Module - IV

Working With PHP

[1] **Arrays =**

- used to store multiple values in a single variable.
- Each value in an array has a position → index.
- eg –

```
$d = array ("m", "key", "moni");
$d [0] = m
$d[1] = key-
$d[2] = moni
```

* 3 types of arrays :-

a) Numeric (A) / indexed (A) : An (A) with numeric key

b) Associative (A) : An (A) where each id key is associated with a value.

c) Multidimensional (A) :
An (A) containing 1 / more arrays within itself.

~~$ Numeric Array:~~

→ Creating Arrays =

I By direct Assignment —

```
<?php
    $col [0] = "Red";
    $col [1] = "green";
?>
```

II By array() construct :
creates a new (A) from the specific of its elements & associated keys.

* $col = array () → creates an (A) with no elements.

* $fr = array ('apple', 'orange');
↓ can be written as

```
$fr [0] = 'apple';
$fr[1] = 'orange';
↓ "
$fr[] = 'apple';
$fr[] = 'orange';
↓ "
$fr = array (0 => 'apple', 1 => 'orange');
$fr[1] = orange.
```

(or)

```
$fr = array ('red' => 'apple', 'yellow' =>
                                 'orange');

$fr['red'] → apple.
```

III By fns returning arrays :

$num = range (1,5) → 1, 2, 3, 4, 5

↓ same as
$num = array (1,2;3,4,5);

★ **list() construct :**
used to assign several array elements to variables in succession.
eg :-
```
$fr = array('app', 'oran');
list($redfr, $orangefr) = $fr;
```
→ assign the str 'app' to $red fr & 'oran' to $orangefr.

⇒ **Multi-Dimensional Arrays =**
* array() construct used to create a multidimensional arrays.
* eg –

```
1) $meals = array(
     'brfast' => array('idly', 'chapati'),
     'lunch' => array('rice', 'biriyani'),
     'Snack' => array('bread', 'nuts'));

   print $meals['lunch'][1] → biriyani.
2) $lunch = array(
     array('chickn', 'beef'),
     array('egg', 'bread'));
```

print $lunch [1] [1] → bread

→ **Deleting from arrays :**
1st call unset().
eg :-
```
$arr[0] = 'wanted';
$arr[1] = 'unwanted';
$arr[2] = 'again';
unset($arr[2]);
print $arr;
  → Array (
      [0] = wanted
      [1] = unwanted
    )
```

⇒ **Array operators :**
* $a + $b – union
* $a == $b – equality (T → same value)
* $a === $b – identity
* $a != $b – Inequality
* $a !== $b – non- identity.

⇒ **Array Functions =**

1) array() → creates an array
eg → normal eg of array.

print_r()
to print human readable info about array
(A)-sngs str assignment use print_r, to get the value if we use print.

2) array_combine () : creates an (A) by using the elements frm 1 keys (A) & 1 values (A).

```
eg → $f = array ("ptr", "Ben");
     $a = array ("35", "37");
     $c = array_combine ($f, $a);
     print_r($c);
→ Array ([ptr] => 35 [Ben] => 37.)
```

3) array_count_values(): counts all values of a(A)

```
eg = $a = array ("A", "cat", "A");
     print_r( array_count_values($a));
  → Array ([A] => 2 [cat] => 1 )
```

4) array_diffs () : compare (A) & returns the diffr.

```
eg = $a1 = array ("a" => "read", "b" => "green",
                  "d" => "yellow");
     $a2 = array ("e" => "red", "f" => "green",
                  "g" = "blue.");
     $r = array_diff ($a1, $a2);
     print_r($r);  → Array ([d] => yellow)
```

5) array_fill() : fill an array with values

```
Stx = array_fill ($startindx, $num, $value)
eg = $a1 = array_fill (3, 4, "bl");
     print_r ($a1);
→ Array ([3] => bl [4] => bl [5] => bl [6] => bl)
```

6) array_pad() : inserts a specified no. of items with a specified value to an array

```
Stx → array_pad (array, size, value)
eg → $a = array ("red", "green");
     print_r (array_pad ($a, 5, "blue"));
  → Array ([0] => red [1] => green [2] => blue
           [3] => blue [4] => blue )
```

7) array_pop () : dlts last element of an (A).

```
eg = $a = array ("r", "g", "o");
     ~~print~~ array_pop ($a);
     print_r ($a);
  → Array ([0] => r [1] => g )
```

→ Iterating through the array :

2 ways { ① foreach loop ② Traversing (A) using list() & each ()

[2] Php Strings =

```
* $str1 = "hello";
  $str2 = "AK";
  print $str1 . $str2;      → hello Ak
```

* concatination with assignment → $any .= $new;

* <u>heredoc syntx</u> :
- In addition to single & double quotes, Php offers anthr way to specify a str → heredoc s/x.
- This s/x turns out to be extremely useful for specifying large chunks of variable mixed txt.
- operator is <<< followed by an identifier
- eg →

```
<?php
    $name = "max";
    $str = <<< Demo
       hello $name <br>
       This is a
              msg for you.
       Demo;
          echo $str;
    ?>
```

→ hello max.
  This is a msg for you.

→ <u>str functions</u> :

1) <u>strlen()</u>:
   echo strlen("Ansar"); → 5

2) <u>strpos()</u> : search for a charctr / txt within a str.
   If a match found, this () returns the character position of 1st match otherwise returns f.
   eg :- echo strpos("Ansar good", "good");
   → 6

3) <u>strstr()</u> : finds the occurnce of a str inside anthr str.
   s/x → strstr($str, $search, $before_search)
   eg → echo strstr("Hello Ak", "Ak");
   → Ak.

4) <u>strcmp()</u> : compares 2 str & return
   → 0 if str1 = str2
   → <0 if str1 < str2
   → >0 " " > "
   eg → echo ("Hello ak", "Hello ak"); → 0
   echo ("Hello ak", hello ak"); → -1
   (h has highest ascii so h > H).
   echo ("hello ak", "Hello ak"); → 1

5) <u>substr()</u> : returns a part of str.
   s/x → substr(str, start, length);
   eg → echo substr("hello world", 6);
   → world

6) <u>strreplace()</u> : replaces a part of a

Str with anthr str.
str → substr_replace (str, replcmnt, start, length)
eg → echo substr_replace ("Hello world", "Earth", 6);
→ Hello Earth.

7) strtolower(): conurt a str into l.case.
eg → echo strtolower ("~~[illegible]~~HELLO");
→ hello.

[3] passing info b/w pgs =

* php will catch the var entered frm 1 pg to nxt.
* Php is gd in the form handling technology that is data passing ().
* In php GET & post method are used to info passing.
* 3 ways to pass info –
1) passing variables b/w pgs using URL:
Here values are visible to the user & not secure way to transfer sensitive data like password, user details, etc.
2) passing var b/w pgs using cookies:
cookies are stored at the user end & values can be passed b/w pgs using php.
3) passing var values b/w pgs using session:
secured way to pass var b/w pgs. In a member login system the user details are verified & once found crt, a new session is created with user id of the member & value stored at srvr end.

I Using URL:
GET & POST methods in HTTP to create dynamically generated pgs & to handle form data.

(1) HTTP GET Method:
* passes arguments frm 1 pg to nxt as part of URL.
* <form action = "" method = "">.
action → URL specifies where to snd the form-data when a form is submitted.
method → GET & POST specifies the HTTP method to use when sending ~~formdata~~
* eg → <form action = "index.php" method ~~[illegible]~~ "GET">
<label for=name> Enter name: </label>

```
<input type="text" name="name" id="name">
<input type="submit">
</form>
<?php
    $name = $_GET["name"];
    echo "your name is : $name";
?>
```

(URL-ൽ key=value കാണിക്കും)

* Should not use HTTP GET method in-
  - You are updating a data source such as dB / spreadsheet.
  - You are dealing with sensitive info like password / credit card details.
  - You hve large amount of data.
  - Your form contains a file upload control.

(2) <u>HTTP POST method</u> :

* when you snd data ~~from~~ from a form to the srvr using Http post, the ~~form~~ form data is snt transparntly is wht is → <u>http headers</u>,
* eg → Same eg as GET method, ~~instead~~ use post instead of GET.
  (url-ൽ key=value കാണില്ല.)

☆ <u>GET</u>	<u>POST</u>
* Info are visible to everyone	not visible
* Displayed in title bar	not displayed
* limited amount of info to be snd (max 2k chars)	8mb of info to be snd.
* not used for sensitive info.	used.
* possible to bookmark the pg	not possible

☆ <u>URL</u> : (eg)

http://localhost/sports.php?sport=cricket & submit=select.

(action: http://localhost/sports.php)

? → denote the following chars constitute a GET str called <u>query str</u>

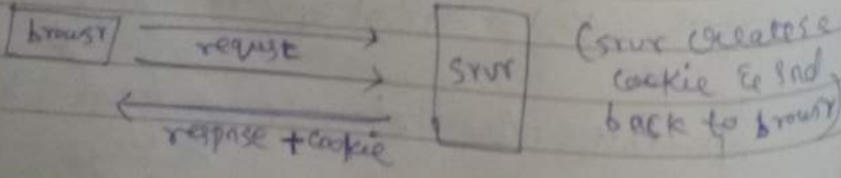
sport = cricket → a var name, an equal to & matching value.

$$\begin{tikzcd}[column sep=small]
? \arrow[d] & \text{sport} \arrow[d] & = & \text{cricket} \arrow[d] & \& \arrow[d] & \text{submit} \arrow[d] & = & \text{select.} \arrow[d] \\
\text{start of params} & \text{key1} & & \text{value1} & \text{separator.} & \text{key2} & & \text{value2}
\end{tikzcd}$$

eg → https://www.domain.com/page?key1=val1 & key2 = val2.

II HTTP cookies :

* whn user makes an http req to yr php prgm through a web browsr, the srvr runs the php intrpreter. The rsult of prgm is a web pg that is snt back to user's browsr.
once yr prgm snds a pg to the user, the php prgm shut dwn bcz its work is dne. Each req frm the user is seen as an entirely new transaction. This behaviour → stateless protocol.
* cookies are small pieces of info that are stored by a user's web browsr.
* used for authentiⁿ, storing site prefrnces, shopping card contnts, etc.
* consist of 1/more name-val pairs containing bits of info.
* cookie without an expiration date exist until the browsr terminates. [illegible] ex. date [illegible] browsr-ൽ store [illegible].
* It is created at srvr side & saved to client browsr.
*



* **php set cookie () :**
  * used to set cookie with http respnse.
  * once a cookie is set, you can access it by $_COOKIE superglobal var.
  * eg → setcookie ("username", "amer akbar", time () + 30 * 24 * 60 * 60);
  * $_COOKIE → used to retrieve a cookie val.
    eg → `<h3> Hello <?php echo $_COOKIE ["username"]; ?>`
    → Hello amer akbar.

$[30 \underset{\text{days}}{\downarrow} * 24 \underset{\text{hrs}}{\downarrow} * 60 \underset{\text{min}}{\downarrow} * 60 \underset{\text{sec}}{\downarrow}] \rightarrow$ cookie expire date.

* to remove cookies — simply call setcookie()
  eg → setcookie ("username", " ", time() − 3600);
  (ex. time in sec → 3600 sec = 1 hr)

III sessions :

* used to store & pass info frm 1 pg to anthr temporarily (until user close the website)
* mainly used in shopping websites where we need to store & pass cart info.
* creates a unique id for each browser

to recognize the user & avoid conflict b/w multiple browsrs.

* session_start() → used to start the session
* session var are ~~got~~ set with the php global var : $_SESSION.
* session_start() must be the very 1st thing in yr doc.
* eg. →

```
<?php
    session_start();
?>
<html>
<body>
<?php
  //set session var.
    $_SESSION["favclr"] = "green";
    $_SESSION["favani"] = "cat";
    echo "s.var are set";
?>
</body></html>
```

* Destroy a session : (2 ways)

session_unset()	session_destroy()
• If you want to destroy a single session var.	In a single call can destroy all the session var
• eg → `<?php unset($_SESSION['counter']); ?>`	`<?php session_destroy(); ?>`

→ Php headers :

* Are bits of info that are snt to a comp bfre anything else like a webpage is snt.
* whn we view a website in a browsr, we will nvr see these headers.
* They come bfre the webpg & all the browsr will display the contnt of webpg.
* 4 basic types of http headers —
  a) General
  b) Request
  c) Response
  d) Entity.
* header() use to snd raw, arbitary HTTP headers.

Stx → header(str, replace, http_response_code)

1) header('Location:')

eg → `<head>`

```
<?php
```

```
    header ('Location://www.google.com');
 exit ;
 ?>                          (pg load ony so it directly
 </head>.                       go to google)
```

2) header ('Expires:')
3) " ('cache-Control:')
4) " ('Content-Type:')
5) " ('Refresh:')