10. Listout PHP style tag.

→ There are ④ ways to add PHP style tag :-

① Canonical PHP tag → <? PHP ... ?>

② short-open (SGML-style) tag → <? ... ?>

③ ASP-style tag → <% ... %>

④ <script language = "PHP"> ... </script>

Long Questions :=

1. Explain Array in detail.

→ PHP array is an ordered map (contains value on the basic of key). It is used to hold multiple values of similar type in a single variable.

Array types :=
1. Indexed Array - numeric array
2. Associative Array
3. Multidimensional Array

1. Indexed (Numeric) Array :=

PHP Array is represented by an index number by default. All elements of array are represented by an index number which starts from 0. Index array can

store numbers, string or any objects.

- There are (2) ways to define :=

(1) $size = array ("Big", "Medium", "short");

(2) $size[0] = "Big";
$size[1] = "Medium";
$size[2] = "short";

- Traversing Indexed Array :=

We can easily traverse array in PHP using foreach loop.

e.x
```
<? PHP
    $size = array (1, 2, 3);
    foreach ($size as $s)
    {
      echo $s;
    }
?>
```

output := 123

- Changing Index of an Array :=

By default index starts from 0 but if we want to change the index we can change it using symbol "⟹".

e.x $sub = array (2 => "java", "wd");
print_r ($sub);

output :: Array ([2] => java [3] => wd)

- Count Length of PHP Indexed Array :=

count() function which returns length of array.

e.x $size = array (1, 2, 3);
echo count ($size);

output :: 3

2. Associative Array :=

PHP allows you to associate name/label with each array elements in PHP using => symbol.

- Define :=

① $salary = array ("jk" => 50000, "dk" => 60000);

② $salary ["jk"] = 50000;
$salary ["dk"] = 60000;

- Traversing PHP Associative Array :=

foreach loop is useful to print associative array.

e.x <? PHP

```
$salary = array ("jk" => 40000, "dk" => 50000);
foreach ($salary as $name => $val)
{
    echo $name."=".val."<br>";
}
?>
```

output :=

jk = 40000
dk = 50000

3. multidimensional Array :-

Also known as Array of Arrays. It allows you to store tabular data in array. Array can be represented in the form of matrix which is represented by row * column.

e.x

```
$emp = array
( array (1, "jk", 80000),
  array (2, "dk", 90000),
  array (3, "lr", 85000)
);

for ($row = 0 ; $row < 3 ; $row++) {
    for ($col = 0 ; $col < 3 ; $col++) {
        echo $emp [$row] [col]." ";
    }
    echo "<br>";
}
?>
```

output :=

jk	80000
dk	90000
lr	85000

2. Explain Session and cookies in detail.

→ Session is used to store and pass information from one page to another temporarily. (until user close the website) session technique is widely used in shopping website where we need to store and pass cart information. e.g username, Price, name from one page to another.

- Starting a PHP session :=

first step is start up a session. After a session is started, session variables can be created to store information. session_start() is used to begin a new session.

- Storing session Data :=

Session data in key-value pairs using the $_SESSION[] superglobal array. The store data can be accessed during lifetime of a session.

- Accessing session Data :=

Data stored in sessions can be easily accessed by firstly calling session_start() and the by passing the corresponding key to the $_SESSION associative array.

- Destorying session Data :=

To delete only a certain session data the unset feature can be used with corresponding session variable in the $_SESSION Array. session_destroy() function is used to completely destroy a session.

e.x <? php

```
session_start();
$_SESSION['ID'] = "88";
$_SESSION['Name'] = "jk";

echo 'The id is :'. $_SESSION['ID']. "<br>";
echo 'The Name is :' . $_SESSION['Name']. "<br>";

if (isset ($_SESSION['ID'])){
    unset ($_SESSION['Name']); }

session_destory();

?>
```

Cookie :=

A cookie is a small file with a maximum size of 4 kB that the web server stores on the client computer. A cookie can only be read from the domain that it has been issued from. Each time when clients sends request to the server, cookie is embedded with request. such way, cookie can be received at the server side.

Setting cookie :=

To set a cookie, the setcookie() function is used. These function needs to be called prior to any output generated by the script otherwise the cookie will not be set.

Syntax :- setcookie (name, value, expire, path, domain, security);

e.x
```
<? php
    Setcookie ("college", "supatej", time() + 3600);
?>
```

If we want to check cookies is set or not before accessing its value, isset() function is used.

Accessing Cookie Value :-

for accessing a cookie value, the PHP $_COOKIE superglobal variable is used. It is associative array that contains a record of all the cookies values sent by the browser in current request. cookie name is used as key.

e.x
```
<? php
    echo "College Name is ". $_COOKIE["college"];
?>
```

output :- College Name is supatej

Deleting Cookie :-

The setcookie() function can be used to delete a cookie. for deleting, setcookie() function is called by passing the cookie name & other arguments or empty string but expiration date is required to be set in the past.

e.x
```
<? php
    Setcookie ("college", "", time() - 60);
?>
```

3. Explain operator of PHP in detail.

→ Operator are used to performing operations on some values. operators are noting but symbols needed to perform operations of various types.

Unary operator :- work on single operands such as ++, --
Binary operator :- work on two operands such as binary +, -, *, /
Ternary operator :- work on three operands such as "?:"

Arithmetic operators :- $a = 29 $b = 4

Operator	Name	Explanation	Example	Answer
+	Addition	sum of operands	$a + $b	33
-	subtraction	difference	$a - $b	25
*	multiplication	Product	$a * $b	116
/	Division	Quotient	$a / $b	7.25
%	Modulus	Remainder	$a % $b	1
**	Exponentiation	Power	$a ** $b	707281

Assignment operators :- $a = 300 $b = 200

operator	Name	Explanation	Example	Answer
=	Assign	Assign the value	$a = $b (200)	200
+=	Add then Assign	Addition ($a = $a + $b)	$a += $b	500
-=	subtract then Assign	subtraction ($a = $a - $b)	$a -= $b	100
*=	multiply then Assign	multiplication ($a = $a * $b)	$a *= $b	60000
/=	Divide then Assign	quotient ($a = $a / $b)	$a /= $b	1.5
%=	Divide then Assign	modulo ($a = $a % $b)	$a %= $b	100

Bitwise operator :=

- Perform bit-level operations on operands.

operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in $a & $b are set 1
|	or	$a | $b	Bits that are 1 in either $a or $b set 1
^	Xor	$a ^ $b	Bits that are 1 in either $a or $b set 0
~	Not	~$a	1 set to 0 and 0 set to 1
<<	shift left	$a << $b	Left shift bits $a $b steps
>>	shift right	$a >> $b	Right shift bits $a $b places

Comparison operators :=

operator	Name	Example	Explanation
==	Equal	$a == $b	True if $a is equal to $b
===	Identical	$a === $b	True if both are equal & same data type
!==	Not identical	$a !== $b	True if $a not equal $b & not same data type
!=, <>	Not equal	$a <> $b	True if $a not equal to $b
<	less than	$a < $b	$a is less than $b
>	greater than	$a > $b	$a is greater than $b
<=	less than or equal	$a <= $b	True if $a is less than or equal $b
>=	greater than or equal	$a >= $b	True if $a is greater than or equal $b
<=>	Spaceship	$a <=> $b	-1 if $a is less than 0 if $a is equal 1 if $a is greater than

Increment / Decrement operator :=

operator	Name	Example	Explanation
++	Increment	++a	Increment the value of $a then return
		a++	return $a, then increment value of $a
--	Decrement	--a	Decrement value of $a then return
		a--	return $a then decrement value of $a

Logical operator :=

operator	Name	Example	Explanation
And &&	And	$a && $b	Return true if both are true
or ||	or	$a or $b	Return true if either $a or $b is true
Xor	Xor	$a xor $b	Return true if any one is true but not both
!	Not	!$a	Return true if $a is not true

String operator :=

operator	Name	Example	Explanation
.	Concatenation	$a.$b	concatenate both
.=	concatenation & Assign	$a.=$b	first concatenate & assign

Array Operators :=

Operator	Name	Example	Explanation
+	Union	$a + $b	Union of $a and $b
==	Equality	$a == $b	Return True if $a & $b have same key/value
!=	Inequality	$a != $b	Return True if $a not equal to $b
===	Identity	$a === $b	Return True if $a & $b have same key/value and same type in same order
!==	Non-identity	$a !== $b	Return True if $a is not identical to $b
<>	Inequality	$a <> $b	Return True if $a not equal to $b

Execution Operator :=

Operator	Name	Example	Explanation
``	Backticks	`dir`	execute the shell command & return result.

Error control operators :=

Operator	Name	Example	Explanation
@	At	@file	Intentional file error