Unit– 2: Arrays and Functions in PHP

2.1. Introduction to PHP Arrays and types of arrays: Indexed, Associative and Multidimensional arrays

❖ PHP Arrays

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

❖ Advantage of PHP Array

- Less Code: We don't need to define multiple variables.
- Easy to traverse: By the help of single loop, we can traverse all the elements of an array.
- Sorting: We can sort the elements of array.

❖ PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

❖ PHP Indexed Array

- PHP index is represented by number which starts from 0.
- We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.
- There are two ways to define indexed array:

1st way:

```
$season=array("summer","winter","spring","autumn");
```

2nd way:

```
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
```

```
?>
```

❖ PHP Associative Array

- We can associate name with each array elements in PHP using => symbol.
- There are two ways to define associative array:

1st way:

```
$salary=array("abc"=>"350000","xyz"=>"450000","pqr"=>"200000");
```

2nd way:

```
$salary["abc"]="350000";
$salary["xyz"]="450000";
$salary["pqr"]="200000";
```

Example

```
<?php
$salary=array("abc"=>"350000","xyz"=>"450000","pqr"=>"200000");
echo "abc salary: ".$salary["abc"]."<br/>";
echo "xyz salary: ".$salary["xyz"]."<br/>";
echo "pqr salary: ".$salary["pqr"]."<br/>";
?>
```

❖ PHP Multidimensional Array

- PHP multidimensional array is also known as array of arrays.
- It allows you to store tabular data in an array.
- PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

```
$emp = array
        (
            array(1,"abc",400000),
            array(2,"xyz",500000),
            array(3,"pqr",300000)
        );
```

Example

```
<?php
        $emp = array
                (
                        array(1,"abc",400000),
                        array(2,"xyz",500000),
                        array(3,"pqr",300000)
```

```
                    );

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

2.2. PHP Strings: single quoted, double quoted, heredoc syntax, nowdoc syntax

- **Single Quoted**

Example:

```
$s1 = 'This is my first string';
echo $s1;
```

- What if you needed to add a single quote inside a single quoted string?
- Just escape that character with a back slash. \'

- **Double Quoted**

- Strings can be declared enclosed by double quotes. Character: ".
- If the string is enclosed in double-quotes ("), PHP will interpret the following escape sequences for special characters:

\n - Newline
\r - Carriage return
\t - Horizontal tab
\v - Vertical tab
\e - Escape
\f - Form feed
\\ - Backslash
\$ - Dollar sign
\" - Double-quote

Example:

```
<?php
        $user = 'vpmp';
        $welcome = "Hello $user \n This will print on next line";
        echo $welcome;
?>
```

Output:

```
Hello vpmp
```

This will print on next line

❖ **Heredoc Syntax**

- By using the syntax, we can display the HTML elements through PHP Script.

Syntax

```
<<<name of string
   //content
name of string
```

Example

```
<?php
   echo <<<VPMP
            welcome to Vpmp Polytechnic
      VPMP ;
?>
```

❖ **nowdoc syntax**

- A nowdoc string is similar to a heredoc string except that it doesn't expand the variables.
- Here's the syntax of a nowdoc string:

```
<?php

$str = <<<'IDENTIFIER'
            place a string here
            it can span multiple lines
            and include single quote ' and double quotes "
IDENTIFIER;
```

2.3. Creating, Manipulating and traversing different types of arrays

- We can traverse an array using loops in PHP.

Example 1: Using FOR Loop

```
<?php
      $a = array("abc", "xyz", "pqr");

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

Example 2: Using FOREACH Loop

```
<?php

    $a=array("Name"=>"abc", "Age"=>25, "Gender"=>"Male");
    foreach($a as $item)
    {
            echo $item."<br>";
     }
?>
```

2.4. User defined function: creating a function, calling a function and returning a value from function

- **PHP User Defined Functions**

Besides the built-in PHP functions, it is possible to create your own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

- **Create a User Defined Function in PHP**

- A user-defined function declaration starts with the word function:

Syntax

```
function functionName()
{
  code to be executed;
}
```

Example

```
<?php
function writeMsg()
{
        echo "Hello world!";
 }
    writeMsg(); // call the function
 ?>
```

- **Returning values**

- To let a function return a value, use the return statement

Example

```
<?php
function sum($n1, $n2 = 0)
```

```
{
    return $n1 + $n2;
}
        $result = sum(100, 50);
        echo "The sum of the two numbers is: ";
        echo $result . "\n";

        $result = sum(200);
        echo "The sum of the two numbers is: ";
        echo $result . "\n";
?>
```

2.5. Function with default arguments, passing arguments by value and reference

❖ **Default values for arguments**

- You can specify default values for arguments.
- If the argument is omitted from the function call the default is used.

```
<?php

    function addFunction($num1, $num2=5)
    {
        $sum = $num1 + $num2;
        return $sum;
    }

    $result = addFunction(10);
    echo "Addition = ".$result;
?>
```

Output:

Addition=15

❖ **Call-by-Value**

- Call by value means passing the value directly to a function.
- The called function uses the value in a local variable.

```
<?php

    function swap($num1, $num2)
    {
        $temp = $num1;
```

```
        $num1=$num2;
        $num2=$temp;
    }

    $num1=10;
    $num2=20;

    echo "Before Swap"."<br>";
    echo "num1=".$num1."<br>";
     echo "num2=".$num2."<br>";

    swap($num1,$num2);

    echo "After Swap"."<br>";
    echo "num1=".$num1."<br>";
     echo "num2=".$num2."<br>";
?>
```

Output:

Before Swap
num1=10
num2=20
After Swap
num1=10
num2=20

❖ Call-by-Reference

- Call by reference means passing the address of a variable where the actual value is stored.

```
<?php
    function swap(&$num1, &$num2)
    {
        $temp = $num1;
         $num1=$num2;
        $num2=$temp;
    }

    $num1=10;
     $num2=20;

    echo "Before Swap"."<br>";
    echo "num1=".$num1."<br>";
     echo "num2=".$num2."<br>";
```

```
    swap($num1,$num2);

    echo "After Swap"."<br>";
    echo "num1=".$num1."<br>";
    echo "num2=".$num2."<br>";
?>
```

Output:

Before Swap
num1=10
num2=20
After Swap
num1=20
num2=10

2.6. Variable scope, accessing global variables inside a function

❖ PHP Variable Scope

- The scope of a variable is defined as its range in the program under which it can be accessed.
- In other words, "The scope of a variable is the portion of the program within which it is defined and can be accessed."
- PHP has three types of variable scopes:
    1. Local variable
    2. Global variable
    3. Static variable

❖ Local variable

- The variables that are declared within a function are called local variables for that function.
- These local variables have their scope only in that particular function in which they are declared.
- This means that these variables cannot be accessed outside the function, as they have local scope.
- A variable declaration outside the function with the same name is completely different from the variable declared inside the function.
- Let's understand the local variables with the help of an example:

```
<?php
  function local_var()
  {
```

```
    $num = 45;  //local variable
    echo "Local variable declared inside the function is: ". $num;
  }
  local_var();
?>
```

❖ Global variable

- The global variables are the variables that are declared outside the function.
- These variables can be accessed anywhere in the program.
- To access the global variable within a function, use the GLOBAL keyword before the variable.
- However, these variables can be directly accessed or used outside the function without any keyword.
- Therefore there is no need to use any keyword to access a global variable outside the function.
- Let's understand the global variables with the help of an example:

```
<?php
  $name = "VPMP Polytechnic";       //Global Variable
  function global_var()
  {
    global $name;
    echo "Variable inside the function: ". $name;
    echo "</br>";
  }
  global_var();
  echo "Variable outside the function: ". $name;
?>
```

2.7. 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. This feature is useful in implementing callbacks, function tables etc.
- Variable functions can not be built eith language constructs such as include, require, echo etc.
- One can find a workaround though, using function wrappers.

Example

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

2.8. Using PHP built-in functions

i. String processing functions:
ii. Mathematical functions:
iii. Date/time function:

❖ **String processing functions:**

1. **Chr**

```
<?php
      echo chr(65);
?>
```

output:
A

2. **Ord**

```
<?php
    echo ord("A")."<br />";
    echo ord("And");
?>
```

Output:
65
65

3. **Strtolower**

```
<?php
     echo strtolower("HELLO");
?>
```

Output:
hello

4. **Strtoupper**

```
<?php
    echo strtolower("hello");
?>
```

Output:
HELLO

5. **Strlen**

```
<?php
     echo strlen("hello hi");
```

Output:
8

```
?>
```

6. ltrim

```
<?php
        $str="Hello";
        echo ltrim($str,"H")."<br>";
        $str="  Hello";
        echo ltrim($str);
?>
```

Output:
ello
Hello

7. Rtrim

```
<?php
      $str="Hello";
      echo rtrim($str,"o")."<br>";
      $str="Hello  ";
      echo rtrim($str);
?>
```

Output:
Hell
Hello

8. Trim

```
<?php
       $str="oHello";
       echo trim($str,"o")."<br>";
       $str="  Hello  ";
     echo trim($str);
 ?>
```

Output:
Hell
Hello

9. Substr

```
<?php

   echo substr("Hello world",3,2);
   echo substr("Hello world",-3,2);

   echo substr("Hello world",3,-2);
   echo substr("Hello world",-3,-2);
?>
```

Output:

lo
rl

lo wor
r

10. Strcmp

```
<?php
   echo strcmp("hello","hello");
   echo strcmp("hello","hi");
```

Output:
0
-1

```
    echo strcmp("hi","hello");                    1
?>
```

11. Strcasecmp

```
<?php                                             Output:
    echo strcmp("hello","hello");                 1
    echo strcasecmp("hello","Hello");             0
?>
```

12. Strpos

```
<?php                                             Output:
echo strpos("hello","e");                         1
echo strpos("hello","l");                         2
?>
```

13. Strrpos

```
<?php                                             Output:
      echo strrpos("hello","l");                  3
?>
```

14. stripos()

```
<?php
    echo stripos("Hello world","w");
?>
```

Output: 6

15. str_replace

```
<?php                                                          Output:
    echo str_replace("world","Aarav","Hello world!");          Hello Aarav!
?>
```

16. Strrev:

```
<?php
    echo strrev("hello");
?>
```

Output:

olleh

17. str_split()

```
<?php
    print_r(str_split("Hello",3));
?>
```

Output: Array ([0] => Hel [1] => lo)

18. str_word_count()

```
<?php
    echo str_word_count("Hello world!");
?>
```

Output: 2

19. join()

```
<?php
    $a = array('Hello,','How','are','you?');
    echo join("  ",$a);
?>
```

Output: Hello, How are you?

20. str_shuffle()

```
<?php
    echo str_shuffle("Hello World");
?>
```

Output: rlH lodWole

❖ **Mathematical functions:**

1. **abs**

```
<?php
    echo abs(-6.7)."<br>";
```

Output

6.7

```
    echo abs(-3)."<br>";                    3
?>
```

2. ceil

```
<?php                                       Output
   echo ceil(0.60)."<br>";                  1
   echo ceil(5)."<br>";                     5
 ?>
```

3. Floor

```
<?php
    echo floor(0.60)."<br>";
    echo floor(5)."<br>";
?>
```

Output:

0

5

4. Round

```
<?php                                       Output
    echo round(5.335,2)."<br>";             5.34
    echo round(0.49);                       0
?>
```

5. Fmod

```
<?php
    echo fmod(5,2);
?>
```

Output:

1

6. **Min**

```
<?php
    echo min(5,2,1,-4);
?>
```

Output:

-4

7. **Max**

```
<?php
    echo max(5,2,1,-4);
?>
```

Output:

5

8. **Pow**

```
<?php
    echo pow(5,2);
?>
```

Output:

25

9. **Sqrt**

```
<?php
    echo sqrt(25);
?>
```

Output:

5

10. **Rand**

```
<?php
    echo rand()."<br>";
```

Output

18269

```
    echo rand(40,500);                    417
?>
```

11. Pi

```
<?php
    echo(pi());
?>
```

Output: 3.1415926535898

12. exp()

```
<?php
    echo(exp(0) . "<br>");
    echo(exp(1) . "<br>");
    echo(exp(10) . "<br>");
?>
```

13. log()

```
<?php
    echo(log(2) . "<br>");
    echo(log(10) . "<br>");
?>
```

14. decbin()

```
<?php
    echo decbin("3") . "<br>";
    echo decbin("1") . "<br>";
    echo decbin("7");
?>
```

15. decoct()

```
<?php
    echo decoct("1") . "<br>";
    echo decoct("9") . "<br>";
?>
```

16. dechex()

```
<?php
    echo dechex("5") . "<br>";
```

```
        echo dechex("16") . "<br>";
        echo dechex("18") . "<br>";
?>
```

17. sin(), cos(), tan()

- **sin():** It returns the sine of a number
- **cos():** It returns the cosine of a number
- **tan():** It returns the tangent of a number

```
<?php
        echo(sin(30) . "<br>");
        echo(cos(0) . "<br>");
        echo(tan(60) . "<br>");
?>
```

18. deg2rad()

```
<?php
        echo deg2rad("0") . "<br>";
        echo deg2rad("30") . "<br>";
?>
```

19. rad2deg()

```
<?php
        echo rad2deg(0);
?>
```

❖ **Date/time function:**

1. getdate()

- The getdate() function returns date/time information of a timestamp or the current local date/time.

```
<?php
        print_r(getdate());
?>
```

Output:

Array ([seconds] => 40 [minutes] => 47 [hours] => 0 [mday] => 31 [wday] => 1 [mon] => 8 [year] => 2015 [yday] => 242 [weekday] => Monday [month] => August [0] =>

1440996460)

2. gettimeofday()

- It returns the current time.

```
<?php
    print_r(gettimeofday());
    echo "<br>";
    echo gettimeofday(true);
?>
```

Output

```
Array ( [sec] => 1679380099 [usec] => 405869 [minuteswest] => 0 [dsttime]
 => 0 )

1679380099.4086
```

3. time()

- The time() function returns the current time in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

```
<?php
    $t=time();
    echo($t . "<br>");
    echo(date("Y-m-d",$t));
?>
```

Output

```
1440996289

2015-08-31
```

4. date_create()

- The date_create() function accepts a date time string and time zone (optional) as parameters and, creates a DateTime object accordingly.

- **Example**

```
<?php
        $date=date_create("2023-04-22");
        echo date_format($date,"d/m/Y");
?>
```

- **Output**

22/04/2023

5. date_format()

- It returns a new DateTime object, and then format the date.

- **Example**

```
<?php
        $date=date_create("2023-04-22");
        echo date_format($date,"d/m/Y");
?>
```

- **Output**

22/04/2023

6. mktime()

- The mktime() function returns the Unix timestamp for a date.
- It contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.
- **Example**

```
<?php
        $d=mktime(11, 10, 50, 3, 24, 2023);
        echo " date is " . date("Y-m-d h:i:s", $d);
?>
```

- **Output:** date is 2023-03-24 11:10:50

7. date_diff()

- The date_diff() function returns the difference between two DateTime objects.
- **Example:**

```
<?php
        $date1=date_create("2023-03-11");
        $date2=date_create("2023-12-11");
        $diff=date_diff($date1,$date2);
        echo $diff->format("%R%a days");
?>
```

- **Output:** -275 days

8. checkdate()

- The checkdate() function is used to validate a Gregorian date.
- **Example**

```
<?php
        var_dump(checkdate(2,29,2003));
        echo "<br>";
        var_dump(checkdate(2,29,2004));
?>
```

- **Output:**

```
bool(false)
bool(true)
```