第 5 章　PHP 函数

本章主题：

- 函数的声明与调用；
- 函数参数；
- 函数返回值；
- 变量函数；
- 匿名函数；
- 日期和时间函数。

函数通常是指能够完成某个特定功能、可被其他程序调用的一段代码。在软件开发中，函数是功能模块化与代码重用的一项技术，有助于降低软件开发和维护的难度、提高软件开发的生产率以及改善软件质量。在 PHP 中，函数也是页面模块化和页面代码重用的一种手段。除了系统内置的函数，用户可以自定义所需的函数。

5.1　函数的声明与调用

这里先介绍函数声明和函数调用的基本格式及用法，更多的细节会在下面各节依次介绍。

5.1.1　函数声明

函数声明以关键字 function 开始，接着给出函数名。函数名后是一对括号，括号内是可选的形参名列表，各形参之间用逗号分隔。最后给出函数体，即调用该函数时要执行的代码，代码放置在一对花括号内。

```
function <function_name>([<$arg> [,<$arg>]*]) {
   // 要执行的代码;
}
```

函数名的命名规则：

（1）函数名不能和已有的函数重名；

（2）函数名称只能包含字母、数字和下画线；

（3）函数名称不能以数字开头；

（4）长度不限，对大小写不敏感。

函数名不区分大小写，但按其声明时的大小写形式来调用它，是一种好的习惯。函数名应尽量精炼且有意义，例如一个用于产生页脚的函数，用 pagefooter 或 page_footer 作为名称是不错的。

PHP 不支持函数重载，即在一个 PHP 页面文件中，不允许存在两个具有相同函数名的函数，即使他们声明有不同的形参。由于系统内置的函数可应用于任何一个 PHP 页面文件中，所以也不能声明一个与系统内置函数同名的自定义函数。

函数体内包含的代码可以是任何合法的 PHP 代码，例如赋值语句、表达式语句、流程控制语句、函数调用语句，甚至是其他函数或类的声明。

自定义函数的作用域是全局的。例如，可以在一个函数外调用声明在该函数内的函数，也可以在一个函数内调用声明在该函数外的函数。

【例 5-1】 编写一个 isLeap 函数，用于判断指定年份$year 是否为闰年。闰年是指能被 400 整除或者能被 4 整除但不能被 100 整除的年份。代码如下：

```
1.  <?php
2.  function isLeap($year) {
3.     return $year%400===0 || $year%4===0 && $year%100!==0;
4.  }
5.
6.  var_dump(isLeap(1990));
7.  ?>
```

下面是代码运行的输出结果：

```
bool(false)
```

这里，第 2~4 行是题目要求编写的函数，第 6 行代码涉及对该函数的调用。当访问一个 PHP 文件时，文件内声明的函数不会自动执行，只有在其他代码调用它时，它才会执行。

5.1.2 函数调用

函数声明应该出现在 PHP 开始标签和结束标签之间，但在所在页面加载时函数并不会立即执行。函数只有在被调用时才会执行。与调用 PHP 系统内置函数一样，可以通过函数名调用用户自定义函数，函数名后的括号内给出要传递给函数形参的值。函数调用格式如下：

```
<function_name>([<expr> [,<expr>]*]);
```

内置函数可以在所有的 PHP 脚本中使用，但对用户自己声明的函数，只能在包含它的页面内使用。这里，并不要求先声明后使用，通常可以在一个函数声明之前调用该函数。

页面中的 PHP 代码可以调用函数，函数内的 PHP 代码也可以调用其他函数，甚至是该函数本身。这种直接或间接调用自身的函数被称为递归函数。

递归的基本思想是不断把问题分解成规模较小的同类问题，直到分解形成的问题因规模足够小而能直接求得解为止。而这个过程中出现的每个问题都将由规模较小的同类问题的解而求得解。

递归思想反映到代码设计上，体现为一个函数直接或间接地调用自身。这里，函数表示问题的解法，函数参数表示问题的规模。一个递归函数的代码一般应包括两部分：一部分是如果问题的规模足够小，问题的解是什么？另一部分是如果问题的规模不是足够小，

那么它应该如何由规模较小的同类问题的解而求得解。

【例 5-2】 声明一个函数 fib，其功能是求斐波纳契数列的第 n 个数。所谓斐波纳契数列是这样一个数列：前两个数都是 1，第三个数是前两个数之和，以后的每个数都是其前两个数之和。代码如下：

```
1.  <?php
2.  echo fib(20);
3.
4.  function fib($n) {
5.     if($n===1 || $n===2) return 1;
6.     return fib($n-1) + fib($n-2);
7.  }
8.  ?>
```

下面是代码运行的输出结果：

```
6765
```

这里，第 4~7 行是题目要求编写的函数，第 2 行代码包含对该函数的调用。该函数采用递归方法实现，函数体分为两部分。第 5 行代码是第一部分：如果问题规模较小，即变量$n 的值为 1 或 2，则问题的解为 1；第 6 行代码是第二部分：将问题分解为两个规模较小的子问题，它们的规模分别为$n-1 和$n-2，并将两个子问题的解的和作为整个问题的解。

一般来说，使用递归的程序代码会更简洁，也更容易理解，但递归代码的执行效率较低，所以应尽量避免使用。

5.2 函 数 参 数

PHP 支持按值传递参数和按引用传递参数两种方式，也具有默认参数值以及可变长参数等特性。

5.2.1 形参与实参

有些函数可能需要从外部接收一些数据，然后再进行数据处理并完成相应的功能，这时可以为函数定义相应的参数，通常称之为形参。形参被定义在函数名之后，括号内部。一个函数的形参数目不限，两个形参之间用逗号分隔。函数形参类似于在函数内定义的局部变量，在函数内有效。

当调用包含形参的函数时，应提供相应的参数值，通常称之为实参。实参是表达式，两个表达式之间用逗号分隔，各表达式从左到右进行计算，计算结果被传递给对应的形参。参数类型既可以是标量类型，也可以是复合类型。通常，实参的数目和类型应与函数形参的数目和所需的类型相一致。如果实参的数目少于形参的数目，系统将给出警告（Warning）信息，未获得值的形参将是未定义的。如果实参的数目多于形参的数目，则多余的实参被忽略。

默认情况下，函数的参数是按值传递的。这意味着，即使实参是变量，当函数对形参

的值进行改变后，也不会影响函数外部实参的取值。

有些情况下，会希望函数对形参的值的改变能同步反映到函数外部的实参变量上。按引用传递可以满足这一要求。所谓按引用传递是指，当实参是变量时，传递的将是变量的引用而非变量的值，其结果是形参和实参变量引用同一个内存单元、代表同一个变量。要实现按引用传递，只需在形参名前加上符号“&”，例如&$arg。

【例 5-3】 按值传递和按引用传递。代码如下：

```
1.  <?php
2.  function pass_arg($a, &$b) {
3.     $a .= ' and something extra.';
4.     $b .= ' and something extra.';
5.  }
6.  $x = "books";
7.  $y = "bags";
8.  pass_arg($x, $y);
9.  echo $x."<br />";
10. echo $y;
11. ?>
```

下面是代码运行的输出结果：

```
books
bags and something extra.
```

在上述函数 pass_arg 中，第 1 个参数按值传递，所以实参$x 和形参$a 是两个独立的变量。第 2 个参数按引用传递，所以实参$y 和形参$b 引用同一个内存单元。当改变形参$b 的值时，实参$y 的值也随之改变。

要实现按引用传递，需要：

（1）形参名前加&；

（2）实参是变量。

5.2.2 参数的默认值

在声明函数时，可以为函数形参指定默认值。形参的默认值必须是常量表达式，通过运算符“=”给其赋值。例如，下面的函数声明：

```
function <function_name>(<$arg1>,<$arg2>=<constant_expression>) { … }
```

其中，形参$arg2 指定了一个默认值。

当调用函数时，如果没有为具有默认值的形参传递值，那么默认值就会自动赋给该形参。例如，在下面的函数调用中，表达式 expr1 的值传递给形参$arg1，而形参$arg2 将取默认值。

```
<function_name>(<expr1>);
```

具有默认值的形参可以有多个，但必须放置在其他形参后面。当调用函数时，对不具

有默认值的形参应该指定相应的实参，对具有默认值的形参，可以指定实参，也可以没有。如果只指定了部分实参，那么各参数将会按照从左到右的顺序进行赋值。也就是说，不能试图给后面的形参传递值，而让前面的形参取默认值。

【例 5-4】 默认参数值。代码如下：

```
<?php
function default_arg($price, $tax = 0.0675) {
   $total = $price + $price * $tax;
   return $total;
}

echo "Total cost: ".default_arg(12.5)."<br />";
echo "Total cost: ".default_arg(12.5, 0.065);
?>
```

下面是代码运行的输出结果：

```
Total cost: 13.34375
Total cost: 13.3125
```

第 1 次调用函数时，只传递了 1 个参数值 12.5。该参数值赋给形参$price，形参$tax 取默认值 0.0675。第 2 次调用函数时，传递了 2 个参数值。第 1 个参数值 12.5 赋给形参$price，第 2 个参数值 0.065 赋给形参$tax。

【例 5-5】 利用函数实现本书 3.4 节介绍的“动态水平导航栏”模块。

考虑调用外部样式表文件/xk/css/xk.css 的方便性，该例在教务选课系统项目 xk 中实现。文件名为 fnavigation_admin.php，保存在项目中 admin 子文件夹下。下面是该文件的代码：

```
<?php
/*
 *  功能：根据应用状态动态呈现导航栏
 *  输入：链入外部样式表<link rel='stylesheet' type='text/css' href='/xk/
  css/xk.css'/>
 *       $name：登录管理员的姓名
 *       $choice：当前页面对应的菜单项序号，取 1、2 或 3
 */
function navigation_admin($name, $choice=1) {
   echo "<div class='nav'>";
   echo "<a class='right' href='exit.php'>退出</a>";
   echo "<span class='right'>".$name."，您好"."</span>";
   echo "<a class='left ".($choice===1 ? "current" : "").
                        "' href='teacher_p.php'>浏览教师信息</a>";
   echo "<a class='left ".($choice===2 ? "current" : "").
                        "' href='course_p.php'>添加课程</a>";
   echo "<a class='left ".($choice===3 ? "current" : "").
                        "' href='opencourse_p.php'>维护开课信息</a>";
```

```
18.    echo "<div style='clear: both'></div>";
19.    echo "</div>";
20. }
21. ?>
22.
23. <link rel='stylesheet' type='text/css' href='/xk/css/xk.css'/>
24. <?php
25. $user = "刘绍军";
26. navigation_admin($user, 2);
27. ?>
```

第 2~20 行是实现动态水平导航栏的函数，其中第 2~7 行是对该函数的注释说明。第 23 ~26 行演示了对函数的调用。

与用 PHP 文件实现页面模块相比，用 PHP 函数实现页面模块时，调用者与函数之间可以通过参数传递数据，而不需要事先在调用文件处为特定变量设置值，所以更加方便，调用者和被调用者之间的耦合度更低，便于开发和维护。

PHP 文件既可以包含 PHP 代码，也可以包含 HTML 等静态代码，而 PHP 函数内只能是 PHP 代码。如果一个页面模块包含太多的 HTML 等静态代码，那么用函数来实现就会增加一些工作量，且会降低代码的可读性。

5.2.3 可变长参数

调用函数时，有时候也可能需要向函数传递数量不等的参数值。PHP 支持可变长参数。在 PHP 5.6 及以后的版本中，可变长参数通过在形参名前加符号“…”实现。可变长形参必须是形参表中最后一个形参，否则会产生一个致命错误（Fatal error）信息。可变长形参可以接收零个或多个实参值，此时这些参数值将被组织成一个数组赋给该形参。

【例 5-6】 可变长形参的使用。代码如下：

```
1.  <?php
2.  function sum(&$sum, ...$numbers) {
3.     foreach($numbers as $n) {
4.        $sum += $n;
5.     }
6.  }
7.
8.  $original = 100;
9.  sum($original, 1, 2, 3, 4, 5);
10. echo "sum=".$original."<br />";
11. sum($original, 10);
12. echo "sum=".$original."<br />";
13. sum($original);
14. echo "sum=".$original;
15. ?>
```

下面是代码运行的输出结果：

```
sum=115
sum=125
sum=125
```

函数 sum 有两个形参，第 1 个形参按引用传递，第 2 个形参是一个可变长形参。例子共 3 次调用函数，第 1 次调用时，传递给形参$numbers 的是一个包含 5 个元素的数组；第 2 次调用时，传递给形参$numbers 的是一个包含 1 个元素的数组；第 3 次调用时，传递给形参$numbers 的是一个空数组。

反过来，你也可以在实参前加符号“…”，称为可变长实参。可变长实参必须是实参表中的最后一个实参，否则会产生一个致命错误（Fatal error）信息。可变长实参的类型应该是数组。这样，当传递参数时，实参数组中的各元素将被自动取出并一一赋给对应的形参。

如果可变长实参数组的元素过少，不能给相关的形参赋值，系统将给出警告（Warning）信息，未获得值的形参将是未定义的。

【例 5-7】 可变长实参的使用。代码如下：

```
1.  <?php
2.  function add($a, $b, $c) {
3.     return $a + $b +$c;
4.  }
5.
6.  $a = array(1, 2, 3);
7.  echo add(...$a)."<br />";
8.  echo add(10, ...$a)."<br />";
9.  ?>
```

下面是代码运行的输出结果：

```
6
13
```

例子共两次调用函数，第 1 次调用时，可变长实参数组$a 中的 3 个元素值 1、2 和 3 分别传递给了形参$a、$b 和$c。第 2 次调用函数时，第 1 个实参 10 传递给形参$a，可变长实参数组$a 中的前两个元素值 1 和 2 分别传递给了形参$b 和$c。

5.3 函数返回值

return 语句可以出现在函数中。当代码执行 return 语句时，控制将从函数中退出并返回到调用者处。

如果需要函数有一个返回值，可以使用带表达式的 return 语句。表达式可以是任意类型的，所以一个函数可以返回数值、字符串等标量类型的值，也可以返回数组、对象等复合类型的值。

【例 5-8】 数组作为参数值和返回值。代码如下：

```
<?php
function getUser($score) {
    $user = array("20090701011", "LiMing", $score[0], $score[1], $score[2]);
    return $user;
}

$score = array(90, 82, 92);
$a = getUser($score);
print_r($a);
?>
```

下面是代码运行的输出结果：

```
Array ( [0] => 20090701011 [1] => LiMing [2] => 90 [3] => 82 [4] => 92 )
```

调用函数时，传递的参数值是一个数组，包含某个学生的三门课的成绩。函数创建了一个新的数组，除了包含成绩，还包含学生的学号和姓名。函数返回新创建的数组。

一个函数中可以包含多个 return 语句，代码一旦执行到 return 语句就从函数中返回。一般情况下，出现在函数体末尾的不带表达式的 return 语句可以被省略。当省略 return 语句或 return 语句不带表达式时，函数返回 NULL 值。

函数可以返回一个值，也可以返回一个引用。要让函数返回一个引用，需要：

（1）在函数声明时，函数名前使用&；

（2）return 语句所带的表达式是变量；

（3）调用函数时，函数名前使用&。

【例 5-9】 函数返回引用。代码如下：

```
<?php
function &myFunc() {
    static $x = 0;
    $x++;
    echo '$x='.$x."<br />";
    return $x;
}

$y = &myFunc();
$y = 10;
myFunc();
?>
```

下面是代码运行的输出结果：

```
$x=1
$x=11
```

这里，$x 是一个静态变量。第 1 次调用函数时，静态变量$x 初始化为 0，接着加 1 后输出。函数返回$x 的引用并赋给变量$y，这样变量$x 和$y 就引用相同的内存单元。当把$y

置为 10 后，变量$x 的值也为 10。第 2 次调用函数时，静态变量$x 不再初始化，加 1 后变为 11，然后输出。

5.4 变 量 函 数

PHP 支持变量函数的概念。这意味着，如果一个变量名后紧跟着括号，PHP 将试着调用与该变量值同名的函数。

【例 5-10】 可变函数。代码如下：

```
1.  <?php
2.  function foo() {
3.     echo "In foo()<br />";
4.  }
5.  function bar($p = "default") {
6.     echo "In bar($p)<br />";
7.  }
8.  function func($func_name) {
9.     $func_name();
10. }
11.
12. func("foo");
13. func("bar");
14. ?>
```

下面是代码运行的输出结果：

```
In foo()
In bar(default)
```

这里，第 9 行代码是一个变量函数调用，根据变量$func_name 的不同取值，调用不同的函数。

变量函数中，变量指定的函数名既可以是用户自定义函数名，也可以是系统内置函数名，但不能是语言结构名。变量函数不适用于语言结构，例如 echo、print、unset、isset、empty、include 和 require 等。

下面代码中，第 4 行代码是合法的，因为 is_null 是一个内置函数；第 6 行代码是不合法的，因为 isset 是一个语言结构。

```
1.  <?php
2.  $var = true;
3.  $a = "is_null";
4.  var_dump($a($var));
5.  $b = "isset";
6.  var_dump($b($var));
7.  ?>
```

如果不存在与变量值同名的自定义函数或系统内置函数，变量函数调用时将产生一个致命错误（Fatal error）信息。为了防止这类错误，可以在调用函数之前使用 PHP 的系统内置函数 function_exists()来判断该函数是否存在。例如：

```
function func($func_name) {
   if(function_exists($func_name)){
      $func_name();
   }
}
```

函数 function_exists()返回布尔型。如果指定的参数值是某个自定义函数或系统内置函数的名称，则函数返回 true；否则，函数返回 false。

5.5 匿 名 函 数

匿名函数是指没有指定名称的函数。在定义匿名函数时，关键字 function 后面直接跟括号和形参表。下面介绍匿名函数的两种用法。

5.5.1 匿名函数作为变量值

可以把匿名函数作为一个表达式赋给一个变量，然后就可以通过该变量来调用匿名函数了。把一个匿名函数赋值给一个变量的语法与普通变量赋值的语法没有区别，最后也要加上分号。

在内部处理中，PHP 系统会自动把匿名函数转换成内置类 Closure 的一个实例对象，然后再把该实例对象赋给变量。

【例 5-11】 匿名函数作为变量值。代码如下：

```
1.  <?php
2.  // 给变量$green 赋一个匿名函数
3.  $greet = function($name) {
4.     echo "Hello ", $name, "<br />";
5.  };                  // 函数末尾需要分号
6.
7.  $greet('World');    // 通过变量调用函数
8.  $greet('PHP');
9.  ?>
```

下面是代码运行的输出结果：

```
Hello World
Hello PHP
```

使用 use 语言结构，匿名函数可以使用父作用域中的变量。其格式如下：

```
<变量>=function(<$arg>) use(<$var>) {…};
```