UNIT - 2
PHP ARRAY FUNCTIONS

Prepared By: Prof. Disha H. Parekh

PHP ARRAY FUNCTION – COUNT

- count(): Returns the number of elements in array.
- Syntax: count ($array_variable);

```
$array = array("IMSC6", "IMCA6", "MSC2");
echo "Number of items in array are "
.count($array);;
```

PHP ARRAY FUNCTION – LIST

- list(): Assigns variables to elements in array.
- Syntax: list(variable1, variable2,….);

```
list($c1,$c2,$c3) = $array;
echo "There are several classes like $c1, $c2, $c3“;
```

PHP ARRAY FUNCTION – IN_ARRAY

- in_array(): helps in searching an array for a particular element.
- Syntax: in_array(search, array, type)

```
if (in_array("IMSC2", $array)) {
        echo "The class does exist in the list"; }
else {
        echo "The class isnt in the list"; }
```

PHP ARRAY FUNCTION – CURRENT, NEXT, END, PREV, EACH

- current(): Returns the current element in an array.
- Syntax: current($variable);

```
echo "Current is " .current($array);
```

PHP ARRAY FUNCTION – SORT

- sort(): sorts the indexed array in ascending order.
- Syntax: sort(array);

```
sort($array);
```

PHP ARRAY FUNCTION – RSORT

- rsort(): sorts the indexed array in descending order.
- Syntax: rsort(array);

```
rsort($array);
```

PHP ARRAY FUNCTION – ASORT

- asort(): Sort an associative array in ascending order, according to the value.
- Syntax: asort(*array,sortingtype);*

```
asort($array);
        foreach ($array as $x => $x_value)
        {
        echo "Key" . $x . "Value" . $x_value.  "<br>";
        }
```

PHP ARRAY FUNCTION – ARRAY_PUSH

- Array_push(): Inserts one or more elements to the end of an array.
- Syntax: array_push(array,value1,value2...
    - *array_push($array, "IMCA2", "MCA4");*
    - *print_r ($array);*

PHP ARRAY FUNCTION – ARRAY_POP

- Array_pop(): Removes one or more elements to the end of an array.
- Syntax: array_pop(array,value1,value2...
    - *array_pop($array);*
    - *print_r ($array);*

PHP ARRAY FUNCTION – ARRAY_MERGE

- Array_merge(): Merges one or more arrays into one array.
- Syntax: array_merge(*array1,array2,array3...*)
    - *$array1 = array("BBA", "MAM", "PGDBA");*
    - *print_r(array_merge($array,$array1));*

PHP ARRAY FUNCTION – ARRAY_REVERSE

- Array_reverse(): returns array in the reverse order.
- Syntax: array_reverse(*array,preserve*);
    - *print_r(array_reverse($array,true));*

PHP ARRAY FUNCTION – ARRAY_SEARCH

- Array_search(): Search an array for the value and return its key
- Syntax: array_search(*value,array,strict*);
    - *echo array_search("IMCA2", $array);*

PHP ARRAY FUNCTION – ARRAY_DIFF

- Array_search(): Search an array for the value and return its key
- Syntax: array_diff(*array1,array2,array3...*);
    - *$difference = array_diff($array, $array1);*
    - *echo "Array 1 is ";*
    - *print_r($array);*
    - *echo "Array 1 is ";*
    - *print_r($array1);*
    - *echo "<br>";*
    - *print_r ($difference);*

UNIT - 2
PHP FILE FUNCTIONS

Prepared By: Prof. Disha H. Parekh

FILE FUNCTIONS: FOPEN()

- The fopen() function opens a file or URL.
- If fopen() fails, it returns FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.
- Syntax: fopen(filename,mode,include_path,context);

FILE FUNCTIONS: FOPEN()

- Possible modes are:
  - "r" (Read only. Starts at the beginning of the file)
  - "r+" (Read/Write. Starts at the beginning of the file)
  - "w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist)
  - "w+" (Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist)
  - "a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist)
  - "a+" (Read/Write. Preserves file content by writing to the end of the file)
  - "x" (Write only. Creates a new file. Returns FALSE and an error if file already exists)
  - "x+" (Read/Write. Creates a new file. Returns FALSE and an error if file already exists)

FILE FUNCTIONS: FOPEN()

- Eg:

FILE FUNCTIONS: FREAD()

- The fread() reads from an open file.
- The function will stop at the end of the file or when it reaches the specified length, whichever comes first.
- This function returns the read string, or FALSE on failure.
- Syntax: fread(file,length)
- Eg:

FILE FUNCTIONS: FWRITE()

- The fwrite() writes to an open file.
- The function will stop at the end of the file or when it reaches the specified length, whichever comes first.
- This function returns the number of bytes written, or FALSE on failure.
- Syntax: fwrite(file,string,length)
- Eg:

FILE FUNCTIONS: FCLOSE()

- The fclose() function closes an open file.
- This function returns TRUE on success or FALSE on failure.
- Syntax: fclose(file)
- Eg:

FILE FUNCTIONS: FILE_EXISTS()

- The file_exists() function checks whether or not a file or directory exists.
- This function returns TRUE if the file or directory exists, otherwise it returns FALSE.
- Syntax: file_exists(path)
- Eg:

FILE FUNCTIONS: IS_READABLE()

- The is_readable() function checks whether the specified file is readable.
- This function returns TRUE if the file is readable.
- Syntax: is_readable(file)
- Eg:

FILE FUNCTIONS: IS_WRITEABLE()

- The is_writeable() function checks whether the specified file is writeable.
- This function returns TRUE if the file is writeable.
- This function is an alias of the is_writable() function.
- Syntax: is_writeable(file)
- Eg:

FILE FUNCTIONS: FGETS()

- The fgets() function returns a line from an open file.
- The fgets() function stops returning on a new line, at the specified length, or at EOF, whichever comes first.
- This function returns FALSE on failure.
- Syntax: fgets(file,length)
- Eg:

FILE FUNCTIONS: FILE()

- The file() reads a file into an array.
- Each array element contains a line from the file, with newline still attached.
- Syntax: file(path,include_path,context)
- Eg:

FILE FUNCTIONS: FILE_GET_CONTENTS()

- The file_get_contents() reads a file into a string.
- This function is the preferred way to read the contents of a file into a string. Because it will use memory mapping techniques, if this is supported by the server, to enhance performance.
- Syntax: file_get_contents(path,include_path,context,start,max_length)
- Eg:

FILE FUNCTIONS: FILE_PUT_CONTENTS()

- The file_put_contents() writes a string to a file.
- This function follows these rules when accessing a file:
  - If FILE_USE_INCLUDE_PATH is set, check the include path for a copy of *filename*
  - Create the file if it does not exist
  - Open the file
  - Lock the file if LOCK_EX is set
  - If FILE_APPEND is set, move to the end of the file. Otherwise, clear the file content
  - Write the data into the file
  - Close the file and release any locks
- This function returns the number of character written into the file on success, or FALSE on failure.
- Syntax: file_put_contents(file,data,mode,context)
- Eg:

FILE FUNCTIONS: FTELL()

- The ftell() function returns the current position in an open file.
- Returns the current file pointer position, or FALSE on failure.
- Syntax: ftell(file)
- Eg:

FILE FUNCTIONS: FSEEK()

- The fseek() function seeks in an open file.
- This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.
- This function returns 0 on success, or -1 on failure. Seeking past EOF will not generate an error.
- Syntax: fseek(file,offset,whence)
- Eg:

FILE FUNCTIONS: REWIND()

- The rewind() function "rewinds" the position of the file pointer to the beginning of the file.
- This function returns TRUE on success, or FALSE on failure.
- Syntax: rewind(file)
- Eg:

FILE FUNCTIONS: REWIND()

- The rewind() function "rewinds" the position of the file pointer to the beginning of the file.
- This function returns TRUE on success, or FALSE on failure.
- Syntax: rewind(file)
- Eg:

UNIT - 2
PHP DATE FUNCTIONS

Prepared By: Prof. Disha H. Parekh

PHP DATE FUNCTION – GETDATE

- getdate(): The getdate() function returns date/time information of a timestamp or the current local date/time.
- Syntax: getdate(*timestamp);*
  - *print_r(getdate());*

PHP DATE FUNCTION – CHECKDATE

- checkdate(): The checkdate() function is used to validate a Gregorian date.
- Syntax: checkdate(*month,day,year);*
  - *var_dump (checkdate(1,28,2019));*

PHP DATE FUNCTION – CHECKDATE

- checkdate(): The checkdate() function is used to validate a Gregorian date.
- Syntax: checkdate(*month,day,year);*
  - *var_dump (checkdate(1,28,2019));*

PHP DATE FUNCTION – DATE_ADD

- date_add(): The date_add() function adds some days, months, years, hours, minutes, and seconds to a date.
- Syntax: date_add(*object,interval);*

```
$date=date_create("04-02-2014");

date_add($date,
date_interval_create_from_date_string("200 days"));

echo date_format($date,"d-M-Y");
```

PHP DATE FUNCTION – DATE_DIFF

- date_diff(): The date_diff() function returns the difference between two DateTime objects.
- Syntax: date_diff(*datetime1,datetime2,absolute);*

```
$date1=date_create("2013-04-23");
$date2=date_create("2013-10-01");
$date_diff=date_diff($date1,$date2);
echo $date_diff -> format ("%R%a days");
```

PHP DATE FUNCTION – DATE_MODIFY

- date_modify(): The date_modify() function modifies the timestamp.
- Syntax: date_modify(*object,modify);*

```
$date = date_create("2019-01-27");
date_modify($date, "-15 days");
echo date_format($date, "Y-M-d");
```

PHP DATE FUNCTION – MKTIME

- mktime(): The mktime() function returns the Unix timestamp for a date.
- Syntax: mktime(*hour,minute,second,month,day,year,is_dst*);

```
echo "The day on 4th February 2014 was ".date("l",
mktime(0,0,0,2,4,2014))
```

PHP DATE FUNCTION – MKTIME

- mktime(): The mktime() function returns the Unix timestamp for a date.
- Syntax: mktime(*hour,minute,second,month,day,year,is_dst*);

```
echo "The day on 4th February 2014 was ".date("l",
mktime(0,0,0,2,4,2014))
```

UNIT – 2
MATH FUNCTIONS

Prepared By: Disha H. Parekh

PHP MATH FUNCTIONS

- PHP abs() function: Return the absolute value of different numbers:
- Syntax: abs(*number*);

```
<?php
    echo(abs(1.2) . "<br>");
    echo(abs(-1.2) . "<br>");
    echo(abs(-2) . "<br>");
    echo(abs(2));

?>
```

PHP MATH FUNCTIONS

- PHP ceil() function: Round numbers up to the nearest integer:
- Syntax: ceil(*number*);

```
<?php
    echo(ceil(0.10) . "<br>");
    echo(ceil(0.20) . "<br>");
    echo(ceil(2) . "<br>");
    echo(ceil(2.1) . "<br>");
    echo(ceil(-3.1) . "<br>");
    echo(ceil(-7.9));
?>
```

PHP MATH FUNCTIONS

- PHP floor() function: Round numbers down to the nearest integer:
- Syntax: floor(*number*);

```
<?php
    echo(floor(0.20) . "<br>");
    echo(floor(0.10) . "<br>");
    echo(floor(2) . "<br>");
    echo(floor(2.1) . "<br>");
    echo(floor(-7.1) . "<br>");
    echo(floor(-3.9));

?>
```

PHP MATH FUNCTIONS

- PHP round() function: The round() function rounds a floating-point number.
- Syntax: round(*number,precision,mode*);

```
<?php
    echo(round(0.60) . "<br>");
    echo(round(0.50) . "<br>");
    echo(round(0.49) . "<br>");
    echo(round(-2.30) . "<br>");
    echo(round(-2.60));

?>
```

PHP MATH FUNCTIONS

- PHP fmod() function: Return the remainder
- Syntax: fmod(*x*,*y*);

```
<?php
    $x = 5;
    $y = 3;
    $result = fmod($x,$y);
    echo $result;
?>
```

PHP MATH FUNCTIONS

- PHP min() function: Returns the lowest value in an array, or the lowest value of several specified values.
- Syntax: min(*array_values*);

```
<?php
    echo(min(5,4,6,9,10) . "<br>");
    echo(min(2,13,78,98,15) . "<br>");

?>
```

PHP MATH FUNCTIONS

- PHP max() function: Returns the largest value in an array, or the largest value of several specified values.
- Syntax: max(*array_values*);

```
<?php
    echo(max(5,4,6,9,10) . "<br>");
    echo(max(2,13,78,98,15) . "<br>");

?>
```

PHP MATH FUNCTIONS

- PHP pow() function: Returns x raised to the power of y.
- Syntax: max(*array_values*);

```
<?php
    echo(pow(2,5) . "<br>");
    echo(pow(-2,5) . "<br>");
    echo(pow(-2,-3) . "<br>");
    echo(pow(-2,-2.2));

?>
```

PHP MATH FUNCTIONS

- PHP rand() function: The rand() function generates a random integer.
- Syntax: rand(*min,max*);

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

PHP MATH FUNCTIONS

- PHP sqrt() function: Return the square root of different numbers:
- Syntax: sqrt(number);

```
<?php
    echo(sqrt(0) . "<br>");
    echo(sqrt(1) . "<br>");
    echo(sqrt(4) . "<br>");
    echo(sqrt(0.144) . "<br>");
    echo(sqrt(-8));

?>
```

UNIT – 2
BUIILT – IN FUNCTIONS

PREPARED BY: DISHA H. PAREKH, DCS, INDUS UNIVERSITY

Prepared By: Disha H. Parekh

PHP BUILT-IN FUNCTIONS

- PHP chr() function: used to return characters from different ASCII values.

```
<?php
    echo chr(52) . "<br>"; // Decimal value
    echo chr(052) . "<br>"; // Octal value
    echo chr(0x52) . "<br>"; // Hex value

?>
```

PHP BUILT-IN FUNCTIONS

- PHP ord() function: The ord() function returns the ASCII value of the first character of a string.
- Syntax: ord(*string*)

```
<?php
    echo ord("h")."<br>";
    echo ord("hello")."<br>";

?>
```

PHP BUILT-IN FUNCTIONS

- PHP strtolower() function: The strtolower() function converts a string to lowercase.
- Syntax: strtolower(*string*)

```
<?php
    echo strtolower("Hello WORLD.");
?>
```

PHP BUILT-IN FUNCTIONS

- PHP strtoupper() function: The strtoupper() function converts a string to uppercase.
- Syntax: strtoupper(*string)*

```
<?php
    echo strtoupper("Hello WORLD!");
?>
```

PHP BUILT-IN FUNCTIONS

- PHP strlen() function: he strlen() function returns the length of a string.
- Syntax: strlen(*string*)

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

PHP BUILT-IN FUNCTIONS

- PHP ltrim() function: The ltrim() function removes whitespace or other predefined characters from the left side of a string.
- Syntax: ltrim(*string,charlist*)

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

?>
```

PHP BUILT-IN FUNCTIONS

- PHP rtrim() function: The rtrim() function removes whitespace or other predefined characters from the right side of a string.
- Syntax: rtrim(*string,charlist*)

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

?>
```

PHP BUILT-IN FUNCTIONS

- PHP strncmp() function: The strncmp() function compares two strings. This function is similar to the strcmp() function, except that strcmp() does not have the length parameter.
- Syntax: strncmp(*string1,string2,length*)

```
<?php
    echo strncmp("Hello world!","Hello earth!",6);
?>
```

PHP BUILT-IN FUNCTIONS

- PHP strncasecmp() function: The strncasecmp() function compares two strings. This function is similar to the strcasecmp() function, except that strcasecmp() does not have the length parameter.
- Syntax: strncasecmp(*string1,string2,length*)

```
<?php
    echo strncasecmp("Hello","Hello",6);
    echo "<br>";
    echo strncasecmp("Hello","hELLo",6);

?>
```

PHP BUILT-IN FUNCTIONS

- PHP strpos() function: The strpos() function finds the position of the first occurrence of a string inside another string.
- Syntax: strpos(*string,find,start*)

```
<?php
    echo strpos("I love php, I love php too!","php");
?>
```

PHP BUILT-IN FUNCTIONS

- PHP strrpos() function: The strrpos() function finds the position of the last occurrence of a string inside another string.
- Syntax: strrpos(*string,find,start*)

```
<?php
    echo strrpos("I love php, I love php too!","php");
?>
```

PHP BUILT-IN FUNCTIONS

- PHP strstr() function: The strstr() function searches for the first occurrence of a string inside another string
- Syntax: strstr(*string,search,before_search*)

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

PHP BUILT-IN FUNCTIONS

- PHP str_replace() function: The str_replace() function replaces some characters with some other characters in a string.
- This function works by the following rules:
  - If the string to be searched is an array, it returns an array
  - If the string to be searched is an array, find and replace is performed with every array element
  - If both find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace
  - If find is an array and replace is a string, the replace string will be used for every find value
- Syntax: str_replace(*find,replace,string,count*)

```
<?php
    echo str_replace("world","Peter","Hello world!");
?>
```

PHP BUILT-IN FUNCTIONS

- PHP strrev() function: The strrev() function reverses a stringIf the string to be searched is an array, it returns an array.
- Syntax: strrev(*string*)
- Example:

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

UNIT - 2
PHP COOKIES

Prepared By: Prof. Disha H. Parekh

PHP COOKIES

- Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.
- There are three steps involved in identifying returning users −
  - Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
  - Browser stores this information on local machine for future use.
  - When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

WHY AND WHEN TO USE COOKIES

- Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer. The path were the cookies are stored depends on the browser.
- Personalizing the user experience – this is achieved by allowing users to select their preferences. The page requested that follow are personalized based on the set preferences in the cookies.
- Tracking the pages visited by a user

CREATING COOKIES

- Basic syntax:

```
<?php
    setcookie(cookie_name, cookie_value, [expiry_time],
    [cookie_path], [domain], [secure], [httponly]);
?>
```

CREATING COOKIES

- Php“setcookie” is the PHP function used to create the cookie.
- “cookie_name” is the name of the cookie that the server will use when retrieving its value from the $_COOKIE array variable. It’s mandatory.
- “cookie_value” is the value of the cookie and its mandatory
- “[expiry_time]” is optional; it can be used to set the expiry time for the cookie such as 1 hour. The time is set using the PHP time() functions plus or minus a number of seconds greater than 0 i.e. time() + 3600 for 1 hour.
- “[cookie_path]” is optional; it can be used to set the cookie path on the server. The forward slash “/” means that the cookie will be made available on the entire domain. Sub directories limit the cookie access to the subdomain.
- “[domain]” is optional, it can be used to define the cookie access hierarchy i.e. www.cookiedomain.com means entire domain while www.sub.cookiedomain.com limits the cookie access to www.sub.cookiedomain.com and its sub domains. *Note it’s possible to have a subdomain of a subdomain as long as the total characters do not exceed 253 characters.*
- “[secure]” is optional, the default is false. It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false.
- “[Httponly]” is optional. If it is set to true, then only client side scripting languages i.e.JavaScript cannot access them.

CREATING COOKIES

- We will create a basic program that allows us to store the user name in a cookie that expires after ten seconds.
- The code below shows the implementation of the above example “cookies.php”.

```
<?php
    setcookie("user_name", “Student_id", time()+ 60,'/');
            // expires after 60 seconds
    echo 'the cookie has been set for 60 seconds';
?>
```

RETRIEVING COOKIE VALUES

- Create another file named “cookies_getvalue.php” with the following code.

```
<?php
    print_r($_COOKIE); //output the contents of the cookie array variable
?>
```

DELETE COOKIES

- If you want to destroy a cookie before its expiry time, then you set the expiry time to a time that has already passed.
- Create a new filed named cookie_destroy.php with the following code.

```
<?php
    setcookie("user_name", "Guru99", time() - 360,'/');
?>
```

EXAMPLE:

- <?php

  // Setting a cookie

  setcookie("username", “ABC XYZ", time()+30*24*60*60);

- ?>
- The PHP $_COOKIE superglobal variable is used to retrieve a cookie value.
- <?php

  // Accessing an individual cookie value

  echo $_COOKIE["username"];

- ?>

EXAMPLE:

- It's a good practice to check whether a cookie is set or not before accessing its value. To do this you can use the PHP isset() function, like this:
- <?php

```
    // Verifying whether a cookie is set or not
  If(isset($_COOKIE["username"]))
  {
      echo "Hi " . $_COOKIE["username"];
  }
  else
  {
      echo "Welcome Guest!";
  }
```

- ?>

EXAMPLE:

- You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past, as shown in the example below:
- <?php

  // Deleting a cookie

  setcookie("username", "", time()-3600);
- ?>

EXAMPLE 2:

- Construct a file called: cookie_page.php

```
<?php
    setcookie("user", "Sonoo");
?>
<html>  <body>
<?php
    if(!isset($_COOKIE["user"]))
    {
        echo "Sorry, cookie is not found!";
    }
    else
    {
        echo "<br/>Cookie Value: " . $_COOKIE["user"];
    }
?>
</body> </html>
```

EXAMPLE 2:

- Construct a file called: cookie_page.php

```
<?php
    setcookie("user", “ABC");
?>
<html>  <body>
<?php
    if(!isset($_COOKIE["user"]))
    {
        echo "Sorry, cookie is not found!";
    }
    else
    {
        echo "<br/>Cookie Value: " . $_COOKIE["user"];
    }
?>
</body> </html>
```

EXAMPLE 2:

- To delete cookie:

```
<?php
   setcookie ("CookieName", "", time() - 3600);
?>
```

UNIT - 2
PHP SESSION

Prepared By: Prof. Disha H. Parekh

PHP SESSION

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it.
- This is much like a Session.
- The computer knows who you are.
- It knows when you start the application and when you end.
- But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

PHP SESSION

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.

PHP SESSION

- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

WHEN TO USE SESSION

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

STARTING A PHP SESSION

- In order to create a session, you must first call the PHP session_start function and then store your values in the $_SESSION array variable.
- Let's suppose we want to know the number of times that a page has been loaded, we can use a session to do that.
- The code below shows how to create and retrieve values from sessions

STARTING A PHP SESSION – SIMPLE EXAMPLE

```
<?php
session_start(); //start the PHP_session function
?>
<html>
<body>
<?php
    $_SESSION[“FName”] = “ABC”;
    $_SESSION[“Lname”] = “XYZ”;
    echo “First Name is: “ .$_SESSION[“FName”] . “<br>”;
    echo “Last Name is: “ .$_SESSION[“LName”] . “<br>”;
?>
</body> </html>
```

STARTING A PHP SESSION

```
<?php
 session_start(); //start the PHP_session function
    if(isset($_SESSION['page_count']))
          {
             $_SESSION['page_count'] += 1;
          }
             else
          {
             $_SESSION['page_count'] = 1;
          }
             echo 'You are visitor number ' . $_SESSION['page_count'];
?>
```

DESTROYING A SESSION VARIABLE

- The session_destroy() function is used to destroy the whole Php session variables.
- If you want to destroy only a session single item, you use the unset() function.
- The code below illustrates how to use both methods.

DESTROYING A SESSION VARIABLE

```
<?php
    session_destroy();
    //destroy entire session
 ?>
```

<u>Another option is unset:</u>

```
<?php
    unset($_SESSION['product_item']);
    //destroy product session item
?>
```

DESTROYING A SESSION VARIABLE

- Session_destroy removes all the session data including cookies associated with the session.
- Unset only frees the individual session variables.
- Other data remains intact.

UNIT - 2
PHP MISC. FUNCTIONS

Prepared By: Prof. Disha H. Parekh

PHP MISC. FUNCTION – DEFINE()

- define(): The define() function defines a constant.
- Syntax: define(*name,value,case_insensitive*)

PHP MISC. FUNCTION – CONSTANT()

- constant(): The constant() function returns the value of a constant.
  - Constants are much like variables, except for the following differences:
  - A constant's value cannot be changed after it is set
  - Constant names do not need a leading dollar sign ($)
  - Constants can be accessed regardless of scope
  - Constant values can only be strings and numbers
- Syntax: constant(constant)

PHP MISC. FUNCTION – INCLUDE()

- include(): The *include* statement includes and evaluates the specified file.
  - The Include() function is used to put data of one PHP file into another PHP file.
  - If errors occur then the include() function produces a warning but does not stop the execution of the script i.e. the script will continue to execute.
- Syntax: include(include_path);

PHP MISC. FUNCTION – REQUIRE()

- require(): The Require() function is also used to put data of one PHP file to another PHP file.
  - If there are any errors then the require() function produces a warning and a fatal error and stops the execution of the script i.e. the script will continue to execute.
- Syntax: require(‘pathname’);

PHP MISC. FUNCTION – HEADER()

- header(): header — Send a raw HTTP header
  - Remember that header() must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP.
- Syntax: header("Location: filename");

PHP MISC. FUNCTION – DIE()

- die(): The die() function prints a message and exits the current script.
  - This function is an alias of the exit() function.
- Syntax: die(*message*);