Module - III

PHP

* Stands for Hypertxt preprocessor.
* Integrated lang (i.e) there is no need for compilation.
* faster than other scripting lang like ASP & JSP
* server-side scripting lang & can be embbed into HTML.
* obj-oriented lang & open-source scripting lang.
* Simple & easy to learn lang.
* php files can contain txt, html, css, js & php code.
* php files have extension '.php'.
* Generate dynamic pg contnts.
* Can create, open, read, write, dlt & close files on Servr & can add, dlt, modify data in yr database.
* runs on various platforms & supports a wide range of databases.
* Features —
  a) performance
  b) opn source
  c) Familiarity with s/x.
  d) Embedded.
  e) platform independnt
  f) Security
* eg :-

* <html>
<body>
<?php
echo "Hello world!";
?>
</body>
</html> → Hello world!

* <?php
$txt = "php";
echo "I love $txt";
?>
→ I love php.

* echo → display o/p of parameters that are passed to it.
* A variable starts with $, followed by the name of variable.
* case sensitivity :
keywords (if, else, while, echo, etc), cls, fns, user defined fns are not case sensitive.
eg:- <?php
echo "hi"; → hi
Echo "Hi"; → Hi
ECHO "helo"; → hello.
?>

All variables names are case sensitive.
eg:- <?php
$color = "red";
echo "car is " . $color . "
";
echo "car is " . $COLOR . "
";
echo "car is " . $Color;
?>
→ car is red
car is
car is

* for str concatination use '.' opr.

[1] php cmnts =

used to describe any line of code so that other developer can understand the code easily.
2 types —
a) single line cmnts :-
// or #
b) multi line cmnts :-
/* hello ak */

→ Server side prgming =

* They are run on the server. & reduces the amount of bugs
* scripts are execute in the Srver & snd back html file to the web browsr.
* ensure high-level security to the source code.
* eg: php, ruby, perl

→ web servr software =

* Fn of web srvr is to store, process & delivr web pgs to clients.
* Commun uses HTTP client.
* eg: google web srvr.
  Apache tom cat.
* 3 Softwares —

a) WApp server:
   W- windows component
   A- Apache
   P- postgresql
   P- Php

b) LApp server:
   L - Linux component.
   A - "
   P - "
   P - "

c) MApp server:
   M - Mac component
   A - "
   P - "
   P - "

→ Include php in html =

1) In <head>
2) In <body>
3) Through ex script file (<script src="">
4) we can use include function — to import php script frm outside the html file.

eg:-
```
<html>
<head>
<?php
    Echo " Hai <br> ";
?>
</head>
<body>
<?php
    echo " Php ";
?>
</body>
</html>
```

```
<?php
    echo "Hi";
?>
```

(can also written as)

I

II
```
<script language="php">
    print "Hi";
</script>.
```

III
```
or <? ... ?>
```

```
<body>
<?php
    $mydate = getdate();
    echo "<h3> Today is $mydate[weekday], $mydate[month] $mydate[mday], $mydate[year] </h3>";
?>.
```

→ Today is ~~April~~ Monday, April 22, 2019.

[2] **Php Datatypes = (8)**

* Supports 8 primitive datatypes that can be categorized in 3 types —

I **Scalar Types: (4)**

* Hold only single value.

var_dump() → returns datatype & value.

boolean eg
```
$error = True;
var_dump($error);
```
→ bool(true)

Scalar → boolean, integr, float, Str

a) **Boolean :**
* Simplest datatype wrk like switch
* hold only 2 value - True (1) or False (0).
* used with condi tional (s).
* If the condition is crt, returns T otherwise F.

b) **Integer :**
* numeric data with a -ve/+ve sign
* hold only whole no.
* eg :-
```
$a = 123;
echo "no.:", $a;
```
→ 123

c) **Float :**
* no. is a non with a decimal point.
* eg :-
```
$a = 1.234;
var_dump($a);
```
→ float(1.234).

d) **Str :**
* non-numeric datatype.
* enclosed either within single quotes / double quotes.

II **Compound Datatypes : hold multiple values**

array, obj (2)

var_dump → return type & its value

a) Array :

* Compound (D) that can store multiple values of same (D) in a single variable
* eg: $likes = array("A", "B", "C");
* eg :-
$clrs = array("red", "green", "blue");
var_dump($clrs);
→ array(3) { [0] => String(3) "Red"
[1] => String(5) "green"
[2] => String(4) "blue" }
* defined as indexed collection of data values.

(b) Obj :

* Are the instance of user-defined class that can store both values & fns.
* They must be explicitly declared.
* Every obj has (pro) & methods corresponding to those of its parent cls.

III Special Types : (2)

Resource

* Are not exact (D) in php
* Basically, these are used to store some () calls.

NULL

* Null is a spl (D) that has only 1 value : NULL
* It is a spl variable holding a refrnce to an ex resource.
* eg: //open a file for reading
$han = fopen("note.txt", "r");
var_dump($han);

* There is a convention of writing it in capital letters as it is case sensitive.
* eg:-
$a = NULL;
var_dump($a);
→ NULL

[3] Php variables =

* In php, variable is declared using a $ sign followed by var_name.
* Syn ⟷ $varname = value;
* Rules —
  - A var must start with a $ sign.
  - can only contain alpha-numeric char & _ (A-z, 0-9, _).
  - varname must start with a letter | _.
  - cannot contain spaces.
* eg :-
<?php
$x = 5;
$y = 2;
$z = $x + $y;

echo $z;
?> → 7

→ Variable scope =

* It is the portion of the prgm within which it is defined & can be accessed.
* 3 types —

a) Local variable :
- declared within a ()
- They hve their scope only in that particular () in which they are declared.

b) Global variable :
- declared outside the ().
- can be accessed anywhere in the prgm.
- To access the global var within a (), use the GLOBAL keyword bfre the var.

c) Static variable :
- It is a feature of php to dlt the var, once it completes its execution & only is freed.
- Sometimes we need to store a var evn after completion of () execution.
- we use static keyword bfre the var to define a var, & this var → static var.
- This var exist only in a local function
- you can declare this by STATIC keyword.

☆ eg:- eg for local var

```
<html>
<body>
<?php
$x = 10;
print "outside fun : $x . <hr>";
function assignx() {
    $x = $x + 1;
    print "inside fun : $x . <hr>";
}
assignx();
print "outside fun : $x";
?>
</body> </html>.
```

→ o. fun 10
i. fun 11
o. fun 10

☆ eg for global var :-

```
<?php
$x = 10;
print "outside fun : $x . <hr>";
function assignx() {
```

```
    global $x;
    $x = $x+1;
    print "Inside fun : $x . <br>";
}
assignx();
print "outside fun : $x";
```

→ o. fun : 10
i. fun : 11
o. fun : 11

☆ eg for static variable :-

```
<?php
    function fact ($num){
        static $f=1;
        $f = $f * $num;
        return $f;
    }
    for($i=1; $i<=3; $i+=1)
        echo $i, "!" = fact($i), "<br>";
?>
```

→ 1! = 1
2! = 2
3! = 6

* Variable types :

```
$flag = TRUE;       // boolean
$str1 = "hello";    // str
$str2 = 'hello';    // str
$num = 12;          // int
```

* Type juggling :

Automatically converts values frm 1 type to anthr whnevr required.
Also → implicit conversion.

eg :-

```
<?php
    $a1 = "25 rupees";
    $a2 = 15;
    $a3 = "10";
    $total = $a1 + $a2 + $a3;
    echo $total;
?>
```

→ 15

* Type casting :

process of explicitly converting values frm 1 datatype to anthr is called Type casting / explicit conversion

eg :-

```
<?php
    $bool_false = false;
    var_dump ($bool_false);   → false
    $int_val = (int) $bool_false;
    var_dump($int_val);   → 0.
```

$var = $Foo + 1.3
2 = $Foo + 1.3
∴ $Foo = 2 + 1.3 = 3.3

implicit conv eg:

```
$var = "0";   // is a str [ASCII→48]
$var += 2;    // var is now an int → 2
$var = $Foo + 1.3;  // Foo is now a float → 3.3
$var = 5 + "10Hello";  // var is int → 15
echo $var;
```

→ 15

explicit conv eg

```
<?php
$float_val = 10.9;
var_dump (float_val);   → float(10.9)
$str = "10.9 float";
var_dump ($str);   → 10.9
```

→ Pre-defined var =

* Are accessible frm anywhere within the executing script & provide you with info.
* Some pre-defined var are also → Superglobals
  eg → $_GET → through GET Method
  $_POST → "
  $_COOKIE → "

→ Defined constants :

* constnt is a value that cannot be modified throughout the execution of prgm

* To define a constnt, use define(), inside parenthesis the name you have chosen for the constnt & its value for the constnt.
* eg → define ("PI", 3.141592);

[4] Output Echo & print =

Echo	print
* statmnt used to display the o/p & used with/without parnthesis.	statmnt used as an alternative to echo to display o/p.
* does not return any value	Always returns an intgr → 1
* can pass multiple str separated by , (comma)	cannot pass multiple argumnts.
* Fastr than print	Slower than echo

[5] PHP operators =

* Symbol used to perform opr on oprands.

I Arithmetic (o):

* used to prform arithmetic opr. like +, -, *, /, %, **.
* eg → $a + $b;
  $a * $b;

II Assignmnt (o):

* used to assign value to diffrnt var.
  = , +=, -=, *=, /=, %=
* eg → $x += $z;

III Logical (o):

* used to perform bit-level opr on oprand.
* !, &&, ||, Xor, And, Or.
* eg → $x || $y;
  $x Xor $z;
* combine othr bodean values to produce a new bodean values

IV comparison (o):

* used for comparison in PHP allow us to compare the values of 2 exp / values.
* == (equality), === (identity → T if 2 oprnd hve same value & same datatype), != , !== (non identity), <, >, <=, >=, ?: (Ternary condtn oprator)
* eg → "123" == 123 → T
  "123" === 123 → F.
  TRUE == "Honk" → T
  " === " → F

V Incrementing Decrmnting (o):

* These (o) add / sub frm a variable
* Supports pre & post incrmnt & decremnt (o).
* ++$x (pre-i),
  $x++ (post-i),
  --$x (pre-d),
  $x-- (post-d)
* eg → $x = 5;
  print $x++; → 5 (x=5)
  " ++$x; → 7 (x=6)

VI String (o):

* Are 2 str (o).
* . → concat "
  $x . $y;
* .= → concat + assignmnt
  $x .= $y;
  (appnds $y to $x)

VII Bitwise (o):

* Allow evaluation & manipulation of specific bits within an int.
* & → And → $x & $y
  | → or → "
  ^ → Xor
  ~ → Not
  << → shift left
  >> → shift right

[String-ൽ \ ഉള്ള ഒരു word ഉണ്ടെങ്കിൽ അത് print-ൽ വരില്ല]

[6] Control-Flow Structures =

* They perform sets of basic instr.. conditionally based on values / exp.
* Also → Branching str...

I Branching / conditional (S) : (decision making)

perform different actions based on the rslts of a logical condi'n at run time.

a) **if (S) :** used to execute a blk of code only if the specified condi evaluates to T.

~~eg: <?php~~

b) **if...else (S) :**

~~c) if...elseif...else (S)~~

eg:-

```
$t = Date ("H");
if ($t < "20") {
        echo "gd day";
}else {
        echo "gd ngt";
}
```

Date ("H") → crnt hr in 24-hr format (00 to 23)

d → 01-31	y = 23	A → AM/PM	N = 1,2,...
D → Sun-Sat	Y = 2023	a → am/pm	
m → 01-12	H → 00-23 (hr)	l → sunday—saturday	
M → Jan-Dec	i → 00-59 (min)	F → January-December	
	s → 00-59 (sec)		

date()

[In 24-hr 20 means 8:00pm (now the time is 7:00p so it returns gd ngt]

* date ("D") → Textual reprsn of Day (sunday ...).
* d → Day of month (01 - 31).
* F → textual reprsn of moth. (January)
* M → short textual repr'n of moth (Jan)

c) **if... elseif ...else (S) :** To check 2/more condi-

```
$t = date ("H");
if ($t <"10") {
        echo "gd morng";
}elseif ($t < "20") {
        echo "gd day";
}else {
        echo "gd night";
}
```

d) **switch (S) :** standard form of if-elseif.

```
$clr = "red";
switch ($clr) {
    case "red" :
```

```
    echo "fav clr is red";
        break;
    case "blue":
        ...
        ...
    default:
        echo "no favclrs";
}
```

II) Loops : (Iterative (s))
used to execute the same block of code again & again until a certain condi- is met.

a) while loop :

```
eg→ $i=1;
    while ($i <=3) {
        echo "Num is ". i . "<hr>";
        $i++;
```

b) Do-while :

```
eg→ $i=1;
    do{
        echo "Num is ". $i . "<hr>";
        $i++;
    }
```

while } entry
for } c·loop

Do-while → exit c·loop

```
    while ($i<=3);
```

c) For loop :
used to execute a block of code for certain no of times.

```
Syn→ for (initial-exp; testing-condition-check;
        loop-end-exp) {
            (s);
    }
```

eg → same as while / do-while with for loop.

d) foreach loop :
used to iterate over arrays.

```
Syn→ foreach ($array as $value) {
        (s);
    }
eg→ $clrs = array("red", "green", "orng");
    foreach ($clrs as $val) {
        echo $val . "<hr>";
    }
```

III) Break & continue :

* Break cmnd exits the innermost loop

construct that contains it.

* continue cmnd skips to the end of the cnt iteration of innermost loop that contain it.

* eg for break →

```
for($x=1; $x<10; $x++) {
    if($x % 2 != 0)
        break;
    print $x.;
}
```

→ ng to print

* eg for continue →

```
for ($x=1; $x<10; $x++) {
    if($x % 2 == 1=0)
        continue;
    print $x.;
}
```

→ 2468

[7] User-Defined ()s =

* A () is a self contained block of code that performs a specific task.

I Creating & Invoking ()s :

* $fx → function fun_name ($arg1, $arg2...) {
    S1;
    S2;
    ;
  }

* Declaration of user-defined () start with the word function, followed by the () name you want to create.

* A () has 4 parts:
  1) function word
  2) function name you want to give
  3) ()'s paramtr list.
  4) () body.

[() name → made up of letters, _, & no~]

* eg →

```
<?php>
    function myfun() {
        echo "Today is ". date('l',
                mktime());
    }
    myfun();
?>.
```

→ Today is wednesday.

* echo "Today is ". date('$I');
  → Today is wednesday (current day in full).

* $time = mktime(12,0,0,1,1,2023);
  echo $time;

mktime(hr, min, sec, month, day, yr);
- returns the unix timestamp corresponding to date & time
- In this eg → January 1, 2023, 12:00:00
- ∴ it returns → 1672531200

II Fns with paramtrs & arg:

* A paramtr is jst like a variable.
* paramtr enable fns to accept input values at run time.
* eg →

```
function getsum($n1, $n2) {
    $sum = $n1 + $n2;
    echo "$sum";
}
getsum(10,20);
```

→ 30.

III optional parmetrs & Default values:

eg →

```
function custfont($font, $size=1.5){
    echo "<p style=\"font-family:
    $font; font-size: {$size}em;
    Hello world </p>";
}
```

```
custfont("Arial", 2);
custfont("Times"); → here size=1.5
```

IV Returning values frm a fn():

* A fn can return a value back to the script that called the fn using the return statemnt.
* eg →

```
function getsum($n1, $n2) {
    $total = $n1 + $n2;
    return $total;
}
echo "Sum is ". getsum(10,20);
```

→ sum is 30.

* A fn cannot return multiple values. However you can obtain similar rslts by returning an array.

V passing arg by value & Refernce:

* In PHP, you can pass arg to a fn in 2 ways. → by value
  → by reference
* By default, fn arg are passed by value so that if the value of the arg within the fn is changed & does not get affected outside of the fn.

* eg →

```
function passvalue ($x) {
    echo "$x befre altering = $x";
    $x+=10;
    echo "$x after altering = $x";
}
$x = 20
echo "$x bfre passing to () = $x";
passvalue($x);
echo "$x after returing frm
        () = $x;
```

→ $x bfre passing to () = 20.
$x befre altering = 20.
$x after altering = 30
$x after returing frm () = 20.
(outside ())

* In PHP, an arg by reference is done by prepending (ampersand) '&' to the arg name in () definition.

VI Functions & variable scope :

* you can use local variable inside a () without worrying about their effects on outside world.

```
function localvar () {
* eg →   $d = 10;
        echo "$d within ()";
        echo "$x within ()";
    }
    $x = 20;
    localvar();
    echo "$x outside ()";
    echo "$d outside ()";
```

→ 10 within ()
within () (no value)
20 outside ()
outside () (no value)

* eg → using global declaration,

```
function localvar () {
    global $x;
    $d = 10;
    echo "---";   → 10
    echo "---";   → 20
}
$x = 20;
localvar ();
echo "---";   → 20
echo "---";   → x
```