Functions

- A function is a self-contained block of code that performs a specific task.
- PHP provides us with two major types of functions:
- Built-in functions : PHP provides us with huge collection of built-in library functions.
- These functions are already coded and stored in form of functions.
- To use those we just need to call them as per our requirement like, var_dump, fopen(), print_r(), gettype() and so on.

Functions

- User Defined Functions : Apart from the built-in functions, PHP allows us to create our own customized functions called the user-defined function
- Using this we can create our own packages of code and use it wherever necessary by simply calling it.

Advantages Of Function

1. Functions reduces the repetition of code within a program — Function allows you to extract commonly used block of code into a single component. Now you can perform the same task by calling this function wherever you want within your script without having to copy and paste the same block of code again and again.
2. Functions makes the code much easier to maintain — Since a function created once can be used many times, so any changes made inside a function automatically implemented at all the places without touching the several files.

Advantages Of Function

3. Functions makes it easier to eliminate the errors — When the program is subdivided into functions, if any error occur you know exactly what function causing the error and where to find it. Therefore, fixing errors becomes much easier.
4. Functions can be reused in other application — Because a function is separated from the rest of the script, it's easy to reuse the same function in other applications just by including the php file containing those functions.

Creating a Function

- The basic syntax of creating a custom function can be give with:

1. Any name ending with an open and closed parenthesis is a function.
2. A function name always begins with the keyword function.
3. To call a function we just need to write its name followed by the parenthesis
4. A function name cannot start with a number. It can start with an alphabet or underscore.
5. A function name is not case-sensitive.

Creating a Function

```
<?php
   function function_name(param_1, ... , param_n)
   {
      statement_1;
      statement_2;
      ...
      statement_m;

      return return_value;
   }
?>
```

Creating a Function

- Example
- <?php
- function myFunction(){
- echo "Example PHP function";
- }
- myFunction();
- ?>

Creating a Function

- Example
- <?php
- function myFunction(){
- echo "Example PHP function";
- }
- myFunction();
- ?>

Passing parameters

- Passing parameters
- <?php Function add($a,$b){
- $total = $a + $b;
- return $total; //Function return, No output is shown
  }
- echo add(55,20); //Outputs 75
- ?>

Returning Values from a Function

- A function can return a value back to the script that called the function using the return statement. The value may be of any type
- <?php function hello(){
- return "Hello World"; // No output is shown
- }

  $txt = hello();//Assigns the return value "Hello World" to $txt
- echo $txt; // Direct Displays "Hello World“
- ?>

Passing parameters

- PHP allows us two ways in which an argument can be passed into a function:
- Pass by Value: On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged. That means a duplicate of the original value is passed as an argument.
- Pass by Reference: On passing arguments as pass by reference, the original value is passed. Therefore, the original value gets altered. In pass by reference we actually pass the address of the value, where it is stored using ampersand sign(&).

Passing parameters

```
<?php
  function valGeek($num) {  // pass by value
    $num += 2;
    return $num;
}
  function refGeek(&$num) {// pass by reference
    $num += 10;
    return $num;
}
  $n = 10;       valGeek($n);
echo "The original value is still $n \n";
refGeek($n);
echo "The original value changes to $n";
  ?>
```

isset() function

- The isset function is used to check if a variable is set or not.
- That means it determines if a variable is assigned a value and is not null.
- Also, the isset PHP function checks if the given variable is not unset by using the unset function.
- If multiple variables are supplied, then this function will return true only if all of the variables are set.
- A variable can be unset with the unset() function.
- Syntax : isset(variable, ...);

variable	Required. Specifies the variable to check
...	Optional. Another variable to check

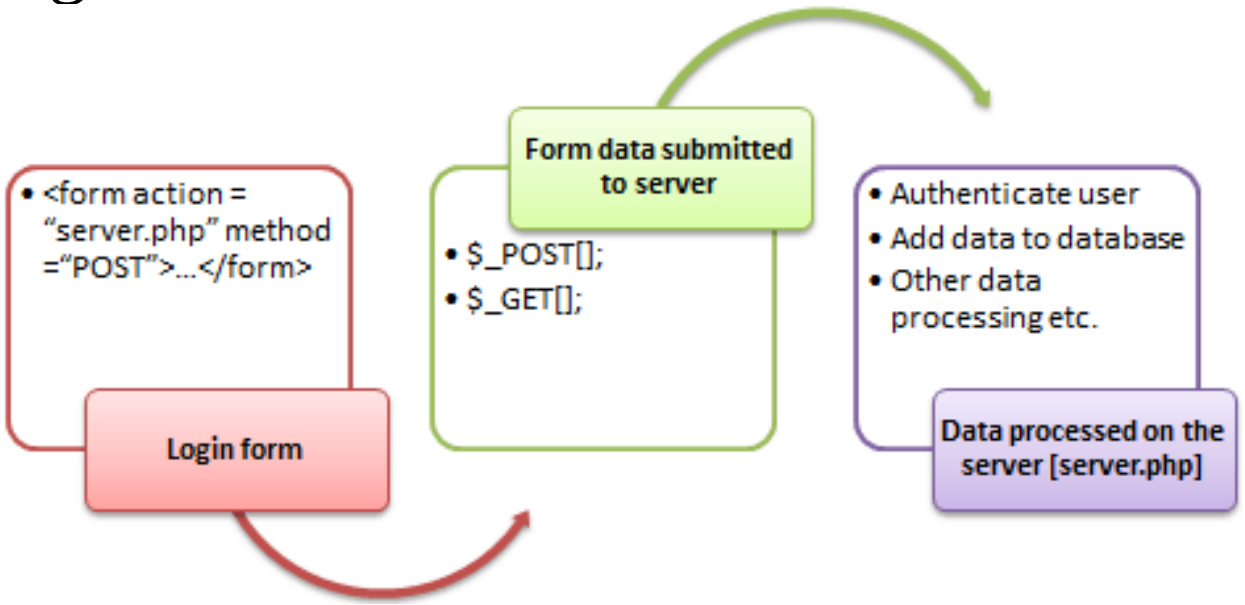
isset() function

```
<?php
$a = 0;
if (isset($a)) {                          // True because $a is set
  echo "Variable 'a' is set.<br>";
}
$b = null;
if (isset($b)) {                          // False because $b is NULL
  echo "Variable 'b' is set.";
}
?>
```

Output : Variable ‘a’ is set.

What is Form?

- When you login into a website or into your mail box, you are interacting with a form.
- Forms are used to get input from the user and submit it to the web server for processing.
- The diagram below illustrates the form handling process.



When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database.
- Create a form
- We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.
- Opening and closing form tags <form>…</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons,checkboxes etc.

Creating forms

- <html>
- <body>
- <form action="registration.php" method="post">
  Name: <input type="text" name="name">
- Email: <input type="text" name="email">
- <input type="submit">
- </form>
- </body>
- </html>

Creating forms

- Notice that there are two attributes within the opening <form> tag:
- The action attribute references a PHP file "process-form.php" that receives the data entered into the form when user submit it by pressing the submit button.
- The method attribute tells the browser to send the form data through POST method.
- Rest of the elements inside the form are basic form controls to receive user inputs.

Creating forms

- Capturing Form Data with PHP
- To access the value of a particular form field, you can use the following super global variables. These variables are available in all scopes throughout a script.

Superglobal	Description
$_GET	Contains a list of all the field names and values sent by a form using the get method (i.e. via the URL parameters).
$_POST	Contains a list of all the field names and values sent by a form using the post method (data will not visible in the URL).
$_REQUEST	Contains the values of both the $_GET and $_POST variables as well as the values of the $_COOKIE superglobal variable.

Creating forms

- When the user fills out and submits the form, then form data will be sent to PHP file: called registration.php.
- registration.php page has following code to print submitted data.
- <html>
- <body> Welcome <?php echo $_POST["name"];
- ?>!
- You email address is <?php echo $_POST["email"]; ?> </body>
- </html>

Form Get/post method and php $_get/$_post

- There are two ways the browser(client) can send information to the web server.
- The GET Method
- The POST Method
- php $_get
- In PHP, the $_GET variable is used to collect values from HTML forms using method get.
- Information sent from an HTML form with the GET method is displayed in the browser's address bar, and it has a limit on the amount of information to send.

Form Get/post method and php $_get/$_post

- php $_get
- <html>
- <body>
- <form action="registration.php" method="get">
  Name: <input type="text" name="name">
- Email: <input type="text" name="email">
- <input type="submit">
- </form>
- </body>
- </html>

Form Get/post method and php $_get/$_post

- When the user clicks on the "Submit button", the URL will be something like this:

- registration.php looks like this:
- <html>
- <body> Welcome <?php echo $_GET["name"]; ?>! You email address is <?php echo $_GET["email"]; ?>
- </body> </html>

the GET method

- The variable names and values will be visible in URL if HTML forms submitted by the GET method.
- The GET method is restricted to send up to 2048 characters only.
- When you submit sensitive information like passwords then should not use this method.
- GET method can't be used, to send binary data like images and Word documents.
- GET method data can be accessed using PHP QUERY_STRING environment variable.
- PHP $_GET associative array is used to access all the sent information by GET method

PHP $_post method

- In PHP, the $_POST variable is used to collect values from HTML forms using method post.
- Information sent from a form with the POST method is invisible and has no limits on the amount of information to send.
- registration.php looks like this:
- Welcome <?php echo $_POST["name"]; ?>! You email address is <?php echo $_POST["email"]; ?>
- When the user clicks on the "Submit button", the URL will be something like

Use of method = post

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header, so security depends on HTTP protocol. By using Secure HTTP, you can make sure that your information is secure.
- PHP $_POST associative array is used to access all the sent information by POST method.
- Variables are not visible in the URL so users can't bookmark your page.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

PHP - File Inclusion

- You can include the content of a PHP file into another PHP file before the server executes it.
- There are two PHP functions which can be used to included one PHP file into another PHP file.
- The include() Function
- The require() Function
- This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

The include() Function

- The include() function takes all the text in a specified file and copies it into the file that uses the include function.
- If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.
- Assume you want to create a common menu for your website. Then create a file menu.php with the following content.
- The basic syntax of the include()
- include("path/to/filename");-Or-
  include "path/to/filename“;

The include() Function

- Lets have a file called even.php with the following code:
- <?php // file to be included
- echo “Welcome to IPL”?>
- Now let us try to include this file into another php file index.php file. We will see that the contents of both the file are shown.
- <?php
- include("even.php");
- echo "
Above File is Included"
- ?>

The require() Function

- The require() function takes all the text in a specified file and copies it into the file that uses the include function.
- If there is any problem in loading a file then the require() function generates a fatal error and halt the execution of the script.
- The basic syntax of the require()
- require("path/to/filename");-Or-
  require "path/to/filename";

The require() Function

- Lets have a file called even.php with the following code:
- <?php // file to be included
- echo “Welcome to IPL"
- ?>
- Now if we try to include this file using require() function this file into a web page we need to use a index.php file. We will see that the contents of both the file are shown.
- <?php
- require("even.php");
- echo "
Above File is Required"
- ?>

Difference Between include and require

- Typically the require() statement operates like include().
- The only difference is — the include() statement will only generate a PHP warning but allow script execution to continue if the file to be included can't be found,
- whereas the require() statement will generate a fatal error and stops the script execution.

Cookies

- A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too.
- A cookie is often used to identify a user. With PHP, you can both create and retrieve cookie values.
- 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

1. Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
2. Browser stores this information on local machine for future use.
3. 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.

Setting a Cookie in PHP

- The setcookie() function is used to set a cookie in PHP.
- Make sure you call the setcookie()function before any output generated by your script otherwise cookie will not set. The basic syntax of this function can be given with:
- setcookie(name, value, expire, path, domain, secure);

Setting a Cookie in PHP

- The parameters of the setcookie() function have the following meanings:

Parameter	Description
name	The name of the cookie.
value	The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.
expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
path	Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.
domain	Specify the domain for which the cookie is available to e.g www.example.com.
secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.

Setting a Cookie in PHP

- Here's an example that uses setcookie() function to create a cookie named username and assign the value value John Carter to it. It also specify that the cookie will expire after 30 days (30 days * 24 hours * 60 min * 60 sec).:

```
<?php
// Setting a cookie
setcookie("username“,"John Carter",
time()+30*24*60*60);
 ?>
```

Accessing Cookies Values

- The PHP $_COOKIE superglobal variable is used to retrieve a cookie value.
- It typically an associative array that contains a list of all the cookies values sent by the browser in the current request, keyed by cookie name.
- The individual cookie value can be accessed using standard array notation,
- Example

```
<?php // Accessing an individual cookie value
echo $_COOKIE["username"];
?> output. John Carter
```

Accessing Cookies Values

- 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:
- Example

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

Removing Cookies

- 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);
?>
```

Session

- A session creates a file in a temporary directory on the server where registered session variables and their values are stored.
- This data will be available to all pages on the site during that visit.
- In a session based environment, every user is identified through a unique number called session identifier or SID.
- This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Starting a PHP Session

- Before you can store any information in session variables, you must first start up the session.
- To begin a new session, simply call the PHP session_start() function.
- It will create a new session and generate a unique session ID for the user.

```
<?php
//Starting session
session_start();
 ?>
```

Starting a PHP Session

- The session_start() function first checks to see if a session already exists by looking for the presence of a session ID.
- If it finds one, i.e. if the session is already started, it sets up the session variables and
- if doesn't, it starts a new session by creating a new session ID..

Storing and Accessing Session Data

- You can store all your session data as key-value pairs in the $_SESSION[] superglobal array.
- The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

```
<?php
//Starting session
session_start(); // Storing data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Parker";
?>
```

Storing and Accessing Session Data

- To access the session data we set on our previous example from any other page on the same web domain — simply recreate the session by calling session_start() and then pass the corresponding key to the $_SESSION associative array..
- <?php // Starting session session_start();//Accessing session data
- echo'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"]; ?>

Destroying a Session

- If you want to remove certain session data, simply unset the corresponding key of the $_SESSIONassociative array, as shown in the following example:
- ```
  <?php // Starting session
  session_start();// Removing session data
  if(isset($_SESSION["lastname"])){
  unset($_SESSION["lastname"]); }
  ```
- ```
  ?>
  ```

Destroying a Session

- to destroy a session completely, simply call the session_destroy() function. This function does not need any argument and a single call destroys all the session data.

```
<?php// Starting session
session_start();//Destroying session
session_destroy();
 ?>
```

Recap: a comparison

	COOKIES	SESSIONS
Where is data stored?	Locally on client	Remotely on server
Expiration?	Variable – determined when cookie is set	Session is destroyed when the browser is closed
Size limit?	Depends on browser	Depends only on server (practically no size limit)
Accessing information?	$_COOKIE	$_SESSION
General use?	Remember small things about the user, such as login name. Remember things after re-opening browser	Remembering varying amount of data about the user in one browsing "session"

Passing information between pages

- Form is an HTML element used to collect information from the user in a sequential and organized manner.
- This information can be sent to the back-end services if required by them, or it can also be stored in a database using DBMS like MySQL.
- Splitting a form into multiple steps or pages allow better data handling and layering of information. This can be achieved by creating browser sessions.
- HTML sessions are a collection of variables that can be used to maintain the state of the form attributes while the user switched between the pages of the current domain. Session entries will be deleted as soon as the user closes the browser or leaves the site.

Passing information between pages

- Simple Way for passing a value
- <html>
- <body>

- <form action="welcome.php" method="post">
  Name: <input type="text" name="name">
  E-mail: <input type="text" name="email">
  <input type="submit">
- </form>

- </body>
- </html>.

Passing information between pages

- Simple Way for passing a value
- Here the welcome.php page
- <html>
- <body>

- Welcome <?php echo $_POST["name"]; ?>
- Your email address is: <?php echo $_POST["email"]; ?>

- </body>
- </html>

String

- A string variable is used to store and manipulate text.

String variables are used for values that contain characters.

- After we have created a string variable we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Consider the following code:

```
<?php
$txt="Hello world!";
echo $txt;
?>
```

Output is: Hello world !

Note: When you assign a text value to a variable, remember to put single or double quotes around the value.

String

The PHP Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to join two string values together.

The example below shows how to concatenate two string variables together:

```
<?php
$txt1="Hello world!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

Hello world! What a nice day!

String

PHP string manipulation function:

1. The strlen() function returns the length of a string, in characters.

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

Length is=12

2. The strpos() function is used to search for a character or a specific text within a string.

If a match is found, it will return the character position of the first match. If no match is found, it will return FALSE.

String

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

Output = 6

3. The strtoupper() function converts a string to uppercase.

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

The output of the code above will be:

HELLO WORLD!

String.

The strtolower() function converts a string to lowercase.

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

The output of the code above will be:

hello world.

String

- The ucfirst() function converts the first character of a string to uppercase.

```
<?php
 echo ucfirst("hello world");
  ?>
```

The output of the code above will be:

Hello world

String

- The ucwords() function converts the first character of each word in a string to uppercase.

```
<?php
 echo ucwords("hello world");
 ?>
```

The output of the code above will be:

Hello World

String

The strcmp() function compares two strings.

Syntax:strcmp(string1,string2)

It returns: 0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

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

The output of the code above will be: **0**

String

- The substr(str,pos) function returns a part of a string.

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

The output of the code above will be: **world!**

```
<?php
 echo substr("Hello world!",6,5);
 ?>
```

The output of the code above will be: **world**

String

The trim() function removes whitespaces and other predefined c

Syntax: trim(string , charlist) where charlist is optional and specifies which characters to remove from the string. If omitted, all of the following characters are removed:

"\0" - NULL
"\t" - tab
"\n" - new line
"\x0B" - vertical tab
"\r" - carriage return
" " - ordinary white space

String

```
<?php
$str = " Hello World! ";
echo "With trim: " . trim($str);
?>
```

The browser output of the code above will be:

With trim: Hello World!

String

```
<?php
$str = "\r\nHello World!\r\n";
echo "With trim: " . trim($str);
?>
```

The browser output of the code above will be:

With trim: Hello World!

Types of Errors

- Error is the fault or mistake in a program.
- Sometimes your application will not run as it supposed to do, resulting in an error. There are a number of reasons that may cause errors, for example:
- The Web server might run out of disk space
- A user might have entered an invalid value in a form field
- The file or database record that you were trying to access may not exist
- The application might not have permission to write to a file on the disk
- A service that the application needs to access might be temporarily unavailable

Types of Errors

- PHP contains basically four main type of errors.

1. Parse error or Syntax Error: A syntax error is a mistake in the syntax of source code, which can be done by programmers due to their lack of concern or knowledge. It is also known as Parse error.

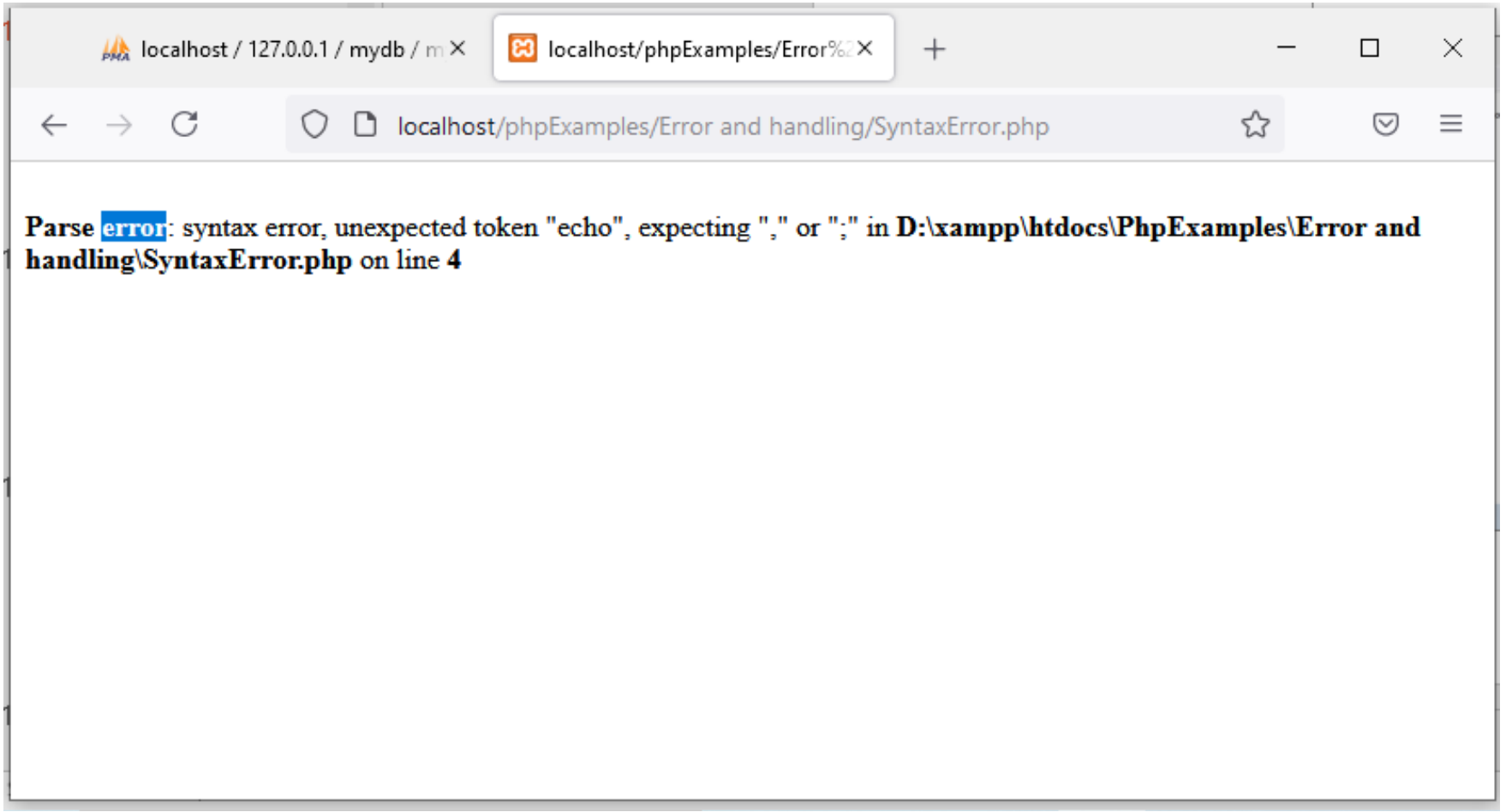
- Compiler is used to catch the syntax error at compile time.

- Parse errors can be caused dues to unclosed quotes, missing or Extra parentheses, Unclosed braces, Missing semicolon etc

Types of Errors

- Common reason of syntax errors are:
- Unclosed quotes
- Missing or Extra parentheses
- Unclosed braces
- Missing semicolon
- Example
- <?php
- echo "Cat";
- echo "Dog"
- echo "Lion";
- ?>

Types of Errors

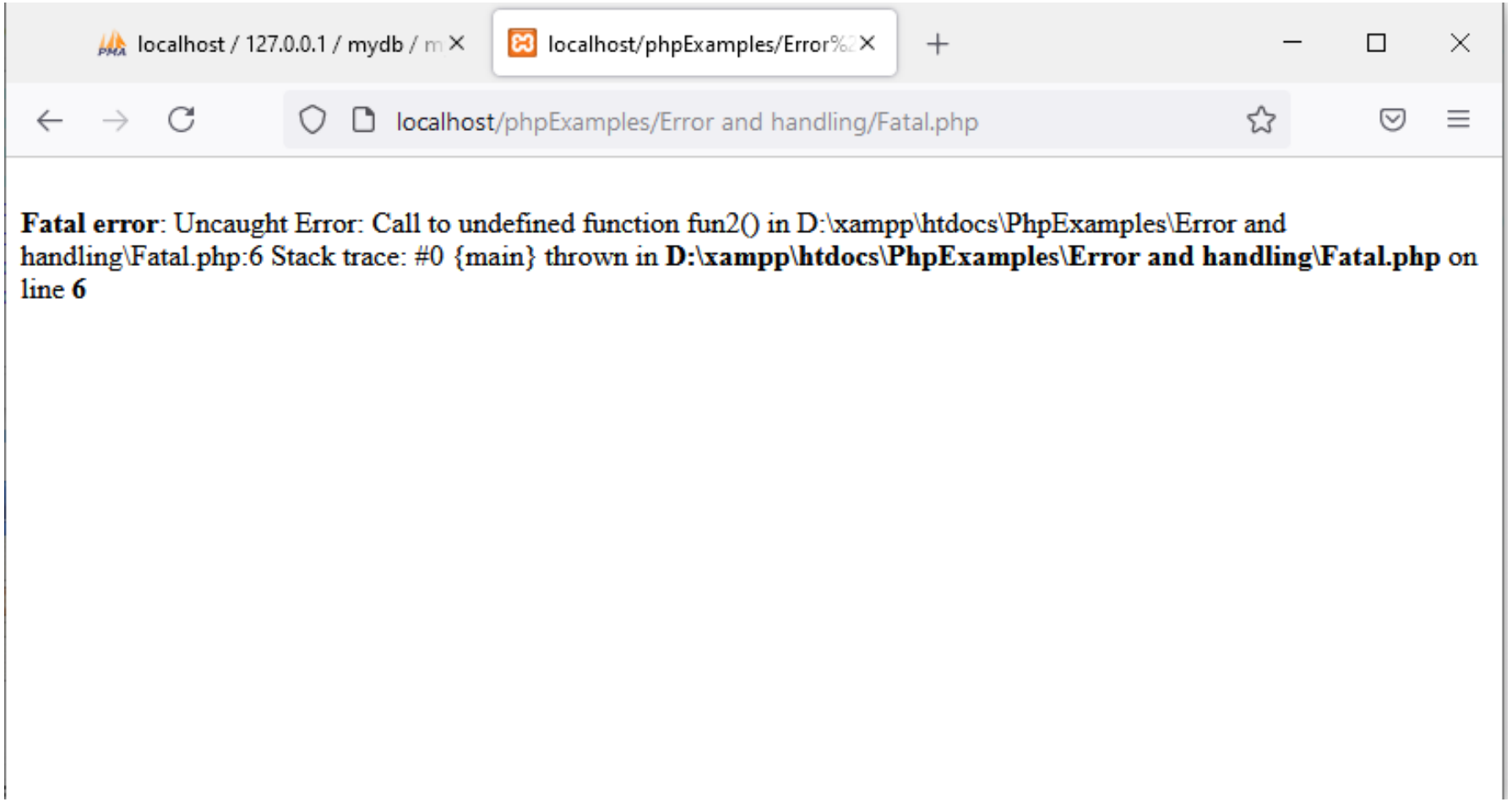


Types of Errors

2. Fatal Error: It is the type of error where PHP compiler understand the PHP code but it recognizes an undeclared function. This means that function is called without the definition of function.

```
<?php
function fun1() {
        echo “Welcome to PHP";
}
fun2();
        echo "Fatal Error !!";
?>
```

Types of Errors



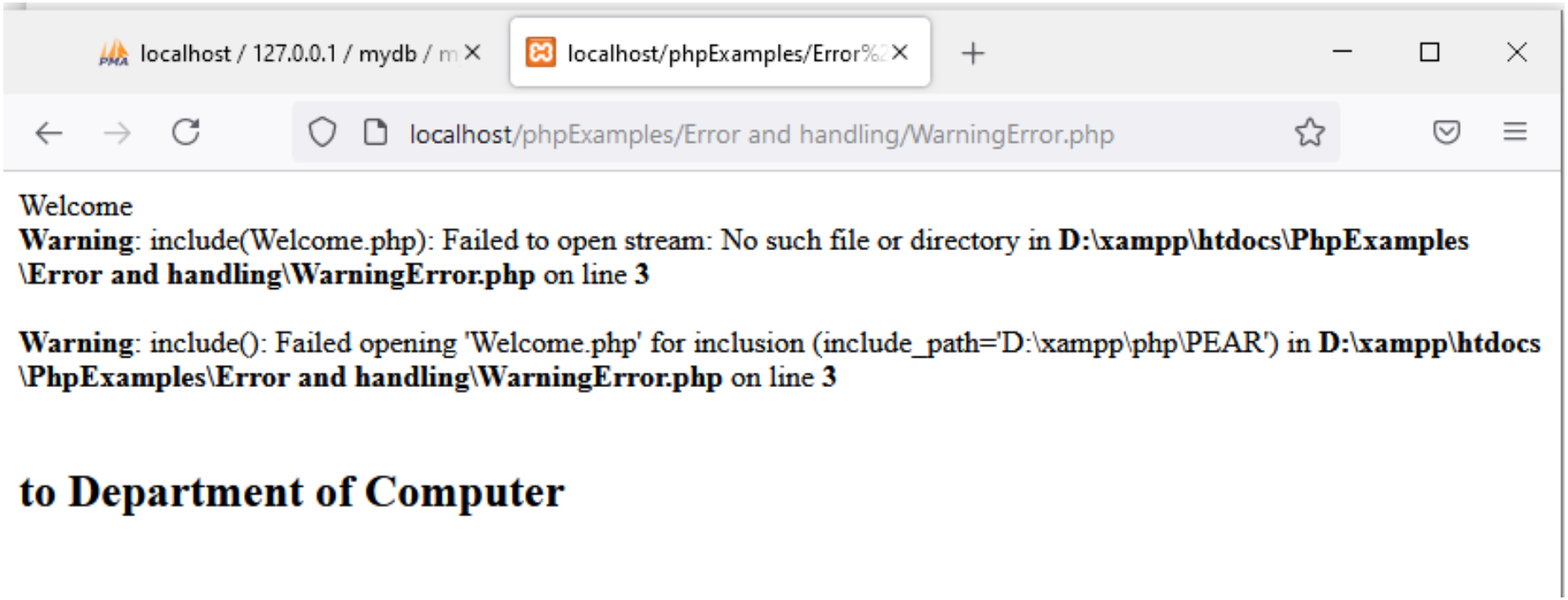
Types of Errors

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

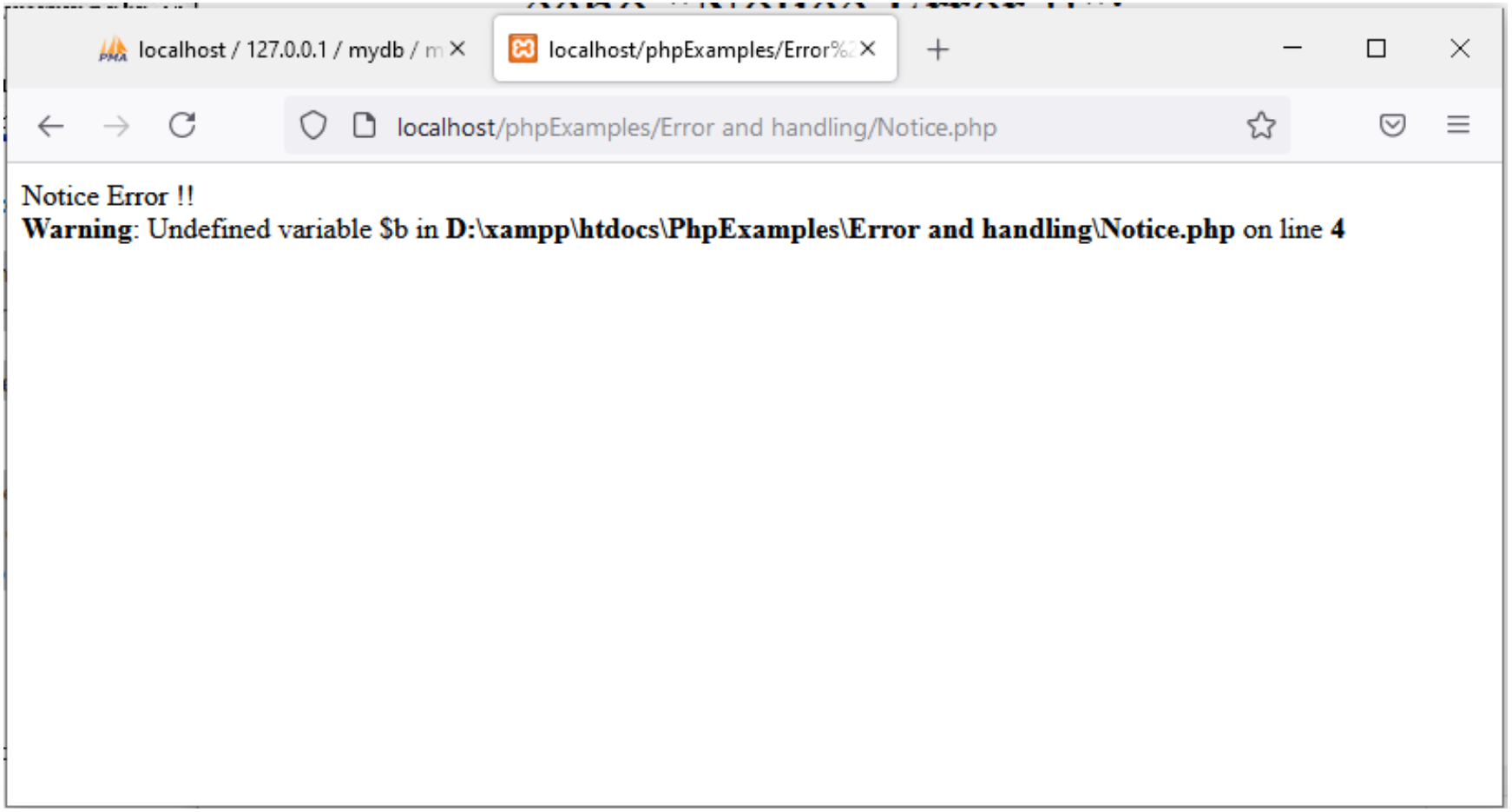
- Example:

```
<?php
echo "Welcome";
include ("Welcome.php");
echo "<br> <h2>to Department of Computer</h2>";
?>
```

Types of Errors



Types of Errors



Types of Errors

4. Notice Error: Notice that an error is the same as a warning error i.e. in the notice error execution of the script does not stop. Notice that the error occurs when you try to access the undefined variable, then produce a notice error.

Example

```
<?php
   $a=“welcome to PHP";
   echo "Notice Error !!";
   echo $b;
?>
```

Error Levels

- Usually, when there's a problem that prevents a script from running properly, the PHP engine triggers an error.
- Each error is represented by an integer value and an associated constant. The following table list some of the common error levels:

Error Level	Value	Description
E_ERROR	1	A fatal run-time error, that can't be recovered from. The execution of the script is stopped immediately.
E_WARNING	2	A run-time warning. It is non-fatal and most errors tend to fall into this category. The execution of the script is not stopped.

Error Levels

Error Level	Value	Description
E_NOTICE	8	A run-time notice. Indicate that the script encountered something that could possibly an error, although the situation could also occur when running a script normally.
E_USER_ERROR	256	A fatal user-generated error message. This is like an E_ERROR, except it is generated by the PHP script using the function trigger_error() rather than the PHP engine.
E_USER_WARNING	512	A non-fatal user-generated warning message. This is like an E_WARNING, except it is generated by the PHP script using the function trigger_error()rather than the PHP. engine
E_USER_NOTICE	1024	A user-generated notice message. This is like an E_NOTICE, except it is generated by the PHP script using the function trigger_error() rather than the PHP engine.

Error Levels

Error Level	Value	Description
E_STRICT	2048	Not strictly an error, but triggered whenever PHP encounters code that could lead to problems or forward incompatibilities
E_ALL	8191	All errors and warnings, except of E_STRICT prior to PHP 5.4.0.

Error Handling

- When an error occurs, depending on your configuration settings, PHP displays the error message in the web browser with information relating to the error that occurred.
- PHP offers a number of ways to handle errors.
- We are going to look at three (3) commonly used methods;

1. Die statements– the die function combines the echo and exit function in one. It is very useful when we want to output a message and stop the script execution when an error occurs.

Error Handling

- Custom error handlers – these are user defined functions that are called whenever an error occurs.

- PHP error reporting – the error message depending on your PHP error reporting settings. This method is very useful in development environment when you have no idea what caused the error. The information displayed can help you debug your application.

Error Handling

- Syntax:
- die($message)
- Example:
- <?php
- // Php code showing default error handling
- $file = fopen(“data.txt", "w");
- ?>
- Note: Run the above code and data.txt file is not present then it will display an run-time error message.
- Warning: fopen(data.txt) [function.fopen]: failed to open stream:
- No such file or directory in C:\webfolder\test.php on line 2

Error Handling

- To avoid that the user gets an error message like the one above, we test if the file exist before we try to access it
- ```
  <?php
  if(file_exists("sample.txt")){
  $file = fopen("sample.txt", "r"); }
  else{
  ```
- ```
  die("Error: The file you are trying
  to access doesn't exist."); } ?>
  ```

Error Handling

2. Custom Error handling: Creating a custom error handler in PHP is quite simple. Create a function that can be called when a error has been occurred in PHP.

- Syntax:
- error_function($error_level, $error_message, $error_file, $error_line, $error_context)
- Parameters: This function accepts five parameters as mentioned above and described below:
- $error_level: It is required parameter and it must be an integer. There are predefined error levels.

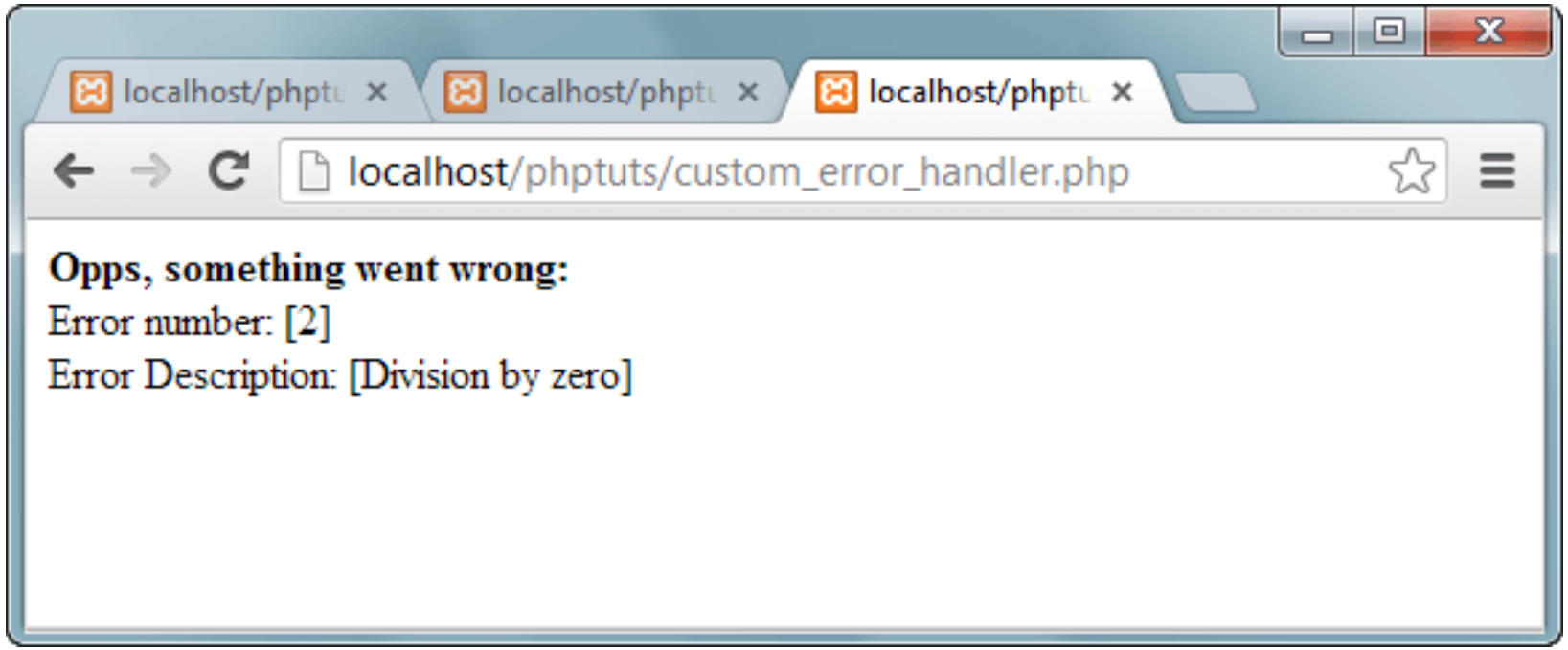
Error Handling

- $error_message: It is required parameter and it is the message which user want to print.
- $error_file: It is optional parameter and used to specify the file in which error has been occurred.
- $error_line: It is optional parameter and used to specify the line number in which error has been occurred.
- $error_context: It is optional parameter and used to specify an array containing every variable and their value when error has been occurred.

Error Handling

- The code below illustrates the implementation of the above example
- <?php
- function my_error_handler($error_no, $error_msg) { echo "Opps, something went wrong:";
- echo "Error number: [$error_no]";
- echo "Error Description: [$error_msg]"; } set_error_handler("my_error_handler");
- echo (5 / 0);
- ?>

Error Handling



Error Handling

- Error reporting function :We will be using the PHP built in function error_reporting function. It has the following basic syntax
- <?php error_reporting($reporting_level); ?>HERE,
- “error_reporting” is the PHP error reporting function
- “$reporting_level" is optional, can be used to set the reporting level. If no reporting level has been specified, PHP will use the default error reporting level as specified in the php.ini file.
- Example : error_reporting(E_WARNING);

Exception Handling

- An error is an unexpected program result that cannot be handled by the program itself.
- Errors are resolved by fixing the program. An example of an error would be an infinite loop that never stops executing.
- An exception is an unwanted or unexpected event, which occurs during the execution of a program.
- i.e at run time, that disrupts the normal flow of the program's instructions.
- An exception is unexpected program result that can be handled by the program itself.
- Examples of exception include trying to open a file that does not exist,Division by zero.

Exception Handling

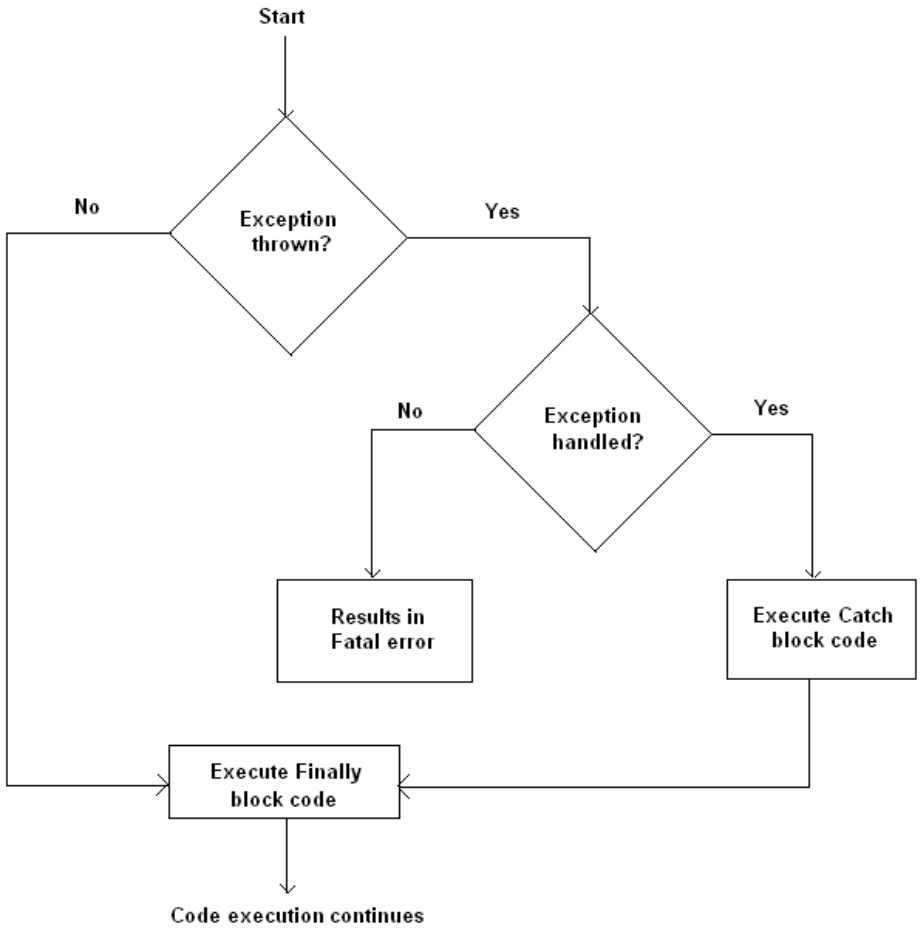
- Exception handling is a powerful mechanism of PHP, which is used to handle runtime errors (runtime errors are called exceptions). So that the normal flow of the application can be maintained.
- The main purpose of using exception handling is to maintain the normal execution of the application.
- PHP offers the following keywords for this purpose:

1. try: It represent block of code in which exception can arise.

Exception Handling

2. catch: It represent block of code that will be executed when a particular exception has been thrown.
3. throw: It is used to throw an exception. It is also used to list the exceptions that a function throws, but doesn’t handle itself.
4. finally: It is used in place of catch block or after catch block basically it is put for cleanup activity in PHP code.

Exception Handling



Exception Handling

```
<?php
   try {
      $error = 'Always throw this error';
      throw new Exception($error);
      // Code following an exception is not executed.
      echo 'Never executed';
   }catch (Exception $e) {
      echo 'Caught exception: ',  $e->getMessage(), "\n";
   }  // Continue execution
   echo 'Hello World';
```

?> OutPut : Caught exception : ‘Always throw this error’

Object Oriented Concepts

- In the above example $e->getMessage function is used to get error message. There are following functions which can be used from Exception class.
- getMessage() − message of exception
- getCode() − code of exception
- getFile() − source filename
- getLine() − source line
- getTrace() − n array of the backtrace()

Object Oriented Concepts

- Class − This is a programmer-defined data type, which includes local functions as well as local data.
- Class is a template for making many instances of the same kind (or class) of object.
- Object − An individual instance of the data structure defined by a class.
- You define a class once and then make many objects that belong to it.
- Objects are also known as instance of a class.

Object Oriented Concepts

- Member Variable − These are the variables defined inside a class.
- This data will be invisible to the outside of the class and can be accessed via member functions.
- These variables are called attribute of the object once an object is created.
- Member function − These are the function defined inside a class and are used to access object data.

Object Oriented Concepts

- Inheritance − When a class is defined by inheriting existing function of a parent class then it is called inheritance.
- Parent class − A class that is inherited from by another class. This is also called a base class or super class.
- Child Class − A class that inherits from another class. This is also called a subclass or derived class.
- purpose of inheritance is;Re-usability– a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database.

Object Oriented Concepts

- Polymorphism − This is an object oriented concept where same function can be used for different purposes.
- For example function name will remain same but it take different number of arguments and can do different task.

Object Oriented Concepts

- Encapsulation – this is concerned with hiding the implementation details and only exposing the methods. The main purpose of encapsulation is to;

1. Reduce software development complexity – by hiding the implementation details and only exposing the operations, using a class becomes easy.
2. Protect the internal state of an object – access to the class variables is via methods such as get and set, this makes the class flexible and easy to maintain.
3. The internal implementation of the class can be changed without worrying about breaking the code that uses the class.

Creating classes and Instantiation

- The class definition starts with the keyword class followed by a class name, then followed by a set of curly braces ({}) which enclose constants, variables (called "properties"), and functions (called "methods") belonging to the class.
- A valid class name (excluding the reserved words) starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- Class names usually begin with an uppercase letter to distinguish them from other identifiers.

Creating classes and Instantiation

- An instance is an object that has been created from an existing class.
- Creating an object from an existing class is called instantiating the object.
- To create an object out of a class, the new keyword must be used.
- Classes should be defined prior to instantiation.
- <?php
- class Myclass {
- // Add property statements here
- // Add the methods here }
- ?>

Creating classes and Instantiation

- Object:
- A class defines an individual instance of the data structure. We define a class once and then make many objects that belong to it. Objects are also known as an instance.

```
<?php
class demo
{
    $a= "hello BCA";
    public function display()
    {
    echo $this->a;
    }
}
$obj = new demo();
  $obj->display();
?>
```