the constructor function.

Destructor:-

Like a constructor function you can define a destructor function using function **\_\_destruct()**. You can release all the resources with-in a destructor.

**Mysql(CH-11)**

Database is a collection of related data to be stored inside server in row and column format. Database is a backend process which is used to process frontend and manipulate with frontend applications. For project or software development we must use backend with a front end. Different types of backend database are:

1. ms-access

2. oracle

3. mysql

4. sqlserver

**What is mysql ?**

* MySQL is a database system used on the web.
* MySQL is a database system that runs on a server.
* MySQL is ideal for both small and large applications.
* MySQL is very fast, reliable, and easy to use.
* MySQL uses standard SQL (structured query language).
* MySQL compiles on a number of platforms.
* MySQL is free to download and use.
* MySQL is developed, distributed, and supported by Oracle Corporation.
* MySQL is named after co-founder Monty Widenius's daughter: My.

**How to work with mysql:-**

If we have installed “xampp” then it contain both “apache” as frontend server and “mysql” as backend server. So we have to start both “apache” as well as “mysql” from xampp control panel. Then open browser and type “localhost/phpmyadmin” on the address bar of the browser. Then the database mysql page was opened. After the database page is opened we can create our own database, select db, insert, update, delete, fetch records inside the database.

**How to connect mysql:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$conn = new mysqli($servername, $username, $password);  
if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
echo "Connected successfully";  
?>

**How to select database:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$conn = new mysqli($servername, $username, $password);  
if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
$sql = "CREATE DATABASE arth";  
if ($conn->query($sql) === TRUE)

{  
echo "Database created successfully";  
}

else

{  
echo "Error creating database: " . $conn->error;  
}  
$conn->close();  
?>

**Table creation:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "arth";  
$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
$sql = "CREATE TABLE login (slno INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,userid VARCHAR(20) NOTNULL, password VARCHAR(30) NOT NULL)";  
if ($conn->query($sql) === TRUE)

{  
echo "Login table created successfully";  
}

else

{  
echo "Error creating table: " . $conn->error;  
}  
$conn->close();  
?>

**Insert data:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "arth";  
$conn = new mysqli($servername, $username, $password, $dbname);  
if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
$sql = "INSERT INTO login (userid, password,)VALUES ('anil@arth', 'abc\*123')";  
if ($conn->query($sql) === TRUE)

{  
echo "New record created successfully";  
}

else

{  
echo "Error: " . $sql . "<br>" . $conn->error;  
}  
$conn->close();  
?>

**Update data:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "arth";  
$conn = new mysqli($servername, $username, $password, $dbname);  
if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
$sql = "UPDATE login SET password='hellohi' WHERE slno=1”;  
if ($conn->query($sql) === TRUE)

{  
echo "Record updated successfully";  
}

else

{  
echo "Error updating record: " . $conn->error;  
}  
$conn->close();  
?>

**Select data:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "arth";  
$conn = new mysqli($servername, $username, $password, $dbname);  
if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
$sql = "SELECT slno,userid, password FROM login";  
$result = $conn->query($sql);  
if ($result->num\_rows > 0)

{  
while($row = $result->fetch\_assoc())

{  
echo "userid: " . $row["userid"]. " - password: " . $row["password"]. "<br>";  
}  
}

else

{  
echo "0 results";  
}  
$conn->close();  
?>

**Delete data:-**

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
$dbname = "arth";  
$conn = new mysqli($servername, $username, $password, $dbname);  
if ($conn->connect\_error)

{  
die("Connection failed: " . $conn->connect\_error);  
}   
$sql = "DELETE FROM login WHERE slno=1";  
if ($conn->query($sql) === TRUE)

{  
echo "Record deleted successfully";  
}

else

{  
echo "Error deleting record: " . $conn->error;  
}  
$conn->close();  
?>

**Insert data using form(signup page):-**

**form.html**

<html>

<head>

<title>abc</title>

</head>

<body>

<h1>Login Info. </h1>

<form action="insert.php" method="POST">

Userid: <input type="text" name="t1">

Password: <input type="text" name="t2">

<input type="submit" value=”submit”/>

</form>

</body>  
</html>

**insert.php**

<html>

<head>

<title>abc</title>

</head>

<body>

<?php

$con = mysql\_connect("localhost","root","");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

$sel=mysql\_select\_db("anil", $con);

 if (!$sel)

{

die('Could not select db: ' . mysql\_error());

}

$sql="INSERT INTO login (userid, password)VALUES('$\_POST[t1]','$\_POST[t2]')";

if (!mysql\_query($sql,$con))

{

die('Error: ' . mysql\_error());

}

echo "1 record added";

mysql\_close($con)

?>

</body>

</html>

**Validation using php and database(signin page):-**

**login.php**

<html>

<head>

<style type="text/css">

input

{

border:1px solid olive;

border-radius:5px;

}

h1

{

color:red;

font-size:2cm;

text-align:center;

text-decoration:underline;

}

table

{

color:blue;

font-size:0.7cm;

}

#x

{

color:white;

background:green;

height:40px;

width:100px;

font-size:0.6cm;

border-radius:10pt;

}

</style>

</head>

<body>

<h1>Login Form</h1>

<form action='#' method='post'>

<table cellspacing='5' align='center'>

<tr><td>Username:</td><td><input type='text' name='name' placeholder="enter your userid" style="height:30px;width:250px;"/></td></tr>

<tr><td>Password:</td><td><input type='password' name='pwd' placeholder="enter your password" style="height:30px;width:250px;"/></td></tr>

<tr><td></td><td><input type='submit' name='submit' value='Submit' id="x"/></td></tr>

</table>

</form>

<?php

session\_start();

if(isset($\_POST['submit']))

{

$a=mysql\_connect("localhost","root","");

if($a)

echo "connected";

else

echo "not connected";

$b=mysql\_select\_db("anil");

if($b)

echo "selected";

else

echo "not selected";

$name=$\_POST['name'];

$pwd=$\_POST['pwd'];

if($name!=''&&$pwd!='')

{

$query=mysql\_query("select \* from login where username='".

$name."' and password='".$pwd."'") or die(mysql\_error());

$res=mysql\_fetch\_row($query);

if($res)

{

$\_SESSION['name']=$name;

header('location:home.php');

}

else

{

echo'You entered username or password is incorrect';

}

}

else

{

echo'Enter both username and password';

}

}

?>

</body>

</html>

**home.php**

<html>

<head>

<title>abc</title>

<style type="text/css">

#header

{

height:28%;

width:100%;

}

#menu

{

height:7%;

width:100%;

background:indigo;

float:left;

}

ul

{

list-style-type:none;

padding:0px;

margin:0px;

}

li

{

display:inline;

margin-left:5px;

margin-right:5px;

margin-top:5px;

margin-bottom:5px;

padding-left:5px;

padding-right:5px;

padding-top:5px;

padding-bottom:5px;

background:white;

border-raduis:10pt;

border-style:solid inset;

border-color:black;

font-size:0.7cm;

}

li a:hover

{

background:red;

}

#link

{

height:60%;

width:25%;

background:grey;

float:left;

}

p a:hover

{

background:lime;

}

#section

{

height:60%;

width:75%;

background:grey;

float:left;

color:white;

font-size:0.7cm;

}

#footer

{

height:5%;

width:100%;

background:indigo;

float:left;

color:white;

font-size:0.4cm;

}

</style>

</head>

<body>

<div id="header">

<img src="aa.jpg" height=100% width=100%">

<h2 align="right"><br>

<?php

echo '<a href="signout.php">Signout</a>';

?>

</h2>

</div>

<div id="menu">

<ul>

<li><a href="">home<a/></li>

<li><a href="">contact<a/></li>

<li><a href="">galary<a/></li>

<li><a href="">blog<a/></li>

<li><a href="">about<a/></li>

<li><a href="">other<a/></li>

</ul>

</div>

<div id="link">

<p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">library</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">news</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">sports</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">events</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">campus</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">faculty</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">student</a></p>

<p p style="background:maroon;font-size:0.5cm;margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;"><a href="" style="color:white;">other</a></p>

</div>

<div id="section">

<?php

session\_start();

$name=$\_SESSION['name'];

echo '<center>welcome :'.$name;

?>

</div>

<div id="footer">

<marquee>this site is designed and maintained by anil</marquee>

</div>

</body>

</html>

**signout.php**

<?php

session\_start();

session\_destroy();

header('location:login.php');

?>

**Updating account password of an user:-**

**update1.html**

<html>

<head>

<title>abc</title>

</head>

<body>

<form name="f" method="POST" action="delete.php">

Enter Username:<input type="text" name="t1">

Enter New Password:<input type="password" name="t2">

<input type="submit" value="delete" name="submit">

</form>

</body>

</html>

**update.php**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "anil";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error)

{

die("Connection failed: " . $conn->connect\_error);

}

$sql = "UPDATE LOGIN SET password=’$\_POST[t2]’ WHERE username='$\_POST[t1]'";

if ($conn->query($sql) === TRUE)

{

echo "Record updated successfully";

}

else

{

echo "Error updating record: " . $conn->error;

}

$conn->close();

?>

**Deleting account of an user:-**

**delete1.html**

<html>

<head>

<title>abc</title>

</head>

<body>

<form name="f" method="POST" action="delete.php">

Enter Username:<input type="text" name="t1">

<input type="submit" value="delete" name="submit">

</form>

</body>

</html>

**delete.php**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "anil";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error)

{

die("Connection failed: " . $conn->connect\_error);

}

$sql = "DELETE FROM login WHERE username='$\_POST[t1]'";

if ($conn->query($sql) === TRUE)

{

echo "Record deleted successfully";

}

else

{

echo "Error deleting record: " . $conn->error;

}

$conn->close();

?>