



ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# Chapter 4.

# Working with Arrays

# Objectives

- To understand the benefits of using arrays in PHP
- To learn how to create and use sequential arrays and their functions
- To learn how to create and use nonsequential arrays and their functions

# Content

1. Benefits of arrays
2. Sequential arrays
3. Non-sequential arrays
4. Multidimensional lists

# Content



## 1. Benefits of arrays

- 2. Sequential arrays
- 3. Non-sequential arrays
- 4. Multidimensional lists

# 1.1. What is an Array?

- An array is a special type of variable.
  - can hold multiple data values
- A sequential array keeps track of these data items by using sequential numbers
  - (e.g., item 0, item 1, item 2, and so on)
- A nonsequential array or associative array keeps track of these data items by using character strings
  - (e.g., item meat, item poultry, item dairy, and so on)

## 1.2. Why Use Arrays?

- *Include a flexible number of list items.*
- *Examine each item more concisely.*
- *Using Loops to Repeat Statements*
- *Use special array operators and functions.*

# Content

1. Benefits of arrays



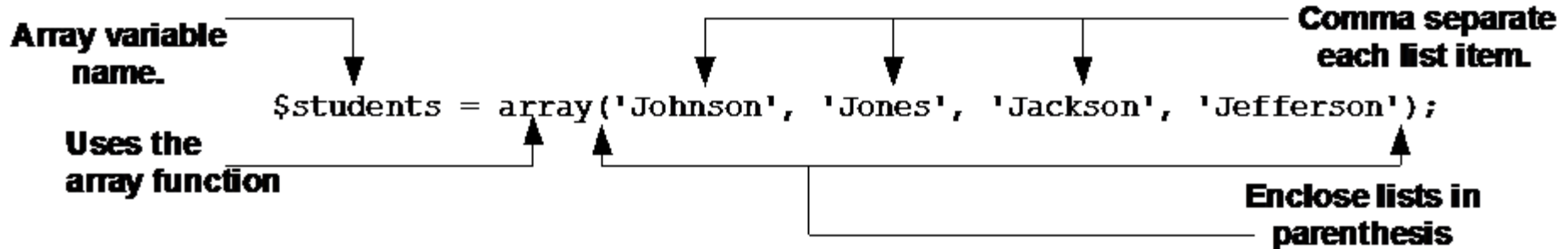
2. Sequential arrays

3. Non-sequential arrays

4. Multidimensional lists

## 2.1. Creating Sequential Arrays

- Use the `array()` function to create an array



- You could also create an array with numerical data
  - `$grades = array(66, 75, 85, 80);`



# Another way to create an array

- You can also create an array by making individual value assignments into the array variable name.
- For example, **`$students[] = 'Johnson';`**

**`$students[] = 'Jones';`**

**`$students[] = 'Jackson';`**

**`$students[] = 'Jefferson';`**

## 2.2. Referencing Sequential Array Items

- To reference individual array items, use an *array name* and *index pair*

`$sports[0] = 'baseball';`

**Array name**      **Index**

- Indices are referenced sequentially:
  - `$names = array('Denise', 'Christopher', 'Matthew', 'Bryant');`
  - `print (" $names[0], $names[1], $names[2],  
$names[3] ");`
- Outputs names sequentially

# Warning: Indices starts with 0

- You might think the arrays in the preceding code would be numbered with indices 1 through 4.
  - By default sequential arrays start with index 0,
  - so the indices above are numbered from 0 to 3.
  - Avoid referencing an item past the end of your array (for example, using `$names[20]` in an array that contains only four items).

# More on Indices ...

- Array indices can be whole numbers or a variable.

```
$i=3;
```

```
$classes = array('Math', 'History', 'Science', 'Pottery');
```

```
$oneclass = $classes[$i-1];
```

```
print "$classes[$i] $oneclass $classes[1] $classes[0]";
```

- This code outputs the following:

**“Pottery Science History Math”**

## 2.3. Changing arrays values

- You can change values in an array as follows:

```
$scores = array(75, 65, 85, 90);  
$scores[3] = 95;  
$average = ($scores[0] + $scores[1] +  
            $scores[2] + $scores[3]) / 4;  
print "average=$average";
```

- The output of the above PHP segment is “average=80”.

# Explicitly Setting Index Values

- You can explicitly sign values to indices

Assign the value of 65 to the item with index 2.

Assign the value of 85 to the item with index 3.

```
$scores = array(1=>75, 2=>65, 3=>85);
```

```
$scores[] = 100;
```

Add item with value 100 to the end of the array.

```
print "$scores[1] $scores[2] $scores[3]
```

- The above outputs “75 65 85 100”.

## 2.4. Using Loops with Sequential Arrays

- Looping statements can be used to iterate through arrays

```
$courses = array ('Perl', 'PHP', 'C', 'Java', 'Pascal', 'Cobol',  
    'Visual Basic');  
  
for ($i=0; $i < count($courses); $i++) {  
    print ("$courses[$i] ");  
  
}
```

- The above repeats 7 times with \$i equal to 0, 1, 2, 3, 4, 5, and 6.
- The above outputs: “Perl PHP C Java Pascal Cobol Visual Basic”.

# Using the foreach statement

- PHP supports the foreach statement as another way to iterate through arrays

**Array Name** —————

```
foreach ($courses as $item) {  
    Set of statements to repeat.  
}
```

**Item variable (\$item)**  
**is automatically set to**  
**next array item**  
**each iteration.**



# foreach statement - example

- Example of foreach command

```
$courses = array('Perl', 'PHP', 'C', 'Java', 'Pascal',  
    'Cobol', 'Visual Basic');  
foreach ($courses as $item){  
    print (" $item ");  
}
```

- The above outputs “Perl PHP C Java Pascal Cobol Visual Basic”.

# Sorting data

- For example the following code segment outputs “1 11 55 91 99 119 911”

```
$courses = array (91, 55, 11, 1, 99, 911, 119);  
sort($courses);  
foreach ($courses as $item) {  
    print "$item ";  
}
```

# Sorting data functions

Effect	Ascending	Descending	User-defined order
Sort array by values, then reassign indices starting with 0	<code>sort( )</code>	<code>rsort( )</code>	<code>usort( )</code>
Sort array by values	<code>asort( )</code>	<code>arsort( )</code>	<code>uasort( )</code>
Sort array by keys	<code>ksort( )</code>	<code>krsort( )</code>	<code>uksort( )</code>

- ◆ User-defined ordering requires that you provide a function that takes two values and returns a value that specifies the order of the two values in the sorted array.
  - ◆ return 1 if the first value is greater than the second
  - ◆ -1 if the first value is less than the second
  - ◆ 0 if the values are the same for the purposes of your custom sort order

# Example

# A Full Script Example

- Consider an example script that enables end-user to select multiple items from a checklist.
  - A survey about menu preferences
  - Will look at how to send multiple items and how to receive them (later)

# A Full Example ...

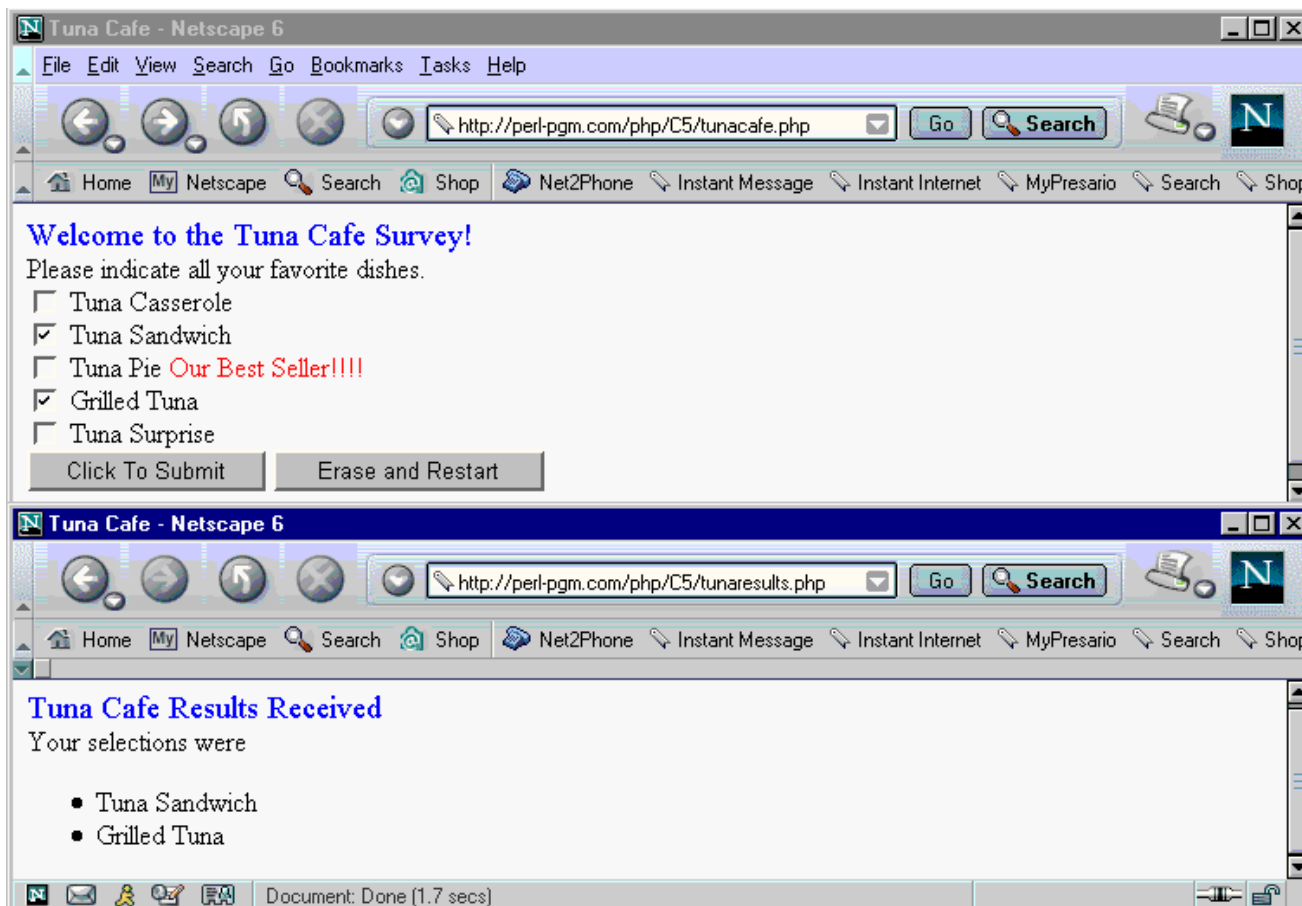
```
1. <html><head><title> Tuna Cafe </title></head>
2. <body> <font size=4 color="blue">
3. Welcome to the Tuna Cafe Survey! </font>
4. <form action="http://webwizard.aw.com/~phppgm/C5/tunaresults.php" method=post>
5. <?php
6. $menu = array('Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie', 'Grilled Tuna',
                  'Tuna Surprise');
7. $bestseller = 2;
8. print 'Please indicate all your favorite dishes.<br>';
9. for ($i=0; $i < count($menu); $i++) {
10.     print "<input type=\"checkbox\" name=\"prefer[]\" value=$i> $menu[$i]";
11.     if ($i == $bestseller) {
12.         print '<font color="red"> Our Best Seller!!!! </font>';
13.     }
14.     print '<br>';
15. }
16. ?>
17. <input type="submit" value="Click To Submit">
18. <input type="reset" value="Erase and Restart">
19. </form></body></html>
```

Create a list of menu items.

This array will be available to the receiving script when the form is submitted.

# The Output ...

The previous code can be executed at  
<http://webwizard.aw.com/~phppgm/C5/tunacafe.php>



# Using Arrays to Receive Multiple Form Element Selections

- Suppose you want to receive these multiple items, set as:

```
print "<input type=\"checkbox\" name=\"prefer[]\"  
value=$i> $menu[$i]";
```

- If the user selects the first and third check box items shown then \$prefer[] would be an array of two items:
  - \$prefer[0], would have a value of 0, and \$prefer[1] would be 2.



# Receiving Script

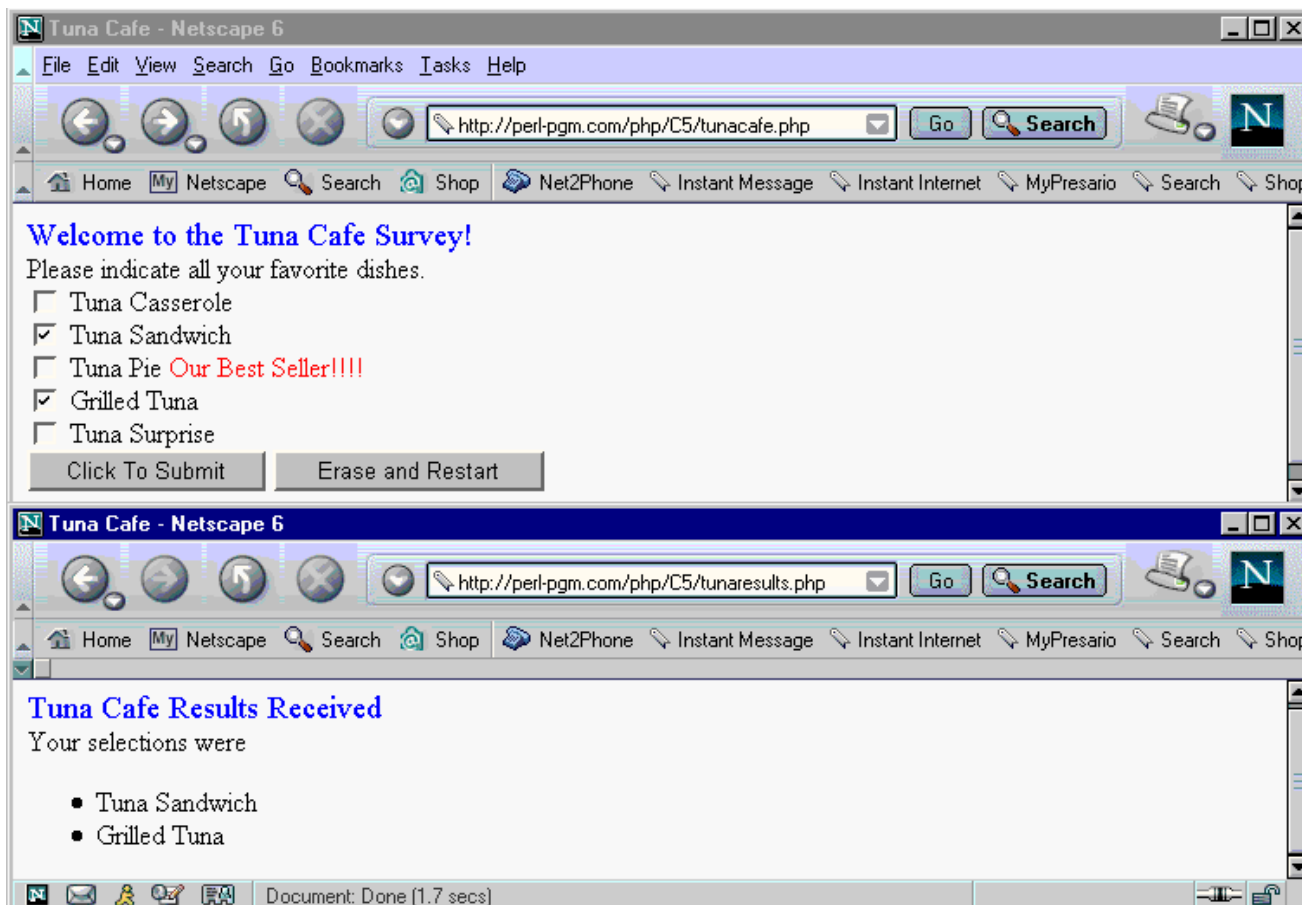
```
1. <html>
2. <head><title> Tuna Cafe </title></head>
3. <body>
4. <font size=4 color="blue"> Tuna Cafe Results Received </font>
5. <?php
6.     $menu = array('Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie',
7.                   'Grilled Tuna', 'Tuna Surprise');
8.     if (count($prefer) == 0 ) {
9.         print 'Oh no! Please pick something as your favorite! ';
10.    } else {
11.        print '<br>Your selections were <ul>';
12.        foreach ($prefer as $item) {
13.            print "<li>$menu[$item]</li>";
14.        }
15.        print '</ul>';
16.    }
17. </body></html>
```

# Receiving Code with REGISTER\_GLOBALS Off

```
1. <html>
2. <head><title> Tuna Cafe </title></head>
3. <body>
4. <font size=4 color="blue"> Tuna Cafe Results Received </font>
5. <?php
6.     $perfer = $_POST["prefer"];
7.     $menu = array('Tuna Casserole', 'Tuna Sandwich', 'Tuna Pie',
                    'Grilled Tuna', 'Tuna Surprise');
8.     if (count($prefer) == 0 ) {
9.         print 'Oh no! Please pick something as your favorite! ';
10.    } else {
11.        print '<br>Your selections were <ul>';
12.        foreach ($prefer as $item) {
13.            print "<li>$menu[$item]</li>";
14.        }
15.        print '</ul>';
16.    }
17. ?>
18. </body></html>
```

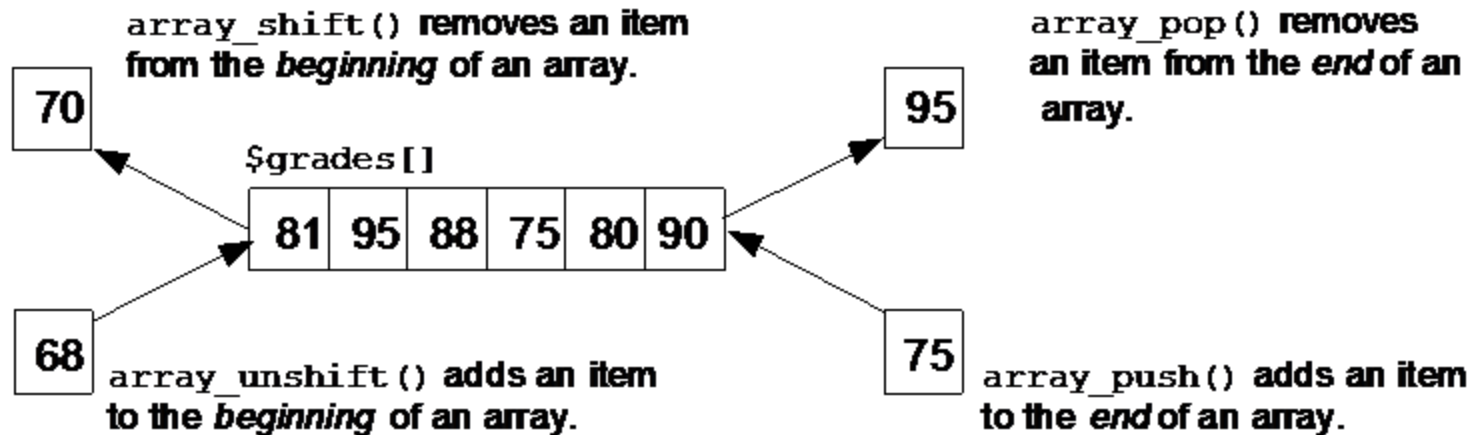
# The Output ...

The previous code can be executed at  
<http://webwizard.aw.com/~phppgm/C5/tunacafe.php>



## 2.5. More Arrays Operations

- Adding and Deleting Items



## a. The `array_shift()` functions

- `array_shift()` accepts an array as an argument, removes the first item, and then returns the removed item.
- For example,

```
$work_week = array('Monday', 'Wednesday', 'Friday');  
$day_off = array_shift($work_week);  
print "Day off = $day_off Work week = ";  
foreach ($work_week as $day) {  
    print "$day ";  
}
```

The above outputs:

“Day off = Monday Work week = Wednesday Friday”

## b. The `array_unshift()` functions

- `array_unshift()` used to *add* an item to the *beginning* of the array.
- It accepts as arguments an array variable and an item to add. For example,

```
$work_week = array('Monday', 'Wednesday', 'Friday');  
array_unshift($work_week, 'Sunday');  
print 'Work week is now = '  
foreach ($work_week as $day) {  
    print "$day "  
}
```

The above outputs:

```
"Work week is now = Sunday Monday Wednesday  
Friday".
```

## c. The `array_pop()` functions

- `array_pop()` accepts an array variable as an argument and returns an item it removed from the end of the array.

- For example,

```
$work_week = array('Monday', 'Wednesday', 'Friday');  
$day_off = array_pop($work_week);  
print "Day off = $day_off Work week = ";  
foreach ($work_week as $day) {  
    print "$day ";  
}
```

The above outputs:

```
"Day off = Friday Work week = Monday  
Wednesday"
```

## d. The `array_push()` functions

- `array_push()` accepts an array variable and an item as arguments and adds the item to the end of an array.
- For example, the following code:

```
$work_week = array('Monday', 'Wednesday', 'Friday');  
array_push($work_week, 'Saturday');  
print 'Work week is now = '  
foreach ($work_week as $day) {  
    print "$day "  
}
```

The above outputs:

```
"Work week is now = Monday Wednesday Friday Saturday"
```



## e. Additional Useful Array Functions

- Use `max()` and `min()` to find the largest and smallest number in an array.

```
$grades = array (99, 100, 55, 91, 65, 22, 16);  
$big=max($grades);  
$small=min($grades);  
print "max=$big small=$small";
```

The above would output:

```
"max=100 small=16".
```

## e. Additional Useful Array Functions (2)

- Use `array_sum()` to return a sum of all numerical values.

- For example,

```
$grades = array (25, 100, 50, 'N/A') ;  
$total=array_sum($grades) ;  
print "Total=$total" ;
```

- The above would output:

```
"Total=175"
```

# Mixing Variable Types

- PHP will try to convert character to numerical values when it can. For example,

```
<?php
$grades = array ('2 nights', '3days', 50, '1 more day');
$total=array_sum($grades);
print "total=$total";
?>
```

- Instead of generating an error message, this code outputs “total=56”.

# Content

1. Benefits of arrays
2. Sequential arrays
- 3. Non-sequential arrays
4. Multidimensional lists

# 3. Non-sequential arrays

- PHP also supports arrays with string-value indices called non-sequential/associative arrays.
  - String-value index is used to look up or provide a cross-reference to the data value
  - For example, the following code creates an associative array with three items

```
$instructor['Science'] = 'Smith';  
$instructor['Math'] = 'Jones';  
$instructor['English'] = 'Jackson';
```

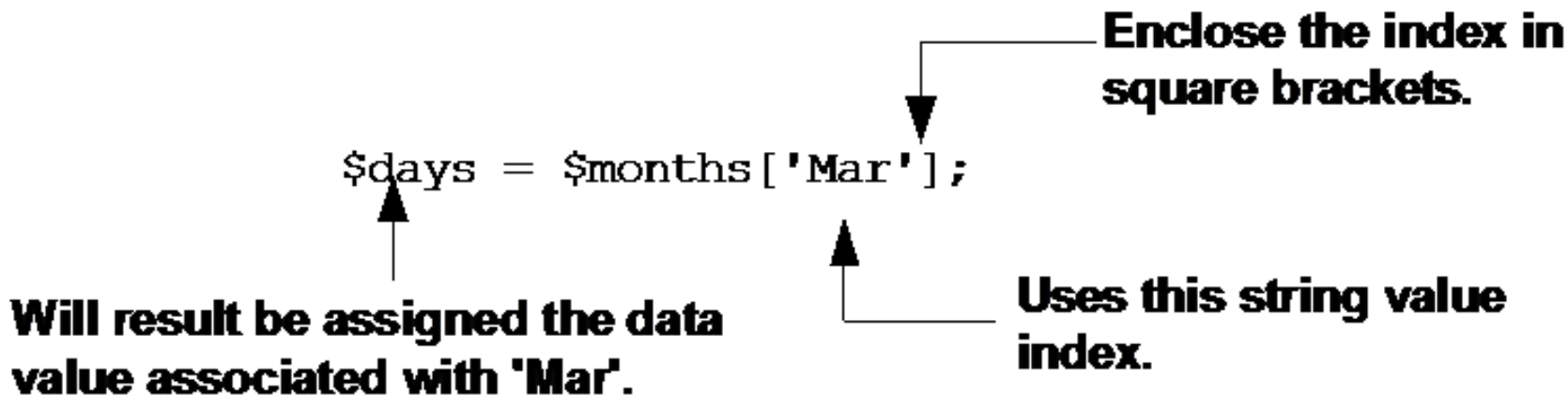
# 3.1. Creating Associative Arrays

- Use the array() function along with the => operator to create an associative array

```
$months = array( 'Jan'=>31, 'Feb'=>28, 'Mar'=>31, 'Apr'=>30,  
                'May'=>31, 'Jun'=>30, 'Jul'=>31, 'Aug'=>31,  
                'Sep'=>30, 'Oct'=>31, 'Nov'=>30, 'Dec'=>31 );
```

## 3.2. Accessing Associative Array Items

- Use a syntax similar to sequential arrays to access items



# WARNING You Cannot Fetch Indices by Using Data Values

- You might be tempted to use a data item to fetch an index from an associative array, as in the following example:
  - `$mon = $months[28];`
- This syntax is incorrect because associative arrays can fetch data values only by using indices (not the other way around)



# Consider the following example ...

- Consider an application that reports distance between Chicago and destination cities

```
<select name="destination" size=3>  
<option> Boston </option>  
<option> Dallas </option>  
<option> Las Vegas </option>  
<option> Miami </option>  
<option> Nashville </option>  
<option> Pittsburgh </option>  
<option> San Francisco </option>  
<option> Toronto </option>  
<option> Washington, DC </option>  
</select>
```

- When user selects destination city the application reports distance from Chicago

# Example script source

1. `<html>`
2. `<head><title> Distance and Time Calculations </title></head>`
3. `<body>`
4. `<?php`
5. `$cities = array ('Dallas' => 803, 'Toronto' => 435, 'Boston' => 848, 'Nashville' => 406, 'Las Vegas' => 1526, 'San Francisco' => 1835, 'Washington, DC' => 595, 'Miami' => 1189, 'Pittsburgh' => 409);`
6. `if (isset($cities[$destination])) {`
7.     `$distance = $cities[$destination];`
8.     `$time = round( ($distance / 60), 2);`
9.     `$walktime = round( ($distance / 5), 2);`
10.     `print "The distance between Chicago and <i> $destination</i> is $distance miles.";`
11.     `print "<br>Driving at 60 miles per hour it would take $time hours.";`
12.     `print "<br>Walking at 5 miles per hour it would take $walktime hours.";`
13. `} else {`
14.     `print "Sorry, do not have destination information for $destination.";`
15. `} ?>`
16. `</body></html>`

Associative array containing destination city and distance.

Check if the input destination city has a value in \$cities[].

Round results to 2 digits to the right of the decimal point.

# Example script source with REGISTER\_GLOBALS Off

1. <html>
2. <head><title> Distance and Time Calculations </title></head>
3. <body>
4. <?php
5. \$destination = \$\_POST["destination"];
6. \$cities = array ('Dallas' => 803, 'Toronto' => 435, 'Boston' => 848, 'Nashville' => 406, 'Las Vegas' => 1526, 'San Francisco' => 1835, 'Washington, DC' => 595, 'Miami' => 1189, 'Pittsburgh' => 409);
7. if (isset(\$cities[\$destination])) {
8.   \$distance = \$cities[\$destination];
9.   \$time = round( (\$distance / 60), 2);
10.   \$walktime = round( (\$distance / 5), 2);
11.   print "The distance between Chicago and <i>\$destination</i> is \$distance miles.";
12.   print "<br>Driving at 60 miles per hour it would take \$time hours.";
13.   print "<br>Walking at 5 miles per hour it would take \$walktime hours.";
14. } else {
15.   print "Sorry, do not have destination information for \$destination.";
16. } ?>
17. </body></html>

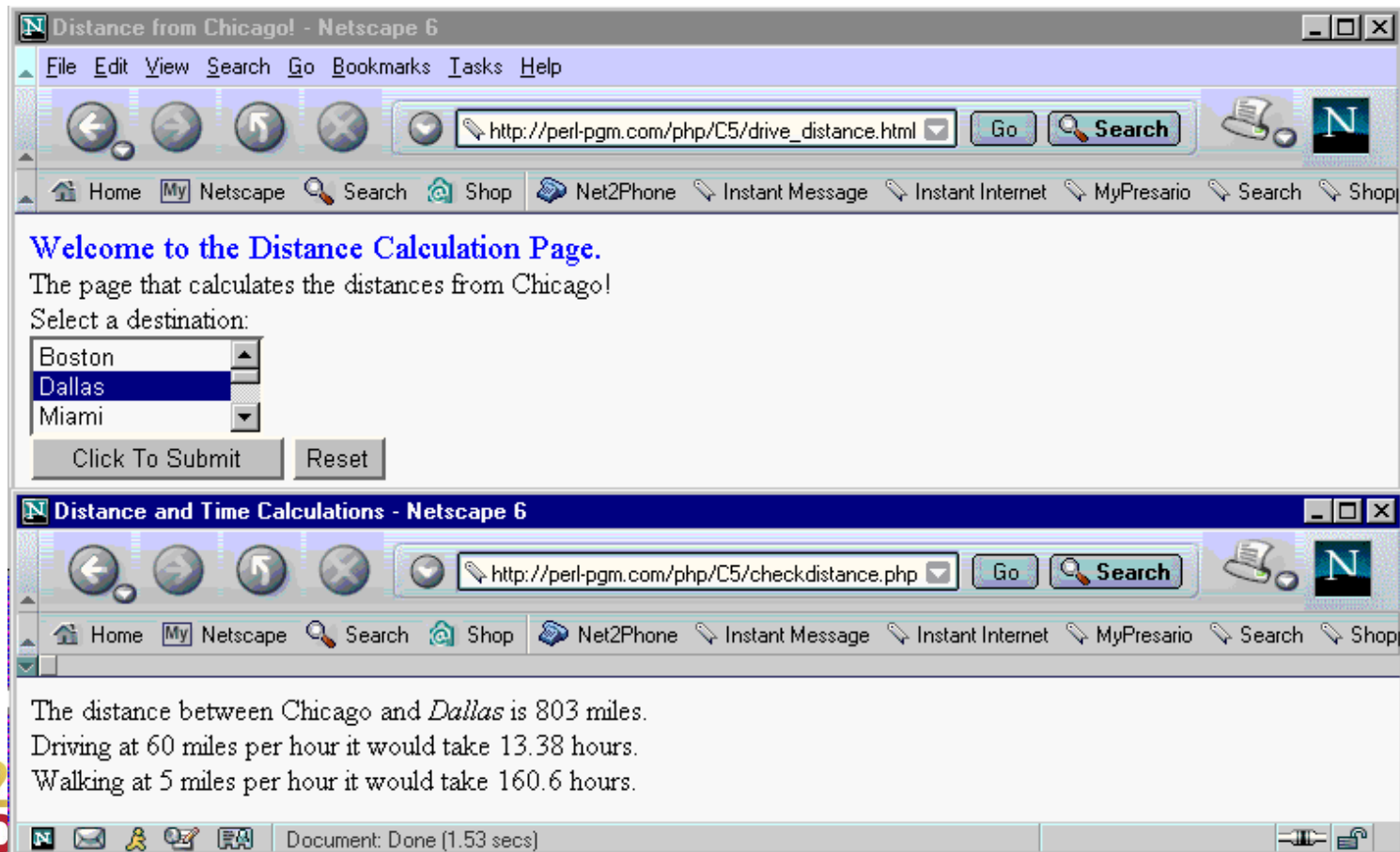
Associative array containing destination city and distance.

Check if the input destination city has a value in \$cities[].

Round results to 2 digits to the right of the decimal point.

# The Output ...

The previous code can be executed at  
[http://webwizard.aw.com/~phppgm/C5/drive\\_distance.html](http://webwizard.aw.com/~phppgm/C5/drive_distance.html)



## 3.3. Using foreach with associative arrays

- You can use foreach to access items from an associative array

**Array Name** →

```
foreach ($courses as $index => $item) {  
    Set of statements to repeat.  
}
```

**Index variable (\$index)**  
is automatically set to next  
array index each iteration.

**Item variable (\$item)**  
is automatically set to next  
array value each iteration.

## 3.3. Using foreach with associative arrays (2)

- Consider the following:

```
$inventory = array('Nuts'=>33, 'Bolts'=>55, 'Screws'=>12);  
foreach ($inventory as $index => $item) {  
    print "Index is $index, value is $item<br> ";  
}
```

- The above outputs:

```
Index is Nuts, value is 33  
Index is Bolts, value is 55  
Index is Screws, value is 12
```

## 3.4. Changing adding/deleting items

- You can change an item by giving it a new value:

```
$inventory = array('Nuts'=> 33, 'Bolts'=> 55,  
    'Screws'=> 12);  
$inventory['Nuts'] = 100;
```

- You can add an item as follows:

```
$inventory = array('Nuts'=>33, 'Bolts'=>55, 'Screws'=>12);  
$inventory['Nails'] = 23;
```

- You can delete an item as follows:

```
$inventory = array('Nuts'=> 33, 'Bolts'=>55, 'Screws'=> 12);  
unset($inventory['Nuts']);
```

## 3.5. Verifying an items existance

- You can use the isset() function to verify if an item exists.

```
$inventory = array('Nuts'=> 33,'Bolts'=>55,'Screws'=> 12);  
if (isset($inventory['Nuts'])) {  
    print ('Nuts are in the list.');} else {  
    print ('No Nuts in this list.');}
```



# Warning indices are case sensitive

- Examine the following lines:

```
$inventory = array( 'Nuts'=> 33, 'Bolts'=>55, 'Screws'=>12) ;  
$inventory['nuts'] = 32;
```

- Results in items 'Nuts', 'Bolts', 'Screws', and 'nuts'

# A Full Application

- Consider an application using the following radio buttons:

```
<input type="radio" name="Action" value="Add" > Add
```

```
<input type="radio" name="Action" value="Unknown" > Unknown
```

```
<br>Enter Index: <input type="text" name="index" size=10>
```

```
Enter Value: <input type="text" name="value" size=10>
```

- It “simulates” adding an inventory item

**That is, it adds it to associative array but does not save to a file or database.**

# PHP Source ...

```
1.<html><head><title>Inventory Add </title>
2.</head><body>
3.<?php
4.$invent = array('Nuts'=>44, 'Nails'=>34, 'Bolts'=>31);
5.if ($Action == 'Add'){
6.    $item=$invent["$index"];
7.    if (isset($invent["$index"])) {
8.        print "Sorry, already exists $index <br>";
9.    } else {
10.        $invent["$index"] = $Value;
11.        print "Adding index=$index value=$Value <br>";
12.        print '-----<br>';
13.        foreach ($invent as $index => $item) {
14.            print "Index is $index, value is $item.<br> ";
15.        }
16.    }
17. } else { print "Sorry, no such action=$Action<br>"; }
18. ?></body></html>
```

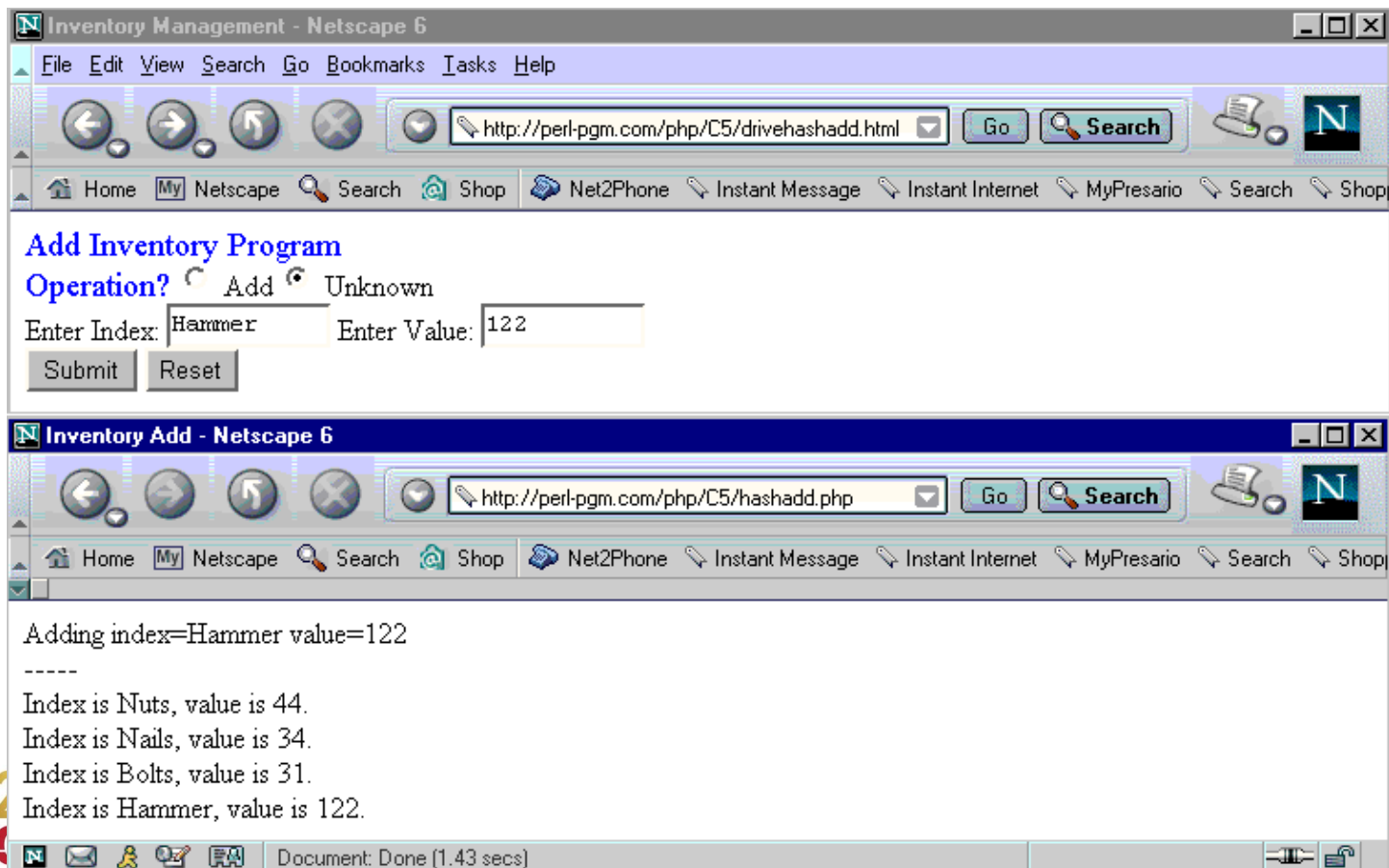
# PHP Source with REGISTER\_GLOBALS Off...

```
1.<html><head><title>Inventory Add </title>
2.</head><body>
3.<?php    $index = $_POST["index"]; $Value = $_POST["Value"];
4.$invent = array('Nuts'=>44, 'Nails'=>34, 'Bolts'=>31);
5.if ($Action == 'Add'){
6.    $item=$invent["$index"];
7.    if (isset($invent["$index"])) {
8.        print "Sorry, already exists $index <br>";
9.    } else {
10.        $invent["$index"] = $Value;
11.        print "Adding index=$index value=$Value <br>";
12.        print '-----<br>';
13.        foreach ($invent as $index => $item) {
14.            print "Index is $index, value is $item.<br> ";
15.        }
16.    }
17. } else { print "Sorry, no such action=$Action<br>"; }
18. ?></body></html>
```

# Would output the following:

The previous code can be executed at

<http://webwizard.aw.com/~phppgm/C5/drivehashadd.html>



## 3.6. Sorting Associative Arrays

- You can sort associative arrays by values or indices.
- Use `asort()` to sort by values:

```
$dest = array('Dallas' => 803, 'Toronto' => 435,  
              'Boston' => 848, 'Nashville' => 406,  
              'Las Vegas' => 1526);  
  
asort($dest);  
  
foreach ($dest as $index => $value) {  
    print " $index = $value ";  
}
```

- The above would output:

```
Nashville = 406  Toronto = 435  Dallas = 803  Boston = 848  
Las Vegas = 1526
```

## 3.6. Sorting Associative Arrays (2)

- Use ksort() to sort by indices:

```
$dest = array ('Dallas' => 803, 'Toronto' => 435,  
              'Boston' => 848, 'Nashville' => 406,  
              'Las Vegas' => 1526);  
  
ksort($dest);  
  
foreach ($dest as $index => $value) {  
    print " $index = $value ";  
}
```

- The above would output:

```
Boston = 848   Dallas = 803   Las Vegas = 1526   Nashville =  
406   Toronto = 435
```

# Content

1. Benefits of arrays
2. Sequential arrays
3. Non-sequential arrays
- 4. Multidimensional lists



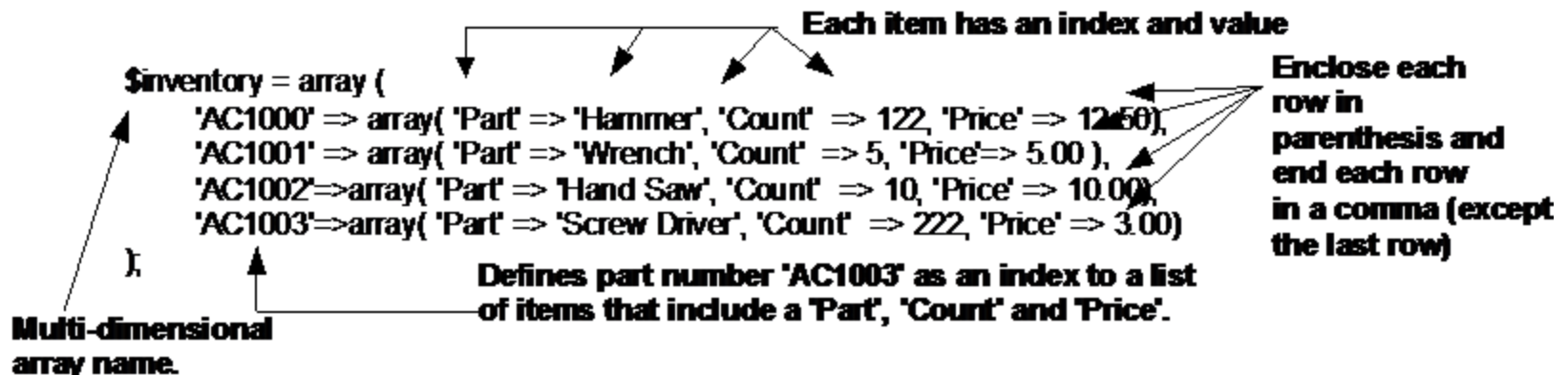
## 4. Multiple dimensional lists

- Some data is best represented using a list of list or a multi-dimensional list.
- For example:

Part Number	Part Name	Count	Price
AC1000	Hammer	122	12.50
AC1001	Wrench	5	5.00
AC1002	Handsaw	10	10.00
AC1003	Screwdriver	222	3.00

# 4.1. Creating Multidimensional Lists

- You can create multidimensional arrays with the `array()` function



`$inventory['AC1000']['Part']` has the value Hammer, `$inventory['AC1001']['Count']` has the value 5, and `$inventory['AC1002']['Price']` has the value 10.00.

# A Full Application

- Application that receives a part number and then returns information about the part
  - Uses the following HTML form:

```
<input type="radio" name="id" value="AC1000"> AC1000
```

```
<input type="radio" name="id" value="AC1001"> AC1001
```

```
<input type="radio" name="id" value="AC1002"> AC1002
```

```
<input type="radio" name="id" value="AC1003"> AC1003
```

# PHP Script Source

```
1.<html><head><title>Inventory Information</title>
2.</head><body>
3.<?php
4. $inventory = array (
    'AC1000'=>array('Part'=>'Hammer','Count'=>122, 'Price'=> 12.50 ),
    'AC1001' => array('Part' =>'Wrench','Count' =>5, 'Price'=>5.00 ),
    'AC1002'=>array('Part' =>'Handsaw','Count' =>10, 'Price'=>10.00 ),
    'AC1003'=>array('Part' =>'Screwdrivers','Count'=>222, 'Price'=>3.00)
    );
5.if (isset($inventory[$id])){
6.    print '<font size=4 color="blue"> ';
7.    print "Inventory Information for Part $id </font>";
8.    print '<table border=1> <th> ID <th> Part <th> Count <th> Price ';
9.    print "<tr> <td> $id </td>";
10.        print "<td> {$inventory[$id]['Part']} </td>";
11.        print "<td> {$inventory[$id]['Count']} </td>";
12.        print "<td> \${$inventory[$id]['Price']} </td></tr>";
13.    } else {
14.        print "Illegal part ID = $id ";
15.    }
16.??> </body></html>
```

# PHP Script Source With REGISTER\_GLOBALS Off

```
1.<html><head><title>Inventory Information</title>
2.</head><body>
3.<?php  $id = $_POST["id"];
4. $inventory = array (
    'AC1000'=>array('Part'=>'Hammer','Count'=>122, 'Price'=> 12.50 ),
    'AC1001' => array('Part' =>'Wrench','Count' =>5, 'Price'=>5.00 ),
    'AC1002'=>array('Part' =>'Handsaw','Count' =>10, 'Price'=>10.00 ),
    'AC1003'=>array('Part' =>'Screwdrivers','Count'=>222, 'Price'=>3.00)
    );
5.if (isset($inventory[$id])){
6.  print '<font size=4 color="blue"> ';
7.  print "Inventory Information for Part $id </font>";
8.  print '<table border=1> <th> ID <th> Part <th> Count <th> Price ';
9.  print "<tr> <td> $id </td>";
10.  print "<td> {$inventory[$id]['Part']} </td>";
11.  print "<td> {$inventory[$id]['Count']} </td>";
12.  print "<td> \${$inventory[$id]['Price']} </td></tr>";
13. } else {
14.  print "Illegal part ID = $id ";
15. }
16.<?> </body></html>
```

# Would output the following ...

The previous code can be executed at

[http://webwizard.aw.com/~phppgm/C5/drive\\_inventory.php](http://webwizard.aw.com/~phppgm/C5/drive_inventory.php)

**Hardware Harry's Inventory Information**

Select part number for more information

☐ AC1000 ☐ AC1001 ☒ AC1002 ☐ AC1003

---

**Inventory Information for Part AC1002**

ID	Part	Count	Price
AC1002	Handsaw	10	\$10