ACTIVITY ANSWER SHEET

Name	Rachel Ana L. Dantes
Section:	3R2

Instructions:

1. Push your output on your GITHUB repository.
2. Use the answer sheet provided save it as PDF file then push it to your GitHub.
3. Answer the ff. problems write it on the answer sheet.
4. Late submissions will no longer be accepted.
5. Caught copying outputs of others will be given sanctions.
6. Failure to follow these instructions will be given sanctions.

Activity 1: Control Structures

1. Write down the syntax in PHP for the ff.

1. if	if (condition) { code to be executed if condition is true; } Ex. <?php $d = date("D"); if($d == "Sat"){ echo "Have a nice weekend!"; } ?>
2. if…else	if (condition) { code to be executed if condition is true; } else { code to be executed if condition is false; } Ex. <?php $d = date("D"); if($d == "Sat"){ echo "Have a nice weekend!"; } else { echo "Have a nice day!"; } ?>
3. if…else if…else	if (condition) { code to be executed if condition is true; } elseif (condition) { code to be executed if first condition is false and this condition is true; } else { code to be executed if all conditions are false; } Ex. <?php $d = date ("D"); If ($d == "Sat"){ echo "Have a nice weekend!"; } elseif ($d == "Sun"){ echo "Have a nice Sunday!"; } else { echo "Have a nice day!"; } ?>
4. switch…case	switch (n) { case *label1*: code to be executed if n=label1; break; case *label2*: code to be executed if n=label2; break;

	case *label3*: code to be executed if n=label3; break; … default: code to be executed if n is different from all labels; } Ex. <?php $today = date("D"); switch($today){ case "Mon": echo "Today is Monday. Clean your house."; break; case "Tue": echo "Today is Tuesday. Buy some food."; break; case "Wed": echo "Today is Wednesday. Visit a doctor."; break; case "Thu": echo "Today is Thursday. Repair your car."; break; case "Fri": echo "Today is Friday. Party tonight."; break; case "Sat": echo "Today is Saturday. Its movie time."; break; case "Sun": echo "Today is Sunday. Do some rest."; break; default: echo "No information available for that day."; break; } ?>
5. for loop	for (init counter; test counter; increment counter) { code to be executed for each iteration; } Ex. <?php for($i=1; $i<=3; $i++){ echo "The number is " . $i . " "; } ?>
6. do while loop	do { code to be executed; } while (condition is true); Ex. <?php $i = 1; do { $i++; echo "The number is " . $i . " "; } while($i <= 3); ?>

7. while loop	while (condition is true) { code to be executed; } Ex. <?php $i = 1; while($i <= 3){ $i++; echo "The number is " . $i . " "; } ?>
8. foreach loop	foreach ($array as $value) { code to be executed; } Ex. <?php $colors = array("Red", "Green", "Blue"); // Loop through colors array foreach($colors as $value){ echo $value . " "; } ?>
9. break statement	<?php $i= array(0=>'Apple',1=>'Pomegranate','Banana', 'Mango', 'Guava', 'Orange', 'Pineapple'); foreach($i as $v) { If ($v=='Mango') { break;//Break statement } echo "Value:". "$v."." "." "; } ?>
10. continue statement	<?php echo '</h2>Example of using Array With Continue statement:</h2> '; $i= array(0=>'Apple',1=>'Pomegranate',' Banana', 'Mango', 'Guava', 'Orange', 'Pineapple'); foreach ($i as $v) { If($v=='Banana') { continue; } echo "The fruit is "."$v"; echo " "." "; } ?>
11. try…catch	<?php function inverse($x) { if (!$x) { throw new Exception('Division by zero.'); } return 1/$x;

```
}

try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo 'Caught exception: ', $e->getMessage(), "\n";
}

// Continue execution
echo "Hello World\n";
?>
```

2. Solve the ff. problem using PHP.

a. Write a program that checks if value is a number (integer).

Sample input: '1' Sample input: 1

Expected output: Not a number Expected output: A number

```
<?php
function isNumber($s)
{
  for ($i = 0; $i < strlen($s); $i++)
    if (is_numeric($s[$i]) == false)
      return false;
  return true;
}
$str = "abc";

if (isNumber($str))
  echo "A number";
else
  echo "Not a number";
?>
```

b. Write a program that checks if a value is positive or negative and odd or even.

Sample input: 0 Sample input: -1

Expected output: Positive & Even Expected output: Negative and Odd

```
<?php
// PHP code to check whether the number
// is Even or Odd in Normal way
function check($number){
        if($number % 2 == 0){
                echo "Positive & Even";
        }
        else{
                echo "Negative and Odd";
        }
}
// Driver Code
$number = 49;
check($number)
?>
```

c. Write a program that checks if a value is palindrome.

Sample input: Anna Sample input: Bogart

Expected output: Palindrome Expected output: Not a Palindrome

```php
<?php
// PHP code to check for Palindrome number in PHP
// Function to check for Palindrome
function Palindrome($number){
	$temp = $number;
	$new = 0;
	while (floor($temp)) {
		$d = $temp % 10;
		$new = $new * 10 + $d;
		$temp = $temp/10;
	}
	if ($new == $number){
		return 1;
	}
	else{
		return 0;
	}
}

// Driver Code
$original = 1431;
if (Palindrome($original)){
	echo "Palindrome";
}
else {
echo "Not a Palindrome";
}
?>
```

d. Write a program to calculate and print the factorial of a number using a for loop.
Sample input: 4
Expected output: 24

```php
<?php
//Driver Code
$n = 4;

$x = 1;
for($i=1;$i<=$n-1;$i++)
{
 $x*=($i+1);
}
echo "$x"."\n";
?>
```

e. Write a PHP program to generate and display the first n lines of a Floyd triangle.
Sample input: 3
Sample output:

```
1
2 3
4 5 6
```

```php
<?php
//Driver Code
$n = 3;

$count = 1;
for ($i = $n; $i > 0; $i--)
{
  for ($j = $i; $j < $n + 1; $j++)
   {
    printf("%4s", $count);
    $count++;
   }
	echo "\n";
```

```
  }
?>
```

Activity 2: PHP Built-in Functions

Write down the functionalities of the ff. built-in functions in PHP.

Array	- stores multiple values in one single variable, the values can be accessed by referring to an index number. There are 3 types of arrays - indexed, associative and multidimensional arrays. **Ex. 1: array_change_key_case() function** changes all keys in an array to lowercase or uppercase. `<!DOCTYPE html>` `<html>` `<body>` `<?php` `$age=array("rachel"=>"25","adrian"=>"37","ken"=>"43");` `print_r(array_change_key_case($age,CASE_UPPER));` `?>` `</body>` `</html>` ========== **Output:** Array ([RACHEL] => 25 [ADRIAN] => 37 [KEN] => 43) **Ex. 2: array_chunk() function** splits an array into chunks of new arrays. `<!DOCTYPE html>` `<html>` `<body>` `<?php` `$cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel");` `print_r(array_chunk($cars,2));` `?>` `</body>` `</html>` ========== **Output:** Array ([0] => Array ([0] => Volvo [1] => BMW) [1] => Array ([0] => Toyota [1] => Honda) [2] => Array ([0] => Mercedes [1] => Opel)) **Ex. 3: array_count_values() function** counts all the values of an array. `<!DOCTYPE html>` `<html>` `<body>` `<?php` `$a=array("A","Cat","Dog","A","Dog");`

	print_r(array_count_values($a)); ?> </body> </html> ========== **Output:** Array ([A] => 2 [Cat] => 1 [Dog] => 2) **Ex. 4: array_intersect_key() function** compares the keys of two (or more) arrays, and returns the matches. <!DOCTYPE html> <html> <body> <?php $a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"pink"); $a2=array("a"=>"red","c"=>"blue","d"=>"pink,"e"=>"maroon""); $result=array_intersect_key($a1,$a2); print_r($result); ?> </body> </html> ========== **Output:** Array ([a] => red [c] => blue [d] => pink) **Ex. 5: array_push() function** inserts one or more elements to the end of an array. <!DOCTYPE html> <html> <body> <?php $a=array("red","green"); array_push($a,"blue","yellow","pink"); print_r($a); ?> </body> </html> ========== **Output:** Array ([0] => red [1] => green [2] => blue [3] => yellow [4] => pink)
Calendar	PHP Calendar - contains functions that simplifies converting between different calendar formats. **Ex. 1: cal_days_in_month() function** returns the number of days in a month for a specified year and calendar <!DOCTYPE html> <html> <body> <?php

```
$d=cal_days_in_month(CAL_GREGORIAN,2,1965);
echo  "There was $d days in February 1965.<br>";

$d=cal_days_in_month(CAL_GREGORIAN,2,2004);
echo  "There was $d days in February 2004.";
?>

</body>
</html>
```

==========

Output:

There was 28 days in February 1965.
There was 29 days in February 2004.

Ex. 2: cal_from_jd() function

converts a Julian Day Count into a date of a specified calendar

```
<!DOCTYPE html>
<html>
<body>

<?php
$d=unixtojd(mktime(0,0,0,6,20,2007));
print_r(cal_from_jd($d,CAL_GREGORIAN));
?>

</body>
</html>
```

==========

Output:

Array ([date] => 6/20/2007 [month] => 6 [day] => 20 [year] => 2007 [dow] => 3 [abbrevdayname] => Wed [dayname] => Wednesday [abbrevmonth] => Jun [monthname] => June)

Ex. 3: cal_info() function

returns information about a specified calendar

```
<!DOCTYPE html>
<html>
<body>

<?php
print_r(cal_info(0));
?>

</body>
</html>
```

==========

Output:

Array ([months] => Array ([1] => January [2] => February [3] => March [4] => April [5] => May [6] => June [7] => July [8] => August [9] => September [10] => October [11] => November [12] => December) [abbrevmonths] => Array ([1] => Jan [2] => Feb [3] => Mar [4] => Apr [5] => May [6] => Jun [7] => Jul [8] => Aug [9] => Sep [10] => Oct [11] => Nov [12] => Dec) [maxdaysinmonth] => 31 [calname] => Gregorian [calsymbol] => CAL_GREGORIAN)

Ex. 4: cal_to_jd() function

converts a date in a specified calendar to Julian Day Count

```
<!DOCTYPE html>
<html>
<body>

<?php
$d=cal_to_jd (CAL_GREGORIAN,6,20,2007);
echo  $d;
?>

</body>
</html>
```

==========

Output:

2454272

Ex. 5: easter_date() function

returns the Unix timestamp for midnight on Easter of a given year

```
<!DOCTYPE html>
<html>
<body>

<?php
echo  easter_date() .  "<br />";
echo  date("M-d-Y",easter_date()) .  "<br />";
echo  date("M-d-Y",easter_date(1975)) .  "<br />";
echo  date("M-d-Y",easter_date(1998)) .  "<br />";
echo  date("M-d-Y",easter_date(2007));
?>

</body>
</html>
```

==========

Output:

Apr-12-2020
Mar-30-1975
Apr-12-1998
Apr-08-2007

Date	date/time functions allow you to get the date and time from the server where your PHP script runs. You can then use the date/time functions to format the date and time in several ways. **Ex. 1: checkdate() function** is used to validate a Gregorian date. <!DOCTYPE html> <html> <body> <?php var_dump(checkdate(12,31,-400)); echo " "; var_dump(checkdate(2,29,2003)); echo " "; var_dump(checkdate(2,29,2004)); ?> </body> </html> ========== **Output:** bool(false) bool(false) bool(true) **Ex. 2: date_add() function** adds some days, months, years, hours, minutes, and seconds to a date. <!DOCTYPE html> <html> <body> <?php $date=date_create("2013-03-15"); date_add($date,date_interval_create_from_date_string("40 days")); echo date_format($date,"Y-m-d"); ?> </body> </html> ========== **Output:** 2013-04-24 **Ex. 3: date_create_from_format()** function returns a new DateTime object formatted according to the specified format <!DOCTYPE html> <html> <body> <?php $date=date_create_from_format("j-M-Y","15-Mar-2013"); echo date_format($date,"Y/m/d"); ?> </body>

	</html> ========== **Output:** 2013/03/15 **Ex. 4: date_create() function** returns a new DateTime object <!DOCTYPE html> <html> <body> <?php $date=date_create("2020-04-25"); echo date_format($date,"Y/m/d"); ?> </body> </html> ========== **Output:** 2020/04/25 **Ex. 5: date_format() function** returns a date formatted according to the specified format <!DOCTYPE html> <html> <body> <?php $date=date_create("2020-01-14"); echo date_format($date,"Y/m/d H:i:s"); ?> </body> </html> ========== **Output:** 2020/01/14 00:00:00
Directory	The directory functions allow you to retrieve information about directories and their contents. **Ex.1: readdir() function** returns the name of the next entry in a directory <!DOCTYPE html> <html> <body> <?php $dir = "/images/"; // Open a directory, and read its contents if (is_dir($dir)){ if ($dh = opendir($dir)){ while (($file = readdir($dh)) !== false){ echo "filename:" . $file . " ";

```
        }
       closedir($dh);
   }
}
?>

</body>
</html>
```

==========

Output:

filename: cat.gif
filename: dog.gif
filename: horse.gif

Ex. 2: getcwd() function

returns the current working directory

```
<!DOCTYPE html>
<html>
<body>

<?php

echo  getcwd();
?>

</body>
</html>
```

==========

Output:

/home/php

Ex. 3: opendir() function

opens a directory handle

```
<!DOCTYPE html>
<html>
<body>

<?php
$dir =  "/images/";

// Open a directory, and read its contents
if  (is_dir($dir)){
    if  ($dh = opendir($dir)){
        while  (($file = readdir($dh)) !== false){
            echo  "filename:"  . $file .  "<br>";
        }
       closedir($dh);
   }
}
?>

</body>
</html>
```

==========

Output:

filename: cat.gif
filename: dog.gif
filename: horse.gif

Ex. 4: rewinddir() function
resets the directory handle created by opendir()

Open a directory, list its files, reset directory handle, list all files once again, then close:

```
<!DOCTYPE html>
<html>
<body>

<?php
$dir =  "/images/";
// Open a directory, and read its contents
if  (is_dir($dir)){
   if  ($dh = opendir($dir)){
      // List files in images directory
      while  (($file = readdir($dh)) !== false){
         echo  "filename:"  . $file .  "<br>";
      }
      rewinddir();
      // List once again files in images directory
      while  (($file = readdir($dh)) !== false){
         echo  "filename:"  . $file .  "<br>";
      }
      closedir($dh);
  }
}

?>

</body>
</html>
```

==========
Output:
filename: cat.gif
filename: dog.gif
filename: horse.gif
filename: cat.gif
filename: dog.gif
filename: horse.gif

Ex. 5: scandir() function
returns an array of files and directories of the specified directory

List files and directories inside the images directory:

```
<!DOCTYPE html>
<html>
<body>

<?php
$dir =  "/images/";

// Sort in ascending order - this is default
$a = scandir($dir);

// Sort in descending order
$b = scandir($dir,1);

print_r($a);
print_r($b);
?>
```

	</body> </html> ========== **Output:** Array ([0] => . [1] => .. [2] => cat.gif [3] => dog.gif [4] => horse.gif [5] => myimages) Array ([0] => myimages [1] => horse.gif [2] => dog.gif [3] => cat.gif [4] => .. [5] => .)
Error	The error functions are used to deal with error handling and logging. It allow us to define own error handling rules, and modify the way the errors can be logged. **Ex. 1: debug_backtrace() function** generates a PHP backtrace; displays data from the code that led up to the debug_backtrace() function; Returns an array of associative arrays. <!DOCTYPE html> <html> <body> <?php function a($txt) { b("Glenn"); } function b($txt) { c("Cleveland"); } function c($txt) { var_dump(debug_backtrace()); } a("Peter"); ?> </body> </html> ========== **Output:** array(3) { [0]=> array(4) { ["file"]=> string(10) "/wwwN1LpUo" ["line"]=> int(10) ["function"]=> string(1) "c" ["args"]=> array(1) { [0]=> string(9) "Cleveland" } } [1]=> array(4) { ["file"]=> string(10) "/wwwN1LpUo" ["line"]=> int(7) ["function"]=> string(1) "b" ["args"]=> array(1) { [0]=> string(5) "Glenn" } } [2]=> array(4) { ["file"]=> string(10) "/wwwN1LpUo" ["line"]=> int(15) ["function"]=> string(1) "a" ["args"]=> array(1) { [0]=> string(5) "Peter" } } } **Ex. 2: debug_print_backtrace() function**

prints a PHP backtrace

```
<!DOCTYPE html>
<html>
<body>

<?php
function a($txt) {
    b("Glenn");
}
function b($txt) {
    c("Cleveland");
}
function c($txt) {
    debug_print_backtrace();
}
a("Peter");
?>

</body>
</html>
```

==========

Output:

#0 c(Cleveland) called at [/wwwhrUK51:10] #1 b(Glenn) called at [/wwwhrUK51:7] #2 a(Peter) called at [/wwwhrUK51:15]

Ex. 3: error_get_last() function

returns the last error that occurred (as an associative array)

- [type] - Describes the error type
- [message] - Describes the error message
- [file] - Describes the file where the error occurred
- [line] - Describes the line where the error occurred

```
<!DOCTYPE html>
<html>
<body>

<?php
echo $test;
print_r(error_get_last());
?>

</body>
</html>
```

==========

Output:

Array ([type] => 8 [message] => Undefined variable: test [file] => /wwwwujMzL [line] => 6)

Ex. 4: error_log() function

sends an error message to a log, to a file, or to a mail account

```
<!DOCTYPE html>
<html>
<body>

<?php
// Send error message to the server log if error connecting to the database
if (!mysqli_connect("localhost","bad_user","bad_password","my_db")) {
```

	error_log("Failed to connect to database!", 0); } // Send email to administrator if we run out of FOO if (!($foo = allocate_new_foo())) { error_log("Oh no! We are out of FOOs!", 1, "admin@example.com"); } ?> </body> </html> **Ex. 5: error_reporting() function** specifies which errors are reported. <!DOCTYPE html> <html> <body> <?php // Turn off error reporting error_reporting(0); // Report runtime errors error_reporting(E_ERROR | E_WARNING | E_PARSE); // Report all errors error_reporting(E_ALL); // Same as error_reporting(E_ALL); ini_set("error_reporting", E_ALL); // Report all errors except E_NOTICE error_reporting(E_ALL & ~E_NOTICE); ?> </body> </html>
File System	The filesystem functions allow you to access and manipulate the filesystem **Ex. 1: basename() function** returns the filename from a path <!DOCTYPE html> <html> <body> <?php $path = "/testweb/home.php"; //Show filename echo basename($path) ." "; //Show filename, but cut off file extension for ".php" files echo basename($path,".php"); ?> </body> </html> ========== **Output:**

home.php
home

Ex. 2: chgrp() function
changes the usergroup of the specified file

```
<!DOCTYPE html>
<html>
<body>

<?php
chgrp("test.txt","admin")
?>

</body>
</html>
```

Ex.3: chmod() function
changes permissions of the specified file

```
<!DOCTYPE html>
<html>
<body>

<?php
// Read and write for owner, nothing for everybody else
chmod("test.txt",0600);

// Read and write for owner, read for everybody else
chmod("test.txt",0644);

// Everything for owner, read and execute for everybody else
chmod("test.txt",0755);

// Everything for owner, read for owner's group
chmod("test.txt",0740);
?>

</body>
</html>
```

Ex. 4: chown() function
changes the owner of the specified file

```
<!DOCTYPE html>
<html>
<body>

<?php
chown("test.txt","charles")
?>

</body>
</html>
```

Ex.5: clearstatcache() function
clears the file status cache

```
<!DOCTYPE html>
<html>
<body>

<?php
```

	//output filesize echo filesize("test.txt"); echo " "; $file = fopen("test.txt", "a+"); // truncate file ftruncate($file,100); fclose($file); //Clear cache and check filesize again clearstatcache(); echo filesize("test.txt"); ?> </body> </html> ========== **Output:** 792 100
Filter	This PHP filters is used to validate and filter data coming from insecure sources, like user input. **Ex. 1: filter_has_var() function** checks whether a variable of a specified input type exist <!DOCTYPE html> <html> <body> <?php if (!filter_has_var(INPUT_GET, "email")) { echo("Email not found"); } else { echo("Email found"); } ?> </body> </html> ========== **Output:** Email not found **Ex. 2: filter_id() function** returns filter ID of a specified filter name <!DOCTYPE html> <html> <body> <?php $echo(filter_id("validate_email")); ?> </body> </html> ========== **Output:** 274 **Ex. 3:filter_input() function**

gets an external variable (e.g. from form input) and optionally filters it

```
<!DOCTYPE html>
<html>
<body>

<?php
if  (!filter_input(INPUT_GET,  "email", FILTER_VALIDATE_EMAIL)) {
       echo("Email is not valid");
}  else  {
       echo("Email is valid");
}
?>

</body>
</html>
```

Ex. 4: filter_input_array() function
gets external variables (e.g. from form input) and optionally filters them

```
<!DOCTYPE html>
<html>
<body>

<?php
$filters =  array  (
   "name"  =>  array  ("filter"=>FILTER_CALLBACK,
                                          "flags"=>FILTER_FORCE_ARRAY,
                                          "options"=>"ucwords"
                                         ),
   "age"     =>  array  (  "filter"=>FILTER_VALIDATE_INT,
                                           "options"=>array("min_range"=>1,"max_rang
e"=>120)
                                         ),
   "email"  => FILTER_VALIDATE_EMAIL
   );
print_r(filter_input_array(INPUT_POST, $filters));
?>

</body>
</html>
```

==========

Output:

```
Array
   (
   [name] => Peter
   [age] => 41
   [email] => peter@example.com
   )
```

Ex. 5: filter_list() function
returns a list of all the supported filter names

```
<!DOCTYPE html>
<html>
<body>

<?php
print_r(filter_list());
?>

</body>
```

	</html> ========== **Output:** Array ([0] => int [1] => boolean [2] => float [3] => validate_regexp [4] => validate_domain [5] => validate_url [6] => validate_email [7] => validate_ip [8] => validate_mac [9] => string [10] => stripped [11] => encoded [12] => special_chars [13] => full_special_chars [14] => unsafe_raw [15] => email [16] => url [17] => number_int [18] => number_float [19] => magic_quotes [20] => callback)
FTP	The FTP functions give client access to file servers through the File Transfer Protocol (FTP); used to open, login and close connections, as well as upload, download, rename, delete, and get information on files from file servers. **Ex. 1: ftp_alloc() function** allocates space for a file to be uploaded to the FTP server `<?php` `// connect and login to FTP server` `$ftp_server = "ftp.example.com";` `$ftp_conn = ftp_connect($ftp_server) or die("Could not connect to $ftp_server");` `$login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);` `$file = "/test/myfile";` `// allocate space` `if (ftp_alloc($ftp_conn, filesize($file), $result))` `{` `echo "Space allocated on server. Sending $file.";` `ftp_put($ftp_conn, "/files/myfile", $file, FTP_BINARY);` `}` `else` `{` `echo "Error! Server said: $result";` `}` `// close connection` `ftp_close($ftp_conn);` `?>` **Ex. 2: ftp_cdup() function** changes to the parent directory on the FTP server `<?php` `// connect and login to FTP server` `$ftp_server = "ftp.example.com";` `$ftp_conn = ftp_connect($ftp_server) or die("Could not connect to $ftp_server");` `$login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);` `// change the current directory to php` `ftp_chdir($ftp_conn, "php");` `// change to the parent directory` `if (ftp_cdup($ftp_conn))` `{` `echo "Successfully changed to parent directory.";` `}` `else` `{` `echo "cdup failed.";` `}`

```
// output current directory name
echo  ftp_pwd($ftp_conn);

// close connection
ftp_close($ftp_conn);
?>
```

Ex. 3: ftp_chdir() function
changes the current directory on the FTP server

```
<?php
// connect and login to FTP server
$ftp_server =  "ftp.example.com";
$ftp_conn = ftp_connect($ftp_server)  or  die("Could not connect to $ftp_server");
$login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);

// change the current directory to php
ftp_chdir($ftp_conn,  "php");

// output current directory name (/php)
echo  ftp_pwd($ftp_conn);

// close connection
ftp_close($ftp_conn);
?>
```

Ex. 4: ftp_chmod() function
sets permissions on the specified file via FTP

```
<?php
// connect and login to FTP server
$ftp_server =  "ftp.example.com";
$ftp_conn = ftp_connect($ftp_server)  or  die("Could not connect to $ftp_server");
$login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);

$file =  "php/test.txt";

// Try to set read and write for owner and read for everybody else
if  (ftp_chmod($ftp_conn,  0644, $file) !== false)
  {
   echo  "Successfully chmoded $file to 644.";
  }
else
  {
   echo  "chmod failed.";
  }

// close connection
ftp_close($ftp_conn);
?>
```

Ex. 5: ftp_close() function
closes an FTP connection

```
<?php
// connect and login to FTP server
$ftp_server =  "ftp.example.com";
$ftp_conn = ftp_connect($ftp_server)  or  die("Could not connect to $ftp_server");
$login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);
```

	// then do something... // close connection ftp_close($ftp_conn); ?>
Libxml	The libxml functions and constants are used together with SimpleXML, XSLT and DOM functions **Ex.1: libxml_clear_errors() function** clears the libxml error buffer <?php libxml_clear_errors() ?> **Ex.2: libxml_disable_entity_loader() function** enables the ability to load external entities <?php libxml_disable_entity_loader(false) ?> **Ex.3: ibxml_get_errors() function** gets the errors from the the libxml error buffer <?php libxml_get_errors() ?> **Ex. 4: libxml_get_last_error() function** gets the last error from the libxml error buffer. <?php libxml_get_last_error() ?> **Ex. 5: libxml_set_external_entity_loader() function** changes the default external entity loader <?php $xml = <<<XML <!DOCTYPE foo PUBLIC "-//FOO/BAR" "http://example.com/foobar"> <foo>bar</foo> XML; $dtd = <<<DTD <!ELEMENT foo (#PCDATA)> DTD; libxml_set_external_entity_loader(function ($public, $system, $context) use($dtd) { var_dump($public); var_dump($system); var_dump($context); $f = fopen("php://temp", "r+"); fwrite($f, $dtd); rewind($f); return $f; }); $dd = new DOMDocument; $r = $dd->loadXML($xml); var_dump($dd->validate()); ?>

Mail	The mail() function allows you to send emails directly from a script **Ex. 1: ezmlm_hash() function** calculates the hash value needed when keeping EZMLM mailing lists in a MySQL database `<?php` `$user = "someone@example.com";` `$hash = ezmlm_hash($user);` `echo "The hash value for $user is: $hash.";` `?>` **Ex. 2: mail() function** allows you to send emails directly from a script `<?php` `// the message` `$msg = "First line of text\nSecond line of text";` `// use wordwrap() if lines are longer than 70 characters` `$msg = wordwrap($msg,70);` `// send email` `mail("someone@example.com","My subject",$msg);` `?>`
Math	The math functions can handle values within the range of integer and float types. **Ex. 1: abs() function** returns the absolute (positive) value of a number `<?php` `echo(abs(6.7) . " ");` `echo(abs(-6.7) . " ");` `echo(abs(-3) . " ");` `echo(abs(3));` `?>` ========== **Output:** 6.7 6.7 3 3 **Ex. 2: acos() function** returns the arc cosine of a number `<?php` `echo(acos(0.64) . " ");` `echo(acos(-0.4) . " ");` `echo(acos(0) . " ");` `echo(acos(-1) . " ");` `echo(acos(1) . " ");` `echo(acos(2));` `?>` ========== **Output:** 0.87629806116834 1.9823131728624 1.5707963267949 3.1415926535898 0 NAN

	Ex.3: acosh() function returns the inverse hyperbolic cosine of a number `<?php` `echo(acosh(7) . " ");` `echo(acosh(56) . " ");` `echo(acosh(2.45));` `?>` ========== **Output:** 2.6339157938496 4.7184191423729 1.5447131178707 **Ex. 4: asin() function** returns the arc sine of a number `<?php` `echo(asin(0.64) . " ");` `echo(asin(-0.4) . " ");` `echo(asin(0) . " ");` `echo(asin(-1) . " ");` `echo(asin(1) . " ");` `echo(asin(2));` `?>` ========== **Output:** 0.69449826562656 -0.41151684606749 0 -1.5707963267949 1.5707963267949 NAN **Ex. 5: asinh() function** returns the inverse hyperbolic sine of a number `<?php` `echo(asinh(7) . " ");` `echo(asinh(56) . " ");` `echo(asinh(2.45));` `?>` ========== **Output:** 2.6441207610586 4.7185785811518 1.6284998192842
Misc	The misc. functions were only placed here because none of the other categories seemed to fit **Ex. 1: connection_aborted() function** checks whether the client has disconnected `<?php` `function check_abort()` `{` `if (connection_aborted())` `error_log ("Script $GLOBALS[SCRIPT_NAME]" .` `"$GLOBALS[SERVER_NAME] was aborted by the user.");` `}` `// Some script to be executed here` `// Call the check_abort function when the script ends` `register_shutdown_function("check_abort");` `?>`

	Ex. 2: connection_status() function returns the current connection status `<?php` `switch (connection_status())` `{` `case CONNECTION_NORMAL:` `$txt = 'Connection is in a normal state';` `break;` `case CONNECTION_ABORTED:` `$txt = 'Connection aborted';` `break;` `case CONNECTION_TIMEOUT:` `$txt = 'Connection timed out';` `break;` `case (CONNECTION_ABORTED & CONNECTION_TIMEOUT):` `$txt = 'Connection aborted and timed out';` `break;` `default:` `$txt = 'Unknown';` `break;` `}` `echo $txt;` `?>` ========== **Output:** Connection is in a normal state **Ex. 3: connection_timeout() function** checks whether the script has timed out `<?php` `connection_timeout()` `?` **Ex. 4: constant() function** returns the value of a constant `<?php` `//define a constant` `define("GREETING","Hello you! How are you today?");` `echo constant("GREETING");` `?>` ========== **Output:** Hello you! How are you today? **Ex. 5: define() function** defines a constant `<?php` `define("GREETING","Hello you! How are you today?");` `echo constant("GREETING");` `?>` ========== **Output:** Hello you! How are you today?
MySQLi	The MySQLi functions allows you to access MySQL database servers. **Ex. 1: affected_rows / mysqli_affected_rows() function** returns the number of affected rows in the previous SELECT, INSERT, UPDATE, REPLACE, or DELETE query. `<?php` `$mysqli = new mysqli("localhost","my_user","my_password","my_db");`

```php
if  ($mysqli -> connect_errno) {
   echo  "Failed to connect to MySQL: "  . $mysqli -> connect_error;
   exit();
}
// Perform queries and print out affected rows
$mysqli -> query("SELECT * FROM Persons");
echo  "Affected rows: "  . $mysqli -> affected_rows;

$mysqli -> query("DELETE FROM Persons WHERE Age>32");
echo  "Affected rows: "  . $mysqli -> affected_rows;
$mysqli -> close();
?>
```

Ex.2: autocommit() / mysqli_autocommit() function

turns on or off auto-committing database modifications

```php
<?php
$mysqli =  new  mysqli("localhost","my_user","my_password","my_db");

if  ($mysqli -> connect_errno) {
   echo  "Failed to connect to MySQL: "  . $mysqli -> connect_error;
   exit();
}

// Turn autocommit off
$mysqli -> autocommit(FALSE);
// Insert some values
$mysqli -> query("INSERT INTO Persons (FirstName,LastName,Age)
VALUES ('Peter','Griffin',35)");
$mysqli -> query("INSERT INTO Persons (FirstName,LastName,Age)
VALUES ('Glenn','Quagmire',33)");
// Commit transaction
if  (!$mysqli -> commit()) {
   echo  "Commit transaction failed";
   exit();
}
$mysqli -> close();
?>
```

Ex. 3: change_user() / mysqli_change_user() function

changes the user of the specified database connection, and sets the current database

```php
<?php
$mysqli =  new  mysqli("localhost","my_user","my_password","my_db");

if  ($mysqli -> connect_errno) {
   echo  "Failed to connect to MySQL: "  . $mysqli -> connect_error;
   exit();
}
// Reset all and select a new database
$mysqli -> change_user("my_user",  "my_password",  "test");

$mysqli -> close();?>
```

Ex. 4: character_set_name() / mysqli_character_set_name() function

returns the default character set for the database connection

```php
<?php
$mysqli =  new  mysqli("localhost","my_user","my_password","my_db");

if  ($mysqli -> connect_errno) {
   echo  "Failed to connect to MySQL: "  . $mysqli -> connect_error;
   exit();
}

$charset = $mysqli -> character_set_name();
```

	echo "Default character set is: " . $charset; $mysqli -> close(); ?> **Ex. 5: close() / mysqli_close() function** closes a previously opened database connection <?php $mysqli = new mysqli("localhost","my_user","my_password","my_db"); if ($mysqli -> connect_errno) { echo "Failed to connect to MySQL: " . $mysqli -> connect_error; exit(); } // ....some PHP code... $mysqli -> close(); ?>
Network	The Network functions contains various network function and let you manipulate information sent to the browser by the Web server, before any other output has been sent. **Ex. 1: checkdnsrr() function** checks DNS records for *type* corresponding to *host* <?php $domain="w3schools.com"; if(checkdnsrr($domain,"MX")) { echo "Passed"; } else { echo "Failed"; } ?> **Ex. 2: closelog() function** closes the connection of system logger. <?php function _log($text) { openlog("phperrors", LOG_PID | LOG_PERROR); syslog(LOG_ERR, $text); closelog(); .... }; ?> **Ex. 3: dns_check_record() function** is an alias of the checkdnsrr() function <?php $domain="w3schools.com"; if(dns_check_record($domain,"MX")) { echo "Passed"; } else { echo "Failed"; }?> **Ex. 4: dns_get_mx() function** is an alias of the getmxrr() function <?php $domain="w3schools.com"; if(dns_get_mx($domain,$mx_details)){ foreach($mx_details as $key=>$value){ echo "$key => $value "; } } ?>

	Ex. 5: dns_get_record() function gets the DNS resource records associated with the specified hostname <?php print_r(dns_get_record("w3schools.com", DNS_MX)); ?>
SimpleXML	SimpleXML is an extension that allows us to easily manipulate and get XML data. It provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout. **Ex. 1: __construct() function** creates a new SimpleXMLElement object <?php $note=<<<XML <note> <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Do not forget me this weekend!</body> </note> XML; $xml=new SimpleXMLElement($note); echo $xml->asXML();); ?> **Ex. 2: __toString() function** returns the string content of an element <?php $xml = new SimpleXMLElement("<note>Hello <to>Tove</to><from>Jani</from>World!</note>"); echo $xml; ?> **Ex.3: addAttribute() function** appends an attribute to the SimpleXML element <?php $note=<<<XML <note> <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Do not forget me this weekend!</body> </note> XML; $xml = new SimpleXMLElement($note); // Add attribute to root element $xml->addAttribute("type","private"); // Add attribute to body element $xml->body->addAttribute("date","2014-01-01"); echo $xml->asXML();; ?> **Ex. 4: addChild() function** appends a child element to the SimpleXML element <?php $note=<<<XML <note> <to>Tove</to> <from>Jani</from>

	<heading>Reminder</heading> <body>Do not forget me this weekend!</body> </note> XML; $xml = new SimpleXMLElement($note); // Add a child element to the body element $xml->body->addChild("date","2014-01-01"); // Add a child element after the last element inside note $footer = $xml->addChild("footer","Some footer text"); echo $xml->asXML(); ?> **Ex. 5: asXML() function** returns a well-formed XML string (XML version 1.0) from a SimpleXML object <?php $note=<<<XML <note> <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Do not forget me this weekend!</body> </note> XML; $xml = new SimpleXMLElement($note); echo $xml->asXML(); ?>
Stream	Streams are the way of generalizing file, network, data compression, and other operations which share a common set of functions and uses. In its simplest definition, a stream is a resource object which exhibits streamable behavior. **Ex. 1: stream_bucket_prepend()** **Ex. 2: stream_context_get_default()** **Ex. 3: stream_context_get_params()** **Ex. 4: stream_context_set_options()** **Ex. 5: stream_copy_to_stream() function** copies data from one stream to another <?php $src = fopen("test1.txt", "r"); $dest = fopen("test2.txt", "w"); // Copy 1024 bytes to test2.txt echo stream_copy_to_stream($src, $dest, 1024) . " bytes copied to test2.txt"; ?>
String	The PHP string functions are part of the PHP core. No installation is required to use these functions. **Ex. 1: addcslashes() function** returns a string with backslashes in front of the specified characters <?php $str = addcslashes("Hello World!","W"); echo($str); ?> ========== **Output:** Hello \World!

	Ex. 2: bin2hex() function converts a string of ASCII characters to hexadecimal values <?php $str = bin2hex("Hello World!"); echo($str); ?> ========== **Output:** 48656c6c6f20576f726c6421 **Ex. 3: html_entity_decode() function** converts HTML entities to characters <?php $str = 'w3schools.com'; echo html_entity_decode($str); ?> ========== **Output:** **(view Source)** w3schools.com **Ex. 4: chop() function** removes whitespaces or other predefined characters from the right end of a string <?php $str = "Hello World!"; echo $str . " "; echo chop($str,"World!"); ?> ========== **Output:** Hello World! Hello **Ex. 5: chr() function** returns a character from the specified ASCII value <?php echo chr(52) . " "; // Decimal value echo chr(052) . " "; // Octal value echo chr(0x52) . " "; // Hex value ?> ========== **Output:** 4 * R
XML Parser	The XML functions lets you parse, but not validate, XML documents. XML is a data format for standardized structured document exchange. More information on XML can be found in our XML Tutorial. **Ex.1: utf8_decode() function** decodes a UTF-8 string to ISO-8859-1 <?php $text = "\xE0"; echo utf8_encode($text) ." "; echo utf8_decode($text); ?> ========== **Output:** à ?

Ex.1: utf8_decode() function

decodes a UTF-8 string to ISO-8859-1

```
<?php
$text =  "\xE0";
echo  utf8_encode($text) ."<br>";
echo  utf8_decode($text);
?>
```

==========

Output:

Ex.2: xml_error_string() function

returns the XML parser error description

```
<?php
// Invalid xml file
$xmlfile =  'test.xml';
$xmlparser = xml_parser_create();

// Open the file and read data
$fp = fopen($xmlfile,  'r');
while  ($xmldata = fread($fp,  4096)) {
   // parse the data chunk
   if  (!xml_parse($xmlparser,$xmldata,feof($fp))) {
      die(  print  "ERROR: "
      . xml_error_string(xml_get_error_code($xmlparser))
      .  "<br>Line: "
      . xml_get_current_line_number($xmlparser)
      .  "<br>Column: "
      . xml_get_current_column_number($xmlparser)
      .  "<br>");
   }
}
xml_parser_free($xmlparser);
?>
```

==========

Output:

ERROR: Mismatched tag
Line: 5
Column: 41

Ex.3: xml_get_current_byte_index() function

returns the byte index for an XML parser.

```
<?php
// Invalid xml file
$xmlfile =  'test.xml';
$xmlparser = xml_parser_create();

// Open the file and read data
$fp = fopen($xmlfile,  'r');
while  ($xmldata = fread($fp,  4096)) {
   // parse the data chunk
   if  (!xml_parse($xmlparser,$xmldata,feof($fp))) {
      die(  print  "ERROR: "
      . xml_error_string(xml_get_error_code($xmlparser))
      .  "<br>Line: "
      . xml_get_current_line_number($xmlparser)
      .  "<br>Column: "
      . xml_get_current_column_number($xmlparser)
      .  "<br>Byte Index: "
      . xml_get_current_byte_index($xmlparser)
      .  "<br>");
   }
}
xml_parser_free($xmlparser);
?>
```

	========== **Output:** ERROR: Mismatched tag Line: 5 Column: 41 Byte Index: 72 **Ex.4: xml_get_error_code() function** returns the XML parser error code <?php // Invalid xml file $xmlfile = 'test.xml'; $xmlparser = xml_parser_create(); // Open the file and read data $fp = fopen($xmlfile, 'r'); while ($xmldata = fread($fp, 4096)) { // parse the data chunk if (!xml_parse($xmlparser,$xmldata,feof($fp))) { die(print "ERROR: " . xml_get_error_code($xmlparser) . " Line: " . xml_get_current_line_number($xmlparser) . " Column: " . xml_get_current_column_number($xmlparser) . " "); } } xml_parser_free($xmlparser); ?> ========== **Output:** ERROR: 76 Line: 5 Column: 41 **Ex.5: xml_parse_into_struct() function** parses XML data into an array <?php $xmlparser = xml_parser_create(); $fp = fopen("note.xml", "r"); $xmldata = fread($fp, 4096); // Parse XML data into an array xml_parse_into_struct($xmlparser,$xmldata,$values); xml_parser_free($xmlparser); print_r($values); fclose($fp); ?> ========== **Output:** Array ([0] => Array ([tag] => NOTE [type] => open [level] => 1 [value] =>) [1] => Array ([tag] => TO [type] => complete [level] => 2 [value] => Tove) [2] => Array ([tag] => NOTE [value] => [type] => cdata [level] => 1) [3] => Array ([tag] => FROM [type] => complete [level] => 2 [value] => Jani) [4] => Array ([tag] => NOTE [value] => [type] => cdata [level] => 1) [5] => Array ([tag] => HEADING [type] => complete [level] => 2 [value] => Reminder) [6] => Array ([tag] => NOTE [value] => [type] => cdata [level] => 1) [7] => Array ([tag] => BODY [type] => complete [level] => 2 [value] => Don't forget me this weekend!) [8] => Array ([tag] => NOTE [value] => [type] => cdata [level] => 1) [9] => Array ([tag] => NOTE [type] => close [level] => 1))
Zip	The Zip files functions allows you to read ZIP files. **Ex.1: zip_close() function**

closes a ZIP file archive opened by the zip_open() function

```
<?php
$zip = zip_open("test.zip");
zip_read($zip);
// some code
zip_close($zip);
?>
```

Ex.2: zip_entry_close() function

closes a ZIP directory entry opened by zip_entry_open()

```
<?php
$zip = zip_open("test.zip");

if ($zip) {
   while ($zip_entry = zip_read($zip)) {
      if (zip_entry_open($zip, $zip_entry)) {
         // some code
         // Close directory entry
        zip_entry_close($zip_entry);
      }
  }
zip_close($zip);
}
?>
```

Ex.3: zip_entry_filesize() function

returns the actual file size of a ZIP directory entry

```
<?php
$zip = zip_open("test.zip");
if ($zip) {
   while ($zip_entry = zip_read($zip)) {
      echo "<p>";
      // Get name of directory entry
      echo "Name: " . zip_entry_name($zip_entry) . "<br>";
      // Get filesize of directory entry
      echo "Filesize: " . zip_entry_filesize($zip_entry);
      echo "</p>";
  }
  zip_close($zip);
}
?>
```

==========

Output:

Name: ziptest.txt
Filesize: 59

Name: htmlziptest.html
Filesize: 124

Ex.4: zip_entry_read() function

reads from an open directory entry

```
<?php
$zip = zip_open("test.zip");

if ($zip) {
   while ($zip_entry = zip_read($zip)) {
      echo "<p>Name: " . zip_entry_name($zip_entry) . "<br>";
      // Open directory entry for reading
      if (zip_entry_open($zip, $zip_entry)) {
         echo "File Contents:<br>";
         // Read open directory entry
        $contents = zip_entry_read($zip_entry);
         echo "$contents<br>";
```

	zip_entry_close($zip_entry); } echo " "; } zip_close($zip); } ?> ========== **Output:** Name: ziptest.txt File Contents: Hello World! This is a test. Name: htmlziptest.html File Contents: **Hello World!** This is a test for the **zip functions** in PHP. **Ex.5: zip_entry_filesize() function** returns the actual file size of a ZIP directory entry <?php $zip = zip_open("test.zip"); if ($zip) { while ($zip_entry = zip_read($zip)) { echo " "; // Get name of directory entry echo "Name: " . zip_entry_name($zip_entry) . " "; // Get filesize of directory entry echo "Filesize: " . zip_entry_filesize($zip_entry); echo " "; } zip_close($zip); } ?> ========== **Output:** Name: ziptest.txt Filesize: 59 Name: htmlziptest.html Filesize: 124
Timezones	List of Timezones supported by PHP which are useful with several PHP functions. **Ex. 1: Africa** **Ex.2: Asia** **Ex. 3: Europe** **Ex.4: Indian** **Ex. 5: Pacific**

Activity 3: Regular Expression

1. Define Regular Expression (RegEx) and provide example programming scenario where you can use

Regular Expression – is a regular expression is an object that describes a pattern of characters. It used to perform pattern-matching and "search-and-replace" functions on text.

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to return the function that created the RegExp object's prototype.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var patt = new RegExp("Im Adrian", "g");
  var res = patt.constructor;
  document.getElementById("demo").innerHTML = res;
}
</script>

<!--
output:
Click the button to return the function that created the RegExp object's prototype.

Try it(Button)

function RegExp() { [native code] }
-->
</body>
</html>
```

2. Solve the ff. problem using Regular Expressions.

a. Write a PHP script that checks if a string contains another string
Sample String: 'The quick brown fox'
Test input: 'Fox'
Expected output: Fox is found the string

```
<?php
$str1 = 'The quick brown fox';
if (strpos($str1,'fox') !== false)
 {
   echo 'Fox is found in the string';
 }
else
```

```
{
  echo 'The specific word is not present.';
}
?>
```

b. Write a PHP script that removes the last word from a string.
Sample String: 'The quick brown fox'
Expected output: ‘The quick brown'

```
<?php
$str1 = 'The quick brown fox';
echo preg_replace('/\W\w+\s*(\W*)$/', '$1', $str1)."\n";
?>
```

c. Write a PHP script to remove nonnumeric characters except comma and dot.
Sample String: '/$123,34.00A#'
Expected output: 123,34.00

```
<?php
$str1 = "$12,334.00A";
echo preg_replace("/[^0-9,.]/", "", $str1)."\n";
?>
```

d. Write a PHP script to extract text (within parenthesis) from a string.
Sample String: ‘The quick brown [fox].’
Expected output: Fox

```
<?php
$my_text = 'The quick brown [fox].';
preg_match('#\[(.*?)\]#', $my_text, $match);
print $match[1]."\n";
?>
```

e. Write a PHP script to remove all characters from a string except a-z A-Z 0-9 or " ".
Sample String: ‘abcde$ddfd @abcd)der]’
Expected output: abcdeddfd abcd der

```
<?php
$string = 'abcde$ddfd @abcd )der]';
echo 'Old string : '.$string."\n";
$newstr = preg_replace("/[^A-Za-z0-9 ]/", '', $string);
echo 'New string : '.$newstr."\n";
?>
```

Activity 4: Error Handling

1. List down the different PHP errors. Provide example code on how to handle these errors.

1. Warning Error

The main reason of warning errors are including a missing file. This means that the PHP function call the missing file.

Basic Error Handling : Using the die() function

Error: File Does Not Exist

if the files does not exist, you will get an error like this:

Warning: fopen(mytestfile.txt) [function.fopen]: failed to open stream:
No such file or directory in **C:\webfolder\test.php** on line **2**

To prevent getting an error like this, we need a custom error handler. A Custom Error Handler is a special function that can be called when an error occurs in PHP.

Function to handle errors:

```
function customError($errno, $errstr) {
   echo "<b>Error:</b>  [$errno] $errstr<br>";
   echo "Ending Script";
   die();
}
```

Also, you need to set an error_handler
Syntax:
set_error_handler("customError");

Sample Code:

```
<?php
//error handler function
function  customError($errno, $errstr) {
   echo  "<b>Error:</b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

2. **Parse Error / Syntax Error**
The parse error occurs if there is a syntax mistake in the script; the output is Parse errors. A parse error stops the execution of the script. There are many reasons for the occurrence of parse errors in PHP. The common reasons for parse errors are as follows:

Common reason of syntax errors are:

- Unclosed quotes
- Missing or Extra parentheses
- Unclosed braces
- Missing semicolon

Ex.
Forgot to close a quote:

```
<?
echo "test;
?>
```

Fix:

```
<?
echo "test’’;
?>
```

Forget a semicolon:

```
<?
if ($test){
echo '1'
}
?>
```

Fix:

```
<?
if ($test){
echo '1';}
?>
```

3. Notice Error

Notice errors are minor errors. They are similar to warning errors, as they also don’t stop code execution. Often, the system is uncertain whether it’s an actual error or regular code. Notice errors usually occur if the script needs access to an undefined variable.

Ex:

```
<?php
$a=’’Defined error’’;
Echo ‘‘Notice Error’’;
```

```
Echo $b
?>
```

Result:

Notice error <b />

<b>Notice<b />: Undefined variable : b in <b> [...] [...] <b /> on line <b>4<b />

Fix:

```
<?php
$a=''Defined error'';
Echo ''Notice Error'';
Echo $a
?>
```

4. Fatal Error

Fatal errors are ones that crash your program and are classified as critical errors. An undefined function or class in the script is the main reason for this type of error.

There are three (3) types of fatal errors:

1. **Startup fatal error** (when the system can't run the code at installation)
2. **Compile time fatal error** (when a programmer tries to use nonexistent data)
3. **Runtime fatal error** (happens while the program is running, causing the code to stop working completely)

Ex.

```
<?php

function add($x, $y)
{
        $sum = $x + $y;
        echo "sum = " . $sum;
}
$x = 0;
$y = 20;
add($x, $y);

diff($x, $y);
?>
```

Result:

```
PHP Fatal error: Uncaught Error:
Call to undefined function (diff)
in  /home/36db1ad4634ff7deb7f7347a4ac14d3a.php:12
Stack trace:
#0 {main}
          thrown in /home/36db1ad4634ff7deb7f7347a4ac14d3a.php:12
```

Fix:

```
function add($x, $y)
{
        $sum = $x + $y;
        echo "sum = " . $sum;
}
$x = 0;
$y = 20;
add($x, $y);
?>
```