PHP two dimensional Array:

Example:

```
<html>
<body>
  <table border="1">
    <tr>
      <th>Title1</th>
      <th>Title2</th>
      <th>Title3</th>
    </tr>
   <?php

      $cars=array(array("ram","sam","beema"),array("harry","poter","bills"),array("dimitri","yong","tong"));

      for($i=0;$i<3;$i++)
      {
           echo "<tr>";
             for($j=0;$j<3;$j++)
             {
              echo "<td>".$cars[$i][$j]."</td>";
             }
           echo "</tr>";
      }


   ?>
  </table>
</body>
</html>
```

Output:

Title1	Title2	Title3
Ram	sam	beema
Harry	poter	bills
dimitri	yong	tong

PHP date and time:

Syntax:

```
$today=date("d-m-Y");
```

Where

d- date
m-month
Y-year

Example:

```
<?php

    $today=date("d-m-Y");
    $today1=date("d.m.Y");
    $today2=date("d,m,Y");
    $today3=date("d:m:Y");

    $today4=date("m:d:Y");

    $year=date("Y");
    $dat=date("d");
    $month=date("m");
    echo $today."<br>";
    echo $today1."<br>";
    echo $today2."<br>";
    echo $today3."<br>";
```

```
    echo $today4."<br>";
    echo $year."<br>";
    echo $dat."<br>";
    echo $month."<br>";
?>
```

Output:

03-11-2017
03.11.2017
03,11,2017
03:11:2017
11:03:2017
2017
03
11

Time:

Syntax:

```
        $today=date("h:i-sa");
```

Where

h-hour
i-min
s-second

Example:

```
<?php
    date_default_timezone_set("Asia/kolkata");
    $today=date("h:i-sa");
    $day=date("l");
    echo $today."<br>";
    echo $day."<br>";
?>
```

Output:

08:01-09am
Friday

Include:

Syntax:

```
    include  'filename';
```

-include is used to include a file to current programme

Example:

```
<html>
<body>

   <?php
     $a=10;
     $b=40;
     include 'sum.php';
     echo "<br>";
     include 'fact.php';
   ?>
</body>
</html>
```

Sum.php

```
<?php
    echo "SUM OF A B:".($a+$b);
?>
```

Fact.php

```
<?php
    $result=1;
```

```
    for($i=$a;$i>0;$i--)
    {
       $result=$result*$i;
    }
    echo "Fact of A:".($result);

?>
```

Output:

```
SUM OF A B:50
Fact of A:3628800
```

File Operations:

PHP Open File - fopen()

A better method to open files is with the fopen() function

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

```
$myfile = fopen(filename, mode) or die("Unable to open file!");
```

The file may be opened in one of the following modes:

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
A	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
X	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

Example:

```
<?php
	$myfile = fopen("1.txt", "r") or die("Unable to open file!");
	echo fread($myfile,filesize("1.txt"));
	fclose($myfile);
 ?>
```

PHP Read Single Line - fgets()

The fgets() function is used to read a single line from a file.

Example:

```
<?php
	$myfile = fopen("1.txt", "r") or die("Unable to open file!");
	echo fgets($myfile);
	fclose($myfile);
?>
```

PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof()

function is useful for looping through data of unknown length.

PHP Read Single Character - fgetc()

The fgetc() function is used to read a single character from a file

Example: How to read contents of file character by character

```
<?php
	$myfile = fopen("1.txt", "r") or die("Unable to open file!");
	// Output one character until end-of-file
	while(!feof($myfile)) {
		echo fgetc($myfile);
	}
	fclose($myfile);
?>
```

PHP Create File - fopen()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Syntax:

```
$myfile = fopen("testfile.txt", "w")
```

PHP Write to File - fwrite()

The fwrite() function is used to write to a file.

The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written

Example:

```
<?php
	$myfile = fopen("2.txt", "w") or die("Unable to open file!");
	$txt = "John Doe\n";
	fwrite($myfile, $txt);
	fclose($myfile);
?>
```

Upload File:

Upload.html is used to upload a file to server
Load.php is used to save the uploaded file to server.

Upload.html

```
<html>
<head>
	<title></title>
</head>
<body>
  <form action="load.php" method="post" enctype="multipart/form-data">
	  <input type="file" name="fileToUpload"  id="fileToUpload">
	  <input type="submit" value="Upload Image" name="submit">
  </form>
</body>
</html>
```

Load.php

```
<?php
	$target_dir = "uploads/";
	$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
	if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file))
	{
		echo " The file  has been uploaded.";
	}
	else
	{
		echo "Sorry, there was an error uploading your file.";
	}
?>
```

PHP Error Handling

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

If the file does not exist you might get an error like this:

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

To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt")) {
  die("File not found");
} else {
  $file=fopen("welcome.txt","r");
}
?>
```

Now if the file does not exist you get an error like this:

```
File not found
```

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

Creating a Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Syntax

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

Parameter	Description

error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()

4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Now lets create a function to handle errors:

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

The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

Now that we have created an error handling function we need to decide when it should be triggered.

Set Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Since we want our custom function to handle all errors, the set_error_handler() only needed one parameter, a second parameter could be added to specify an error level.

Example

Testing the error handler by trying to output variable that does not exist:

```
<?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);
?>
```

The output of the code above should be something like this:

```
Error: [8] Undefined variable: test
```

Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?php
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below");
}
?>
```

The output of the code above should be something like this:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
- E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

Example

In this example an E_USER_WARNING occurs if the "test" variable is bigger than "1". If an E_USER_WARNING occurs we will use our custom error handler and end the script:

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

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

//trigger error
$test=2;
```

```
if ($test>=1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

The output of the code above should be something like this:

```
Error: [512] Value must be 1 or below
Ending Script
```

Exception:

Example:

```
<?php

  function read($i)
  {
     if($i==0)
     {
       throw new Exception("div error");
     }
     $result=10/$i;
     return $result;
  }

  try
  {

    echo read(0);
  }
  catch(Exception $e)
  {
     echo   $e->getMessage();
  }
?>
```

Output:

div error

creating custom exception in php

-you need to extend Exception class in php to create a custome exception

Example:

```
<?php
   class EvenNumberException extends Exception
   {
        var $errorinfo="";
        function __construct($info)
        {
          $this->errorinfo=$info;
        }
        function sendError()
        {
          return $this->errorinfo;
        }
   }
   function give($i)
   {
     if(($i%2)!=0)
     {
```

```
        throw new EvenNumberException("even number exception".$i);
      }
      else
      {

      }
    }
    try
    {
       give(7);
    }
    catch(EvenNumberException $e)
    {
      echo $e->sendError();
    }
?>
```