Q1.What is PHP?

Ans: PHP is a server side scripting language. that is used to develop Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

PHP scripts can only be interpreted on a server that has PHP installed.

What is a Scripting Language?

A script is a set of programming instructions that is interpreted at runtime.

A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments.

The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.

Server side scripts are interpreted on the server while client side scripts are interpreted by the client application.

PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages.

Programming Language Vs Scripting Language

PROGRAMMING LANGUAGE	SCRIPTING LANGUAGE
Has all the features needed to develop complete applications.	Mostly used for routine tasks
The code has to be compiled before it can be executed	The code is usually executed without compiling
Does not need to be embedded into other languages	Is usually embedded into other languages.

Q2. PHP Advantages and Disadvantages?

Ans:

Advantages of PHP:

- **It's easy to learn and use:** One of the main reasons PHP became so commonplace is that it is relatively simple to get started with. Even without extensive knowledge or experience in web development, most people could create a web page with a single PHP file in a relatively short period of time. The syntax is simple and command functions are easy to learn, meaning the barriers to entry with PHP are lower than with many other languages.

- **It's open source (and therefore free!):** This also helps developers get started with PHP - it can be installed quickly and at zero cost. There is also open access to a wide range of PHP frameworks, such as Laravel and Symfony. This feature is also appealing to companies as it helps control the costs of web development.

- **It's versatile:** One of the major benefits of PHP is that it is platform independent, meaning it can be used on Mac OS, Windows, Linux and supports most web browsers. It also supports all the major web servers, making it easy to deploy on different systems and platforms at minimal additional cost.

- **It enjoys strong community support:** As a veteran scripting language that is widely used, PHP now has a large and loyal community base to support it. There are tons of tutorials, FAQs, and tips to help new PHP developers and to continue pushing the boundaries of what the language can achieve through regular updates.

- **It’s fast and secure:** Two things that every organization wants their website or application to be are fast and secure. PHP uses its own memory and competes well on speed, especially when using the newer versions. There have been questions in the past about PHP security, though it is important to note that it is not inherently more or less secure than other programming languages. One important benefit is that because of its widespread use and community support there are now many tools, frameworks and best practices to help fix vulnerabilities and protect against cyberattacks.

- **It is well connected with databases:** PHP makes it easy to connect securely with almost any kind of database. This gives developers more freedom when choosing which database is best suited for the application being built.

Disadvantages of PHP?

- **Declining Popularity:** While PHP is a powerful tool supported by a large community and plentiful reference documentation, there are easier programming languages for web apps. For this reason, novice developers prefer learning Python as their first language and rarely consider adding PHP to their skillset. Currently, PHP dominates the web development segment, but this most likely will change in the future. The number of specialists will decline eventually, and there will be a lack of novice developers that offer basic skills at a low price, so the costs of products built on PHP will probably rise.

- **Lack of dedicated libraries for modern needs:** For example, machine learning is a hot trend nowadays, and it is definitely going to keep its popularity in the near future. While PHP has its set of libraries, it cannot compete with Python in developing web applications empowered by machine learning. Currently, PHP cannot offer equally fast and effective alternatives to Python's TensorFlow, Scikit-learn, Theano, and Keras. This way, if your app requires ML functionality or may require it in the future when your business scales up, PHP is not the best choice.

- **Security is not the best:** Since PHP is open source code, it also means that it has an ASCII text file, which is easily accessible. This basically means that whatever code you write, the general public can easily view it, along with all the bugs your code has. This also means that they can use those bugs against you. Ergo, security is not the best.

- **Poor Error Handling:** Error handling is the process of identifying errors in a program and optimizing it. However, It is a popular belief among developers that PHP has a poor quality of errors handling. It lacks good debugging tools that are essential to identify errors and debug, results in poor application maintenance.

Q3. Use of PEAR in PHP?

Ans:

- PEAR stands for **P**HP **E**xtension and **A**pplication **R**epository, a library of reusable PHP components, built under the same standard, that can be used in any project.
- PEAR was first released in 1999, and it has been for a long time the de facto official dependency manager for PHP. But, currently, is considered deprecated, and the ruling dependency manager is now Composer. So, is important to understand that PEAR should not be used for new projects.
- To understand what PEAR is and why it was created, we first have to review briefly the history of PHP.
- In the past, many PHP projects used to rely in monolithic frameworks. These frameworks were composed by components designed to perform specific operations. But each framework had their components programmed *it's way*, that is, with their own coding style, interface declaration, etc. Every framework was reinventing the wheel for problems that were solved, but in other frameworks. And, **if developers wanted to use a specific component, they were forced to use some framework**, which is an overkill for most of the cases. **And these components were not interoperable with other frameworks**.
- To solve this problem, PEAR (**P**HP **E**xtension and **A**pplication **R**epository) was developed, a PHP component development and distribution environment. It was developed with three main objectives:
- Promote a well structured code library.
- Maintain a distribution and maintenance system of code packages.
- Promote a standard coding style.

- With PEAR, the developers had a repository of components that would provide solutions for many areas: encryption, databases, mathematic operations, web services, and many more. And being every component usable individually, without the need of using a huge framework for very particular solutions.

Q4. What is SESSION and use of SESSION in PHP?

Ans:

- PHP session is used to store and pass information from one page to another temporarily (until user close the website).
- PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

How to store and access Session Data?

To store and get session data we use $_SESSION[] superglobal.

Example:

$_SESSION["user"] = “the-digital-oceans";

To Get Session data :

Example:

echo $_SESSION["user"];

Q5. What is an Array? Types of Arrays in PHP

Ans:

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Types of Arrays

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

Indexed Arrays: The array which is accessed by indexed number.

Example:

```
$cars = array("Volvo", "BMW", "Toyota");

$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Associative Arrays: The array which is accessed by key.

Example:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Multidimensional Array

- A multidimensional array is an array containing one or more arrays.
- PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

Example:

```
$cars = array (
      array("Volvo",22,18),
      array("BMW",15,13),
      array("Saab",5,2),
      array("Land Rover",17,15)
     );


<?php
    echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
    echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
    echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
    echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
    ?>
```

Q6. Commonly used Array functions in PHP?

Ans:

1. **count()** : Returns the number of elements in an array
2. **sort():** Sorts an indexed array in ascending order
3. **rsort():** Sorts an indexed array in descending order
4. **ksort():** Sorts an associative array in ascending order, according to the key
5. **krsort():** Sorts an associative array in descending order, according to the key
6. **in_array():** Checks if a specified value exists in an array
7. **asort():** Sorts an associative array in ascending order, according to the value
8. **arsort():** Sorts an associative array in descending order, according to the value
9. **array_key_exists():** Checks if the specified key exists in the array
10. **array_keys():** Returns all the keys of an array
11. **array_reverse():** Returns an array in the reverse order
12. **array_search():** Searches an array for a given value and returns the key
13. **array_unique():** Removes duplicate values from an array
14. **array_merge():** Merges one or more arrays into one array

Q7. Commonly used String functions in PHP?

Ans:

1. **strlen() :** to find the length of strings.
2. **str_replace():** is used to replace one string with another string.
3. **substr():** is used to extract the string.
4. **strtolower():** is used to convert string into lower case.
5. **strtoupper():** is used to convert string into uppercase.
6. **str_word_count() :** is used to count words from the string.
7. **strrev():** is used to reverse the string.
8. **strpos():** to locate a string's initial position within another string.
9. **ucwords():** is used to convert the first letter of the word in upper case.
10. **ucfirst():** to convert the first letter in upper case of string or sentence.
11. **lcfirst():** to convert the first letter in lower case of string or sentence.
12. **wordwrap():** to wrap a given string to a certain number of characters.
13. **trim():** is used for trimming the white space from the start and end of the strings.
14. **strip_tags():** to clear/remove the HTML tags from the string.

Q8. Include functions in PHP?

Ans: There are 4 function in PHP to include file, which are following:

- **Include():** if we use include function and we give the wrong file name or path, then it will throw warning message and rest of the code will working fine.
- **require():** As its name suggests , require means mandatory, if we give the wrong file name or path, it will throw fatal error and rest of the code will not work.
- **Include_once():** this function is an improved version of include() function, if we include one file multiple time in a same page by mistake then it will only include your file one time not multiple time.
- **require_once():** this function is an improved version of require() function, it includes the file only one time.

Q9. Form Validation in PHP?

Ans: Form validation is crucial part of our web development , there are two types of form validation:

- **Client side validation:** In this we use JavaScript or jQuery to validate our form.
- **Server side validation:** In this we use PHP to validate our form.

```
<?php   $nameErr = $emailErr = $mobileErr =""; $name = $email = $mobile = "";
if (isset($_POST["submit"])) {
if (empty($_POST["name"])) {
$nameErr = "Name is required";
}
else {
$name = $_POST["name"];     // check if name only contains letters and whitespace
if (!preg_match("/^[a-zA-Z-' ]*$/",$name))
{
$nameErr = "Only letters and white space allowed";
}
}

 if (empty($_POST["email"])) {
$emailErr = "Email is required";   }
else {     $email = $_POST["email"];
 if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {       $emailErr = "Invalid email format";     }   }
```

```
if (empty($_POST["mobile"])) {
 $mobileErr = "Mobile is required";
}
else {     $mobile = $_POST["mobile"];
if (!preg_match('/^[0-9]{10}+$/', $mobile)) {
 $mobileErr = "Only letters and white space allowed";     }   }

 }   ?>

<!DOCTYPE html>
<html lang="en">
<head>
<title>Form Validation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
<script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>
   <style>
.error              {                          color:#f00;              }
</style>
 </head>
```

```
<div class="container">
<h2 class="text-center mt-5 text-primary">CONTACT US</h2>
<form method="post" class="needs-validation" enctype="multipart/form-data">
<div class="form-group">
<label for="uname">NAME:</label>
<input type="text" class="form-control"  placeholder="Enter username" name="name" value="<?php
      if(isset($_POST['name'])){echo $_POST['name'];} ?>" >
<span class="error">* <?php echo $nameErr;?></span>
</div>

<div class="form-group">
<label for="uname">EMAIL:</label>
 <input type="email" class="form-control"  placeholder="Enter Email" name="email" value="<?php
      if(isset($_POST['email'])){echo $_POST['email'];} ?>">
<span class="error">* <?php echo $emailErr;?></span>          </div>

<div class="form-group">
<label for="uname">MOBILE NUMBER:</label>
<input type="text" class="form-control"  placeholder="Enter Mobile no" name="mobile" value="<?php
      if(isset($_POST['mobile'])){echo $_POST['mobile'];} ?>" >
<span class="error">* <?php echo $mobileErr;?></span>          </div>

<button type="submit" name="submit" class="btn btn-primary p-3 btn-block my-5">Submit</button>
</form>

</div>      </body>   </html>
```

Q10. File Upload in PHP?

Ans:

```
if(isset($_POST[‘submit’]))
 {
$filename = $_FILES[‘doc’][‘name’];
$tempfile = $_FILES[‘doc’][‘tmp_name’];
move_uploaded_file($tempfile,"images/".$filename);
 }

<form method="post" enctype="multipart/form-data">
<div class="form-group">
<label for="email">Document:</label>
<input type="file" class="form-control" name="doc">
</div>
<button type="submit" name="submit" class="btn btn-primary">Submit</button>
</form>
```

Q11. How to Upload only pdf file with 2MB size?

Ans:

```
if(isset($_FILES['image']))    {
$upload="";
$file_name = $_FILES['image']['name'];
$file_size =$_FILES['image']['size'];
$file_tmp=$_FILES['image']['tmp_name'];
$file_type=$_FILES['image']['type'];
$file_ext=end(explode('.',$_FILES['image']['name']));

if($file_ext!="pdf" && $file_size < 2097152)
 {
move_uploaded_file($file_tmp,"uploads/".$file_name);
$upload='<div class="alert alert-success">
<strong>Success!</strong> File Upload Successfully.   </div>';
}
else
{
$upload=' <div class="alert alert-danger">     <strong>Error:</strong> File Extension should be .pdf or File Size
      less than 2MB.   </div>';
}
}
```

Q12. Multiple File Upload in PHP?

Ans:

```
if(isset($_POST['submit']))
      {

      $length= count($_FILES['doc']['name']); for($i=0;$i<$length;$i++)
      {
      $file_name = $_FILES['doc']['name'][$i];
      $file_tmp = $_FILES['doc']['tmp_name'][$i];
      move_uploaded_file($file_tmp,"uploads/".$file_name);
      }

      }


  <form method="post" enctype="multipart/form-data">
      <input type="file" name="doc[]" multiple> <input type="submit" name="submit">
      </form>
```

Q13. PHPMailer in PHP?

Ans: PHPMailer is a library which is used to send mail in PHP, It has some extra features as compared to mail() function of PHP.

By Using PHPMailer we can:

- Send mail by using local server
- Send mail by using SMTP like Gsuit, Microsoft office 365 mail.

Q14. How to send html email in PHP?

Ans: By using PHP mail() function we can easily send HTML email with using headers.

Example:

```
$to = “clientmailid";
$subject = "HTML email";

        $message = "
        <html>
        <head>
        <title>HTML email</title>
        </head>
        <body>
        <p>This email contains HTML Tags!</p>
        <table>
        <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        </tr>
        <tr>
        <td>John</td>
        <td>Doe</td>
        </tr>
        </table>
        </body>
        </html>
        ";

        // Always set content-type when sending HTML email
        $headers = "MIME-Version: 1.0" . "\r\n";
        $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

        // More headers
        $headers .= 'From: <webmaster@example.com>' . "\r\n";

        mail($to,$subject,$message,$headers);
```

Q15. Implode() and explode() in PHP?

Ans: Implode and Explode function of PHP is very useful to convert the array into string and string into an array for example when we collect data by using checkbox input then we receive data in an array format then we have two choices to get checkbox value either by using a loop or we can convert the array into a string by using implode function.

Implode() function

Implode() function is used to convert the array into a string, it uses any symbol to join an array. It is commonly used to convert checkbox values into strings to store easily in a database.

Example:

```
if(isset($_POST['submit']))
{
$price=implode("#",$_POST['price']);
echo $price;
}
<input type="checkbox" class="form-control" name="price[]">1000
<input type="checkbox" class="form-control" name="price[]">2000
<input type="checkbox" class="form-control" name="price[]">3000
```

Explode() function

Explode() function is used to convert a string into an array or we can say that show implode() function value individually then we use explode.

Example:

```
<?php $number = '1,2,3,4,5';
 $arr = explode(',',$number);
foreach($arr as $i) echo($i.'<br>');
?>
```

Q16. Difference between unset session and session destroy?

Ans: Sometimes web developers want to destroy a particular session of the website and sometimes they want to destroy all sessions of the website but they got confused between the **session_destory()** and **session_unset()** method. So this blog has a solution for you:

session_destroy()

session_destroy() function is used to destroy all the sessions of the website.

Example:

```
<?php session_destroy(); ?>
```

session_unset()

session_unset() function is used to destroy specific sessions of the website, it does not destroy all the sessions like **session_destory().**

Example:

```
<?php session_unset($_SESSION['username']); ?>
```

Q17. Difference between $ and $$?

Ans:

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.

The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

Example:

```
$var = "Hello";
$Hello = “The Digital Oceans";

echo $var . "\n";
echo $$var;
```

Output:

```
Hello
The Digital Oceans
```

Q18. Cookie in PHP?

Ans: PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.

setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Example:

```
setcookie("CookieName", "CookieValue");/* defining name and value only*/
setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)
```

$_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

Example:

```
$_COOKIE["CookieName"];
```

Q19. TYPES OF COOKIE IN PHP?

Ans: There are two types of cookies, they are:

- **Session Cookie**: This type of cookies are temporary and are expire as soon as the session ends or the browser is closed.

Example:

```
setcookie("CookieName", "CookieValue");
```

- **Persistent Cookie**: To make a cookie persistent we must provide it with an expiration time. Then the cookie will only expire after the given expiration time, until then it will be a valid cookie.

Example:

```
setcookie("CookieName", "CookieValue", time()+1*60*60);
```

Q20.Different Types of errors in PHP?

Ans: Basically, an error is a mistake in a program that may be caused by writing incorrect syntax or incorrect code. An error message is displayed on your browser containing the filename along with location, a message describing the error, and the line number in which error has occurred.

There are usually different types of error. In PHP, mainly four types of errors are considered:

- Syntax Error or Parse Error
- Fatal Error
- Warning Error
- Notice Error

Syntax Error or Parse 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.

Example:

```
echo "Alex: Hie! I'm Alex. </br>";
echo "Bob: I'm Bob. How are you?"
```

In this it will throw syntax error or parse error because in second statement we have not used semicolon at the end of the statement.

Fatal Error

A fatal error is another type of error, which is occurred due to the use of undefined function. The PHP compiler understands the PHP code but also recognizes the undefined function. This means that when a function is called without providing its definition, the PHP compiler generates a fatal error.

Example:

```
function addition($f1, $f2) {
    $sum = $f1 + $f2;
    echo "Addition:" . $sum;
}

$f1 = 23;
$f2 = 56;

//call the function that is not defined
//Generate fatal error
subtraction();
//echo "Fatal Error";
```

In the above example, we have declared function add but we are trying to call subtraction(), so it will throw fatal error , because this function does not exist.

Warning Error

A warning is generated when the programmer tries to include a missing file. The PHP function calls that missing file which does not exist. The warning error does not stop/prevent the execution of the program.

The main reason behind generating a warning error is to pass an incorrect number of parameters to a function or to include a missing file.

Example: include missing file

```
<?php
   /*-------------------warning error------------------*/
   $cmpny = ‘the digital oceans';
   echo "Warning Error: ";

   //include a file in the code
   include (‘header.php');
?>
```

In the above example, we are trying to include a file which does not exist, so it will throw warning message failed to open a stream.

Notice Error

Notice error is same as warning error. When program contains something wrong, the notice error occurs. But it allows/continue the execution of the program with a notice error. Notice error does not prevent the execution of the code. **For example** - access to undefined variable.

Generally, notice error occurs when we try to use or access a variable which is undefined.

Example:

```
<?php
   /*------------------notice error-------------------*/
   $data = “hello";
   echo $data;
   echo $data1;
?>
```

In the above example, we are trying to print a variable $data1, which does not exist. So it will throw notice error.

Q21. Superglobals in PHP?

Ans: Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- **$GLOBALS:** It is used to access global variable.
- **$_SERVER:** variable which holds information about headers, paths, and script locations
- **$_REQUEST:** It works same as $_POST and $_GET
- **$_POST:** It is used to access form data.
- **$_GET:** It is used to get data from url. For example: to edit data we get id from url.
- **$_FILES:** It is used to get data related to file, like filename, file size etc.
- **$_ENV:** It is used to set environment data like base url, database related info, mail related info like .env file in laravel.
- **$_COOKIE:** It is used to get cookie value.
- **$_SESSION:** It is used to get session value.

Q22. Difference between echo and print?

Ans:

Echo()

- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

Print()

- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

Q23. What is php.ini file?

Ans: At the time of PHP installation, **php.ini** was a special file provided as a default configuration file.

- It's a very essential configuration file that controls what a user can or cannot do with the website.
- Each time PHP is initialized, the php.ini file is read by the system.
- Sometimes you need to change the behaviour of PHP at runtime, then this configuration file is to use.
- All the settings related to registering global variables, **uploading maximum size, display log errors, resource limits, the maximum time to execute** a PHP script and others are written in a file as a set of directives that helps in declaring changes.
- The php.ini file is the default configuration file for running applications that require PHP. It is used to control variables such as **upload sizes, file timeouts, and resource limits.**
- php.ini file is the configuration file. It is always checked when the server gets started or HTTP is restarted in the module and it configures the website to know what a user can do or can't do with a website.
- It also helps with the administration of the web server easily.

Q24. What is the most used method for password hashing?

Ans: Back in the day, passwords were stored using an MD5 or SHA1 hash.
For two reasons:

- MD5 and SHA algorithms are too weak for today's computational power.
- Simple, *not-salted* hashes are vulnerable to "rainbow tables" and dictionary attacks.

If an attacker steals an MD5 or SHA hash, he or she can easily find out the original password too.
In other words, these hashes are almost as insecure as plain text passwords.
The solution is to use a **secure hashing function:** *password_hash()*.

Example:

```
$password = "thedigitaloceans@@123";
password_hash($password, PASSWORD_BCRYPT);
```

To verify the hashed password: PHP provides an inbuilt function called **password_verify()** to match the hashed password to the original passwords.

Example:

```
$password = " thedigitaloceans@@123";

$hashed_password =
'$2y$10$MQU3vDgoN10.JxyJ1m9UQOEqFy.Jg3D8tmHdZUAAkcpGFRwkbbLfi';

if (password_verify($password, $hashed_password)) {
   echo 'Password is valid!';
} else {
   echo 'Invalid password.';
}
```

Q25. Little Bit Knowledge of OOP Concepts & PDO in PHP?