05 PHP II

PHP II

- Controls
- Using Arrays
- String Manipulation
- Workshop

Controls

- Branching
  - if
  - if…else
  - if else if (if…elseif)
  - switch
- Looping
  - for
  - while
  - do…while
  - foreach
- break;
- continue;

Controls

Branching

- if

```
if(condition){
        statement1;
        statement2;
}
```

- if…else

```
if(condition){
        statement1;
        statement2;
}else{
        statement3;
        statement4;
}
```

- if else if (if…elseif)

```
if(condition){
        statement1;
        statement2;
}elseif(condition){
        statement3;
        statement4;
}else{
        statement5;
        statement6;
}
```

Controls

- Branching
  - Switch

```
switch(variable){
        case constant:
                statement1;
                statement2;
                break;
        case constant:
                statement3;
                statement4;
                break;
                …
                …
                …
        case constant:
                statement11;
                statement12;
                break;
        default:
                statement13;
}
```

```
$choice = 3;
switch($choice){
        case 1:
                $day = 'Sunday';
                break;
        case 2:
                $day = 'Monday';
                break;
        case 3:
                $day = 'Tuesday';
                break;
        case 4:
                $day = 'Wednesday';
                break;
        case 5:
                $day = 'Thursday';
                break;
        case 6:
                $day = 'Friday';
                break;
        case 7:
                $day = 'Saturday';
                break;
        default:
                echo 'Invalid Input';
}
```

Controls

- Looping
  - for

```
for(initialization; condition; increment/decrement){
        statement1;
        statement2;
}
statement3;
```

  - while

```
initialization;
while(condition){
        statement1;
        statement2;
        increment/decrement;
}
statement3;
```

  - do...while

```
do{
        statement1;
        statement2;
}while(condition);
statement3;
```

Controls

Looping

- foreach

```
foreach(array_expression as $value){
        statement1;
        statement2;
}
```

```
<?php
  $fruit=array("apple", "banana", "orange");
  foreach($fruit as $value){
    echo "Value: $value<br>";
  }
?>
```

```
Value: apple
Value: banana
Value: orange
```

```
<?php
  $fruit=array("apple", "banana", "orange");
  for($i=0; $i<count($fruit); $i++){
    echo "Value: $fruit[$i]<br>";
  }
?>
```

Controls

- break;

```
<?php
  $fruit=array("apple", "banana", "orange");
  foreach ($fruit as $value){
    if($value == "banana"){
      break;
    }
    echo "Value: $value<br>";
  }
  echo "Bye Bye!"
?>
```

```
Value: apple
Bye Bye!
```

- continue;

```
<?php
  for ($i=1; $i<6; $i++){
      if($i == 3)
          continue;
      echo "$i";
      echo '+';
      echo '- ';
  }
?>
```

```
1+- 2+- 4+- 5+-
```

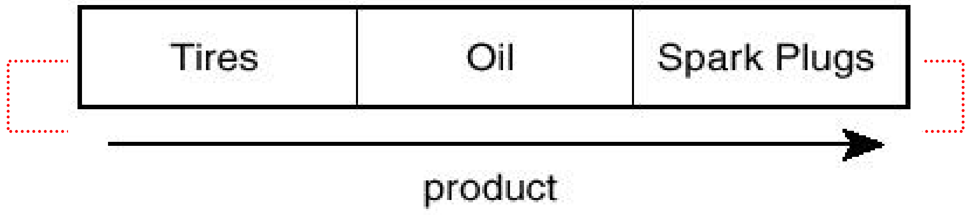
Using Arrays

- What is and Array?
- Numerically Indexed Arrays
- Associative Arrays
- Multidimensional Arrays
- Sorting Arrays
- Reordering Arrays
- Other Array Manipulations

Using Arrays

- What is and Array?
  - A variable that stores a set or sequence of values.
  - One array can have many elements.
  - Each element can hold a single value, such as text or numbers, or another array.
  - An array containing other arrays is known as a multidimensional array.
  - For example, a list of three products stored in an array format and one variable, called $product which stores the three values.

Figure: 8.1



Using Arrays

Numerically Indexed Arrays

- Initializing Numerically Indexed Arrays
  - To create the array called **product** contains "Tires", "Oil", and "Spark Plugs", use the following line of PHP code:
    - $product = array('Tires', 'Oil', 'Spark Plugs');
  - If you want an ascending sequence of numbers stored in an array, you can use the **range()** function
    - $numbers = range(1,10); ⇐===⇒ $numbers = array(1,2,3,4,5,6,7,8,9,10);
- Accessing Array Contents
  - To access the contents using the variable name and a key or index.
  - The key or index indicates which stored values we access.
  - The index is placed in square brackets after the name.
  - For example:
    - echo $product[0]; ===➔ Tires
    - echo $product[1]; ===➔ Oil
    - echo $product[2]; ===➔ Spark Plugs
    - $product[3] = 'Fuses'; // adding a new element to array "product"

Using Arrays

Numerically Indexed Arrays (cont.)

- Using Loops to Access the Array
  - The array is indexed by sequence of numbers, starting from 0 (zero).
  - Using *for* loop to access its content

```
for($i=0; $i<4; $i++)
    echo $product[$i]."<br>";
```

===➔

```
Tires
Oil
Spark Plugs
Fuses
```

  - We can also use the *foreach* loop, specially designed for use with arrays.

```
foreach ($product as $current)
    echo $current.' ';
```

===➔

```
Tires Oil Spark Plugs Fuses
```

Using Arrays

Associative Arrays

- Initializing an Associative Array
  - The following code creates an associative array with product names as keys and prices as values.
    - $product = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4);
- Accessing the Array Elements
  - We access the contents using the variable name and key, so we can access the information we have stored in the prices array as follows:
    - $product['Tires'], $product['Oil'], and $product['Spark Plugs ']
  - An array can be created and initialized with one element, and then added two more element as follows:

```
- $product = array( 'Tires'=>100 );             $product['Tires'] = 100;
- $product['Oil'] = 10;               <===>     $product['Oil'] = 10;
- $product['Spark Plugs'] = 4;                  $product['Spark Plugs'] = 4;
```

Using Arrays

Associative Arrays (cont.)

- Using Loops with Associative Arrays
  - The indices in this associative array are not numbers, we cannot use a simple counter in a for loop to work with the array.
  - We can use the **foreach** loop or the **each()** and **list()** constructs.
  - An example of using **foreach** loop

```
foreach($product as $key => $value)
    echo $key.'=>'.$value.'<br/>';
```

  - An example of using **each()**

```
while($element = each($product)){
    echo $element['key'];
    echo ' - ';
    echo $element['value'];
    echo '<br/>';
}
```

// ‘key’ and ‘value’ are keywords

Using Arrays

Associative Arrays (cont.)

- An example of using **list()**
  - list($name, $price) = each($product); //No $ sign in front of list
  - /* This line uses each() to take the current element from $product return it as an array, and make the next element current. It also uses list() to turn the 0 and 1 elements from the array returned by each() into two new variables called $name and $price */
  - // To display the whole list, we will code it as follows

```
while(list($name, $price) = each($product))
    echo "$name - $price<br/>";
```

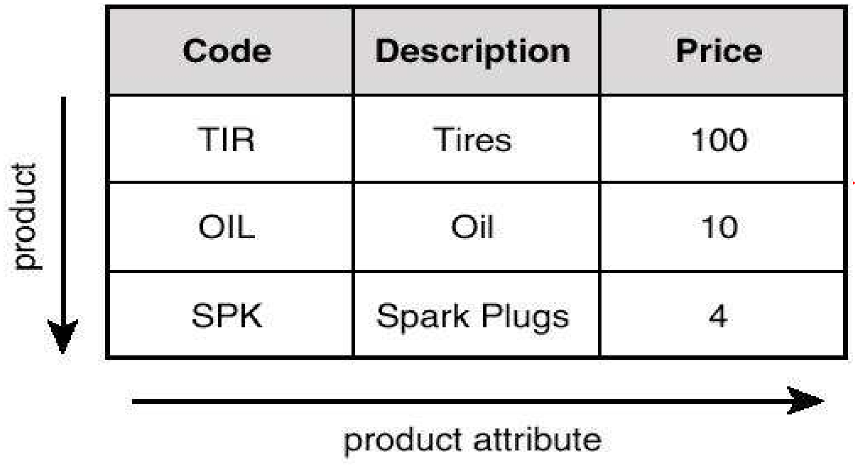
- **Note:** When using each(), the array keeps track of the current element. If we want to use the array twice in the same script, we need to set the current element back to the start of the array using the function **reset()**.

```
reset($product);
```

Using Arrays

Multidimensional Arrays

- Arrays do not have to be a simple list of keys and values –each location in the array can hold another array.
- This way, we can create a two-dimensional array.
- You can think of two dimensional array as a matrix, or grid, with width and height or rows and columns.



Code	Description	Price
TIR	Tires	100
OIL	Oil	10
SPK	Spark Plugs	4

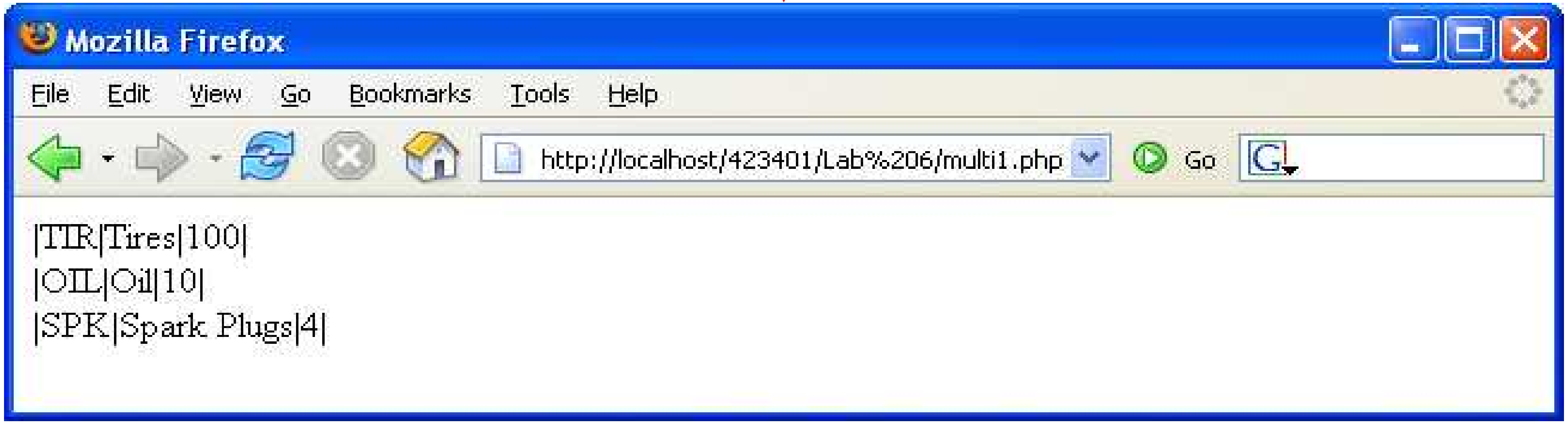
```
$product = array(array('TIR', 'Tires', 100 ),
                 array('OIL', 'Oil', 10 ),
                 array('SPK', 'Spark Plugs', 4));
```

Using Arrays

Multidimensional Arrays (cont.)

- Display all Data in a Multidimensional Array

```
$product = array(array('TIR', 'Tires', 100),
                 array('OIL', 'Oil', 10),
                 array('SPK', 'Spark Plugs', 4));
echo '|'.$product[0][0].'|'.$product[0][1].'|'.$product[0][2].'|<br/>';
echo '|'.$product[1][0].'|'.$product[1][1].'|'.$product[1][2].'|<br/>';
echo '|'.$product[2][0].'|'.$product[2][1].'|'.$product[2][2].'|<br/>';
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/423401/Lab%206/multi1.php
Go

```
|TIR|Tires|100|
|OIL|Oil|10|
|SPK|Spark Plugs|4|
```

Using Arrays

Multidimensional Arrays (cont.)

- Display all Data in a Multidimensional Array Using for Loop

```
$products = array(array('TIR', 'Tires', 100),
                  array('OIL', 'Oil', 10),
                  array('SPK', 'Spark Plugs', 4));
echo '|'.$product[0][0].'|'.$product[0][1].'|'.$product[0][2].'|<br/>';
echo '|'.$product[1][0].'|'.$product[1][1].'|'.$product[1][2].'|<br/>';
echo '|'.$product[2][0].'|'.$product[2][1].'|'.$product[2][2].'|<br/>';
```

```
$product = array(array('TIR', 'Tires', 100),
                 array('OIL', 'Oil', 10),
                 array('SPK', 'Spark Plugs', 4));
for($row=0; $row<3; $row++){
   for($column=0; $column<3; $column++){
      echo '|'.$product[$row][$column];
   }
   echo '|<br/>';
}
```

Using Arrays

Multidimentional Arrays (cont.)

- Using Associative Array in a Multidimensional Array

```
$product = array(array(Code => 'TIR',
                       Description => 'Tires',
                       Price => 100),
                 array(Code => 'OIL',
                       Description => 'Oil',
                       Price => 10),
                 array(Code => 'SPK',
                       Description => 'Spark Plugs',
                       Price => 4));

for ($row=0; $row<3; $row++){
   echo'|'.$product[$row]['Code'].'|'.$product[$row]['Description'].
       '|'.$product[$row]['Price'].'|<br/>';
}
//   We can use list() as follows:
//   for($row=0; $row<3; $row++){
//      while (list($key, $value) = each($product[$row])){
//         echo "|$value";
//      }
//      echo '|<br/>';
//   }
```

Using Arrays

Sorting Arrays

- Using **sort()** for One-dimensional Array
  - $product = array('Tires', 'Oil', 'Spark Plugs');
  - sort($product);
  - foreach($product as $item)
  -    echo $item.', ';   ===➔ Oil, Spark Plugs, Tires,

- Using **asort()** and **ksort()** to Sort Associative Arrays
  - The function **asort()** orders the array according to the <u>value</u> of each element, while **ksort()** will sort by the <u>key</u> rather than value.
    - $product1 = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4);
    - $product2 = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4);
    - asort($product1);
    - foreach($product1 as $key => $value)
    -    echo $key.' => '.$value.', ';   ===➔ Spark Plugs => 4, Oil => 10, Tires => 100
    - ksort($product2);
    - foreach($product2 as $key => $value)
    -    echo $key.' => '.$value.', ';   ===➔ Oil => 10, Spark Plugs => 4, Tires => 100,

Using Arrays

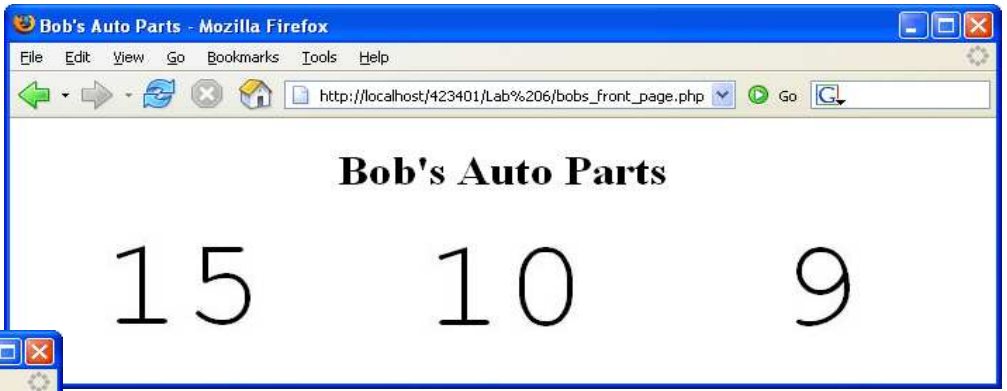
Sorting Arrays (cont.)

- Sorting in Reverse
  - The three different sorting functions: (**sort()**, **asort()**, and **ksort()**) all sort an array into ascending order.
  - Each of these functions has a matching reverse sort function to sort an array into descending order. The reverse versions are called **rsort()**, **arsort()**, and **krsort()**.
    - The **rsort()** function sorts a single dimensional numerically indexed array into descending order.
    - The **arsort()** function sorts a one-dimensional associative array into descending order using the value of each element.
    - The **krsort()** function sorts a one-dimensional associative array into descending order using the key of each element.

Using Arrays

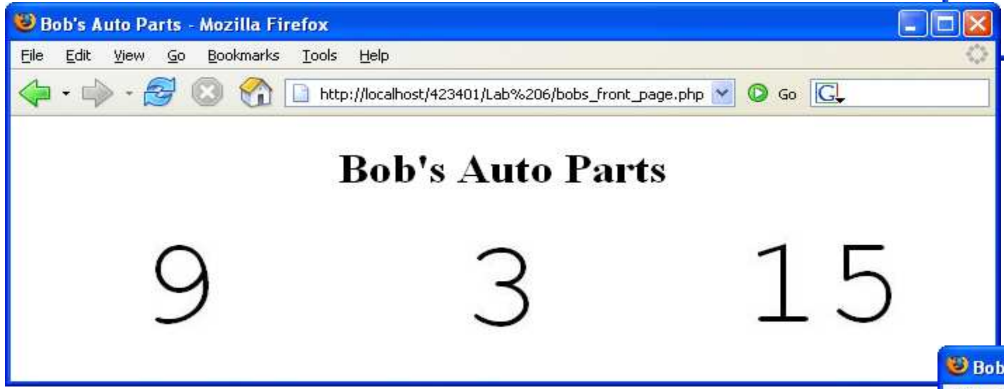
Reordering Arrays

- The function shuffle() randomly reorders the elements of your array.


Bob's Auto Parts - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/423401/Lab%206/bobs_front_page.php
Go

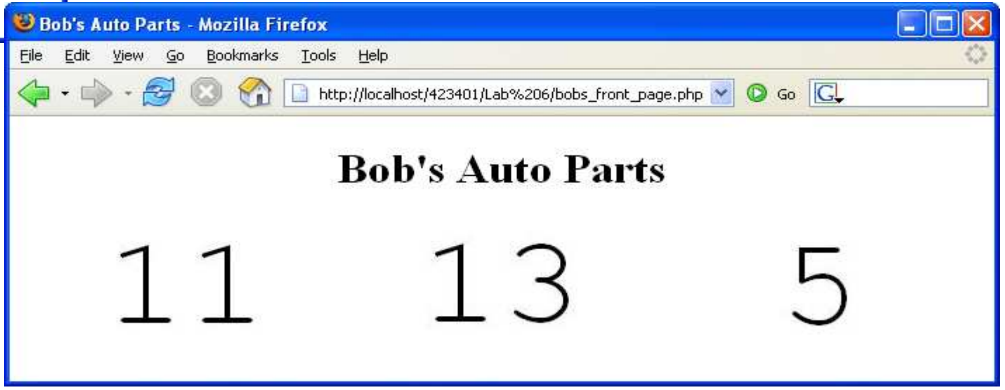
Bob's Auto Parts

15 10 9


Bob's Auto Parts - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/423401/Lab%206/bobs_front_page.php
Go

Bob's Auto Parts

9 3 15


Bob's Auto Parts - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/423401/Lab%206/bobs_front_page.php
Go

Bob's Auto Parts

11 13 5

Using Arrays

Reordering Arrays

- The function array_reverse() gives you a copy of your array with all the elements in reverse order.
  - The array_reverse() returns a modified copy of the array.
    - // The following code will create an empty array called “numbers:, and then use
    - // the for loop and array_push( ) function to add one element to the end of the
    - // array.

```
$numbers = array( );
for($i=10; $i>0; $i--)
    array_push($numbers, $i);
```

- // Another way
- //The following code will first create an array called numbers containing 1 to 10

```
$numbers = range(1,10);
$numbers = array_reverse($numbers);
```

Using Arrays

- Other Array Manipulations
  - Navigating Within an Array: each(), current(), reset(), end(), next(), pos() , and prev()
    - If we create a new array, the current pointer is initialized to point to the first element in the array.
    - Calling **current($array_name)** returns the current element. **pos()** Alias of current().
    - Calling **each($array_name)** returns the current element before advancing the pointer.
    - Calling **next($array_name)** advances the pointer and then returns the new current element.
    - The **prev($array_name)** moves the current pointer back one and then returns the new current element.
    - Calling **reset($array_name)** returns the pointer to the first element in the array.
    - Calling **end($array_name)** sends the pointer to the end of the array.

Using Arrays

Other Array Manipulations

- Example

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
echo $mode."<br>";
$mode = next($transport);     // $mode = 'bike';
echo $mode."<br>";
$mode = next($transport);     // $mode = 'car';
echo $mode."<br>";
$mode = prev($transport);     // $mode = 'bike';
echo $mode."<br>";
$mode = reset($transport);    // $mode = 'foot';
echo $mode."<br>";
$mode = end($transport);      // $mode = 'plane';
echo $mode."<br>";
?>
```

```
foot
bike
car
bike
foot
plane
```

Using Arrays

Other Array Manipulations (cont.)

- Counting Elements in an Array: count(), sizeof(), and array_count_values()
  - The function **count()** counts the number of elements in an array.
  - The function **sizeof()** has exactly the same purpose.
  - The **array_count_values($array)** counts how many times each *unique* value occurs in the array “array”.

```
$array = array(4, 5, 1, 2, 3, 1, 2, 1);
$ac = array_count_values($array);
```

```
creates an array called $ac that contains
key value
4      1
5      1
1      3
2      2
3      1
```

Using Arrays

Other Array Manipulations (cont.)

- Converting Arrays to Scalar Variables: extract()
  - The purpose of **extract()** is to take an array and create scalar variables with the names of the keys in the array. The values of these variables are set to the values in the array.

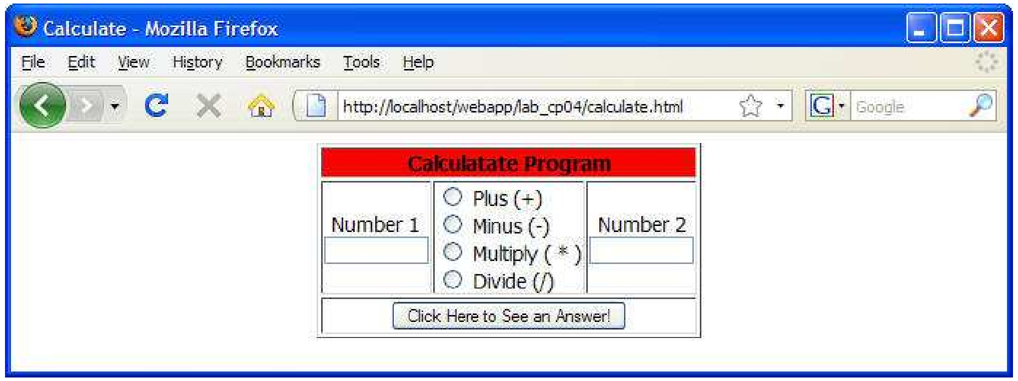
```
$a = array( 'key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3');
extract($a);
echo "$key1 $key2 $key3";
```

```
value1 value2 value3
```

Workshop

PHP II: Using Operators & Control

- Laboratory 5
  - 107(1-3)

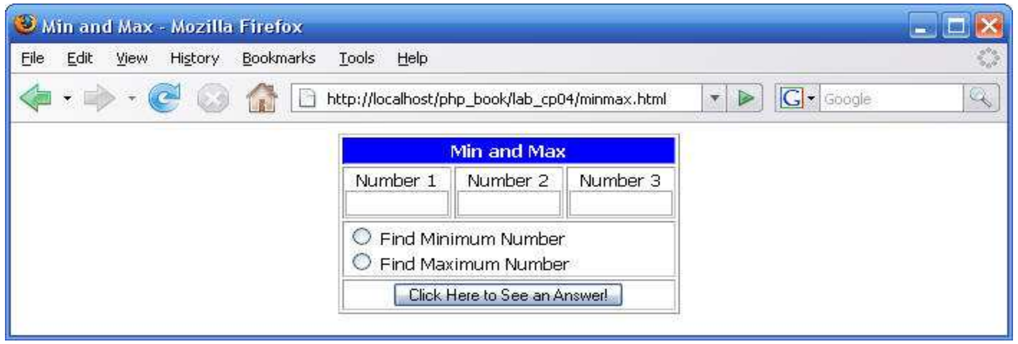


Calculate - Mozilla Firefox

http://localhost/webapp/lab_cp04/calculate.html

Calkulatate Program

Number 1	○ Plus (+) ○ Minus (-) ○ Multiply (*) ○ Divide (/)	Number 2



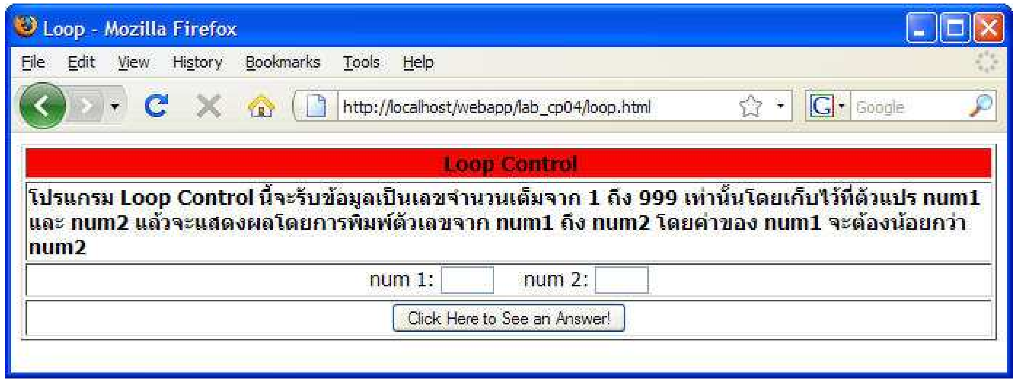
Min and Max - Mozilla Firefox

http://localhost/php_book/lab_cp04/minmax.html

Min and Max

Number 1	Number 2	Number 3

○ Find Minimum Number
○ Find Maximum Number



Loop - Mozilla Firefox

http://localhost/webapp/lab_cp04/loop.html

Loop Control

โปรแกรม Loop Control นี้จะรับข้อมูลเป็นเลขจำนวนเต็มจาก 1 ถึง 999 เท่านั้นโดยเก็บไว้ที่ตัวแปร num1 และ num2 แล้วจะแสดงผลโดยการพิมพ์ตัวเลขจาก num1 ถึง num2 โดยค่าของ num1 จะต้องน้อยกว่า num2

num 1: num 2: