## INTRODUCTION TO PHP

- ✓ PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- ✓ PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- ✓ PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

**What is PHP?**

- ✓ PHP is an acronym for "PHP: Hypertext Preprocessor".
- ✓ PHP is a widely-used, open source scripting language.
- ✓ PHP scripts are executed on the server.

**What is a PHP File?**

- ✓ PHP files can contain text, HTML, CSS, JavaScript, and PHP code.
- ✓ PHP files have extension ".php"

**What Can PHP Do?**

- ✓ PHP can generate dynamic page content
- ✓ PHP can create, open, read, write, delete, and close files on the server
- ✓ PHP can collect form data
- ✓ PHP can send and receive cookies
- ✓ PHP can add, delete, modify data in your database

**Why PHP?**

- ✓ PHP runs on various platforms (Windows, Linux, UNIX, Mac OS X, etc.)
- ✓ PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- ✓ PHP supports a wide range of databases

**PHP Environment Setup**

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

1. **Web Server** – PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
2. **Database** – PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
3. **PHP Parser** – In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

**Basic PHP Syntax:**

- ✓ A PHP script can be placed anywhere in the document.
- **Canonical PHP tags:**

```
<?php
// PHP code goes here
?>
```

- **Short-open (SGML-style) tags:**

```
<?...?>
```

- **HTML script tags:**

```
<script language="PHP">...</script>
```

- ✓ The default file extension for PHP files is ".php".

**Example:**

```
<html><body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body></html>
```

**OUTPUT:**
**My first PHP page**
Hello World!

- ✓ **Comments in PHP:**

```
// this is a single-line comment
# this is also a single-line comment
/* this is a multiple-lines comment block that spans over multiple lines */
```

- ✓ **Displaying in PHP:** In PHP there are two basic ways to get output: echo and print.

## 1.1 Declaring variables

In PHP, a variable starts with the $ sign, followed by the name of the variable.

**Example:**

```
<?php
$txt = "Hello world!"; // string
$x = 5; //int
$y = 10.5; //float
echo $txt."<br>". $x."<br>";
print $y;
?>
```

**OUTPUT:**

```
Hello world!
5
10.5
```

**PHP Variables**

- ✓ A variable starts with the $ sign, followed by the name of the variable
- ✓ A variable name must start with a letter or the underscore character
- ✓ A variable name cannot start with a number
- ✓ A variable name can contain alpha-numeric characters and underscores (A-z, 0-9, and _ )
- ✓ Variable names are case-sensitive ($age and $AGE are two different variables)

**1.2 Data types**

Variables can store data of different types, and different data types can do different things. PHP supports the following data types:

- ✓ String
- ✓ Integer
- ✓ Float (floating point numbers - also called double)
- ✓ Boolean
- ✓ Array
- ✓ NULL

**Integer:** An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

**NOTE:** The PHP var_dump() function returns the data type and value:

**Example**

```
<?php
$x = 5985;
var_dump($x);
?>
```

**OUTPUT:**
```
int(5985)
```

**Float:** A float (floating point number) is a number with a decimal point or a number in exponential form. In the following example $x is a float. The PHP var_dump() function returns the data type and value:

**Example**

```
<?php
$x = 10.365;
var_dump($x);
?>
```

**OUTPUT:**
```
float (10.365)
```

**Boolean:** A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

## 1.3 Operators Expressions

PHP language supports following type of operators.

- ✓ Arithmetic Operators (+,-,*,/,%)
- ✓ Comparison Operators (==,!=,<,>,<=,>=)
- ✓ Logical (or Relational) Operators(&&,||)
- ✓ Assignment Operators(=)
- ✓ Conditional (or ternary) Operators (?:)
- ✓ Increment/Decrement Operators(++,--)

## 1.4 Strings:

A string is a sequence of characters, like "Hello world!"

**Note:** A string can be any text inside quotes. You can use single or double quotes:

**Example**

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo "<br>";
echo $y;
?>
```

**OUTPUT:**

```
Hello world!
Hello world!
```

### String Functions

1. **strlen()** function returns the length of a string

```
<?php
echo strlen("Hello world!"); //outputs 12
?>
```

2. **str_word_count()** function counts the number of words in a string

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

3**. strrev()** function reverses a string

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

4. **strpos()** function searches for a specific text within a string

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

5. **str_replace()** function replaces some characters with some other characters in a string

```
<?php
echo str_replace("world", "Dolly", "Hello world!");
// outputs Hello Dolly!
?>
```

**PHP Date and Time:** The PHP date() function is used to format a date and/or a time.

**Syntax**

date(*format ,timestamp*)

| Parameter | Description |
|---|---|
| Format | Required. Specifies the format of the timestamp |
| timestamp | Optional. Specifies a timestamp. Default is the current date and time |

Here are some characters that are commonly used for dates:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters.

**Example:**

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

Here are some characters that are commonly used for times:

- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)

- a - Lowercase Ante meridiem and Post meridiem (am or pm)

  **Example:**

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

**1.5 Control Structures**

**Conditional Statements:** In PHP we have the following conditional statements:

✓ **if Statement :** executes some code if one condition is true

**Syntax:**

```
if (condition) {
  code to be executed if condition is true;
 }
```

**Example:**

```
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}  ?>
```

✓ **if...else Statement:** executes some code if a condition is true and another code if that condition is false

**Syntax:**

```
if (condition) {
   code to be executed if condition is true;
} else {
   code to be executed if condition is false;
}
```

**Example:**

```
<?php
$age = 15;
if($age < 18)
   echo 'Child'; // Display Child if age is less than 18
else
   echo 'Adult'; // Display Adult if age is greater than or equal to 18
?>
```

Using the ternary operator the same code could be written in a more compact way:

```
<?php
$age = 15;
echo ($age < 18) ? 'Child' : 'Adult';
?>
```

- ✓ **if...elseif....else Statement:** executes different codes for more than two conditions

**Syntax:**

```
if (condition) {
   code to be executed if this condition is true;
} elseif (condition) {
   code to be executed if this condition is true;
} else {
   code to be executed if all conditions are false;
}
```

**Example:**

```
<?php
$d = date("D");
if($d == "Fri")
        echo "Have a nice weekend!";
 elseif($d == "Sun")
        echo "Have a nice Sunday!";
 else
        echo "Have a nice day!";
 ?>
```

- ✓ **switch Statement:** selects one of many blocks of code to be executed.

**Syntax:**

```
switch (n) {
   case label1:      //code to be executed if n=label1;
      break;
   case label2:     //  code to be executed if n=label2;
      break;
    default:      //code to be executed if n is different from all labels;
}
```

**Example:**

```
<?php
$favcolor = "red";
switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
} ?>
```

**OUTPUT:**
```
Your favorite color is red
```

In PHP, we have the following looping statements:

- ✓ **while -** loops through a block of code as long as the specified condition is true

**Syntax:**

while (*condition is true*) {
  *code to be executed*;
}

**Example:**// php program to find reverse of a number

```
<?php
$num = 23456;
$revnum = 0;
while ($num > 0)
{
        $rem = $num % 10;
        $revnum = ($revnum * 10) + $rem;
        $num = ($num / 10);
}
echo "Reverse number of 23456 is: $revnum";  ?>
```

- ✓ **do...while -** loops through a block of code once, and then repeats the loop as long as the specified condition is true.

  **Syntax:**

  ```
  do {
     code to be executed;
  } while  (condition is true);
  ```

**Example:** // Program to print 1 to 5

```
<?php
$i = 0;
do{
echo $i + 1 . "<br>";
$i++;
} while ($i < 5);
?>
```

- ✓ **for -** loops through a block of code a specified number of times

  **Syntax:**

  ```
  for (init counter; test counter; increment counter) {
     code to be executed;
  }
  ```

**Example:** // php program to find factorial of a number

```
<?php
$num = 4;
$fact = 1;
for ($x=$num; $x>=1; $x--)
{
  $fact = $fact * $x;
}
echo "Factorial of $num is $fact";
?>
```

- ✓ **foreach -** loops through a block of code for each element in an array. This loop is only used for arrays.

**Syntax:**

```
foreach ($array as $value) {
   code to be executed;
}
```

**Example:**

```
<?php
$array = array( 1, 2, 3, 4);
foreach( $array as $value )
{
    echo "Value is $value <br />";
}
?>
```

**OUTPUT:**

```
Value is 1
Value is 2
Value is 3
Value is 4
```

**1.6 Functions**

- ✓ A function is a block of statements that can be used repeatedly in a program.
- ✓ A function will not execute immediately when a page loads.
- ✓ A function will be executed by a call to the function.
- ✓ There are two types of functions
    - Predefined functions
    - User-defined functions

**User-defined functions:**

A user-defined function declaration starts with the word function:

**Syntax:**

```
function functionName() {
   code to be executed;
}
```

**Note:** A function name can start with a letter or underscore (not a number).

**Example:**

```
<?php
function writeMsg() {
   echo "Hello world!";
}
writeMsg();
?>
```

**OUTPUT:**

```
Hello world
```

**Passing Parameters:**

- ✓ **Call by value**: In call by value, actual value is not modified if it is modified inside the function.

  **Example :**

```
<?php
 function swap($first, $second)
{
        $temp = $first;
        $first = $second;
        $second = $temp;
 }
 $a = 5;
$b = 7;
 swap ($a, $b);
 echo "First number ".$a." <br> Second number ".$b;
?>
```

**OUTPUT:**

```
First number 5
Second number 7
```

- ✓ **Call by reference:** In call by reference, actual value is modified if it is modified inside the function.

  **Example:**

```
<?php
function swap(&$first, &$second)
{
        $temp = $first;
        $first = $second;
        $second = $temp;
}
$a = 5;
$b = 7;
swap ($a, $b);
echo "First number ".$a." <br> Second number ".$b;
?>
```

**OUTPUT:**

```
First number 7
Second number 5
```

**PHP Functions returning value**

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

**Example:**

```
<?php
function addFunction($num1, $num2) {
  $sum = $num1 + $num2;
  return $sum;
}
$c = addFunction(10, 20);
   echo "Returned value from the function : $c";
?>
```

**OUTPUT:**

```
Returned value from the function: 30
```

**1.7 Arrays**

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables it’s easy to define an array of 100 length.

There are three different kinds of arrays and each array value is accessed using an ID which is called array index.

1. **Numeric array –** An array with a numeric index. Values are stored and accessed in linear fashion.
2. **Associative array –** An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
3. **Multidimensional array –** An array containing one or more arrays and values are accessed using multiple indices

**Numeric Array:**

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

There are two ways to create arrays.

```
$cars=array (“BEAT”,”BMW”,”ALTO”);
                  Or
$cars[0]=”BEAT”;
$cars[1]=”BMW”;
$cars[2]=”ALTO”;
```

**Example :**

```
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);
        foreach( $numbers as $value ) {
   echo "Value is $value <br />";
}
 /* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";
 foreach( $numbers as $value ) {
   echo "Value is $value <br />";
}     ?>
```

**OUTPUT:**

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

**Associative Arrays:**

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

$rollno=array("divya"=>"1","diya"=>"2","riya"=>"3");

Or

$rollno['divya']="1";

$rollno['diya']="2";

$rollno['riya']="3";

**Example:**

```
 <?php
/* First method to associate create array. */
$salaries = array("Rikki" => 2500, "Nikki" => 1500, "zara" => 500);
 echo "Salary of Rikki  is ". $salaries['Rikki'] . "<br />";
 echo "Salary of Nikki is ".  $salaries['Nikki']. "<br />";
 echo "Salary of zara is ".  $salaries['zara']. "<br />";
 /* Second method to create array. */
```

```
    $salaries['Rikki'] = "high";
    $salaries['Nikki'] = "medium";
    $salaries['zara'] = "low";
     echo "Salary of Rikki is ". $salaries['Rikki'] . "<br />";
    echo "Salary of Nikki is ".  $salaries['Nikki']. "<br />";
    echo "Salary of zara is ".  $salaries['zara']. "<br />";
  ?>
```

**OUTPUT:**

```
Salary of Rikki is 2000
Salary of Nikki is 1000
Salary of zara is 500
Salary of Rikki is high
Salary of Nikki is
medium
```

**Multidimensional Arrays:**

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

**Example:**

```
 <?php
  $marks = array( "aaa" => array ("WT" => 35,"CD" => 30,"DP" => 39 ),
                 "bbb" => array ("WT" => 30,"CD" => 32,"DP" => 29 ),
                 "ccc" => array ("WT" => 31,"CD" => 22,"DP" => 39 ) );

  /* Accessing multi-dimensional array values */
  echo "Marks for aaa in WT : " ;
  echo $marks[' aaa '][' WT'] . "<br />";
  echo "Marks for bbb in maths : ";
  echo $marks['bbb']['CD'] . "<br />";
  echo "Marks for ccc in DP : " ;
  echo $marks['ccc']['DP'] . "<br />";
?>
```

**OUTPUT:**

```
Marks for aaa in WT : 35
Marks for bbb in CD: 32
Marks for ccc in DP: 39
```

**Sort Functions for Arrays**

- ✓ sort() - sort arrays in ascending order
- ✓ rsort() - sort arrays in descending order
- ✓ asort() - sort associative arrays in ascending order, according to the value
- ✓ ksort() - sort associative arrays in ascending order, according to the key
- ✓ arsort() - sort associative arrays in descending order, according to the value
- ✓ krsort() - sort associative arrays in descending order, according to the key

**Example:** // Program for displaying numbers in ascending order using sort()

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
$arrlength = count($numbers);
for($x = 0; $x < $arrlength; $x++) {
   echo $numbers[$x];
   echo "<br>";
}?>
```

```
OUTPUT:
2
4
6
11
22
```

**Example:** // Program for displaying numbers in descending order using rsort()

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
$arrlength = count($numbers);
for($x = 0; $x < $arrlength; $x++) {
   echo $numbers[$x];
   echo "<br>";
}
?>
```

```
OUTPUT:
22
11
6
4
2
```

**PHP File Inclusion**

There are two PHP functions which can be used to include one PHP file into another PHP file.

- ✓ The include() Function
- ✓ The require() Function

**Syntax**

```
include 'filename';
or
require 'filename';
```

**Example:**

**even.php**

```
<?php
echo "Hello III CSE"
?>
```

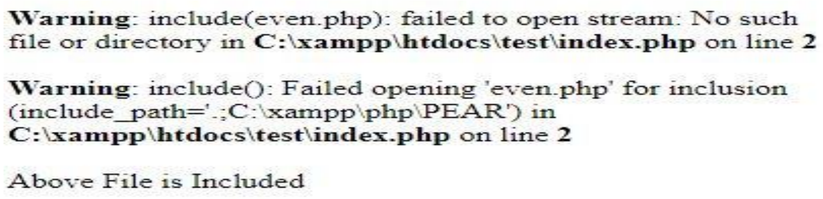
```
OUTPUT:
Hello III CSE
Above File is Included
```

**index.php**

```
<?php
  include("even.php");
  echo "<br>Above File is Included"
?>
```

**include() VS require():** The both function acts as same and produce same results, but if by any chance a fatal error is arised, then the difference comes into the surface.

Now if we don't have a file named **even.php**, then in the case of the include(), following output will be shown with warnings about missing file, but atleast the output will be shown from the index.php file:

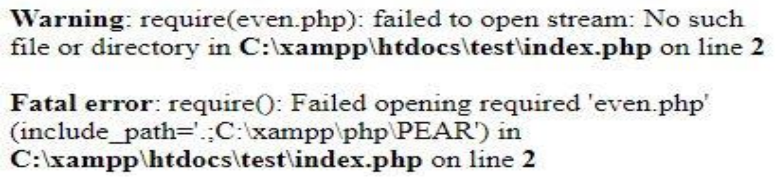


```
Warning: include(even.php): failed to open stream: No such
file or directory in C:\xampp\htdocs\test\index.php on line 2

Warning: include(): Failed opening 'even.php' for inclusion
(include_path='.;C:\xampp\php\PEAR') in
C:\xampp\htdocs\test\index.php on line 2

Above File is Included
```

In the case of the require(), if the file PHP file is missing, a **fatal error** will rise and no output is shown and the execution halts.



```
Warning: require(even.php): failed to open stream: No such
file or directory in C:\xampp\htdocs\test\index.php on line 2

Fatal error: require(): Failed opening required 'even.php'
(include_path='.;C:\xampp\php\PEAR') in
C:\xampp\htdocs\test\index.php on line 2
```

**PHP Global Variables - Superglobals**

- ✓ **$GLOBALS:** It is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

  **Example:**

```
<?php
$x = 75;
$y = 25;
function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z;
?>
```

**OUTPUT:**
100

- ✓ **$_SERVER:** It is a PHP super global variable which holds information about headers, paths, and script locations.

  **Example:**

  ```
  <?php
  echo $_SERVER['PHP_SELF'];  //Returns filename of the currently executing script
  echo "<br>";
  echo $_SERVER['SERVER_NAME'];// Returns the name of the host server
  echo "<br>";
  echo $_SERVER['SERVER_ADDR'];  //Returns the IP address of the host server
  echo "<br>";
  echo $_SERVER['SCRIPT_NAME'];// Returns the path of the current script
  echo "<br>";
  echo $_SERVER['HTTP_HOST'];             //returns host header
  ?>
  ```

- ✓ **$_REQUEST:** It is used to collect data after submitting an HTML form.

  **Example:**

  ```
  <html><body>
  <form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
   Name: <input type="text" name="fname">
   <input type="submit"></form>
  <?php
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
      $name = htmlspecialchars($_REQUEST['fname']); // collect value of input field
    if (empty($name))
      echo "Name is empty";
   else
      echo $name;
    }?>
  </body></html>
  ```

- ✓ **$_POST:** It is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.
- ✓ **$_GET:** can also be used to collect form data after submitting an HTML form with method="get". $_GET can also collect data sent in the URL.

- ✓ **$_FILES:** This superglobal variable represents an associative array and is used to upload files to the php scripts.
- ✓ **$_ENV:** Contains all environment variables set by your system or shell for the script.
- ✓ **$_COOKIE:** It represents an associative array and is used to fetch user information sent by the web server. Whenever a user requests a page, a cookie is sent to the computer and to fetch the information of that cookie, we use $_COOKIE variable.
- ✓ **$_SESSION:** Contains all variables stored in a user's session.

## 1.8 Reading data from web form controls like text boxes radio buttons, lists etc.,

### GET vs. POST

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters.

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

**Example:**

**Data.html**

```
<html> <body>
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form></body></html>
```

**welcome_get.php**

```
<html><body>
Welcome
<?php echo $_GET["name"]; ?><br>
Your email address is:
<?php echo $_GET["email"]; ?>
</body></html>
```

**For Check boxes we use foreach loop**

```
foreach($_GET['c1'] as $selected
```

**Example: registration form.html**

```
<html>
<body bgcolor="cyan">
<table align="center"><caption>Student Registration Form</caption>
<form action="post" action="info.php" >
<tr><td>User name:</td>
<td><input type="text" name="userid"></td></tr>
<tr><td>Password:</td>
<td><input type="password" name="psw"></td></tr>
<tr><td>Mobile number:</td>
<td><input type="text" name="phnum"></td></tr>
<tr><td>Email id:</td>
<td><input type="text" name="mailid"></td></tr>
<tr><td>Gender:</td>
<td><input type = "radio" name = "r1" value = "male"> Male
<input type = "radio" name = "r1" value = "female">Female
</td></tr>
<tr><td>Select Place:</td>
<td><select><option value="1">Telangana</option>
        <option value="2">Andra Pradhesh</option>
        <option value="3">Tamil Nadu</option>
       </select></td></tr>
<tr><td>Languages Known:</td>
<td>
<input type = "checkbox" name = "c1" value = "c1"> Telugu
<input type = "checkbox" name = "c1" value = "c2"> English
<input type = "checkbox" name = "c1" value = "c3"> Hindi
</td></tr>
<tr><td>Address:</td>
<td><textarea rows = "5" cols = "50" name = "description"></textarea></td></tr>
<tr><td><input type = "submit" name = "submit" value = "Submit" /></td>
<td><input type = "reset" name = "reset"  value = "Reset" /></td></tr>
</form></table></body></html>
```

**info.php**

```
<html><body>
Welcome <?php echo $_POST["userid"]; ?><br>
Your Password is<?php echo $_POST["psw"]; ?><br>
Your Mobile Number is<?php echo $_POST["phnum"]; ?><br>
Your email address is: <?php echo $_POST["mailid"]; ?><br>
Gender is: <?php echo $_POST["r1"]; ?><br>
Place is:<?php echo $_POST["place"]; ?><br>
Languages known:
 <?php
foreach($_POST['c1'] as $selected){
echo $selected;}
?><br>
Address is<?php echo $_POST["description"]; ?>
</body></html>
```