UNIT – III

WORKING WITH PHP ARRAYS & FUNCTIONS

3.1 Arrays

- **3.1.1 Definition**

- An array is a data structure that stores multiple values in a single variable.

- In array elements are stored in the form of Key-Value pair.

- Each element in the array can be identified using its key along with name of the array.

- There are two types of arrays:
  - Indexed/Numeric arrays and
  - Associative arrays

3.1 Arrays

- **3.1.2 Types of Array**

- There are two types of arrays:
  - **Numeric/Indexed arrays :** In numeric array, each element having numeric key associated with it, that is starting from 0.
  - **Associative arrays :** In Associative array, each element having key in the form of String associated with it.

3.1 Arrays

- Numeric arrays :

- Arrays with a numeric key (index) is known as Numeric or Indexed Array.

- **Creating Numeric Array**
- There are two ways to create indexed arrays:
  1. using array()
  2. using array identifier

3.1 Arrays

- **3.1.3 Creating Array using array() function**

- Numeric array can be created using array() function.

- **Syntax :**

- $array_name = array (Value1, Value2, Value3, ……);

- In Numeric array, each elements are assigned a numeric Key value starting from 0 for first element and so on.

3.1 Arrays

- **3.1.3 Creating Array using array() function**

- **Example :**

```
<?php
    $MyArray = array("A", "B", "C");
    print_r ( $MyArray );
?>
```

Output:

```
Array( [0] => A [1] => B [2] => C )
```

```
<?php
    $MyArray = array("A", "B", "C");
    echo $MyArray[1];
?>
```

Output:

```
B
```

3.1 Arrays

- **3.1.4 Creating Array using array identifier**

- PHP allows you to create an array using array identifier as shown below:

- **Syntax :**

- $array_name [] = Value1 ;
- $array_name [] = Value2 ;
- $array_name [] = Value3 ;

3.1 Arrays

- **3.1.4 Creating Array using array identifier**

- **Example :**

```
<?php
      $ MyArray [] = “A”;
      $ MyArray [] = “B” ;
      $ MyArray [] = “C” ;
   print_r ( $MyArray );
?>
```

Output:

Array([0] => A [1] => B [2] => C)

```
<?php
      $ MyArray [] = “A”;
      $ MyArray [] = “B” ;
      $ MyArray [] = “C” ;
   echo  $MyArray [1];
?>
```

Output:

B

3.1 Arrays

- **3.1.5 <u>Start Index value :</u>**

- By default starting index of an array is 0. It is incremented by 1 for the each successive elements of an array.

- However it is also possible to define start index of an array explicitly while creating array.

3.1 Arrays

- **3.1.5 Start Index value :**

- **Example :**

```
<?php
      $MyArray = array(10 => “A”, “B”, “C”);
    print_r ( $MyArray );
?>
```

Output:

Array([10] => A [11] => B [12] => C)

```
<?php
      $ MyArray [10] = “A”;
      $ MyArray [] = “B” ;
      $ MyArray [] = “C” ;
     print_r ( $MyArray );
?>
```

Output:

Array([10] => A [11] => B [12] => C)

3.1 Arrays

Adding more elements to array :

- Once you create an array using either array() function or using an array identifier, you can add more elements to an array using array identifier.

- **Example:**

```
<?php
    $MyArray = array("A", "B", "C");
    print_r ( $MyArray );
    $ MyArray [] = "D" ;
    $ MyArray [] = "E" ;
    echo "<br/>";
    print_r ( $MyArray );
?>
```

Output:

```
Array( [0] => A [1] => B [2] => C )
Array( [0] => A [1] => B [2] => C  [3] => D  [4] => E )
```

3.1 Arrays

Adding more elements to array :

- **Example:**

```
<?php
    $ MyArray [10] = “A”;
    $ MyArray [] = “B” ;
    $ MyArray [] = “C” ;
    print_r ( $MyArray );

    $ MyArray [] = “D” ;
    $ MyArray [] = “E” ;
    echo “<br/>”;
    print_r ( $MyArray );
?>
```

Output:

```
Array( [10] => A [11] => B [12] => C )
Array( [10] => A [11] => B [12] => C  [13] => D  [14] => E )
```

3.2 Associative Array

- In Associative array, each element having key in the form of String associated with it.
- You can use array() function to create an associative array.
- **Syntax :**
- $array_name = array (Key1 => Value1, Key2 => Value2, Key3 => Value3, ……);

3.2 Associative Array

- **Example - 1 :**

```
<?php
    $MyArray = array ( “Sachin” => 60, “Sehwag” => 80, “Virat” => 120);
    print_r ( $MyArray );
?>
```

Output:

Array([Sachin] => 60 [Sehwag] => 80 [Virat] => 120)

3.2 Associative Array

- **Example – 2 :**

```
<?php
    $MyArray = array ( “Sachin” => 60, “Sehwag” => 80, “Virat” => 120);
    echo “Run Scored by Virat : ” .  $MyArray [“Virat”];
?>
```


Output:

Run Scored by Virat : 120

3.2 Associative Array

- **Example – 3 :**

```
<?php
    $MyArray[“Sachin”]  = 60 ;
    $MyArray[ “Sehwag”] = 80;
    $MyArray[“Virat”]  = 120;

    echo “Run Scored by Virat : ” .  $MyArray [“Virat”];
?>
```

Output:

Run Scored by Virat : 120

3.2 Associative Array

- **for each statement**

- The Foreach loop is used to loop over all the elements of an array.

- **Syntax:**

- ```
  foreach (array as value)
  {
    code to be executed;
  }

  foreach (array as key => value)
  {
    code to be executed;
  }
  ```
  ```

3.2 Associative Array

- **for each statement**

- The example below demonstrates the Foreach loop that will print the values of the given array:

```
<?php
        $person = array('name' => 'Andrew', 'age' => 21);

        foreach ($person as $value)
        {
                echo $value  .  "<br />";
        }
?>
```

Output:

```
Andrew
21
```

3.2 Associative Array

- **for each statement**

- An alternative form of For each loop gives you access to the current key:

```
<?php
        $person = array(“Name” => “Andrew”, “Age” => 21);

        foreach ($person as $key => $value)
        {
                echo  $key  .  “ is “  .  $value  .  "<br />";
        }
?>
```


Output:

Name is Andrew
Age is 21

3.3 User Defined Function

• 3.3.1 Defining Function

- In PHP user defined function can be defined using function statement.

- **Syntax:**

```
function function_name ( Argument_LIST )
{
    //Statement Block;
}
```


- In Above syntax, function_name can be any name that you want to define as function.

- Argument list is a collection of variables, separated by commas.

3.3 User Defined Function

- **3.3.1 Defining Function**

- **Example:**

- ```
  <?php
    function display ()
    {
        echo “Name : Mayur Thakkar”
    }
  ?>
  ```
  ```

3.3 User Defined Function

- **3.3.2 Calling Function**

- Once you define a function in your script, you can call it any number of times from any where in the script.

- **Syntax:**

```
function_name();
```

- **Example:**

```
display();
```

3.3 User Defined Function

- **3.3.2 Calling Function**

- **Example:**

- ```
  <?php
      function display ()
      {
        echo "Name : Mayur Thakkar";
      }

      display();
  ?>
  ```

Output:

Name : Mayur Thakkar
  ```

3.3 User Defined Function

• 3.3.3 Passing arguments to Function

- Sometimes it is required to pass arguments to the function while calling the function.

- **Example:**

- <?php

```
    function display ($name)
    {
      echo "Name : "  .  $name;
    }

      display("Mayur  Thakkar");
?>
```

Output:

Name : Mayur Thakkar

3.3 User Defined Function

• 3.3.4 Returning values from function

- Sometimes it is required that the function should return a value to the point from which the function is called:
- **Example:**
- <?php

```
    function sum ($a, $b)
    {
      echo "Name : "  .  $name;
      $c = $a + $b;

      return  $c
    }

      echo "Sum = " .   sum(3,5);
?>
```

3.3 User Defined Function

• 3.3.5 Variable Scope

- The scope of a variable is the part of the script where the variable can be used.

- PHP has three different variable scopes:
  - local
  - Global

3.3 User Defined Function

- **Accessing *local* variable**

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function.

- **Example:**

```
<?php

        function myTest()
        {
                $x = 5;                 // local scope
                echo " X  = ".  $x ;
        }

        myTest();

        // using x outside the function will generate an error
         echo " X  = ".  $x ;
?>
```

3.3 User Defined Function

- **Accessing variable with *global* statement**

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

```
<?php

        $x = 5;      // global scope
        function myTest()
        {
              echo " X  = ".  $x ; // using x inside the function will generate an error
        }

        myTest();


         echo " X  = ".   $x ;
?>
```

3.3 User Defined Function

- **Accessing variable with *global* statement**

- The global variable can be accessed within function using global keyword.

- **Example:**

```
<?php

        $x = 5;      // global scope
        function myTest()
        {
              global $x;
             echo " X  = ".   $x  . "<br/>";
        }

        myTest();

         echo " X  = ".   $x ;
?>
```

3.3 User Defined Function

- **Accessing variable with *global* statement**

  - PHP also stores all global variables in an array called $GLOBALS[*index*].
  - The *index* holds the name of the variable.
  - **Example:**

```
<?php

        $x = 5;      // global scope
        function myTest()
        {
            echo  $GLOBALS[“x”]  .  “<br/> “;
        }

        myTest();

         echo “ X  = “.   $x ;
?>
```

3.4 User Defined Function

• 3.4.1 Setting Default values for Arguments

- When you define a function that accepts arguments, you must pass that many arguments while calling the function. If you pass wrong number of arguments at the time of calling the function then it will generate a error message.

- In PHP, you can define a function having default arguments. So if you don't pass the value for that argument then it will consider the default value for that argument.

- But if you pass the explicit value for that argument then it will overwrite the default argument value.

- The default arguments must be specified after all non default arguments in the function.

3.4 User Defined Function

- **3.4.1 Setting Default values for Arguments**

- **Syntax:**

```
function function_name ($arg1 , $arg2=Value)
{
}
```

- **Example:**

```
<?php
    function Area ($radius , $pi=3.14)
    {
       $a = $pi * $r * $r;
       return $a;
    }

    $r=3;
    echo  "Area of Circle: " .  Area($r);
?>
```

Output:

Area of Circle: 28.26

3.4 User Defined Function

- **3.4.2 Passing arguments with values**

- When you pass arguments to the function, the values of the passed arguments are copied into the argument variables declared inside the argument list of function definition.

- Thus called function works with copies of argument instead of original passed arguments. So any changes made to these variables in the body of the function are local to that function and are not reflected outside it.

3.4 User Defined Function

- **3.4.2 Passing arguments with values**

- **Example:**

```
<?php
    function swap ($a , $b)
    {
       $c = $a;
       $a = $b;
       $b = $c;
    }

    $a = 2;
    $b = 5;

    echo  “Before Swap: ”  .  “a = $a  and b = $b ”  . “<br/>”;
    swap($a,$b);
    echo  “After Swap: ”  .  “a = $a  and b = $b ”;

?>
```

Output:

Before Swap: a = 2 and b = 5
After Swap: a = 2 and b = 5

3.4 User Defined Function

• 3.4.3 Passing arguments with Reference

- In order to work with original variable, that are passed as an argument within called function, we have to pass arguments by reference instead of passing arguments by value.

- When arguments are passed by reference, the function works with original variable instead of copies of that variable.

- In order to pass arguments by reference, each argument in the function definition should be proceeded by an ampersand (&) sign.

3.4 User Defined Function

- **3.4.3 Passing arguments with Reference**

- **Example:**

```
<?php
    function swap ( &$a , &$b)
    {
       $c = $a;
       $a = $b;
       $b = $c;
    }

    $a = 2;
    $b = 5;

    echo "Before Swap: " . "a = $a and b = $b " . "<br/>";
    swap($a,$b);
    echo "After Swap: " . "a = $a and b = $b ";

?>
```

Output:

Before Swap: a = 2 and b = 5
After Swap: a = 5 and b = 2

3.5.1 String Functions

- String functions allow you to manipulate the string in various ways.
- You can perform different operations on string using these functions.

- Different string functions in PHP are as below:
  - chr
  - Ord
  - Strtolower
  - Strtoupper
  - Strlen
  - Ltrim
  - Rtrim
  - Trim
  - Substr
  - Strcmp
  - strrev

3.5.1 String Functions

- **chr :** Returns a one-character string containing the character specified by ASCII value.

- **Syntax:**

  string chr (ASCII – VALUE)

- The parameter ASCII is the ASCII code of a character.

- **Example:**

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

Output:

A

3.5.1 String Functions

- **ord :** Returns the ASCII value of the first character of string.

- **Syntax:** int ord (string)

- **Example:**

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

Output:

65
65

3.5.1 String Functions

- **strtolower** : returns string with all alphabetic characters converted to lowercase.

- **Syntax:** string strtolower (string)

- **Example:**


```
<?php
        echo strtolower("Hello") . "<br/>";
        echo strtolower("HELLO") . "<br/>";
?>
```

Output:

hello
hello

3.5.1 String Functions

- **strtoupper :** Returns string with all alphabetic characters converted to uppercase.

- **Syntax:** string strtoupper (string)

- **Example:**


```
<?php
        echo strtoupper(“Hello”) . “<br/>”;
        echo strtoupper(“hello”) . “<br/>”;
?>
```

Output:

HELLO
HELLO

3.5.1 String Functions

- **strlen :** returns the length of the given string.

- **Syntax:** int strlen (string)

- **Example:**

```
<?php
        echo strlen("Hello");
?>
```


Output:

5

3.5.1 String Functions

- **ltrim :** removes whitespace from the beginning of a string.

- **Syntax:** string ltrim (string)

- **Example:**

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


Output:

Hello World

3.5.1 String Functions

- **rtrim :** removes whitespace from the end of a string.

- **Syntax:** string rtrim (string)

- **Example:**

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


Output:

Hello World

3.5.1 String Functions

- **trim :** removes whitespace from the beginning and end of a string.

- **Syntax:** string trim (string)

- **Example:**

```
<?php
        echo rtrim(“    Hello World     ”) ;
?>
```

Output:

Hello World

3.5.1 String Functions

- **substr :** return part of a string.

- **Syntax:** string substr (string, int start [, int length])

- **Example:**

```
<?php
        echo substr( “Hello World” , 6 ) . “<br/>”;
        echo substr( “Hello World” , 0, 5 ) ;
?>
```

Output:

World
Hello

3.5.1 String Functions

- **strcmp :** accepts two string (comma separated) as input to compare and returns an int (integer).

- **Syntax:** int strcmp (str1 , str2)

- Return Values:
  - Returns < 0 if str1 is less than str2;
  - Returns > 0 if str1 is greater than str2, and
  - Returns 0 if they are equal.

- **Example:**

```
<?php
        echo strcmp( “A” , “A” ) . “<br/>”;
        echo strcmp( “A” , “B” ) . “<br/>”;
?>
```

Output:

0
-1

3.5.1 String Functions

- **strrev :** Reverse a string.

- **Syntax:** string strrev (string)

- **Example:**

```
<?php
        echo strrev( “Hello” ) ;
?>
```

Output:
olleH

3.5.2 Date Functions

- Date function allows you to display date and time in different format and manipulate it.
- Different date functions in PHP are as below:
  - date
  - getdate
  - checkdate

3.5.2 Date Functions

- **date :** returns a string in the specified format for a date and time.

- **Syntax:** string date(format)

- **Example:**

```
<?php
        echo date(d/m/y)  .  “<br/>”;
        echo date(D/M/Y)  .  “<br/>”;
        echo date(d/m/y  h:i:s  a)  .  “<br/>”;
        echo date(D/M/Y  h:i:s  A)  .  “<br/>”;
?>
```

Output:

15/09/16
Thu/Sep/2016
15/09/16 05:41:30 am
Thu/Sep/2016 05:41:30 AM

3.5.2 Date Functions

- **getdate :** returns an array with date, time information for an unix timestamp.

- **Syntax:** getdate(timestamp)

- **Example:**

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


Output:

Array ([seconds] => 30 [minutes] => 41 [hours] => 5 [mday] => 15 [wday] => 4 [mon] => 9 [year] => 2016 [yday] => 258 [weekday] => Thursday [month] => September [0] => 1473918090)

3.5.2 Date Functions

- **checkdate :** check the validity of a given date.

- **Syntax:** boolean checkdate (int month, int day, int year)

- **Example:**

```
<?php
        echo checkdate(2,28,2016) . “<br/>”;
        echo checkdate(28,2,2016) . “<br/>”;
?>
```

Output:

1

3.5.3 Time Functions

- Different time functions in PHP are as below:
  - time
  - mktime

3.5.3 Time Functions

- **time :** return the current Unix timestamp.

- **Syntax:** time(void)

- **Example:**

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


Output:
1473918090

3.5.3 Time Functions

- **mktime :** is used to get unix timestamp for a date.

- **Syntax:** mktime(hour,minute,second,month,day,year)

- **Example:**

```
<?php
    echo mktime(0,0,0,1,2,1970);
?>
```

Output:
86400

3.5.4 Math Functions

- Math functions allow to perform various operations on numeric values.
- Different math functions in PHP are as below:
  - abs
  - ceil
  - floor
  - round
  - fmod
  - min
  - max
  - pow
  - sqrt

****