**ACTIVITY ANSWER SHEET**

|  |  |
| --- | --- |
| Name | **ABATO, LEO CARLIE A.** |
| Section: | **3R1** |

**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;  } |
| 2. if…else | if (condition) {      code to be executed if condition is true;  } else {      code to be executed if condition is false;  } |
| 3. if…else if…else | if (condition) {      code to be executed if this condition is true;  } elseif (condition) {      code to be executed if this condition is true;  } else {      code to be executed if all conditions are false;  } |
| 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;  } |
| 5. for loop | for (init counter; test counter; increment counter) {      code to be executed;  } |
| 6. do while loop | do {      code to be executed;  } while (condition is true); |
| 7. while loop | while (condition is true) {      code to be executed;  } |
| 8. foreach loop | foreach ($array as $value) {      code to be executed;  } |
| 9. break statement | switch (n)  {  case text1:  code executed if n=text1;  break;  case text2:  code executed if n=text2;  break;  default:  code executed if n is different from both text1 and text2;  } |
| 10. continue statement | <?php  Loop (While, do-while, for,)  {  conditions  {  continue;//continue statement  }  code executed;  }  ?> |
| 11. try…catch | //TRY to do something.  try{      //Obviously, 1 is never going to be equal to 2...      if(1 !== 2){          //Throw an exception.          throw new Exception('1 is not equal to 2!');      }  }  //CATCH the exception if something goes wrong.  catch (Exception $ex) {      //Print out the exception message.      echo $ex->getMessage();  } |

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  $input = '1';  if(is\_integer($input)){  // return \*\*TRUE\*\* if it is numeric      echo "A number";  }else{      echo "Not a number";  }  ?>  Output: Not a number  <?PHP  $input = 1;  if(is\_integer($input)){  // return \*\*TRUE\*\* if it is numeric      echo "A number";  }else{      echo "Not a number";  }  ?>  Outpu: 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  $number = 0;  $x = $number % 2;  if($number >= 0){      echo 'Positive';  }else{      echo 'Negative';  }  if($x != 0){      echo ' and Odd';  }else{      echo ' & Even';  }  ?>  Output: Positive & Even  <?php  $number = -1;  $x = $number % 2;  if($number >= 0){      echo 'Positive';  }else{      echo 'Negative';  }  if($x != 0){      echo ' and Odd';  }else{      echo ' & Even';  }  ?>  Output: Negative and Odd |

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  function Palindrome($string){      if (strrev($string) == $string){          return 1;      }      else{          return 0;      }  }    // Driver Code  $input = "ANNA";  if(Palindrome($input)){      echo "Palindrome";  }  else {  echo "Not a Palindrome";  }  ?>  Output: Palindrome  <?php  function Palindrome($string){      if (strrev($string) == $string){          return 1;      }      else{          return 0;      }  }    // Driver Code  $input = "BOGART";  if(Palindrome($input)){      echo "Palindrome";  }  else {  echo "Not a Palindrome";  }  ?>  Output: 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  $number = 4;  $x = 1;  for($i=1;$i<=$number-1;$i++)  {   $x\*=($i+1);  }  echo "$x"."\n";  ?>  Output: 24 |

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  $numOfLines = 1;  for ($i = 1; $i <= 3; $i++) {      for ($j = 1; $j <= $i; $j++) {          echo $numOfLines . "&nbsp;";          $numOfLines++;          if ($j == $i) {              echo "&nbsp;";              echo "<br/>";          }      }  }  ?>  Output:  1  2 3  4 5 6 |

**Activity 2: PHP Built-in Functions**

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

|  |  |
| --- | --- |
| Array | * *allows you to access and manipulate arrays. Simple and multi-dimensional arrays are supported* |
| Calendar | * *an extension that contains functions that simplifies converting between different calendar formats.* |
| Date | * *allows 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.* |
| Directory | * *allows you to retrieve information about directories and their contents.* |
| Error | * used to deal with error handling and logging. It also allows us to define own error handling rules and modify the way the errors can be logged. |
| File System | * allows you to access and manipulate the filesystem. |
| Filter | * used to validate and filter data coming from insecure sources, like user input. |
| FTP | * gives client access to file servers through the File Transfer Protocol (FTP). It used to open, login and close connections, as well as upload, download, rename, delete, and get information on files from file servers. |
| Libxml | * used together with SimpleXML, XSLT and DOM functions. |
| Mail | * allows you to send emails directly from a script. |
| Math | * handles values within the range of integer and float types. |
| Misc | * place if none of the other categories seemed to fit. |
| MySQLi | * allows you to access MySQL database servers. |
| Network | * contains various network function and let you manipulate information sent to the browser by the Web server, before any other output has been sent. |
| SimpleXML | * 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. It also turns an XML document into a data structure you can iterate through like a collection of arrays and objects. |
| Stream | * is 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. That is, it can be read from or written to in a linear fashion, and may be able to fseek() to an arbitrary location within the stream. |
| String | * is a function that is used to manipulate strings. |
| XML Parser | * used to read and update, create and manipulate an XML document. |
| Zip | * it allows you to read ZIP files. |
| Timezones | * are useful with several PHP date functions. |

**Activity 3: Regular Expression**

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

|  |
| --- |
| Regular expressions are powerful pattern matching algorithm that can be performed in a single expression.  Regular expressions use arithmetic operators such as (+,-,^) to create complex expressions.  Regular expressions help you accomplish tasks such as validating email addresses, IP address etc.  <?php  function\_name('/pattern/',subject);  ?>  The first example uses the preg\_match function to perform a simple pattern match for the word guru in a given URL.  <?php  $my\_url = "www.guru99.com";  if (preg\_match("/guru/", $my\_url))  {  echo "the url $my\_url contains guru";  }  else  {  echo "the url $my\_url does not contain guru";  }  ?> |

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

|  |
| --- |
|  |

b. Write a PHP script that removes the last word from a string.

Sample String: 'The quick brown fox'

Expected output: ‘The quick brown'

|  |
| --- |
| $txt = "The quick brown fox";  $str= preg\_replace('/\W\w+\s\*(\W\*)$/', '$1', $txt);  echo $str;  Output: The quick brown |

c. Write a PHP script to remove nonnumeric characters except comma and dot.

Sample String: '/$123,34.00A#'

Expected output: 123,34.00

|  |
| --- |
| <?php  $testString = "/$123,34.00A#";  echo preg\_replace("/[^0-9,.]/", "", $testString);  ?>  Output: 123,34.00 |

d. Write a PHP script to extract text (within parenthesis) from a string.

Sample String: ‘The quick brown [fox].’

Expected output: Fox

|  |
| --- |
| $fullString = "The quick brown [fox].";  $start = strpos('(', $fullString);  $end = strlen($fullString) - strpos(')', $fullString);  $shortString = substr($fullString, $start, $end);  Output: Fox |

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.'';  $newstr = preg\_replace("/[^A-Za-z0-9 ]/", '', $string);  echo 'New string : '.$newstr."\n";  ?>  Output: abcdeddfd abcd der |

**Activity 4: Error Handling**

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

**PHP debug\_backtrace() Function**

<?php  
function a($txt) {  
    b("Glenn");  
}  
function b($txt) {  
    c("Cleveland");  
}  
function c($txt) {  
    var\_dump(debug\_backtrace());  
}  
a("Peter");  
?>

**PHP debug\_print\_backtrace() Function**

<?php  
function a($txt) {  
    b("Glenn");  
}  
function b($txt) {  
    c("Cleveland");  
}  
function c($txt) {  
    debug\_print\_backtrace();  
}  
a("Peter");  
?>

# PHP error\_get\_last() Function

<?php  
echo $test;  
print\_r(error\_get\_last());  
?>

# PHP error\_log() Function

<?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");  
}  
?>

# PHP error\_reporting() Function

<?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);  
?>

# PHP restore\_error\_handler() Function

<?php  
// A user-defined error handler function  
function myErrorHandler($errno, $errstr, $errfile, $errline) {  
    echo "<b>Custom error:</b> [$errno] $errstr<br>";  
    echo " Error on line $errline in $errfile<br>";  
}  
  
// Set user-defined error handler function  
set\_error\_handler("myErrorHandler");  
  
$test=2;  
  
// Trigger error  
if ($test>1) {  
    trigger\_error("A custom error has been triggered");  
}  
  
// Restore previous error handler  
restore\_error\_handler();  
  
// Trigger error again  
if ($test>1) {  
    trigger\_error("A custom error has been triggered");  
}  
?>

# PHP restore\_exception\_handler() Function

<?php  
// Two user-defined exception handler functions  
function myException1($exception) {  
    echo "[" . \_\_FUNCTION\_\_ . "]" . $exception->getMessage();  
}  
function myException2($exception) {  
    echo "[" . \_\_FUNCTION\_\_ . "]" . $exception->getMessage();  
}  
  
set\_exception\_handler("myException1");  
set\_exception\_handler("myException2");  
  
restore\_exception\_handler();  
  
// Throw exception  
throw new Exception("This triggers the first exception handler...");  
?>

# PHP set\_error\_handler() Function

<?php  
// A user-defined error handler function  
function myErrorHandler($errno, $errstr, $errfile, $errline) {  
    echo "<b>Custom error:</b> [$errno] $errstr<br>";  
    echo " Error on line $errline in $errfile<br>";  
}  
  
// Set user-defined error handler function  
set\_error\_handler("myErrorHandler");  
  
$test=2;  
  
// Trigger error  
if ($test>1) {  
    trigger\_error("A custom error has been triggered");  
}  
?>

# PHP set\_exception\_handler() Function

<?php  
// A user-defined exception handler function  
function myException($exception) {  
    echo "<b>Exception:</b> ", $exception->getMessage();  
}  
  
// Set user-defined exception handler function  
set\_exception\_handler("myException");  
  
// Throw exception  
throw new Exception("Uncaught exception occurred!");  
?>

# PHP trigger\_error() Function

<?php  
if ($usernum>10) {  
    trigger\_error("Number cannot be larger than 10");  
}  
?>