

Chapter 4

Arrays

Arrays

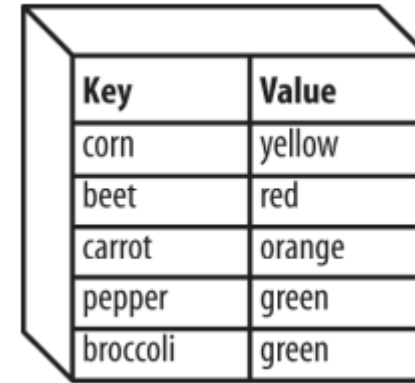
- Arrays are **collections of related values**, such as the data submitted from a form, the names of students in a class, or the populations of a list of cities.
- An array is made up of **elements**.
- Any string** or **number** value can be an array element **key**, such as corn, 4, -36, or Salt Baked Squid.

Creating an Associative Array

Associative Arrays: Each element has a **key** and a **value**.

- **For example**, an array holding information about the colors of vegetables has vegetable names for keys and colors for values
- Arrays and other non-scalar values **can't be keys**, but they can be element values.
- **Scalar values** are simple values such as 6, 98.76, "Saeed". Arrays or objects are non-scalar values

```
$vegetables['corn'] = 'yellow';  
$vegetables['beet'] = 'red';  
$vegetables['carrot'] = 'orange';
```



Key	Value
corn	yellow
beet	red
carrot	orange
pepper	green
broccoli	green

```
$vegetables = array('corn' => 'yellow',  
                   'beet' => 'red',  
                   'carrot' => 'orange');  
  
$dinner = array(0 => 'Sweet Corn and Asparagus',  
               1 => 'Lemon Chicken',  
               2 => 'Braised Bamboo Fungus');  
  
$computers = array('trs-80' => 'Radio Shack',  
                  2600 => 'Atari',  
                  'Adam' => 'Coleco');
```

Shortcut to Create Arrays

```
$vegetables = ['corn' => 'yellow', 'beet' => 'red', 'carrot' => 'orange'];  
  
$dinner = [0 => 'Sweet Corn and Asparagus',  
           1 => 'Lemon Chicken',  
           2 => 'Braised Bamboo Fungus'];  
  
$computers = ['trs-80' => 'Radio Shack', 2600 => 'Atari', 'Adam' => 'Coleco'];
```

Creating an array element by element

```
// An array called $vegetables with string keys
$vegetables['corn'] = 'yellow';
$vegetables['beet'] = 'red';
$vegetables['carrot'] = 'orange';

// An array called $dinner with numeric keys
$dinner[0] = 'Sweet Corn and Asparagus';
$dinner[1] = 'Lemon Chicken';
$dinner[2] = 'Braised Bamboo Fungus';

// An array called $computers with numeric and string keys
$computers['trs-80'] = 'Radio Shack';
$computers[2600] = 'Atari';
$computers['Adam'] = 'Coleco';
```

Creating a Numeric Array

- ⋮ If you create an array with [] or array() by specifying only a list of values instead of key/value pairs, the PHP engine automatically assigns a numeric key to each value.
- ⋮ The keys start at 0 and increase by one for each element.
- ⋮ PHP automatically uses **incrementing numbers for array keys** when you create an array or add elements to an array with the **empty brackets**

```
$dinner = array('Sweet Corn and Asparagus',  
               'Lemon Chicken',  
               'Braised Bamboo Fungus');  
print "I want $dinner[0] and $dinner[1].";
```

```
// Create $lunch array with two elements  
// This sets $lunch[0]  
$lunch[] = 'Dried Mushrooms in Brown Sauce';  
// This sets $lunch[1]  
$lunch[] = 'Pineapple and Yu Fungus';  
  
// Create $dinner with three elements  
$dinner = array('Sweet Corn and Asparagus', 'Lemon Chicken',  
               'Braised Bamboo Fungus');  
// Add an element to the end of $dinner  
// This sets $dinner[3]  
$dinner[] = 'Flank Skin with Spiced Flavor';
```

Finding the Size of an Array

- ⋮ The **count()** function tells you the number of elements in an array
- ⋮ An **empty array** (an array with no elements in it), **count()** returns 0.
- ⋮ An **empty array** also evaluates to **false** in an **if()** test expression.
 - For example: `$x = [];` **if(\$x)** `print "$x is an empty array"`

```
$dinner = array('Sweet Corn and Asparagus',  
               'Lemon Chicken',  
               'Braised Bamboo Fungus');  
  
$dishes = count($dinner);  
  
print "There are $dishes things for dinner.";
```

Example 4-7 prints:
There are 3 things for dinner.

Looping Through Arrays - foreach

⋮ The **foreach()** construct lets you run a code block once for each element in an array.

```
$meal = array('breakfast' => 'Walnut Bun',  
             'lunch' => 'Cashew Nuts and White Mushrooms',  
             'snack' => 'Dried Mulberries',  
             'dinner' => 'Eggplant with Chili Sauce');  
print "<table>\n";  
foreach ($meal as $key => $value) {  
    print "<tr><td>$key</td><td>$value</td></tr>\n";  
}  
print '</table>';
```

Example 4-8 prints:

```
<table>  
<tr><td>breakfast</td><td>Walnut Bun</td></tr>  
<tr><td>lunch</td><td>Cashew Nuts and White Mushrooms</td></tr>  
<tr><td>snack</td><td>Dried Mulberries</td></tr>  
<tr><td>dinner</td><td>Eggplant with Chili Sauce</td></tr>  
</table>
```


Looping Through Arrays - foreach

- Inside the foreach(), changing the values of \$key and \$value **doesn't affect the elements** in the actual array.
- If you want to change the array element values, use the **\$key variable as an index** into the array.

```
foreach ($meals as $dish => $price) {  
    // $price = $price * 2 does NOT work  
    $meals[$dish] = $meals[$dish] * 2;  
}  
  
// Iterate over the array again and print the changed values  
foreach ($meals as $dish => $price) {  
    printf("The new price of %s is \$.2f.\n", $dish, $price);  
}
```

Example 4-10 prints:

```
The new price of Walnut Bun is $2.00.  
The new price of Cashew Nuts and White Mushrooms is $9.90.  
The new price of Dried Mulberries is $6.00.  
The new price of Eggplant with Chili Sauce is $13.00.
```

Looping Through Arrays - foreach

Example 4-11. Using foreach() with numeric arrays

```
$dinner = array('Sweet Corn and Asparagus',  
               'Lemon Chicken',  
               'Braised Bamboo Fungus');  
foreach ($dinner as $dish) {  
    print "You can eat: $dish\n";  
}
```

Example 4-11 prints:

```
You can eat: Sweet Corn and Asparagus  
You can eat: Lemon Chicken  
You can eat: Braised Bamboo Fungus
```

Looping Through Arrays - foreach

- ⋮ If elements of a numeric array were added in a different order than how their keys would usually be ordered, this could produce **unexpected results**.

```
$letters[0] = 'A';  
$letters[1] = 'B';  
$letters[3] = 'D';  
$letters[2] = 'C';  
  
foreach ($letters as $letter) {  
    print $letter;  
}
```

Example 4-14 prints:

ABDC

```
for ($i = 0, $num_letters = count($letters); $i < $num_letters; $i++) {  
    print $letters[$i];  
}
```

This prints:

ABCD

Locate a particular **key** in an array

⋮ To check for an element with a certain **key**, use **array_key_exists()**

```
$meals = array('Walnut Bun' => 1,
               'Cashew Nuts and White Mushrooms' => 4.95,
               'Dried Mulberries' => 3.00,
               'Eggplant with Chili Sauce' => 6.50,
               'Shrimp Puffs' => 0); // Shrimp Puffs are free!
$books = array("The Eater's Guide to Chinese Characters",
               'How to Cook and Eat in Chinese');

// This is true
if (array_key_exists('Shrimp Puffs',$meals)) {
    print "Yes, we have Shrimp Puffs";
}
// This is false

if (array_key_exists('Steak Sandwich',$meals)) {
    print "We have a Steak Sandwich";
}
// This is true
if (array_key_exists(1, $books)) {
    print "Element 1 is How to Cook and Eat in Chinese";
}
```

Locate a particular **value** in an array

⋮ To check for an element with a particular **value**, use `in_array()`,

```
$meals = array('Walnut Bun' => 1,
               'Cashew Nuts and White Mushrooms' => 4.95,
               'Dried Mulberries' => 3.00,
               'Eggplant with Chili Sauce' => 6.50,
               'Shrimp Puffs' => 0);
$books = array("The Eater's Guide to Chinese Characters",
               'How to Cook and Eat in Chinese');

// This is true: key Dried Mulberries has value 3.00
if (in_array(3, $meals)) {
    print 'There is a $3 item.';
}
// This is true
if (in_array('How to Cook and Eat in Chinese', $books)) {
    print "We have How to Cook and Eat in Chinese";
}
// This is false: in_array() is case-sensitive
if (in_array("the eater's guide to chinese characters", $books)) {
    print "We have the Eater's Guide to Chinese Characters.";
}
```

Locate a particular **value** in an array

· The `array_search()` function, if it finds an element, it **returns the element key** instead of true.

```
$meals = array('Walnut Bun' => 1,  
              'Cashew Nuts and White Mushrooms' => 4.95,  
              'Dried Mulberries' => 3.00,  
              'Eggplant with Chili Sauce' => 6.50,  
              'Shrimp Puffs' => 0);  
  
$dish = array_search(6.50, $meals);  
if ($dish) {  
    print "$dish costs \$6.50";  
}
```

Example 4-17 prints:

Eggplant with Chili Sauce costs \$6.50

Modifying Arrays

- You can operate on individual array elements just like regular scalar variables, using arithmetic, logical, and other operators.

```
$dishes['Beef Chow Foon'] = 12;
$dishes['Beef Chow Foon']++;
$dishes['Roast Duck'] = 3;

$dishes['total'] = $dishes['Beef Chow Foon'] + $dishes['Roast Duck'];

if ($dishes['total'] > 15) {
    print "You ate a lot: ";
}

print 'You ate ' . $dishes['Beef Chow Foon'] . ' dishes of Beef Chow Foon.';
```

Example 4-18 prints:

You ate a lot: You ate 13 dishes of Beef Chow Foon.

Using element arrays in a string

- Inside strings, don't put quotes around the element key

```
$meals['breakfast'] = 'Walnut Bun';  
$meals['lunch'] = 'Eggplant with Chili Sauce';  
$amounts = array(3, 6);  
  
print "For breakfast, I'd like $meals[breakfast] and for lunch,\n";  
print "I'd like $meals[lunch]. I want $amounts[0] at breakfast and\n";  
print "$amounts[1] at lunch.";
```

- If you have an array key that has **whitespace** or **other punctuation** in it, interpolate it **with curly braces**

```
$meals['Walnut Bun'] = '$3.95';  
$hosts['www.example.com'] = 'website';  
  
print "A Walnut Bun costs {$meals['Walnut Bun']}. \n";  
print "www.example.com is a {$hosts['www.example.com']}. ";
```

Example 4-20 prints:

```
A Walnut Bun costs $3.95.  
www.example.com is a website.
```


Remove elements by `unset()`

⋮ To remove an element from an array, use `unset()`

```
unset($dishes['Roast Duck']);
```

⋮ Removing an element with `unset()` is **different** than just setting the element value to **0** or the **empty** string.

Print all of the values in an array

- Use the **implode()** function to print all of the values in an array at once

```
$dimsum = array('Chicken Bun','Stuffed Duck Web','Turnip Cake');  
$menu = implode(', ', $dimsum);  
print $menu;
```

Example 4-21 prints:

Chicken Bun, Stuffed Duck Web, Turnip Cake

To implode an array with no delimiter, use the empty string as the first argument to `implode()`:

```
$letters = array('A','B','C','D');  
print implode('', $letters);
```

This prints:

ABCD

Creating Multidimensional Arrays

- Use the `array()` construct or the `[]` to create **arrays that have more arrays as element values**

```
$meals = array('breakfast' => ['Walnut Bun', 'Coffee'],
               'lunch'      => ['Cashew Nuts', 'White Mushrooms'],
               'snack'      => ['Dried Mulberries', 'Salted Sesame Crab']);

$lunches = [ ['Chicken', 'Eggplant', 'Rice'],
              ['Beef', 'Scallions', 'Noodles'],
              ['Eggplant', 'Tofu'] ];
```

[illegible]

Accessing Multidimensional Arrays

```
print $meals['lunch'][1];           // White Mushrooms
print $meals['snack'][0];           // Dried Mulberries
print $lunches[0][0];              // Chicken
print $lunches[2][1];              // Tofu
print $flavors['Japanese']['salty']; // soy sauce
print $flavors['Chinese']['hot'];   // mustard
```

Iterating through a multidimensional array

```
$flavors = array('Japanese' => array('hot' => 'wasabi',  
                                     'salty' => 'soy sauce'),  
                'Chinese'   => array('hot' => 'mustard',  
                                     'pepper-salty' => 'prickly ash'));  
  
// $culture is the key and $culture flavors is the value (an array)  
foreach ($flavors as $culture => $culture_flavors) {  
    // $flavor is the key and $example is the value  
    foreach ($culture_flavors as $flavor => $example) {  
        print "A $culture $flavor flavor is $example.\n";  
    }  
}
```

Example 4-31 prints:

```
A Japanese hot flavor is wasabi.  
A Japanese salty flavor is soy sauce.  
A Chinese hot flavor is mustard.  
A Chinese pepper-salty flavor is prickly ash.
```

Exercises

1. According to the US Census Bureau, the 10 largest American cities (by population) in 2010 were as follows:

- New York, NY (8,175,133 people)
- Los Angeles, CA (3,792,621)
- Chicago, IL (2,695,598)
- Houston, TX (2,100,263)
- Philadelphia, PA (1,526,006)
- Phoenix, AZ (1,445,632)
- San Antonio, TX (1,327,407)
- San Diego, CA (1,307,402)
- Dallas, TX (1,197,816)
- San Jose, CA (945,942)

Define an array (or arrays) that holds this information about locations and populations. Print a table of locations and population information that includes the total population in all 10 cities.

2. Modify your solution to the previous exercise so that the rows in the result table are ordered by population. Then modify your solution so that the rows are ordered by city name.
3. Modify your solution to the first exercise so that the table also contains rows that hold state population totals for each state represented in the list of cities.

Exercises

4. For each of the following kinds of information, state how you would store it in an array and then give sample code that creates such an array with a few elements. For example, for the first item, you might say, “An associative array whose key is the student’s name and whose value is an associative array of grade and ID number,” as in the following:

```
$students = [ 'James D. McCawley' => [ 'grade' => 'A+', 'id' => 271231 ],  
              'Buwei Yang Chao' => [ 'grade' => 'A', 'id' => 818211 ] ];
```

- The grades and ID numbers of students in a class
- How many of each item in a store inventory are in stock
- School lunches for a week: the different parts of each meal (entrée, side dish, drink, etc.) and the cost for each day
- The names of people in your family
- The names, ages, and relationship to you of people in your family