PHP NOTES

21/09/21 – Pracs 1

Extended regular expressions—Support all common regular expression operations. Regular expressions are mainly used for pattern matching, pattern substitutions, and general string manipulation. - eg trusha@gmail.com needs pattern matching and @ and . are compulsory

(a+b)^+@(a+b)^+.(a+b)^+ ----- regular expression for email id

Running a php script:

step 1: The client requests for the .php script (when localhost is typed)

step 2: the web server receives the request and recognises that the file is a php script using .php extension

step 3: the server then hands it over ot the php parser and interpreter for further processing

step 4: the php interpreter reads the instructions between the php tags

<?php

?>

executes them and passes the results back to the server which in turn sends it to the browser

step 5: browser displays output on screen

- the echo stmt is used to display outputs on the screen
- all the php code must be enclosed within the php tags
- every php stmt must end with a ';'
- single line comments are written using // and multiple lines comments are written using /* */

HANDLING ERRORS IN A PHP SCRIPT

• the execution of the script stops in between due to errors

• either a warning msg is displayed on the screen or the execution of the script stops at the point of error with a notification of error msg.

ESCAPE SEQUENCE CHARACTERS

• used to display certain characters on the screen

• to display ", we use \" in the code

• to display ', we use \' in the code

• newline character \n --- will not work for html, we use

```
<html>
<body>
<?php
echo "introduction to php <br>" ;
echo "If you view the page source <hr> you will find a newline in this string.";
?>
</body>
</html>
```

28/09/21 – Pracs 2

WORKING WITH VARIABLES AND CONSTANTS:

- Variable and constant is same as that in C programming
- A variable is a symbolic name associated with a value which can be changed during the execution of a program.

- Variables can take numeric or alphanumeric values.
- Constants store values that cannot be changed during the execution of the program

NAMING RULES AND CONVENTIONS FOR VARIABLES

- Every variable name must start with a $ sign. Eg: $var
- A variable name must start with a letter of underscore. Eg. $_var(valid), $1var(invalid)
- Variables can only contain alphanumeric characters and underscores. E.g.: $my_var (valid), $23#(invalid)
- Variables should not contain spaces. Can separate using underscore
- There is no limit on the length of a variable name.

- The print statement can also be used in place of echo but they are not the same.
- Echo can take multiple arguments separated by commas, but print cannot take multiple arguments. E.g.: echo $x, $y; → Valid
  print $x; → Valid
  print $x, $y; → Invalid
- You can also assign values to variables by using the assign-by reference method. In this method the new variable points to another variable and the changes done in any of the variables affect both the variables. The variable name must precede by the & symbol.
- Any change made to other variables affects the variable passed by reference.
- PHP allows you to set the name of a variable dynamically.
- A value assigned to a variable itself becomes a variable.

DESTROYING VARIABLES

- PHP allows you to destroy variables or an element from an array when they are no longer required using the unset function

USING CONSTANTS

- Constants are identifiers that store values which cannot be changed during the execution of the script.
- Constants can be used to store data like configuration settings whose values don't change during the execution of the script. Constants can also be used to store error codes and flags.
- The constants are case sensitive and their names do not start with a $ sign. By convention, all constant names should be in capital letters.
- Syntax: define ("CONSTANT_NAME", constant_value);
- If a constant is redefined in the code it may show an error msg or it may print the value that was assigned for the first time.
- Constants can only hold numbers, strings and Boolean values. It cannot hold arrays or objects.
- E.g. of built-in constants: M_PI (3.14) and PHP_VERSION (prints current version number of php)

12/10/21 – Pracs 4

DATA TYPES IN PHP

- A data type refers to the type of data that the variable can hold
- There are 8 data types
  - integers
  - Floating point numbers
  - Strings
  - Boolean
  - Arrays
  - Objects
  - Resource
  - Null
- It is not necessary to specify the data type of a variable in php.
- PHP automatically decides the data type according to the value assigned to the variables
- There are two categories of data types
  1. Scalar/Primitive
  2. Compound

- Scalar data types can only hold a single value. Integers, floating point number, string and Boolean are scalar data types
- Compound data types can store a collection of values – Arrays, Objects

INTEGERS

- Represent a whole number without any fractional component.
- The permissible range of integers is decided by the operating system (-2,147, 483, 648 to +2,147, 483, 647)
- Integers can be written in decimal, octal or hexadecimal format.
- Hexadecimal numbers are preceded by 0 and x. E.g. 0xFEFEFE

FLOATING POINT NUMBERS

- Real numbers with decimal places
- PHP includes two types of FPN
  1. Simple numeric with a decimal point
  2. FPN written in scientific notation E.g.: 3.14E-3

STRINGS

- Are text of any length which can be enclosed in single quotes or double quotes
- Strings inside double quotes are parsed while strings inside single quotes are not parsed.

```
$name = “India”

Echo “We are $name”

Echo ‘We are $name’
```

Output: We are India
We are $name

BOOLEAN

- Represents a true or false value
- All conditions return a true or false Boolean value based on the condition being tested.
- But some statements always return a false value
- E.g.: 1. the keyword “false” can never become true.
  2. the number 0 and 0.0 can never return true,

3. “” or “0” will always be false
4. NULL value will never be true.
5. Object with no values or functions

ARRAYS

- Represent a variable that stores a collection of related data elements
- Each element can be accessed using its index position
- The positions are specified numerically or alphabetically.

OBJECT

- Objects allow you to store data as well as information to process that data
- The data elements stored within an object are called its properties or attributes of the object
- To declare an object, first u must declare a class of the object. Then you need to instantiate the object.
- Objects also allow you to create your own data types.

RESOURCE

- A resource represents a special data type which stores references to functions and resources external to php.
- E.g.: Database Call

NULL VARIABLE

- Can only store null value which is also a keyword.
- A variable of the null data type is a variable that has no value assigned to it.
- When no value is assigned to a variable it is automatically assigned to a NULL value.

DETERMINING VARIABLE DATA TYPE

- PHP automatically determines the data type of a variable from the value that the variable holds.

- If the value of the variable changes in the program then php will automatically set the appropriate new data type

PRACS CLASS _ 2/11/21

TYPE CASTING:

- You can convert the data type of a variable to another data type using type casting.
- Converting integer to string data type.
- An example of automatic type casting is the addition operation.
- If either of the operands is a float, then both the operands are evaluated as floats and the result will also be a float
- E.g. $x = “3”; //string
  $x+=3; //x becomes an integer
  $x = $x + 1.3; //x becomes float, value = 7.3
- You can explicitly type cast a variable by enclosing the name of the desired type in brackets before the variable which is to be type casted.
- E.g. $x = (Boolean) $x;
  $x = (float) $x;
  settype($x, “bool”);

OPERATORS IN PHP

Type of operators are assignment operators, arithmetic operators, string operators, comparison, logical, increment, decrement, arithmetic assignment.
Learn short notes under each of these 7 types

E.g. $x = 9;
$x = $y;
$x + $y;
-$x (negation)

String concatenation operator is used to combine values to create a string.

E.g.

```
<?php
    echo “We are”.”India”;
$color = “black”;
echo “My favourite color”.$color;
```

Comparison operators return either true or false.
!=, <>, !== -> The first two are “not equal to” third one is “not identical to” <, > , ==, <=, =>
!== returns true if the first operand is not equal to the second operand in both value and type.
E.g. (“3” !== 3) – returns true

Logical operators are also known as Boolean operators because they evaluate parts of an expression and return either true or false.
E.g AND, OR, NOT

Increment, decrement - $x++ is post increment, returns value of x first and then increments it by 1.++$x – increments the value of x by 1 and then returns the value.
$x—and --$x;

```
$x = 19;
$y = $x++;
echo $y;       //19
$y = ++$x;
echo $y; //21
$y = $x--;
echo $y;       //21
$y = --$x;
echo $y; //19
```

Arithmetic assignment operators:

```
$x = 9;
$x += 2; echo $x;    //11
$x -= 3; echo $x;    //8
$x *= 2; echo $x;    //16
$x /= 2; echo $x;    //8
$x %= 2; echo $x;   //0
$x = “We are”;
$x .= “India”
echo $x; //We are India
```

Operator precedence

Echo 3*3/2; //4.5
echo 3/3*2 //2

CONTROLLING THE PROGRAM FLOW

Conditional statements: if, else, if-else ladder, switch
Switch statement – allows u to evaluate a condition and match the condition's value to the statements given in the case labels. If a match is found, the program executes the associated statement. If no matching label is found, it executes the optional default statement. You can also test for other conditions like greater than and less than relationships.
If the expression matches with the values specified in more than one case statement then only the first one will get executed.
E.g.

```
$x = -9;
switch($x):
{
        case $x<0:
                echo  "negative number";
                break;
        case $x>0:
                echo  "positive number";
                break;
```

Looping statements: for, while, do-while, foreach
While – the value of the condition is checked each time at the beginning of the loop. So even if this value changes during execution it does not stop until the end of the process.
Do-while – always executes the block of statements at least once and then checks the condition.
For – it is used when you know exactly how many times u want to execute the block of statements.
for(initialization; test-condition; incr/decr)
Foreach loop – it is a variation of the for loop and allows you to iterate over elements in an array.

There are two variations of foreach stmt:

1. Foreach (array as value)
```
{
}
```
2. foreach (array as key => value)
```
{
}
```
E.g. $email = array('gupta', 'dixit');
```
foreach ($email as $value)
{
    echo "Hello".$value."<bool>";
}
```
$person.array('Name'=> 'Gupta', 'Age' => 23, 'Address' => 'Goa')
```
foreach($person as $key => $value)
{
    echo $key."is".$value."</br>";
}
```
Break, continue and exit statements – the break stmt in a loop skips the remaining stmts in the loop body and breaks it.

Continue – skips all remaining stmts in the loop for the current iteration but returns to the top of the loop and allows it to continue running

Exit statement – used to stop a program from running;

E.g.
```
for($i=0; $i<5; $i++)
{
    if($i == 2)
        exit;
    echo "The number is".$i;
}
```

WORKING WITH FUNCTIONS, ARRAYS AND FILES

1. USER DEFINED FUNCTIONS IN PHP

- In php there are more than 700 built in functions and you can also create your own user defined functions.
- function <function_name> (arg-list)

```
{
    //CODE
    Return;
```

}

Naming conventions for function name

-Function names are not case sensitive in php. Therefore, fname(), Fname() and FNAME() refer to the same function.

-Function names can contain only letters from the ascii character set, digits and underscore.

-Function name cannot begin with a digit

-Two functions cannot have the same name in php because php doesn't support function overloading.

-Reserved keywords cannot be used as function names

E.g

```
Function Book_name($name)
{
        echo "Book name".$name
}
book_name("PHP")
```

```
<?php
        function get_year()
        {
                $year = date("y");
                return $year;
        }
        echo "year is: ".get_year()
?>
```

Variable scope in functions:

```
<?php
function change_score()
{
        $score = 20;
}
$score = 40;
echo "Score is".$score;
change_score();
echo "Score is".$score;
?>
```

Output = 40, 40

A local variable is restricted to the function space alone and cannot be manipulated outside the function in which it is declared.

Global variable scope:

```
<?php
function change_score()
{
        global $score = 20;
}
$score = 40;
echo “Score is”.$score;
change_score();
echo “Score is”.$score;
?>
```

Output = 40, 20

```
<?php
function change_score(&$score)
{
        $score = $score + 20;
}
$score = 40;
echo “Score is”.$score;    //40
change_score($score);
echo “Score is”.$score;    //60
?>
```

Built-in functions in php

-String manipulation functions:

-strlen() – returns length of the string

-int strlen(string)

-The return value is the length of the string on success and 0 if the string is empty.

-The input parameter is a string whose length is to be calculated

explode()

```
<?php
$string = “PHP learning solution”;
$element = explode(“ “, $string);
echo $element[0]; //PHP
echo $element[1]; //learning
echo $element[2]; //solution
?>
```

The explode() function breaks a string into an array
The input parameters are the separator and the string
If the separator is an empty string the return value of explode() will be false, else it will return an array of characters.
If the separator is not contained in the string then an empty array is returned.

Implode()

Used to join array elements into a string

```
<?php
$string = array(‘php’, ‘learning’);
echo implode(““, $string); //phplearning  --- no space
?>
```

Input parameters are the separator which is optional and the input string or array.

Str_repeat()

```
<?php
$string = “php”;
echo str_repeat($string, 5);       //phpphpphpphpphp
?>
```

It repeats the given string. Input parameters are the string and the number of times to repeat it.

Strrev()

```
<?php
$string = “Index”;
```

```
echo strrev($string);        //xednI
?>
```

Date and time functions:

The date and time built-in functions in php provides the ability to read the system data in various formats and also to manipulate it.

date(format, timestamp); //format is mandatory, timestamp is optional

d: specifies the day of the month from 1-31

D: specifies the textual representation of the day using 3 letters

i – specifies the minutes with leading zeroes

s: specifies seconds

Y – specifies year

a: specifies am or pm

F – full representation of the month

h: specifies the 12 hour format from 1 to 12

e.g

```
<?php
        echo date (“l”);              //Thursday
        echo date(“l d \of F Y h: i: sa”);  //Thursday 25 of November 2021
12:50:13pm
?>
<?php
        var dump(checkdate(11, 31, 2022));    //boolean(false)
        var dump(checkdate(2, 28, 2022));     //boolean(true)
?>
```

Checkdate(month, day, year) – validates a date

Var dump displays structured information about expressions that includes its type and value.

Returns true or false depending on whether the specified day is valid or invalid.

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form name="form1" method="" action="" enctype="" >
```

```
        First Name:
        <input type="text" name="firstname"><br><br>
        Last Name:
        <input type="text" name="lastname"><br><br><br>

        Gender
        <br>
        <input type="radio" name="gender" value="male"> Male <br>
        <input type="radio" name="gender" value="female"> Female <br><br><br>

        Hobbies
        <br>
        <input type="checkbox" name="hobbies" value="reading">Reading <br>
        <input type="checkbox" name="hobbies" value="gaming">Gaming <br>
        <input type="checkbox" name="hobbies" value="watchingmovies">Watching
Movies <br><br>

        Address
        <br>
        <select name="address" id="">
            <option value="margao">Margao</option>
            <option value="panjim">Panjim</option>
            <option value="vasco">Vasco</option>
        </select>
        <br><br>
        <input type="submit" name="submit" id="">

    </form>
</body>
</html>
```

Working with the <form> Tag and Form Elements:

The <form> tag is used to create an HTML form for user Input.
The <form> tag contains four attributes:
Name —Specifies the name of the form
Action —Specifies the address where you want to send the data, such as an e-mail address, a Web saver address, or any other form
Method —Specifies the http method used to pass the form's content to the Web server
enctype —Specifies how the form content should be encrypted

```
<html>
```

```
    <body>
        <form name="" action="" method="get/post" enctype="">
            First Name:
            <input type="text" name="firstname" />
            <br/>
            Last Name:
            <input type="text" name="lastname" />
        </form>
    </body>
</html>
```

Processing a Web Form:

The information entered in a webform is sent to a web server for processing.
The processing of a web form may include the following activities:

1. Submitting the form data
2. Retrieving the form data
3. Validating the form data

Submitting the form data:

The data in a Web form can be sent to the server by using either the get() or post() method.

Using the get() Method:

After you press the Submit button on a Web form, the get() method sends the form data to the server by attaching it at the end of the URL.
The attached form data is called query_string.
The get() method allows you to send only a limited amount of information at a time.
The information sent by using the get() method is displayed in the address bar of the Web browser.

```
<html>
    <body>
        <form name="Form1" action="hyper.php" method="get">
         Enter your Name:
         <input type="text" name="name" />
         <br/>
         Enter your age:
         <input type="text" name="age" />
         <br/>
         <input type="submit" name="button" value="Submit"/>
        </form>
    </body>
```

```
</html>
```

Since the information entered is displayed in the URL, this method should not be used when sending passwords or other sensitive information.

Using the post() Method:

The post() method appends all the form data into the HTTP header message body. You can send any amount of data using the post() method. Information sent through this method is not displayed in the URL of the Web browser.

```
<html>
    <body>
        <form name="Form1" action="hyper.php" method="post">
         Enter your Name:
         <input type="text" name="name" />
         <br/>
         Enter your age:
         <input type="text" name="age" />
         <br/>
         <input type="submit" name="button" value="Submit"/>
        </form>
    </body>
</html>
```

Retrieving the Form Data

PHP provides various functions to retrieve the submitted form data. The following are some of the commonly used functions to retrieve data:

The $_GET[] function

The $_POST[] function

The $_REQUEST[] function

In addition, you can use the $_SERVER [‘REQUEST_METHOD‘] method to determine the method used to send the form data to the server.

The $_GET[] Function

The $_GET[] function is a built-in function used to retrieve values from a form sent through the get() method.

This function only retrieves data sent through the get() method.

In the succeeding example, we create a file named welcome.php and use the $_GET[] function to retrieve the data from another file, index.php The retrieved data is displayed on the welcome.php page.

Put this in index.php:

```
<html>
    <body>
        <form name="Form1" action="welcome.php" method="get">
        Enter your Name:
        <input type="text" name="name" />
        <br/>
        Enter your age:
        <input type="text" name="age" />
        <br/>
        <input type="submit" name="button" value="Submit"/>
        </form>
    </body>
</html>
```

Put this in welcome.php:

```
<html>
    <body>
        welcome <?php echo $_GET["name"]; ?>
        you are <?php echo $_GET["age"]; ?> years old
    </body>
</html>
```

The $_POST[] Function

The POST function is a built-in function used to retrieve the values from a form sent through the post() method.

```
<html>
    <body>
        <form name="Form1" action="welcome.php" method="post">
        Enter your Name:
        <input type="text" name="name" />
        <br/>
        Enter your age:
        <input type="text" name="age" />
        <br/>
        <input type="submit" name="button" value="Submit"/>
        </form>
    </body>
</html>
```

```
<html>
    <body>
        welcome <?php echo $_POST["name"]; ?>
        you are <?php echo $_POST["age"]; ?> years old
    </body>
</html>
```

The $_REQUEST[] Function

The REQUEST function can be used to collect form data sent with both get() and post() methods.

Put this in index.php Same will work with both get and post

Put this in welcome.php

```
<html>
    <body>
        welcome <?php echo $_REQUEST["name"]; ?>
        you are <?php echo $_REQUEST["age"]; ?> years old
    </body>
</html>
```

The SERVER['REQUEST_METHOD'] method:

The method that a form uses to send data to the server is specified with the method attribute of the <form> tag. You can find which method has been used to send data to the server by using the $_SERVER['REQUEST_METHOD'] method.

Put this in index.php Same will work with both get and post.

```
<html>
    <body>
        <?php
         if ($_SERVER['REQUEST_METHOD'] == "POST") {
           echo " get() function" ;
         } else {
           echo " post() function" ;
    }

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

Validating a Form Form:

validation is the process of checking that a form has been filled in correctly before it is processed.

When using PHP, the data sent in a Web form is verified at the server end.

For example, if our form has a box for the user to type their email address, we might want our form handler to check that they have filled In their address before we deal with the rest of the form.

There are two main methods for validating forms: server-side using PHP, and client-side (using JavaScript). Server-side validation is more secure. Process of validating a Web form

The data entered in a form is sent to the server. The data sent is then verified at the server end. When the server encounters an incorrect data element being sent through a Web form, an error message is displayed.
The server also verifies if the data entered in a form element is in the correct format. For example, a Date text box on a Web form must accept data only in the date format
Server side form validation is usually performed with PHP, ASP, or CGI.

Displaying an Error Message:
when a Field is Empty The most basic type of form validation is to enforce that a particular field must contain a value. In the case of a text input that is submitted with no value entered, the element in $_POST is still created, but it contains an empty value. A Web form must ensure that a user has entered relevant data in all mandatory form elements. In case a user leaves a mandatory form element blank, an error message can be displayed instructing the user to enter relevant data in the mandatory form elements.

Index.php

```
<html>
    <body>
        <form name="Form1" action="formvalidate.php" method="post">
        Name:
        <input type="text" name="name" />
        <br/>
        Password:
        <input type="text" name="age" />
        <br/>
        <input type="submit" name="button" value="Submit"/>
        </form>
    </body>
</html>
```

Formvalidate.php:

```
<?php
  $flag="OK";
  $msg=NULL;
  if(!$_POST['username']){
      $msg="Please enter username\n";
      $flag="NOT OK";
  }
  if(!$_POST['password']){
    $msg="Please enter username\n";
    $flag="NOT OK";
  }
```

```php
  if($flag=="NOT OK"){
      echo $msg;
      echo"<input type='button' value='back' onclick='history.go(-1)'>";
  }
  else{
      echo "All entities are correct";
  }
?>
```

Note that if a user submits a Web form without entering relevant data in a form element, the associated $_POST clement is still created; however, it contains an empty value.

```php
<?php
  $flag="OK";                          // This is the flag and we set it to OK
  $msg="";                            // Initializing the message to hold the error messages
  if(!$_POST['userid'])                    //this check Check for empty Name field
  {
        $msg = ("Please enter user id.")</BR>";
        $flag="NOTOK";                        //setting the flag to error flag.
  }
  if( !$_POST['password'])
  {
        $msg=" Please enter the password ";
        $flag="NOTOK";   //setting the flag to error flag.
  }
  if($flag =="NOTOK")
  {
        // if flag is set to NOTOK, then there is display the back button for the user to
        go navigate back and correct the entries echo "<h1>". $msg."</h1>"."<input
        type='button' value='back' onClick='history.go(-1)'>";
  }
  else
  {
  echo "all entries are correct and let us proceed with the database checking etc ..";
}
```

Enforcing Data Rules:

A Web form may contain specific form elements that accept data in a specific format; for example the Date field. You can verify whether or not the data entered in these specific fields is in the right format by enforcing data rules. Consider a case wherein a Web form contains a text box named Telephone. The Web form would need to enforce data rules to ensure that the data entered in the Telephone text box is in the numerical format.

We will often want to ensure not only that data is entered into required fields but also that the quality of the data is good enough before proceeding. For instance, we might want to check that an email address or a phone number has the right format or not.

Let's now validate a text box so that only characters can be entered in it. Ex: in "userid" field other than characters nothing else is allowed. For this purpose, we use "ereg()" regular expression function of PHP.

```php
<?php
  $flag="OK";
  $msg=NULL;
  if(!$_POST['userid']){
      if(!ereg('[A-Z a-z]', $_POST['userid'])){
       $msg = "Please use only lower or upper case letters";
       $flag="NOTOK";
      }
  }
  else {
    $msg = "Please enter userid";
   $flag="NOTOK";
}
  if(!$_POST['password']){
    $msg="Please enter username\n";
    $flag="NOT OK";
  }
  if($flag=="NOT OK"){
      echo $msg;
      echo"<input type='button' value='back' onclick='history.go(-1)'>";
  }
  else{
      echo "All entities are correct";
  }
?>
```

Using PHP and MYSQL

MySQL is a Relational Database Management System (RDBMS). PHP is used to create dynamic Web pages. The dynamic content does not always require a database; however in most cases there is need of a database. Web based applications, such as content management systems, blogs, and emails use databases to store information.

The most widely used database in PHP is MySql. To access the MySql database server, you need to check its configuration.

Checking Configuration

While accessing a database you can avoid problems, such as connectivity errors, by checking the configuration of the MySQL database server. You can run the following code to check the configuration of Mysql in PHP:

<?php

```
phpinfo();
?>
```
When you scroll down, the configuration of mysql appears. If you do not find the MySql section in the list of configurations, then you cannot access MySql using the current configuration.

Connecting to a Database:

In PHP, you can use database by establishing a connection to the MySql database server. PHP requires the following information to connect to a database: a hostname database username password

code to establish connection with the MySql database server in PHP:

```
<?php
mysqli_connect('localhost', '',  '') or die(mysql_error());
echo "Connection to the server was successful!";
?>
```
It displays the text, Connection to server was successfull, if the connection with MySql is established successfully, else it displays the corresponding error. The mysql_error() function is used to display the MySql errors.

Suppose, you need to provide database connectivity to 150 Web pages. You can accomplish this task by writing hardcode on each Web page; however, this is not appreciated because if you want to change any of the information, such as hostname, username, and password, then you have to change it on every Web page.

You can perform this task by creating a single file, which contains the code to connect to the database and you can call this file on whichever page the connectivity is required.

You can call this file by using the following code:

```
<?php
include ‘filename’;
?>
```
Where, filename refers to the name of the file that contains code to connect to the database.

Selecting a database:

The data is stored in a database from where it can be retrieved. A database server can have many databases, therefore while working with the database it is necessary to select the database on which you want to work.

Code to create a database in the database server:

```
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$name1->query("CREATE DATABASE teslin") or die(mysql_error());
echo "Database created";
mysqli_close($name1); ?>
```

how to select a database.

MySql database server provides a function mysql_select_db() to select a database.

```
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$dbase_name = "teslin";
mysqli_select_db($name1, $dbase_name) or die(mysql_error());
echo "Database is selected";
mysqli_close($name1);
?>
```

Adding table in a database.

example:

```
CREATE TABLE Employee_Information
(
Emp_ID integer NOT NULL,
Emp_Name varchar (40) NOT NULL,
Emp_Address varchar (80) NOT NULL,
Emp_PhoneNumber varchar (15) NOT NULL,
PRIMARY KEY (Emp_ID)
)
```

```
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$dbase_name = "teslin";
```

```php
mysqli_select_db($name1, $dbase_name) or die(mysql_error());

$query = " CREATE TABLE Employee_Information
(
  Emp_ID integer NOT NULL,
  Emp_Name varchar (40) NOT NULL,
  Emp_Address varchar (80) NOT NULL,
  Emp_PhoneNumber varchar (15) NOT NULL,
  PRIMARY KEY (Emp_ID)
  )";
$name1 -> query($query);
echo "Table successfully created";

mysqli_close($name1);
?>
```

first the connection with the database is established using the mysql_connect().
Next, the corresponding database is selected using the mysql_select_ db( ).
The query to create a table is stored in the $query variable and executed using the mysql_query() function. If the query is executed successfully, the echo statement prints a message, Table successfully created else it prints the corresponding error message.
various datatypes used in MySql:
CHAR()
VARCHAR()
INT
FLOAT
DATE
TIME

Altering a table in a database:

The structure of an existing table can be altered using SQL queries. You can drop, add, modify and change the name of a column using the alter table query.
Following syntax is used to delete a column of a table:
Dropping a Column(used to delete an entire column):
alter table Employee_Information drop column Emp_PhoneNumber
Adding a Column(used to add new column in table):
alter table Employee_Information add column Emp_PhoneNumber varchar(10)

Changing a Column (used to change the column name):
alter table Employee_Information change Emp_PhoneNumber Emp_Phone_Number varchar(10)

Modifying a column (used to modify the definition of a column):
alter table Table_Name modify Column_Name New Datatype

Following example shows how to modify a column of a table:
alter table Employee_Information modify Emp_Phone_Number varchar (20)

```php
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$dbase_name = "teslin"; mysqli_select_db($name1, $dbase_name) or
die(mysql_error());

$query = " alter table Employee_Information change Emp_PhoneNumber
Emp_Phone_Number varchar(10)";

$name1 -> query($query);
echo "Column name changed";

mysqli_close($name1);
?>
```

Inserting data in a table:

```php
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$dbase_name = "teslin";
mysqli_select_db($name1, $dbase_name) or die(mysql_error());

$query = " INSERT INTO Employee_Information (Emp_ID,
Emp_Name,Emp_Address,Emp_Phone_Number) VALUES (1002, 'david', 'margao',
'12345')";

$name1 -> query($query);
echo "DAta entered";

mysqli_close($name1);
?>
```

Modifying data in a table:

The data stored in a table often needs to be modified to keep the information updated. Following syntax is used to update data in a table:

UPDATE table name
SET column1 = value1, column2 = value2
WHERE condition

•UPDATE -Specifies that the query is used to update data
•SET -Specifies new values to be inserted into the table
•WHERE -Specifies the condition that selects the rows to be affected

```php
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$dbase_name = "teslin";
mysqli_select_db($name1, $dbase_name) or die(mysql_error());

$query = " UPDATE Employee_Information SET Emp_Name = 'GEC' where Emp_ID =
1002";

$name1 -> query($query);
echo "Data modified";
mysqli_close($name1); ?>
```

Retrieving Data:

```php
<?php
$name1 = new mysqli('localhost', 'root',  '');
echo "Connection to the server was successful <br/>";
$dbase_name = "teslin";
mysqli_select_db($name1, $dbase_name) or die(mysql_error());

$query = " SELECT * FROM Employee_Information";

$data = $name1 -> query($query);
print "<table border cellpadding = 3>";
while ($info = mysqli_fetch_array( $data )) {
  print "<tr>";
  print "<th>Emp ID:</th> <td>" .$info['Emp_ID'] . "</td>";
  print "<th>NAme:</th> <td>" .$info['Emp_Name'] . "</td>";
  print "<th>Address:</th> <td>" .$info['Emp_Address'] . "</td>";
  print "<th>Phone:</th> <td>" .$info['Emp_Phone_Number'] . "</td>";
}
print "</table>";
```

```
mysqli_close($name1);
?>
```