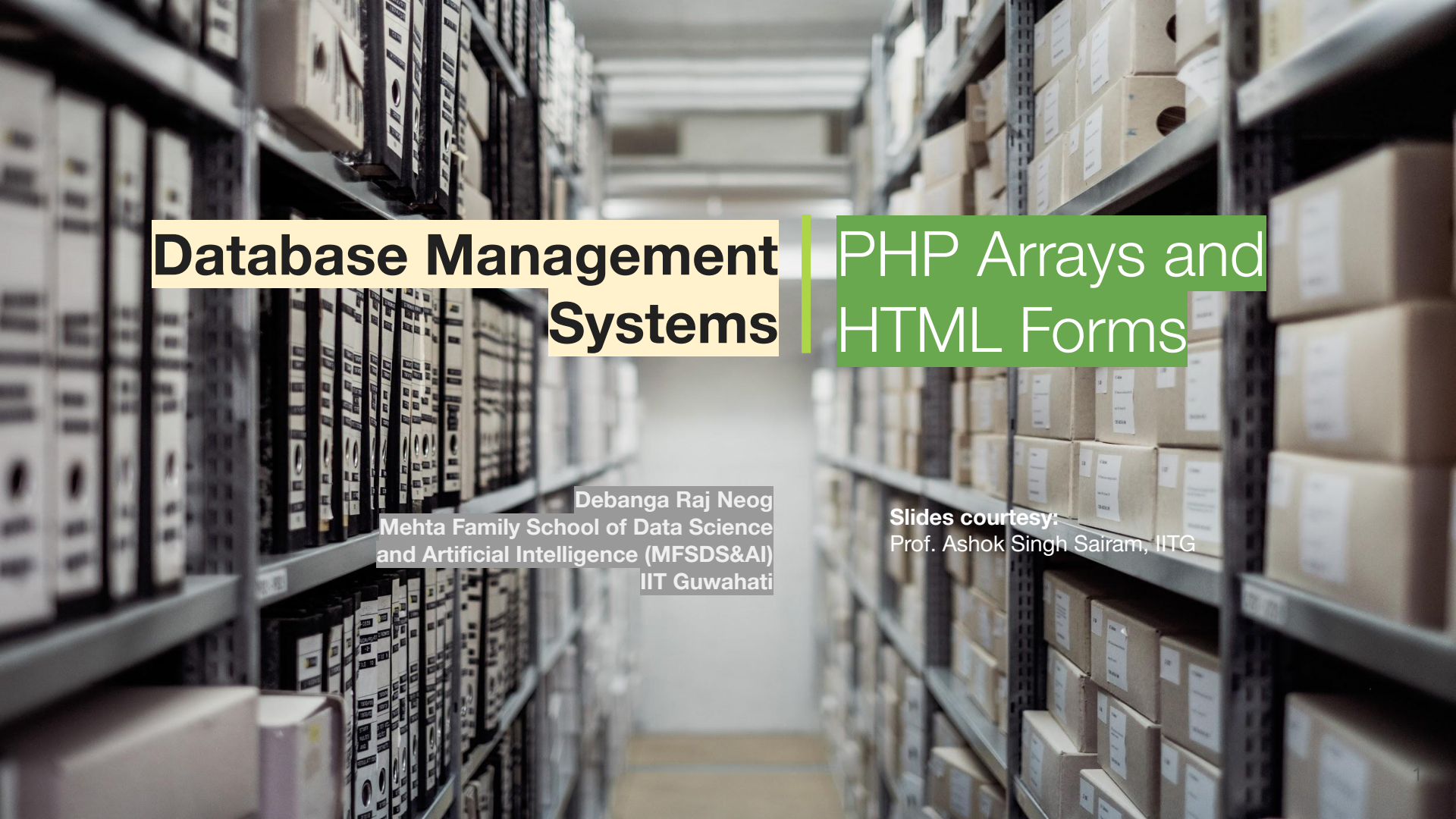


Database Management Systems

PHP Arrays and HTML Forms

Debanga Raj Neog
Mehta Family School of Data Science and Artificial Intelligence (MFSDS&AI)
IIT Guwahati

Slides courtesy:
Prof. Ashok Singh Sairam, IITG

Lecture Plan

- PHP Arrays
  - Array types – Numeric and Associative
  - Printing arrays
  - Looping through an array
  - Array functions
- HTML Forms
  - Syntax of HTML forms
  - GET and Post method
  - Processing HTML forms using PHP

PHP Arrays: Numerically indexed arrays (1)

- The keyword **array** is a constructor
- Integer indices starting from 0

PHP Arrays: Numerically indexed arrays (2)

- The keyword **array** is a constructor
- Integer indices starting from 0
- Creating arrays
  - Start from an empty array and add items

```
<?php
        $words=array();
        $words[]="Hello";
        $words[]="World";
        echo $words[1];
?>
```

Output: World

PHP Arrays: Numerically indexed arrays (3)

- The keyword **array** is a constructor
- Integer indices starting from 0
- Creating arrays
  - Start from an empty array and add items
  - Copy one array to the other using the “=“

```
<?php
	$words=array(“hello”,”world”);
	$greet=$words;
?>
```

PHP Arrays: Numerically indexed arrays (4)

- The keyword **array** is a constructor
- Integer indices starting from 0
- Creating arrays
  - Start from an empty array and add items
  - Copy one array to the other using the “=“
  - Use the range() function to auto populate an array

```
<?php
        $numbers = range(1,10);
?>
$numbers[0] is assigned 1,
$numbers[1] is assigned 2,
and so on
```

PHP Arrays: Associative arrays

- In associative arrays, we associate a key (or index) with each value

  key=>value

PHP Arrays: Associative arrays

- In associative arrays, we associate a key (or index) with each value

  key=>value

```
<?php
    $DA512H=array("instructor"=>"Debanga","course
name"=>"DBMS");
    echo $DA512H['course name'] ."\t". $DA512H['instructor'], "\n";
?>
```

Output

DBMS Debanga

Dumping Arrays

- print_r(): shows information about a variable, good for debugging

Dumping Arrays

- print_r(): shows information about a variable, good for debugging
- Array values will presented as **key** and **value**.

Dumping Arrays

- print_r(): shows information about a variable, good for debugging
- Array values will presented as **key** and **value**.

```
<pre>
Text in a pre element
is displayed in a fixed-width
font, and it preserves
both      spaces and
line breaks
</pre>
```

```
<?php
$DA512H=array("instructor"=>"Debanga","course
name"=>"DBMS");
echo("<pre>\n");
print_r($DA512H);
echo("</pre>");
?>
```

Output

```
Array
(
    [instructor] => Debanga
    [course name] => DBMS
)
```

var_dump vs. print_r

- var_dump displays structured information about variables/expressions including its **type**.

var_dump vs. print_r

- var_dump displays structured information about variables/expressions including its **type**.

```
<?php
	$DA214=array("instructor"=>"Debanga",
"course name"=>"DBMS");
	echo("<pre>\n");
	var_dump($DA214);
	echo("</pre>");
?>
```

Output

```
array(2) {
  ["instructor"]=>   string(5) "Debanga"
  ["course name"]=>   string(4) "DBMS"
}
```

Looping through an array

- Numerically indexed arrays: As the array is indexed by numbers we can use a **for** loop
- Loop **foreach**: Iterate through each item of an array
  - specially designed for arrays

Looping through an array

- Numerically indexed arrays: As the array is indexed by numbers we can use a **for** loop
- Loop **foreach**: Iterate through each item of an array
  - specially designed for arrays

```
<?php
	$words=array("Hello","World");
	for($i=0;$i<2;$i++)
		echo $words[$i],"\n"	;
?>
```

```
<?php
	$words=array("Hello","World");
	foreach ($words as $singleword)
		echo $singleword,"\n"	;
?>
```

Looping through an Array (2)

- **foreach** loop in Associative arrays
  - can iterate through each item as in case of numerically indexed arrays
  - we can incorporate the keys as well, $key=>$value

Looping through an Array (2)

- **foreach** loop in Associative arrays
  - can iterate through each item as in case of numerically indexed arrays
  - we can incorporate the keys as well, $key=>$value

```
<?php
	header('Content-type: text/plain');
	$DA214=array("instructor"=>"Debanga","cname"=>"DBMS");
	foreach($DA214 as $k => $val) {
		echo "Key: " , $k , " Value: " , $val , "\n";
	}
?>
```

Output

Key: instructor Value: Debanga
Key: cname Value: DBMS

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array

```
<?php
$number = array(‘one' => 1, ‘two' => 2);
if (array_key_exists(‘one', $number)) {
   echo "The key element ‘one’ is in the array";
}
?>
```

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key (variable) is set in the array

```
<?php
$number = array(‘one' => 1, ‘two' => 2);
if (isset($number[‘one‘]) {
   echo "The element ‘one’ is in the array";
}
?>
```

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key is set in the array
- **count**($ar): Count the number of elements in the array

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key is set in the array
- **count**($ar): Count the number of elements in the array
- **is_array**($ar): Returns TRUE if variable ar is an array

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key is set in the array
- **count**($ar): Count the number of elements in the array
- **is_array**($ar): Returns TRUE if variable ar is an array
- **sort**($ar): Sorts the array values (losses keys)

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

Output: 2, 4, 6, 11, 22

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key is set in the array
- **count**($ar): Count the number of elements in the array
- **is_array**($ar): Returns TRUE if variable ar is an array
- **sort**($ar): Sorts the array values (losses keys)
- **ksort**($ar): Sorts the array by keys

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

Output: Ben, Joe, Peter

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key is set in the array
- **count**($ar): Count the number of elements in the array
- **is_array**($ar): Returns TRUE if variable ar is an array
- **sort**($ar): Sorts the array values (losses keys)
- **ksort**($ar): Sorts the array by keys
- **asort**($ar): Sorts the array by values, keep association

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

Output: 35, 37, 43

Array Functions

- **array_key_exists**($key,$ar): Returns TRUE if key is set in array ar
- **isset**($ar[‘key’]): Returns TRUE if key is set in the array
- **count**($ar): Count the number of elements in the array
- **is_array**($ar): Returns TRUE if variable ar is an array
- **sort**($ar): Sorts the array values (losses keys)
- **ksort**($ar): Sorts the array by keys
- **asort**($ar): Sorts the array by values, keep association
- **shuffle**($ar): Shuffles the array into random order

PHP - Dealing with the Client

- PHP allows us to use HTML forms
- Forms collect data from the client and send it to the backend application for processing
- Forms require technology at the server to process them
- PHP is a feasible and good choice for the processing of HTML forms

Anatomy of HTML form Tag

- The HTML form tag is used to create an HTML form

```
<form action = "Script URL" method = "GET|POST">
        form elements like input, text area etc.
</form>
```

Anatomy of HTML form Tag

- The HTML form tag is used to create an HTML form
  - **action**: backend script to process data passed from the client

```
<form action = "Script URL" method = "GET|POST">
        form elements like input, text area etc.
</form>
```

Anatomy of HTML form Tag

- The HTML form tag is used to create an HTML form
  - **action**: backend script to process data passed from the client
  - **method**: The method specifies how the data will be uploaded
    - **GET** method sends all form input in the URL requested, using name=value pairs separated by ampersands (&)

```
<form action = "Script URL" method = "GET|POST">
        form elements like input, text area etc.
</form>
```

Anatomy of HTML form Tag

- The HTML form tag is used to create an HTML form
  - **action**: backend script to process data passed from the client
  - **method**: The method specifies how the data will be uploaded
    - **GET** method sends all form input in the URL requested, using name=value pairs separated by ampersands (&)
      - E.g. process.php?name=trevor&number=345
      - Is visible in the URL shown in the browser

```
<form action = "Script URL" method = "GET|POST">
        form elements like input, text area etc.
</form>
```

Anatomy of HTML form Tag

- The HTML form tag is used to create an HTML form
  - **action**: backend script to process data passed from the client
  - **method**: The method specifies how the data will be uploaded
    - **GET** method sends all form input in the URL requested, using name=value pairs separated by ampersands (&)
      - E.g. process.php?name=trevor&number=345
      - Is visible in the URL shown in the browser
    - **POST** method sends all contents of a form with basically hidden headers (not easily visible to users)

```
<form action = "Script URL" method = "GET|POST">
        form elements like input, text area etc.
</form>
```

$_GET and $_POST

- All form values are placed into an array

$_GET and $_POST

- All form values are placed into an array
- PHP loads the values for the URL parameters into an array called **$_GET** and the POST parameters into an array called **$_POST**

$_GET and $_POST

- All form values are placed into an array
- PHP loads the values for the URL parameters into an array called **$_GET** and the POST parameters into an array called **$_POST**
- There is another array called $_REQUEST which merges GET and POST data

Example: $_GET and $_POST

- Assume a form contains one textbox called “txtName” and the form is submitted using the post method, invoking process.php

```
<form action=“process.php" method="post" >

 <input type=“text"  name=“txtName">
</form>
```

Example: $_GET and $_POST

- Assume a form contains one textbox called “txtName” and the form is submitted using the post method, invoking process.php
- process.php could access the form data using:
  - $_POST[‘txtName’]

```
<form action=“process.php" method="post" >

 <input type=“text"  name=“txtName">
</form>
```

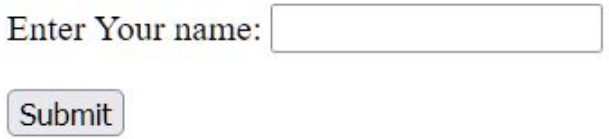
Example: $_GET and $_POST

- Assume a form contains one textbox called “txtName” and the form is submitted using the post method, invoking process.php
- process.php could access the form data using:
  - $_POST[‘txtName’]
- If the form used the get method, the form data would be available as:
  - $_GET[‘txtName’]

```
<form action=“process.php" method=“get" >

 <input type=“text"  name=“txtName">
</form>
```

HTML Forms: A simple example

```
<form action="" method="post" id="numbers">
   <p>  Enter Your name: <input type = “text” name =
“my_name”> <p>

<input type="submit" value="Submit" name="submit">
</form>
```

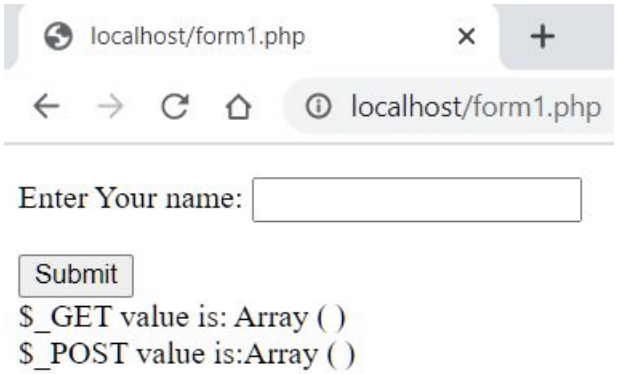


HTML Forms: A simple example

```
<form action="" method="post" id="numbers">
   <p>  Enter Your name: <input type = “text” name =
“my_name”> <p>

<input type="submit" value="Submit" name="submit">
</form>

<?php
 echo '$_GET value is: ';
 print_r($_GET);
 echo "<br>";
 echo  '$_POST value is:';
 print_r($_POST);
  ?>
```

HTML Forms: A simple example

```
<form action="" method="post" id="numbers">
   <p>  Enter Your name: <input type = “text” name =
“my_name”> <p>

<input type="submit" value="Submit" name="submit">
</form>

<?php
 echo '$_GET value is: ';
 print_r($_GET);
 echo "<br>";
 echo  '$_POST value is:';
 print_r($_POST);
  ?>
```



HTML Forms: A simple example (2)

```
<form action="" method=“get" id="numbers">
   <p>  Enter Your name: <input type = “text” name =
“my_name”> <p>

<input type="submit" value="Submit" name="submit">
</form>

<?php
 echo '$_GET value is: ';
 print_r($_GET);
 echo "<br>";
 echo  '$_POST value is:';
 print_r($_POST);
  ?>
```

HTML Form Controls

- Different type of form controls use to collect data using HTML form
  - **Text Input Controls**
  - Hidden Controls
  - **Submit and Reset Button**
  - Checkboxes Controls
  - Radio Box Controls
  - Select Box Controls
  - File Select boxes
  - Clickable Buttons

Courtesy: https://www.tutorialspoint.com/html/html_forms.htm

Text Input Controls

- <input> tag is used for items that require only one line of user input, such as search boxes or names

```
<form action = "Script URL" method = "GET|POST">
        First name: <input type = "text" name = "first_num“ id="first_num"
        required="required" value="0" />
</form>
```

Text Input Controls

- <input> tag is used for items that require only one line of user input, such as search boxes or names
  - **type**: indicates the type of input control – text, number, password

```
<form action = "Script URL" method = "GET|POST">
        First name: <input type = "text" name = "first_num“ id="first_num"
        required="required" value="0" />
</form>
```

Text Input Controls

- <input> tag is used for items that require only one line of user input, such as search boxes or names
  - **type**: indicates the type of input control – text, number, password
  - **name**: give a name to the control, can be used at server side to get the value

```
<form action = "Script URL" method = "GET|POST">
        First name: <input type = "text" name = "first_num“ id="first_num"
        required="required" value="0" />
</form>
```

Text Input Controls

- <input> tag is used for items that require only one line of user input, such as search boxes or names
  - **type**: indicates the type of input control – text, number, password
  - **name**: give a name to the control, can be used at server side to get the value
  - **id**: specify a unique id for an HTML element

```
<form action = "Script URL" method = "GET|POST">
    First name: <input type = "text" name = "first_num“ id="first_num"
    required="required" value="0" />
</form>
```

Text Input Controls

- <input> tag is used for items that require only one line of user input, such as search boxes or names
  - **type**: indicates the type of input control – text, number, password
  - **name**: give a name to the control, can be used at server side to get the value
  - **id**: specify a unique id for an HTML element
  - **required**: Boolean attribute, field must be filled out before submitting form

```
<form action = "Script URL" method = "GET|POST">
        First name: <input type = "text" name = "first_num“ id="first_num"
        required="required" value="0" />
</form>
```

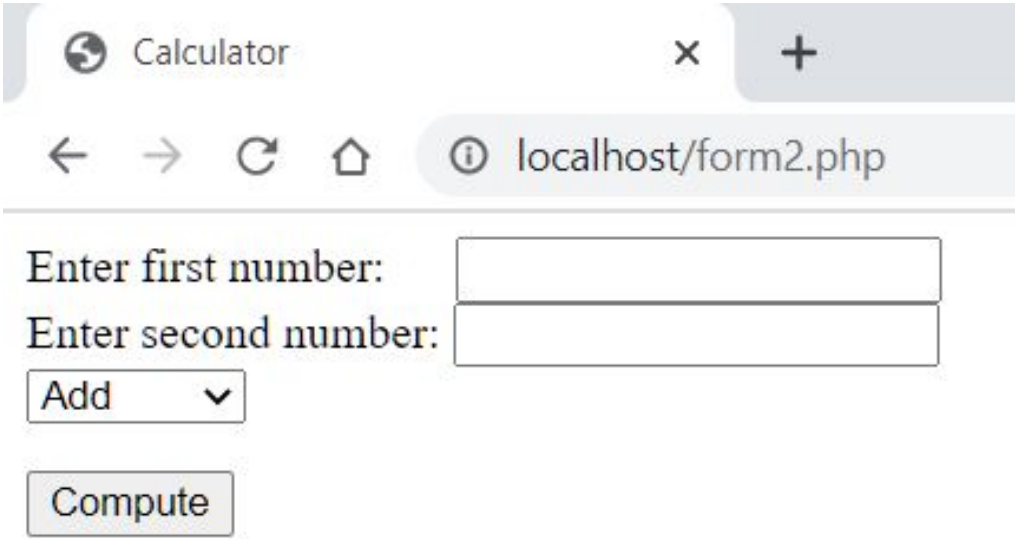
Text Input Controls

- <input> tag is used for items that require only one line of user input, such as search boxes or names
  - **type**: indicates the type of input control – text, number, password
  - **name**: give a name to the control, can be used at server side to get the value
  - **id**: specify a unique id for an HTML element
  - **required**: Boolean attribute, field must be filled out before submitting form
  - **value**: used to provide an initial value to the control

```
<form action = "Script URL" method = "GET|POST">
        First name: <input type = "text" name = "first_num“ id="first_num"
        required="required" value="0" />
</form>
```

HTML Form: A simple calculator

```
<form action="" method="post" id="numbers">
  <div>
    Enter first number:
    <input type="number" name="first_num" id="first" required="required" value="0">
  </div>
  <div>
    Enter second number:
    <input type="number" name="second_num" id="second" required="required" value="0">
  </div>

  <select name ="operator">
    <option value = "Add" selected>Add</option>
    <option value = "Subtract">Subtract</option>
    <option value = "Multiply">Multiply</option>
    <option value = "Divide">Divide</option>
  </select>
    <p> <span text-align:center >
    <input type="submit" value="Compute"> </span> </p>
</form>
```


Calculator
localhost/form2.php
Enter first number:
Enter second number:
Add
Compute

Processing POST data

```
        ……..
</form>
<?php
 if (isset($_POST['first_num']) && isset($_POST['second_num']) && isset($_POST['operator'])) {
        $first_num=$_POST['first_num'];
        $second_num=$_POST['second_num'];
        $operator=$_POST['operator'];   $result = '“”;
 }
?>
```

Processing POST data

```
        ……..
</form>
<?php
 if (isset($_POST['first_num']) && isset($_POST['second_num']) && isset($_POST['operator'])) {
        $first_num=$_POST['first_num'];
        $second_num=$_POST['second_num'];
        $operator=$_POST['operator'];   $result = '“”;
  if (is_numeric($first_num) && is_numeric($second_num)) {









    }

  }
?>
```

Processing POST data

```
        ……..
</form>
<?php
 if (isset($_POST['first_num']) && isset($_POST['second_num']) && isset($_POST['operator'])) {
        $first_num=$_POST['first_num'];
        $second_num=$_POST['second_num'];
        $operator=$_POST['operator'];   $result = '“”;
  if (is_numeric($first_num) && is_numeric($second_num)) {
   switch ($operator) {









         }  }

 }
?>
```

Processing POST data

```
        ……..
</form>
<?php
 if (isset($_POST['first_num']) && isset($_POST['second_num']) && isset($_POST['operator'])) {
        $first_num=$_POST['first_num'];
        $second_num=$_POST['second_num'];
        $operator=$_POST['operator'];   $result = '“”;
  if (is_numeric($first_num) && is_numeric($second_num)) {
   switch ($operator) {
      case "Add":
        $result = $first_num + $second_num;
         break;




} }
        echo "Result:<input type='text' value=$result />";
 }
?>
```

Processing POST data

```
        ……..
</form>
<?php
 if (isset($_POST['first_num']) && isset($_POST['second_num']) && isset($_POST['operator'])) {
        $first_num=$_POST['first_num'];
        $second_num=$_POST['second_num'];
        $operator=$_POST['operator'];   $result = '“”;
  if (is_numeric($first_num) && is_numeric($second_num)) {
   switch ($operator) {
      case "Add":
        $result = $first_num + $second_num;
         break;
      case "Subtract":
        $result = $first_num - $second_num;
         break;
         ……..……
         } }
       echo "Result:<input type='text' value=$result />";
 }
?>
```

Summary

- Numeric-indexed arrays: Indexed by integers
- Associative arrays: Indexed by strings
- Printing Arrays
  - print_r: Shows information about an array
  - var_dump: displays structured information
- Array functions
- Syntax of HTML forms
- GET method sends form input in the URL using name=value pairs separated by &
- POST method sends all contents of a form with basically hidden headers
- The $_REQUEST variable is a superglobal variable, which can hold the content of both $_GET and $_POST variable.
- When to use GET or POST? https://www.w3schools.in/php/get-post