PHP

→ PHP is a server side scripting language
→ Abbrevation of PHP is (Personal Home Page) in olden days
Now, it is Hypertext Preprocessor
→ PHP is inverted by Rasmos, it belongs to Zend org^
→ features of PHP
→ It support cross-platform
→ It support cross-server
→ At the time of developing & testing of project → web server
→ At the time of deploying we use Application server
→ It support cross-database.

Software for PHP:
→ go to google search engine → type download XAMP
WAMP (windows Apache MySQL PHP) → work only on windows
LAMP (Linux " " ") → work only on Linux os
XAMP (cross " " ") → work on diff. os
XAMPP (cross " " " Perl) →

3306 is → mysql Port no
1521 is → oracle "
8080 is → Tomcat "

→ PHP is partially case sensitive, In case of variables it is case sensitive & In case of fuctions it is case insensitive.

PHP versions
1.0 → not server side scripting language
2.0 → partially server side scripting language
3.0 → purely server side scripting language

Disadvantages
→ It is open source (security)
→ we can't develop large scale applications
→ extension of PHP is .php
→ open source means we can see source code & modify to our requirements.
→ latest version of PHP is 8.1
server → high config. system

older days
zend org^

web server

AMP
on windows
on linux os
diff. os

variables
pt is

modify

→ we can change the port no at the time of installation &
in INI file
→ php.exe is interpreter name
→ we write php code inside following php tags:
1) <? php code ?> (universal style tag)
2) <? code > (canonical tag)
3) <% code > (ASP tag)
4) <script type="text/php"> (script tag).
→ In php we can display o/p in 5 diff. ways.
1) print() → by using this we can display only one strn
this function inside the brackets & returns boolean
value
ex: <? php
print $a=10;
print 'welcome to php $a';
?>
2) echo() → by using this fuction we can display multiple
strngs
ex: echo 'hi', 'welcome', '
', $a; → hi welcome
10
3) var_dump() : this function displays variable value
along with its datatype.
ex: $a=10;
var_dump($a); → int(10)
4) printf() : by using this function we can display the o/p
with the help of format specifiers
ex: $name = "isa";
$age = 10;
printf("%s has %d years exp", $name, $age);
o/p: isa has 10 years exp.
5) print_r() : by using this fuction we can display elements
from the array & properties from the obj

variable:

Rules for declaring a variable:
→ All variable names should be in lowercase.
→ static variable name should start with "s"
→ global variable name should start with "g"
→ variable names in PHP should start with "$"

Comments in PHP
→ It is a small description about the pgm ignored by PHP engine
1) single line comment (//)(C#)
2) Multiline comment (/*.....*/)

1) Local variable :- the variable which is declared inside the function
scope :- within the function
2) Global variable :- the variables which are declared outside the function.
scope :- within the pgm.
3) static variable :- once your php function 'execution has been completed we can retain the value by using static variables.
→ we can declare the static variable by using "static" keyword & "s" characters.
4) variable in variables : Declaring the variable inside the another variable.

$a="hello";
$$a="world";
$a → hello
$$a → hello world.

$a=10;
unset($a);

5) globals variable (super global variables) → these variables will be accessed from

$a=10;
function f(){
$b=20;
echo $b; → 20
echo $a; → error undefine
}
f();

→ There are 2 ways to access global variable value

1). $GLOBALS (super global variable)

```
$a=10;
Function f(){
    $a=11;
    echo $a; → 11
    echo $GLOBALS['a']; → 10
}
```

```
$a=10;
Function f(){
    global $a;
    echo $a; → 10
}
```

(or)

variable in variables

```
$x = "scott";
$y = "x"
echo $$y; → scott
          ↓
          $x → scott
```

static variable :- this variables are used to maintain previous values.

→ we can assign a value only once to the static variable

→ By using static keyword we declare the static variable

Ex :-
```
Function f(){
    static $a=100;
    $a++;
    echo $a;
}
f(); → 101
f(); → 102
f(); → 103
```

reference variable :- It refers value of another variable

Actual variable & reference variable these address will be same.

Ex :-
```
$x=100;
$y = &$x;
$y=200;
echo $y; → 200
```

super global variables

→ these variables can be accessible from anywhere within the project.

1) $GLOBALS : it is used to access global variable val
2) $_GET : it is used to get posted values of get() method
3) $_POST : it is used to get posted values of post() method
4) $_REQUEST : it is used to get posted values of ~~both~~ get(), post(), cookies & ~~queries statements~~ querystring strings.
5) $_SERVER : by using ~~this~~ this variable we get the info. about the server & browser like server & browser IP address
6) $_SESSION : } → by using this two variables we can
7) $_COOKIE : } maintain state of an application.
8) $_FILES : this variable is used to get info. of the uploaded files.
9) $_ENV : this variable is used to get os variables like computer name,

→ All super global variables return "Array Datatype values"

Datatype :
→ it specifies type & size of the data
→ PHP support 3 datatypes :
1) scalar/primitive datatypes
ex : int, float, string, boolean
2) compound/Derived datatypes : In this value will not be stored directly rather address will be stored
→ memory is allocated at run time.
ex : Arrays, objects
3) special datatype
ex : Null datatype
resources datatypes :-
→ these datatype refers to the external resources like database corection, FTP Correction, file pointers.

scalar datatype :
ex : $x = TRUE;
echo $x; → 1
print is_bool($x);

lobal variable val
es of get() method
of post() method
lues of both
queystring

e we get
like server &

es we can
plication;
f. of the

variables like

Datatype values"

will not be
be stored

eg like
ters.

→ by using is_bool() we can check whether the variable is boolean or not

→ (bool) → by using this fn we can convert other datatype to boolean datatype.

Eg: $x = "10";
print (bool) $x; → 1
print (int/integer) $x; → 10
print is_int ($x); →

Null datatype:

→ In 3 cases we can say the variable is null
1) if variable is not set with any value
2) if variable is set with null value
3) if variable is unset

Eg: $x;
$y = null;
$z = 10;
unset ($z);

Ex-2: $x = 100;
$y = "20";
$total = $x + $y; → 120
echo $total;

$x = 100;
$y = "pro pixel";
$total = $x + $y
echo $total;
o/p: 120 A non-numeric value encountered

Ex: $a = "da";
$b = ($a === null);
var_dump ($b); → boolean (false

Type conversion/casting: converting one type to other.

1) (array) → converting to array datatype
2) (bool)/(Boolean) → converting to boolean datatype
3) (int)/(integer) → " " integer "
4) (object) → converting to object datatype
5) (float) → " " float "
6) (string) → " " string "

Ex: $a = 'hi';
$b = (int) $a;
echo $b; → 0

$amount = '200 usd';
echo (int) $amount; → 200

Ex: $s = NULL; → 0
echo (int) $s;

Type Juggling

→ juggling occur during the comparision of variables of different datatypes.
→ PHP will be convert to the comparable type.

Ex: $a = 100;
$b = '200';
$total = $a + $b; / $a - $b; → -100.
echo $total; → 300

operators

operator: it is the symbol to perform operations on the operands
→ PHP support following operators:
1) Arithmetic operators (+, -, *, /, %.)
2) Assignment "
3) Bitwise "
4) Comparison " (==, ===, !==, !=, <, <=, >, >=)
→ identical operator
5) Inc & Dec "
6) logical "
7) Concatination "

Comparision operators

Ex: $a = 20;
$b = '20';
echo $a == $b; → true(1)
echo $a === $b; → false(0)

Ex: function f1() {
return false;
}
f1() or f2("Hello"); → Hello
false or true

logical operators

Ex) $a = TRUE or FALSE;
var_dump($a); → bool(1)
$a = (TRUE OR FALSE);
var_dump($a); → bool(1)

"or" has highest priority than "="
highest priority for

Concatination operator (.) :- By using . operator we can combine two strings.

control structures / control flow stmts

→ by using control flow stmts we control the execution of pgm.

→ It specifies the order in which the stmts should be execute.

→ PHP support 3 types of control flow stmts:

1) conditional stmts
   a) if
   b) switch

2) iterative stmts
   a) for
   b) while
   c) do while

3) jump / transfer stmts
   a) break
   b) continue
   c) return

1) conditional stmts :- for diff. conditions we perform diff. actions.

syntax: if (condition)
{ stmts; }

(or) if (condition):
stmts ← syntax
endif;

→ if we are mixed the php code with Html we use this)

Ex: $a = 10;
$b = 20;
$c = 30;
if ($a > $b and $a > $c):
echo $a. " is the big no";
elif ($b > $c):
echo $b. " is big no";
else:
echo $c. " is big no";
endif;

Ternary / conditional / question mark operator (? :)

→ It is the shortend form of if else.

syntax:

var = condition ? val-1 : val-2
(condition → true → val-1; condition → false → val-2)

Ex: $user_login = FALSE;
$result = $user_login ? "logout" : "login";
echo $result; → login

switch

→ from the multiple conditions if we are expecting only one value then go with switch case

syntax: switch(expression)
{
case caselabel:
stmts;
break;
——
default:
stmts;
}

(or)

switch(expression):
case caselabel:
stmts
break;
——
default:
stmts;
end switch;

Ex: $role = "Admin";
$message = "";
switch($role) {
case 'Admin':
$message = "welcome ita";
break;
case 'tester':
$message = "welcome tester";
break;
default:
$message = "invalid";
break;
}
echo $message; → welcome ita

Iterative statements

for loop: It is used when we know the fixed no. of iterations in advance

syntax: for(initialization; condition; inc/dec) {
stmts;
}

for each : it is used when we want to read the element from a collection / complex data.

Syntax :- foreach ($array as $variable) {
    logic;
}

Ex :- $arr = array (10,11,12,13);
foreach ($arr as $i){
  echo $i; → 10111213 (or) echo $i."
"; → 10
11
12
13
}

syntax :- foreach ($array as $key => $element) {
    logic;
}

Ex :- $arr = array(
    'name' => 'Ram';
    'rollno' => 21;
    'add' => "Hyd";
)
foreach ($arr as $key => $value)
{
    echo $key." ".$value."
"; → name Ram
rollno 21
add Hyd
}

while loop :- it is used when we don't know fixed no. of iterations in advance
→ while loop is called as entry control loop

Ex :- $a = 20;
$b = 65;
echo $a & ~$b; → 20

Constat :- it is a variable in php. which value doesn't change
→ there are 8 magic constants. it's starts with double underscore & end with double underscore
→ there are 2 ways to define a constant in php :-
1) define()
2) by using const keyword

1) by using define()

Ex: define('isa', 'welcome'); o/p: welcome
echo isa;

2) by using const keyword

Ex: const a='welcome';
echo a; → welcome

→ the scope of constant is global.

Ex: define('isa', 'welcome');
function f1(){
echo isa;
}
f1(); → welcome
?

Array constant

Ex: define('cars', ['bmw', 'audi', 'ford']);
echo cars[2];

magic constants:

→ these are the pre-defined constants in php, which are used on the bases of their use

→ there are 9 magic constants in php. it should start with (__) & end with (__).

→ only 8 magic constants are going to start & end with (__)

1) __LINE__ : it returns current line no. of line

Ex: echo '<h3>Example for __LINE__ </h3>';
echo __LINE__;

2) __FILE__ : it returns current file destiny with filename

3) __dir__ : it returns current file destiny upto foldername

4) __FUNCTION__ : It returns the name of the function where this magic constant is included.

Ex : echo 'Hello' . __FUNCTION__ . "
";

```
function fun1()
{
echo __FUNCTION__ . "<br>";
}
fun1();
```

5) __CLASS__ : It returns the current classname

Ex : class one

```
{
public function getclassname()
{
return __CLASS__;
}
}
$obj = new one();
echo $obj->getclassname();
```

6) __METHOD__ : It returns the methodname

Ex : return __METHOD__;

7) __NAMESPACE__ : It returns the name of namespace
→ It is a collection of classes, functions & variable.
→ using namespace keyword we create a namespace.
→ It should be declared on top of all.

Ex : <?php

```
namespace root;
{
return __NAMESPACE__;
}
}
```

8) __TRAIT__ : In php, only single level inheritance is supported.
→ If you want to use the class for multiple times, declare it as TRAIT so, that class can be used for multiple purpose.

Ex: trait ex {
    function f1() {
        echo "TRAIT";
    }
}
class company {
    use ex;
}
$obj = new company();
$obj->getclassname();

9) classname::class : it will return fully qualified classname.

Arrays : It is a collection of heterogeneous elements.
php supports 3 types of arrays:
1) Indexed Array / Numbered Array:
2) Associative Array
3) Multi-Dimensional Array

1) Indexed / Numbered Array : An array with numeric values, where values are stored numerically
→ In php, we can create array in 2 ways:
1) array() 2) []

Ex: $arr = array(10,7,5);
print_r($arr); → o/p: Array([0]=>10 [1]=>12 [2]=>15)

Ex: $arr = array(10,11,12);
for($i=0; $i<count($arr); $i++){
    echo $arr[$i]; → 101112
}

2) Associative Array : An array with string index, instead of linear storage, it will be stored with help of key

Ex: $arr = array(
    'p1' → 'c';
    'p2' → 'java'
);

$keys = array_keys($arr);
print_r($keys);

→ using array_keys, we get all the keys from array

3) Multi-Dimensional Array: It is also called as array of arrays

→ It's elements are considered as array. We are creating array in array instead of single element.

Ex: $arr = array(
array(),
array()
);

```
$arr = array("name" => "John
for($i=0; $i<count($arr); $i++)
{
foreach($arr[$keys[$i] as $key => $value)
{
echo $key."=".$value."<br>";
}
}
```

Array_change_key_case() :- By using this fn we change all the cases to the given lowercase/uppercase

1) ucfirst() → converting 1st characters to the uppercase

```
$arr = array(
    "name" => "John"
    "mob" => 9999999999,
);
print_r(array_change_key_case($arr, CASE_LOWER]
                                    CASE_UPPER.
```

sort() : it will arrange the elements in ascending order.

Ex: $arr = array(10, 2, 12, 20, 30);
sort($arr);
print_r($arr); → Array([0] => 2 (0 12 20 50)

rsort() : elements will be arranged in descending order

Ex: rsort($arr)
print_r($arr); → Array([0] => 30 [1] => 20 ----]

asort() : values will be arranged in ascending order.

Ex: asort($arr);
print_r($arr);

arsort() : values will arranged in descending order

ksort() : keys will be arranged in ascending order

krsort() : keys will be arranged in descending order

Ex: $arr = array("Peter" => "35", "Ben" => "32", "Joe" => "43");
sort($arr); O/P: Array([0] => 35 [1] => 37 [2] => 43)
print_r($arr);

array_sum() : it returns sum of the values in a array

array_sum() : it return product of all elements in an array

array_pop() : it will delete the last element & return it

Ex : from an array

```
$arr = array(10, 5, 20, 30);
echo array_pop($arr);
print_r($arr);
```

array_unshift() : it will add the element to the array at first / front

array_shift() : it will remove 1st element of an array

array_push() : it adds element to the array at end.

~~array~~ explode() : this fn divides the array element based on i/p value

ex : explode("delimiter", "given string");

→ $a = "welcome";
$arr = explode('c', $a); → Array([0] => wel [1] => ome)
print_r($arr);

Ex : $str = "welcome to php";
$arr = explode(" ", $str);
print_r($arr);

implode() : it combines array elements as a string.

Ex : $s = array(10, 20, 30);
echo implode($s); → 102030.

list() : By using this fn we can assign array elements into variables

ex : list($a, $b, $c) = array(10, 20, 30);
echo $b; → 20
" $b; → 10.

in_array() : this fn checks wether i/p value is available/not.

Ex : $arr = array(10, 20, 30);
echo in_array(10, $arr); → 1
echo in_array(300, $arr); → No o/p

array_search() : this fn search i/p value in an array if the value is found, it returns key of its element

ex : $arr = array(10, 20, 30);
echo array_search(30, $arr); → 2

array_slice() : this fn will returns part of an array

ex : $arr = array(11, 12, 13, 33, 22, 44);
print_r(array_slice($arr, 0, 3));
o/p : Array [0 → 11, 1 → 12, 2 → 3]

array_count_values() : it will returns how many times the specific element has been repeated.

ex : $arr = array(11, 12, 4, 14, 11);
print_r(array_count_values($arr));

→ merge : this fn will combine two given arrays

ex : $arr1 = array(11, 12, 13);
$arr2 = array(22, 3, 4);
$arrnew = array_merge($arr1, $arr2);
print_r($arrnew);

→ array_reverse() : this fn display the array elements in the reverse order

ex : $arr = array(11, 12, 13)
print_r(array_reverse($arr));

string functions

→ string is a collection of characters

1) strtolower()
2) strtoupper()

ex : $s = "ME"
echo strtolower($s); re

3) ucfirst() → converting 1st character to capital.
Ex: echo ucfirst($a); → hlelcome

4) ucwords() → converting 1st character of each word to capital. Ex: echo ucwords($a); → hlelcome To Iza.

5) ord() → it returns ASCII value for given character
Ex: echo ord('a'); → 97

6) chr() → it returns a character for given ASCII code value
Ex: echo chr(98); → b

7) str_repeat() → it repeats the string based on i/p no
Ex: echo str_repeat($a, 3); → how many times you want to repeat

8) trim() → eliminating the spaces from front & back
ltrim() → it removes left side spaces
rtrim() → it removes right side spaces
chop() → it is same like rtrim()

9) str_reverse() →

10) str_cmp() → it returns integer value i.e.,
0 → both strings are equal
1 → 1st string is greater than 2nd string
-1 → 1st string is less than 2nd string.
→ it considers case sensitive
strcasecmp() → it ignores case sensitive
Ex: $a = "hi";
$b = "Hi";
echo strcmp($a, $b); → 1

11) strlen() → it returns no. of characters in a given string

12) explode() → divide

13) implode() → combine

14) join() → It is same like implode

15) strpos() → it returns position & character of a given string.
Ex: $a = "welcome"
echo strpos($a, 'e'); → 1
echo strpos($a, 'e', 2) → 6

16) strrpos() → $a = "welcome"
reverse echo strpos($a, 'e'); → 6

→ str_pads() :
under this fn we have 3 constants
1) str_pad_left
2) str_pad_right
3) str_pad_both

Ex : $s = "hi";
echo str_pad($s, 20, '*', STR_PAD_BOTH);

→ count_chars() : It counts the characters based on the mode
→ It supports 3 modes i.e; 0, 1 & 2
0 → it returns all character ASCII code values 0-255
1 → it returns ASCII code for given character in string
2 → it returns ASCII code value for remaining character except the given string ASCII code values

Ex : $s = "hi";
print_r(count_chars($s, '2'));
(string)

mode 3 → it returns a character in ascending order

→ str_word_count() :
$s = "welcome to ira";
mode (0) → Display the no. of words in given string
o/p ⇒ 3
mode (1) →
o/p : Array ([0] ⇒ welcome [1] ⇒ to [2] ⇒ ira)
mode (2)
o/p : Array ([0] ⇒ welcome [8] ⇒ to [11] ira)

→ similar_text() : It returns similarity b/w two strings
Ex : $s1 = "hlo";
$s2 = "hi";
echo similar_text($s2, $s1);

→ str_split() : it split the string as array element based on the number.
Ex : $s1 = "it is array";
$arr = str_split($s1, 3); → Array ([0] ⇒ it [1] ⇒ is [2] ⇒ array)
print_r($arr);

→ strip_tags() : It strip (remove) html tags in given string.
Ex : $s1 = "welcome to
 isa";
echo strip_tags ($s1); → welcome to isa.

PHP to MYSQL :

→ In the php file & least inversion we are using mysqli extension & pdo (php data object) we connect to database
→ pdo is used extension we can connection php to 12 different database
→ By using mysqli extension we can connect to only mysql database

mysqli_connect() :

syntax :
mysqli_connect ($host, $username, $password)

BY - $servername = "localhost";
$username = "root";
$password = "root";
$conn = new mysqli ($servername, $username, $password);
if ($conn -> connect_error) {
die ("connection failed :". $conn -> connect_error);
}
echo "connected success";

connect() : this fn opens the connection to mysql database.
→ It requires 3 parameters :
1) servername 2) username 3) password.

connect_error() : It returns error description from the last connect

extensions of sql :

mysql → deprecated in latest version

mysqli

pdo

→ error no : 1045 → this error no will displayed when connection of database is failure

→ echo is faster than print bcoz echo will not return a value but print will return a value.

Ex :

```
$con = mysqli_connect("localhost", "root", "root");
if ($con == NULL){
   echo "not connected successfully". mysqli_connect_error();
}
else{
   echo "connected successfully";
}
```

mysqli_connect_error(); → description of connection failure

→ mysqli_~~correct~~query() will execute sql queries on the correction object

→ It takes two parameters:-
  1) correction object 2) sql query

Ex: $sql = "create dbase isadatabase123";
  if(mysqli_query($conn, $sql)
  {
   echo "correction succ";
  }
  else {
   echo "not succ". $.mysqli_correct_error();
  }

mysqli_affected_rows() :- It is going to display the no. of affected rows

Ex: $conn = mysqli_correct("localhost", "root", "root", "isadatebase123");
  $sql = "insert into tblemp values (100, 'isa')";
  if (mysqli_query ($conn, $sql) {
   echo "success". mysqli_affected_rows ($conn);
  }
  else {
   echo "not succ";
  }

→ insert into emp values (100, 'isa'); } two ways
→ insert into emp(empno) values (100); } to give values

mysqli_fetch_array() → this fn fetches the records from the connection object

→ It has 3 modes:-
  1) MYSQLI_ASSOC
  2) MYSQLI_NUM (index no)
  3) MYSQLI_BOTH

Ex: $sql = "select * from tblemp";
  $result = mysqli_query ($conn, $sql);
  $row = mysqli_fetch_array ($result, MYSQLI_ASSOC);
  printf ("%s %s \n", $row['empno'], $row['ename']);
  $row = mysqli_fetch_array ($result, MYSQLI_NUM);
  printf ("%s %s \n", $row[0], $row[1]);
o/p: 100 isa 100 isa

→ default mode is = MYSQLI_NUM
→ returntype is array datatype

mysqli_fetch_row() : this fn will return record from the connection object.

Ex: while($row = mysqli_fetch_row($result))
{
printf("%s %s %s\n", $row[0], $row[1]);
}

cookie's and session

→ cookie is a small amount of memory used by the web server at the client browser
→ cookie is created at the server side & stored at the client browser
→ If you requesting to the server for first time the cookie will attached to your request. through the response of the server the info will be saved in the cookie. for the second time request, the request will be send to the cookie. for the info. if the info will be available that will be response back.

→ cookie → will store small amount of data i.e., 2kB (depending on browser type & version you are using)
→ not secure
→ it stores only string values
→ session → will store large amount of data.
→ secure.
→ it stores different datatype values

→ Cookies are less secured bcoz it can be edited & deleted
→ cookie will store only string datatype values
→ cookie will store only less amount of data
→ In php cookie will be used before HTML tags
→ there are 2 types of cookie's:
1) InMemory Cookie (temporary cookie) : it is saved in browser process memory. once, the browser is close the cookie will be erased / deleted
2) persistent (permanent cookie : it is saved permanently in the local hard disk.

→ once life span has been completed. automatically the os will going to delete

→ the info will not pass directly from one page to another. In this situation we use concept of cookie & session.
→ HTTP protocol is a state less protocol.
→ state means maintaing state of value throughout the website
→ by using set cookie() function we create a cookie.
eg: setcookie(cookiename, cookievalue)
setcookie("cookiename", "isa")
→ by using $_COOKIE super global variable we can access value from the cookie
eg: $val = $_COOKIE['cookiename']; echo $val;
permanent
→ setcookie("cookiename", "isa", time() + 3600); 1x60x60

session:
→ session can store all datatype values.
→ session's are more secure. It can't be deleted/updated
→ In session, we can store unlimited of data.
→ by using $_SESSION super global variable we can store a value inside the session & we can read the value
→ $_SESSION is known as "Associative Array"
eg: $_SESSION['sessionname'] = value;
$_SESSION['sessionname'] = 'isa';
echo $_SESSION['sessionname']; → isa.
→ by using session_start() function we start the session after start we have to create a session by using
$_SESSION
→ by using session_destroy() function we destroy a session manually
→ If the application is not interactive with the webserver for 20 minutes automatically the session will expire. but we can change session time out

Ex: <?php 1st page

```
setcookie('cookieira', 'isa', time()+1);
?>
<html>
<body>
<form action="second.php">
<input type="submit">
</form>
</body>
</html>
```

2nd page

```
<?php
echo $_COOKIE['cookieira'];
```

1st page

Ex: <?php

```
session_start();
$_SESSION['sessionid'] = 'isasession';
```

2nd page

```
<?php
session_start();
$_SESSION['sessionid'];
```

1st page

Ex: <form action="second.php" method="get">

```
Name <input type="text" name="name"><br>
Age <input type="text" name="age"><br>
<input type="submit">
</form>
```

2nd page

```
<?php
if($_POST['name'] || $_POST['age'])
{
echo "welcome to". $_POST['name']. "<br>";
echo "ur age". $_POST['age'];
}
?>
```

Ex: echo $_SERVER['SERVER_NAME']; → localhost

echo $_SERVER['PHP_SELF']; → /second.php(web page name)

echo $_SERVER['HTTP_POST']; → localhost:8000

2nd page

<?php
echo $_cookie
['cookieira'];

">
">
>

">";

lhost

d.php(web page name)
ost : 8000

Functions

-> A group of stmts under one name is called function.
-> Advantage is code reusability
syntax for creating = function:

```
function function_name(parameters)
{
logic;
return value;
}
```

(function function_name(parameters) → function prototype / signature.)

-> PHP support 2 types of fn's:
1) predefined fn's 2) user-defined fn's
-> In php function names are in case insensitive. It can start alphabets / underscore.

parameters: these are the i/p to the fn. while calling the fn we have to pass values to the parameters.
-> no. of parameters & values should be balanced.
& position are important

Ex:
```
function f1($user, $name)
{
echo 'welcome to --'.{$user." ".$name};
}
f1('Hlo', 'ira'); → Welcome to -- hlo ira.
```

default arguments:
-> passing default values to the positional arguments

-> from 7.0 version of php we can declare datatype before the variables.

Ex:
```
function f1($age=10)
{
echo 'welcome'.$age;
}
f1() → 10
f1('ira'); → ira
```

variable length arguments

→ we can pass n no. of values to the arguments
→ this topic available in c, c++, python, php.
→ we can write variable length args by using three dots (...) before the parameter.
→ this fuction is called as variadic function.
→ we can pass positional & variable length args simultaneously but variable length arg must be the last arg.
→ the returntype of this fn is "array" datatype

Ex:
```
function sumnumbers(...$s)
{ $sum=0;
  for(int $i=0; $i<=$s.length; $i++)
  {
   $sum=$sum+$i;
  }
  echo ~~$sumnumbers~~; $sum;
}
sumnumbers(1,2,3,4); → 10
```

```
$len=count($s);
$i<$slen;
```

return statement

→ function will return a value.
→ return keyword should be the last stmt inside fn
→ after return stmt if we write any stmts those stmts will not be consider

Ex:
```
function f1($a,$b)
{
  return $a+$b;
}
echo f1(5, "5 days"); → 10 & warning for nonnumerical value encountered
```

sol^n is:
```
declare (strict_types=1);
```

→ by using strict mode the string value will not be converted to the integer & we get exception

→ **class:**
→ class is a collection of data members & member fn's
→ class is a logical representation of data
→ class doesn't hold any value.

syntax for creating a class :

```
class Classname
{
  variables;
  constructors;
  methods
}
```

object :

→ object is a physical representation of data
→ It is an instance of a class.
→ with the help of object we can access variables & methods of a class.

syntax :

```
$reference_variable = new classname();
```

methods

→ methods are used to logic/ functionality
→ By using function keyword we create method in php

syntax :

```
access modifiers function methodname()
{
statements;
}
```

→ method name can be anything
→ method may/ may not return a value.

constructor :

→ It is used to initialize values to parameters.
→ the name of the constructor should be __construct()
→ once, we create object for class the constructor will be called.
→ constructor will not return a value.

Ex :

```
class Ex
{
 var $i = 10;
public function __construct()
{
 echo 'Hlo constructor';
}
function m1()
{
 echo 'ion method';
}
}
```

```
$obj = new Ex();
$obj -> m1()
$obj -> $i;
```

o/p : Hlo constructor
ion method
10