

# Introduction to Web Development (4340704)

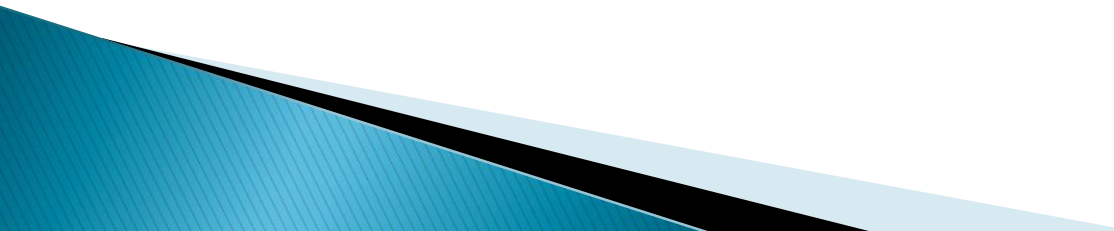
Computer Department  
AVPTI Rajkot

# Unit – II

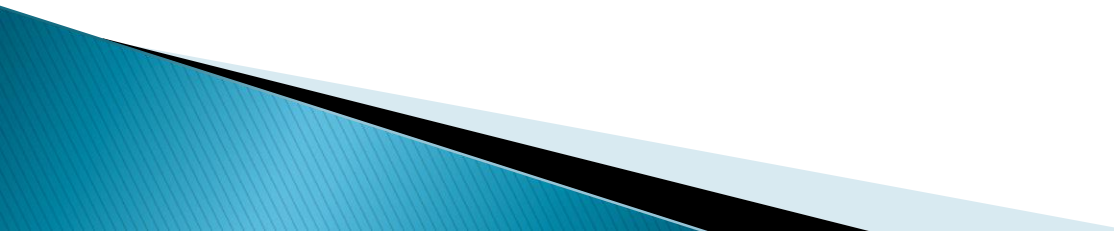
## Array and Functions in PHP



# Unit Outcomes

- ▶ Use different types of arrays for a given application.
  - ▶ Create a custom user defined function for a given requirement.
  - ▶ Use PHP in-built functions to perform string operations, simple mathematical operations and to process date and time.
- 

# PHP Array

- ▶ An array is a special variable or data structure, which can hold more than one value at a time, at single place.
  - ▶ Arrays are useful when we want to store a group of data.
  - ▶ If you have a list of items (a list of city names, for example), storing the city and then search the specific city from that list is become difficult when list is long.
  - ▶ To store all these values in single variable we have to create array and you can access any value using index number of array.
- 

# Types of Array.

- ▶ In PHP, there are three types of arrays:
- ▶ **Indexed arrays** – Arrays with a numeric index
- ▶ **Associative arrays** – Arrays with named keys
- ▶ **Multidimensional arrays** – Arrays containing one or more arrays
- ▶ In PHP, the `array()` function is used to create an array.
- ▶ Syntax for indexed array  
`array(value1,value2,value3,etc.);`
- ▶ Syntax for associative arrays:  
`array(key=>value,key=>value,key=>value,etc.);`

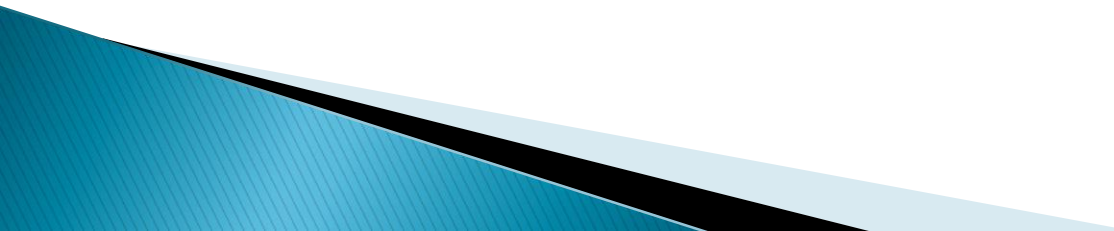
# Indexed arrays(One dimensional array or vector array)

- ▶ In this array values are stored in sequential manner with starting index 0.

## Syntax to create index array:

- ▶ `$arrayname[index]="value";`
- ▶ `$arrayname=array("value1","value2","value3");`
- ▶ **Following are the two ways to create Index array.**
- ▶ `$cities = array("rajkot", "baroda", "surat");`
- ▶ `$cities[0] = " rajkot ";`  
`$cities[1] = " baroda ";`  
`$cities[2] = " surat ";`

# Continue...

- ▶ The **count()** function is used to return the length (the number of elements) of an array.
  - ▶ **count(\$arrayname);**
  - ▶ It will return the number of elements of an array.
  - ▶ You can print value of index array one by one element or by using loop.
- 

# Example

```
<?php
```

```
$cities = array("Rajkot", "baroda", "surat");
```

```
echo $cities[0]."<br>";
```

```
echo $cities[1]."<br>";
```

```
echo $cities[2]."<br>";
```

```
for($i=0;$i<3;$i++)
```

```
{
```

```
    echo $cities[$i]."<br>";
```

```
}
```

```
$c=count($cities);
```

```
echo "$c" . "<br>";
```

```
echo count($cities);
```

```
?>
```





# PHP Associative Arrays

- ▶ In associative array there is unique ID key which is specified by the programmer to reference the each value stored in array.

## **Syntax to create associative array:**

- ▶ `$arrayname['keyname']="value";`
- ▶ `$arrayname=array("keyname1"=>value1",  
("keyname2"=>value2"));`
- ▶ **Following are the two ways to create associative array.**
- ▶ `$cities = array(" rajkot "=>"03",  
"baroda"=>"06", "surat"=>"05");`

# Continue...

- ▶ `$cities['rajkot'] = "03";`  
`$cities['baroda'] = "06`  
`$cities['surat'] = "07";`
  - ▶ To loop through and print all the values of an associative array, you could use a **foreach loop**.
  - ▶ foreach loop is used to accessed arrays.
  - ▶ **`foreach($array as $variable)`**  
**`{`**  
**`echo $variable;`**  
**`}`**
- For each pass the value of the current array element assigned to `$value` an the array pointer is moved by one and in the next pass next element will be processed.

# Example

```
<?php  
$cities = array("rajkot"=>"03", "baroda"=>"06",  
"surat"=>"05");
```

```
foreach ($cities as $c )  
{  
    echo "$c<br>";  
}
```

```
foreach ($cities as $c => $p)  
{  
    echo "$c: $p <br>";  
}
```

```
?>
```



# Multidimensional Arrays

- ▶ A multidimensional array is an array containing one or more arrays.
- ▶ IN associative array you can store values with one key to each value but if there are more than one key for each value than you have to use multi dimensional array to store these values.
- ▶ A two-dimensional array is **an array of arrays** while a three-dimensional array is **an array of arrays of arrays**

# Continue...

- ▶ `$cars = array`  
    (  
        `array("Volvo",22,18),`  
        `array("BMW",15,13),`  
        `array("Land Rover",17,15)`  
    );
- ▶ Here the two-dimensional `$cars` array contains three arrays, and it has two indices: row and column.
- ▶ To get access to the elements of the `$cars` array we must point to the two indices (row and column).
- ▶ **The dimension of an array indicates the number of indices you need to select an element.**
- ▶ For a two-dimensional array you need two indices to select an element
- ▶ For a three-dimensional array you need three indices to select an element

# Example

```
<?php
$cars = array (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Land Rover",17,15)
);

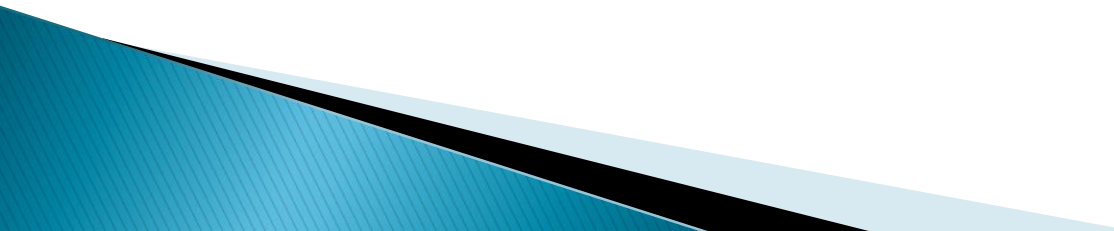
for ($row = 0; $row < 3; $row++)
{
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++)
    {
        echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
}
?>
```

# PHP Strings

- ▶ Strings can be seen as a stream of characters.
- ▶ A string is a sequence of characters.
- ▶ Strings in PHP are surrounded by either double quotation marks, or single quotation marks.
- ▶ Everything inside quotes, single (‘ ‘) and double (“ “) in PHP is treated as a string.
- ▶ 

```
<?php  
    echo "Hello";  
    echo 'Hello';  
?>
```
- ▶ Both echo statment will print Hello in output.

# Single quoted & Double quoted string

- ▶ Double quotes process special characters, single quotes does not.
  - ▶ Double quoted strings perform action on special characters.
  - ▶ When there is a variable in the string, it returns the *value* of the variable.
  - ▶ Single quoted strings does not perform such actions, it returns the string like it was written, with the variable name.
- 



# Continue...

- `<?php`  
    `$name = "Hiren";`  
    `echo "Hello $name";`  
    `?>`
- The output is Hello Hiren
- `<?php`  
    `$name = "Hiren";`  
    `echo 'Hello $name';`  
    `?>`
- The output is Hello \$name

# heredoc syntax

- ▶ It is a special syntax to define strings in PHP.

- ▶ Syntax

```
<<<name of string (identifier)
```

```
//content
```

```
name of string;
```

- ▶ <?php

```
$str = <<<strname
```

```
Hi.
```

```
How r u?.
```

```
I am fine.
```

```
strname;
```

```
echo $str;
```

```
?>
```

- ▶ This method is convenient for specifying long strings, containing both line feeds and multi-type quotes.
- ▶ Heredoc strings are like double-quoted strings without escaping and without the double-quotes.
- ▶ Any variables in the string will get substituted for their respective values.

# nowdoc syntax

- ▶ The nowdoc's syntax is similar to the heredoc's syntax except that the identifier which follows the <<< operator needs to be enclosed in **single quotes**.

- ▶ Syntax

```
<<<'name of string' (identifier)
```

```
//content
```

```
name of string;
```

- ▶ <?php

```
$str = <<<'strname'
```

```
Hi.
```

```
How r u?.
```

```
I am fine.
```

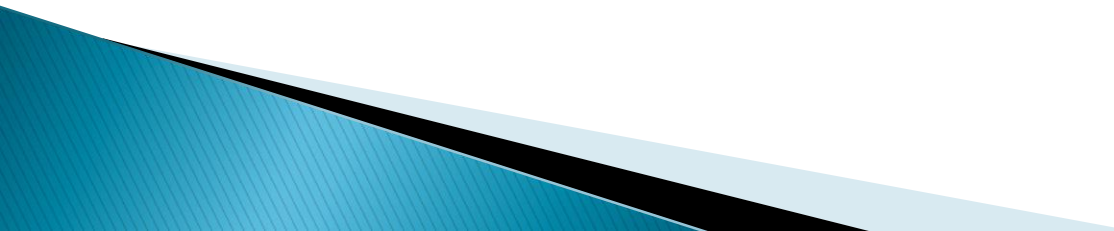
```
strname;
```

```
echo $str;
```

```
?>
```

- ▶ Nowdoc strings are like single-quoted strings without escaping and without the double-quotes.
- ▶ Any variables in the string will Not get substituted for their respective values.

# PHP Functions

- ▶ A **function** is a piece of code which takes one or more input in the form of parameter and does some processing and returns a value.
  - ▶ Each function perform some specific task.
  - ▶ Function can be used repeatedly in a program.
  - ▶ A function will not execute immediately when a page loads but it will be executed by a call to the function.
  - ▶ There are two main category of function one is **inbuilt function** and other is **user define function**.
  - ▶ PHP is very power full language because of its inbuilt functions.
  - ▶ it has more than 1000 built-in functions.
  - ▶ Foe examples **string functions**, **date and times functions** and **mathematical functions** etc.
- 

# PHP User Defined Functions

- ▶ Besides the built-in PHP functions, we can create our own functions.
- ▶ A user defined function declaration starts with the word "function".
- ▶ Syntax
- ▶ **function** *functionName()*  
{  
    *code to be executed;*  
}
- ▶ A function name can start with a letter or underscore (not a number).
- ▶ Give the function a name that reflects what the function does.
- ▶ Function names are NOT case-sensitive.
- ▶ To call the function, just write its name.

# Continue...

- ▶ **Return value**
  - If your Function perform calculation and you want to return value than using *return* statement you can return the value from your function. the function return value at point from where you call your function.
- ▶ **Function with arguments**
  - Information can be passed to functions through arguments. An argument is just like a variable.
  - Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
  - `function functionName(arg1,arg2,...,argn) {  
    code to be executed; }`

# Example

```
<?php
function printmsg()
{
    echo "Hello world!";
}
printmsg();
?>
```

```
<?php
function add($x, $y)
{
    $z = $x + $y;
    return $z;
}
echo add(5,10)."<br>";
$ans=add(5,10);
echo $ans;
?>
```

# Default arguments

- ▶ When we are defining function we can set default argument in function.
- ▶ Once we have define default argument there is no need to pass value for that argument while you are calling function.

```
<?php
function add($n1=30,$n2=10)
{
    $n3=$n1+$n2;
    echo "Addition is: $n3<br/>";
}
add(); //30+10
add(20); //20+10
add(40,40); //40+40
?>
```



# Passing Arguments by value and reference

- ▶ In PHP, arguments to a function can be passed by value or passed by reference.
- ▶ By default, values of actual arguments are passed by value to formal arguments which become local variables inside the function.
- ▶ Hence, modification to these variables doesn't change value of actual argument variable.

```
<?php
```

```
function swap($arg1, $arg2) {  
    $temp=$arg1;  
    $arg1=$arg2;  
    $arg2=$temp;  
    echo $arg1.$arg2."<br>"; // 20 and 10  
}  
$arg1=10;  
$arg2=20;  
swap($arg1, $arg2);  
echo $arg1.$arg2."<br>"; // 10 and 20  
?>
```

# Continue...

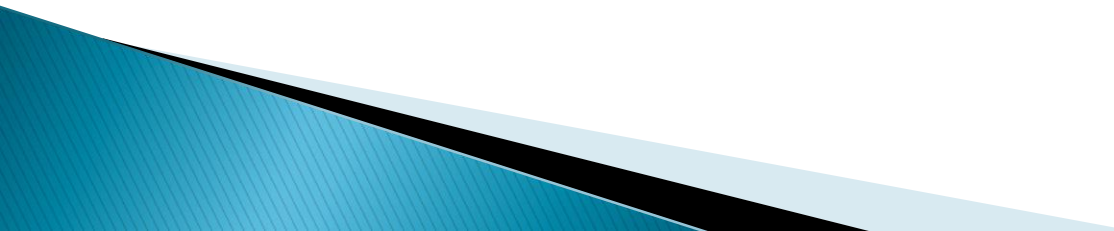
- ▶ When arguments are passed by reference, change in value of formal argument is reflected in actual argument variable because the former is a reference to latter.
- ▶ Thus pass by reference mechanism helps in indirectly manipulating data in global space.

```
<?php
```

```
function swap(&$arg1, &$arg2) {  
    $temp=$arg1;  
    $arg1=$arg2;  
    $arg2=$temp;  
    echo $arg1.$arg2."<br>"; // 20 and 10 }  
$arg1=10;  
$arg2=20;  
swap($arg1, $arg2);  
echo $arg1.$arg2."<br>"; // 20 and 10  
?>
```

There is no reference sign on a function call – only on function definitions.

# PHP Variable Scope

- ▶ In PHP, variables can be declared anywhere in the script.
  - ▶ The scope of a variable is the part of the script where the variable can be referenced/used.
  - ▶ PHP has three different variable scopes:
    - local
    - global
    - static
- 

# Continue...

- ▶ A variable declared **outside** a function has a **GLOBAL SCOPE** and can only be accessed outside a function.

```
<?php
$x = 5; // global scope
function myTest() {
    echo "<p>Variable x inside function is:
    $x</p>"; // not print 5
}
myTest();

echo "<p>Variable x outside function is:
$x</p>"; // print 5
?>
```

# Continue...

- ▶ A variable declared **within** a function has a **LOCAL SCOPE** and can only be accessed within that function.
- ▶ You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

```
<?php
```

```
function myTest() {
```

```
    $x = 5; // local scope
```

```
    echo "<p>Variable x inside function is: $x</p>";
```

```
    // print 5}
```

```
myTest();
```

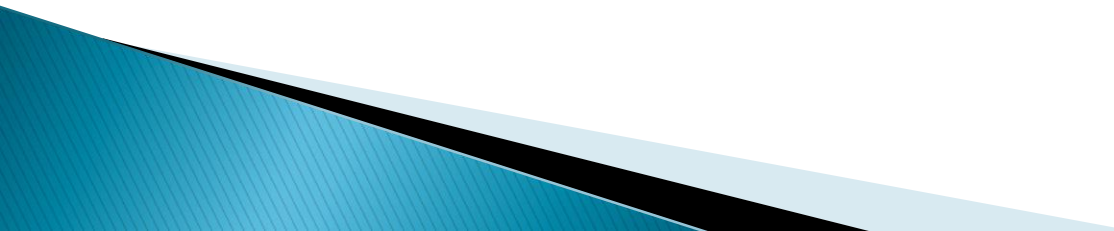
```
echo "<p>Variable x outside function is: $x</p>";
```

```
// not print 5
```

```
?>
```



# PHP The static Keyword

- ▶ Static Variables in PHP are variables that retain their value even after the function or method in which they are defined has finished execution.
  - ▶ Unlike regular variables, static variables are not destroyed and recreated each time the function is called.
  - ▶ They preserve their value and state across multiple calls.
  - ▶ static keyword can be used to declare the such variable.
- 

# Continue...

```
<?php
function myTest() {
    static $x = 0;
    echo $x;    // print 0 1 2 and without static
                keyword print 0 0 0

    $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

# *global* keyword

- The global keyword is used to access a global variable from within a function.
- To do this, use the global keyword before the variables (inside the function).

```
<?php
$x = 5; // global scope
function myTest()
{
    global $x;
    echo $x; // print 5
}
myTest();
echo $x; // print 5
?>
```



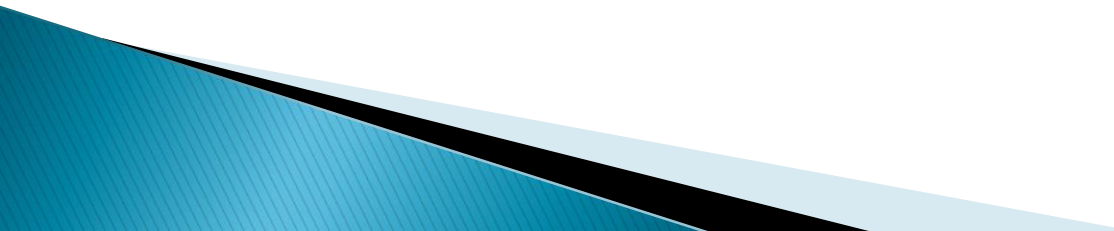
# Variable function

- ▶ If name of a variable has parentheses (with or without parameters in it) in front of it, PHP parser tries to find a function whose name corresponds to value of the variable and executes it.
- ▶ Such a function is called variable function(Value of Variables and Function name Same).
- ▶ Variable functions won't work with language constructs such as echo, print, unset(), isset() etc.

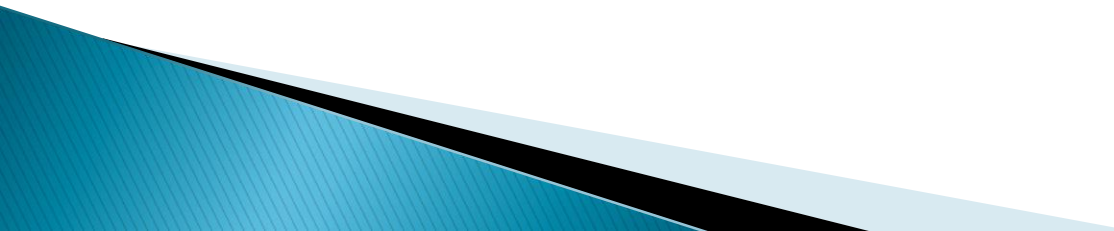
# Continue...

```
<?php  
function hello(){  
    echo "Hello World";  
}  
$var="Hello";  
$var();  
?>
```

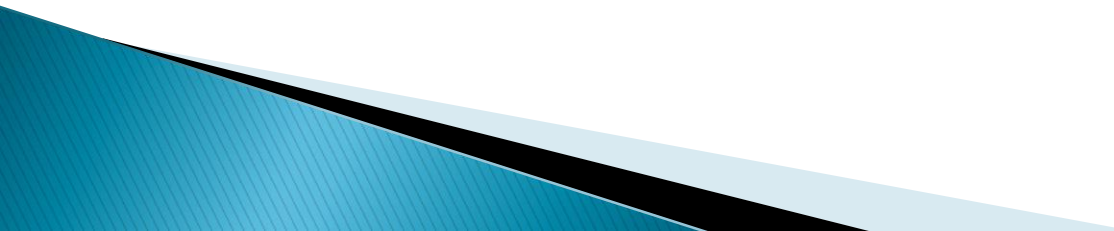
Value of a variable matches with function name. The function is thus called by putting parentheses in front of variable



# PHP Inbuilt Function

- ▶ Built-in functions in PHP are pre-defined functions that are available for use without the need for explicit declaration or definition.
  - ▶ These functions provide a wide range of functionality, from basic operations like string manipulation and mathematical calculations to more advanced tasks such as database connectivity and file handling.
  - ▶ PHP's extensive collection of built-in functions makes it a powerful language for web development and other applications.
- 

# Continue...

- ▶ String Manipulation: PHP offers numerous built-in functions for string operations.
  - ▶ Mathematical Calculations: PHP includes built-in functions for performing mathematical calculations.
  - ▶ Date and Time Manipulation: PHP provides functions for working with dates and times.
- 

# String processing functions

- `chr()`, `ord()`, `strlen()`, `trim()`, `ltrim()`, `rtrim()`,  
`join()`, `substr()`, `str_replace()`, `str_split()`,  
`str_word_count()`, `strcmp()`, `strcasecmp()`,  
`strpos()`, `stripos()`, `strrev()`, `strtolower()`,  
`strtoupper()`, `str_shuffle()`

# Mathematical functions

- `abs()`, `ceil()`, `floor()`, `round()`, `rand()`, `min()`, `max()`, `pi()`, `pow()`, `sqrt()`, `exp()`, `log()`, `decbin()`, `decoct()`, `dechex()`, `sin()`, `cos()`, `tan()`, `deg2rad()`, `rad2deg()`
- **Date/time function**
- `getdate()`, `gettimeofday()`, `time()`, `date_create()`, `mktime()`, `date_format()`, `date_diff()`, `checkdate()`