第4章 函数

+ 程序在开发的过程中，我们常常需要重复进行某种操作或者处理，如果每次执行相同的操作都要重新写一遍相同的代码，那样会造成系统资源极大的浪费。而且可维护性也会很差，就比如说一个操作需要改进，程序中有多处使用相同的代码，那么改动一个小小的地方，都要费很大的力气。写很多重复的代码也会降低开发的效率。函数就是为解决这些问题而设计的。

4.1 为什么使用函数

- 函数会把一个完整功能的多条语句封装成一个模块，只要需要使用这个功能，我们只需要调用这个模块就可以了。而且模块的内的数据一般情况下和外部的数据是相对独立的，也就提高了软件的可靠性。使用了函数后，一个完整的程序基本就是多种模块，结合我们前面学习的流程控制，就可以完成了。
- 下面通过输出0~100间的所有数、101~200的所有数，就这样以100的范围输出直到输出到1000结束，那么我们用for循环来实现的话，就要使用十个for循环来实现，如下所示。

4.1 为什么使用函数

```php
<?php
    for($a=1;$a<=100;$a++){
        echo "$a<br/>";
    }
    for($a=101;$a<=200;$a++){
        echo "$a<br/>";
    }
    for($a=201;$a<=300;$a++){
        echo "$a<br/>";
    }
    ......
    ......
    ......
    for($a=901;$a<=1000;$a++){
        echo "$a<br/>";
    }
?>
```

未使用函数

```php
<?php
    function shu($x,$y){
        for($a=$x;$a<=$y;$a++){
            echo "$a<br/>";
        }
    }
    shu(1,100);
    shu(101,200);
    shu(201,300);
    ......
    ......
    ......
    shu(901,1000);
?>
```

使用函数

4.1 为什么使用函数

+ 这里就有读者要说了，那完全可以使用一个for循环实现1~1000的输出，那么我们要是每100个数范围内随机排列呢。那就必定写这么多了吧。我们可以看到使用函数的程序简单明了很多，这还是代码很少的函数，如果是好几十换行的代码的情况下，节省的精力就非常可观了。如果现在我们又有一个要求，就是要增加输出我们现在要的是输出每100数范围内所有数相加的和，两种方法的代码改写如下。

+ 这里的优势就相当明显了，只需要在函数里面做一点修改就可以，在不使用函数的写法里，读者可以看到修改的地方就很多了。

```php
<?php
    for($a=1,$num =0 ;$a<=100 ;$a++){
        $num +=$a ;
        echo "$a<br/>";
    }
    echo "以上输出的数字总和是$num <br/>";
    for($a=101,$num =0 ;$a<=200 ;$a++){
        $num +=$a ;
        echo "$a<br/>";
    }
    echo "以上输出的数字总和是$num <br/>";
    for($a=201,$num =0 ;$a<=300 ;$a++){
        $num +=$a ;
        echo "$a<br/>";
    }
    echo "以上输出的数字总和是$num <br/>";
    ......
    ......
    ......
    for($a=901,$num =0 ;$a<=1000 ;$a++){
        $num +=$a ;
        echo "$a<br/>";
    }
    echo "以上输出的数字总和是$num <br/>";
?>
```

不使用函数

框中均为修改过的内容

```php
<?php
    function shu ($x,$y){
        for($a=$x,$num =0;$a<=$y ;$a++){
            $num +=$a ;
            echo "$a<br/>";
        }
        echo "以上输出的数字总和是$num <br/>";
    }
    shu(1,100);
    shu(101,200);
    shu(201,300);
    ......
    ......
    ......
    shu(901,1000);
?>
```

使用函数

+ 这里的优势就相当明显了，只需要在函数里面做一点修改就可以，在不使用函数的写法里，读者可以看到修改的地方就很多了。

4.2 使用函数

+ 在前面我们已经使用案例充分体现了函数的优越性，接下来我们就要逐步学习使用函数了。本章希望读者深入理解函数的精髓，函数是为了程序编写的简单化，高效率而来的。因此学习过程中也不会有很难的知识，我们也尽量讲得通俗简单，让读者不会感到压力。

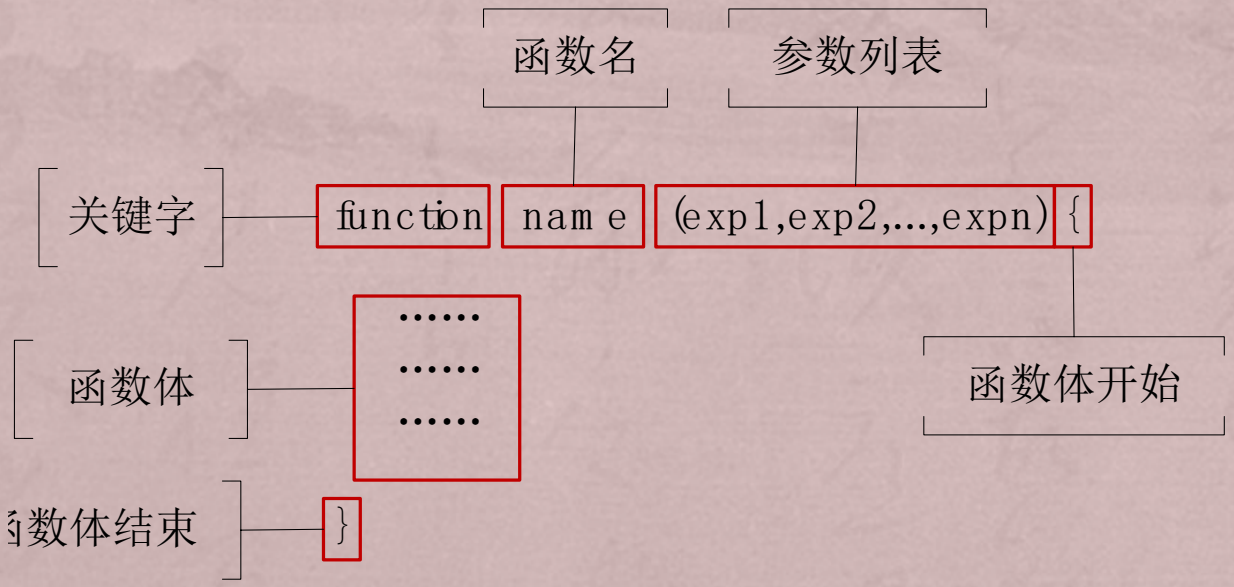
4.2.1 定义和调用函数

+ 在上一节中，我们已经通过一个示例切实体验到了使用函数的优越之处，函数的使用也是大势所趋。本节的目标就是通过下面的学习，让读者学会使用函数。

1.函数的定义

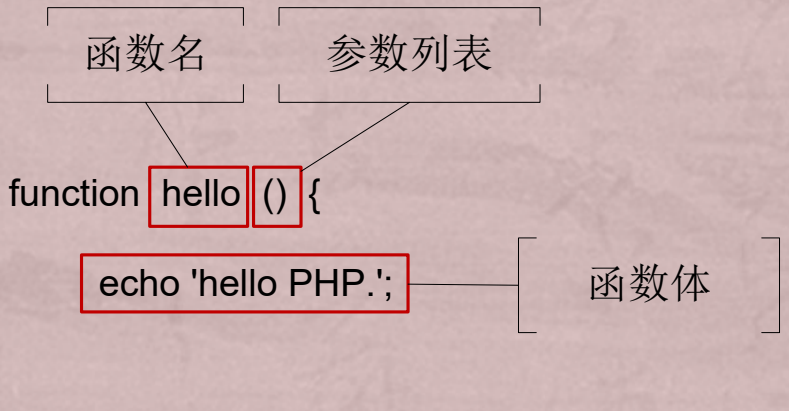
- PHP中使用function关键字来定义一个函数，函数定义的语法比较简单，主要由函数名、参数列表和函数体组成。如图所示。

- 在图中，函数名要符合PHP的命名规范。
- 参数列表中的参数可以是一个或者多个也可以没有，视情况而定。函数可以没有参数，但是不可以省略小括号。
- 函数体是函数的主体，用于函数功能的实现。
- 函数体里面可以是任何代码，包括循环语句、分支语句或者是另一个函数。



1.函数的定义

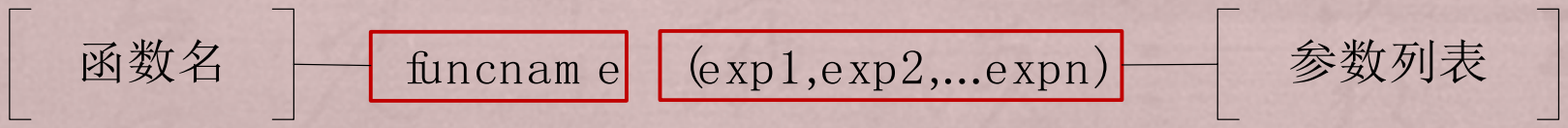
+ 下面我们就定义一个简单的函数，它的功能就是输出一句话。



+ 上面的代码就是定义了一个名为hello的函数。这个函数的功能就是输出一句话，不需要参数。这里提醒读者，没有参数也不可以省略小括号。

2.函数的调用

- 前面我们已经学会了如何定义函数，光定义了函数是没有任何作用的，只有调用了它，函数才会工作。函数的调用是非常简单的只需要写出函数名并且传入对应的参数就可以了。语法如图所示。
- 在图中需要注意的是PHP函数名虽然对大小写不敏感，但是也建议与定义函数名大小写保持一致。



2.函数的调用

+ （1）调用前面我们定义的函数hello。
+ 在PHP函数先定义后调用，也可以先调用，后定义，比如把上面的程序改成如下这样也可以正常使用。

```
+ <?php
+ hello();     //这里在函数定义前调用了函数
+ function  hello(){
+     echo 'hello PHP.';
+ }
+ ?>
```

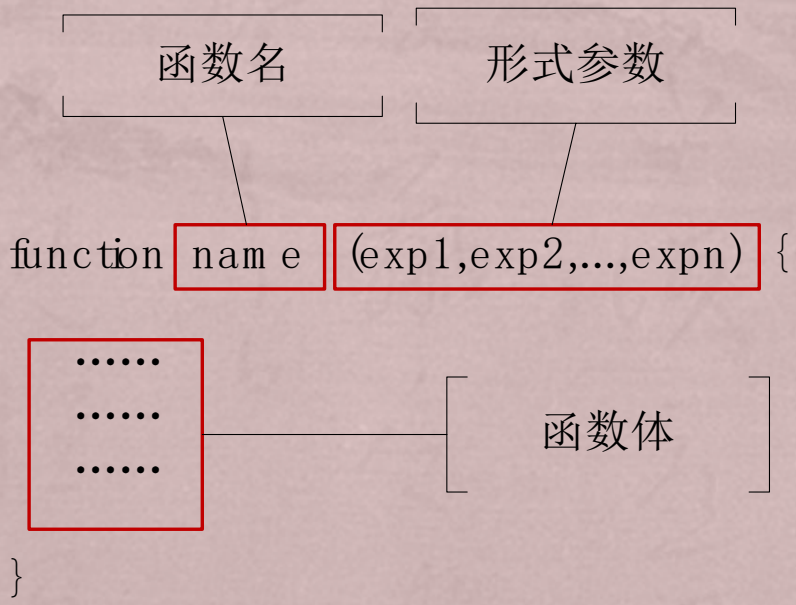
+ 上面的程序也会输出正确的运行结果。

4.2.2 函数的参数

+ 通过函数调用，我们可以很轻松实现输出一个语句“hello PHP”。但很多时候我们需要让函数输出不同的内容。这个时候，我们就需要在调用的时候，通知函数我们要输出的内容。这个时候，就需要使用到参数。参数是函数内部和函数外部进行数据交换的端口，函数中数据的传入都是由参数来完成的。根据参数使用的位置，参数分为形式参数和实际参数。

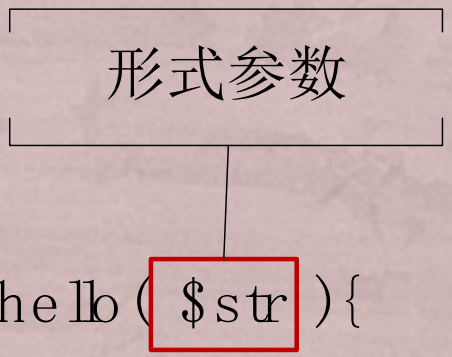
1.形式参数

- 形式参数就是我们定义函数时候的参数，就像它的名字一样，它只是一个形式，而不是一个具体。因为函数体中需要使用外部传入的参数，为了参数可以正确地传递进来，就需要通过形式参数与函数体里面的数据进行传递。



1.形式参数

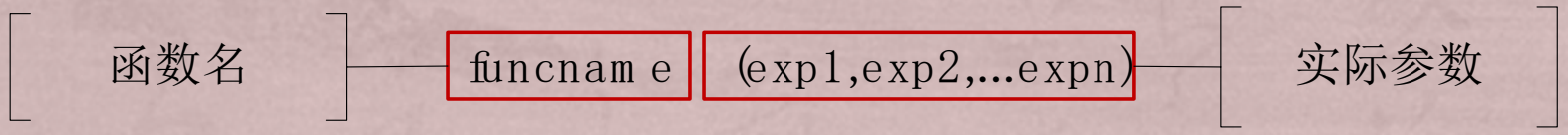
+ （1）演示一个具体函数的形式参数。




```
function hello( $str ){
    echo 'hello $str.';
}
```

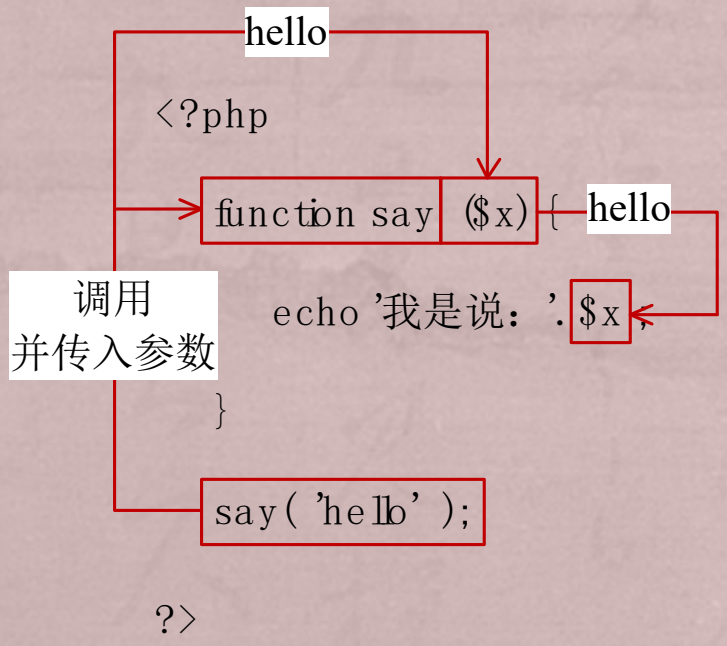
2.实际参数

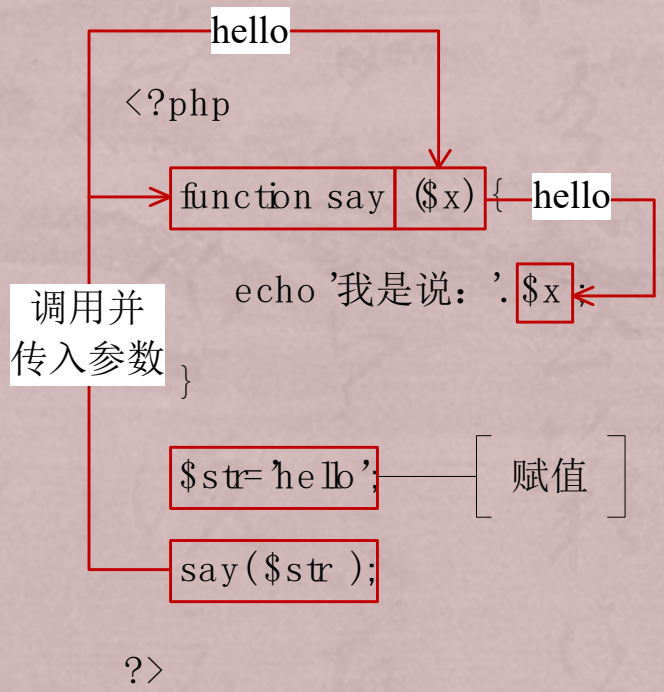
- 实际参数就是我们调用函数时候在参数列表里传入的参数，它会替换形式参数在函数体中对应的变量值，函数的参数可以接受常量和变量。



2.实际参数

+ 展示函数调用的实际参数。



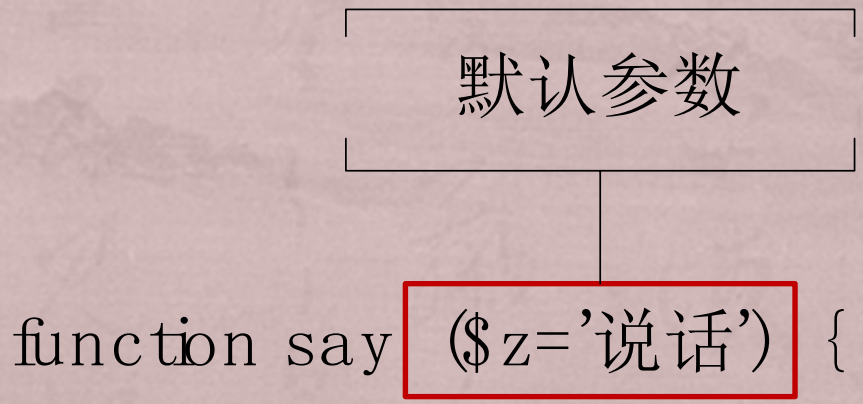


2.实际参数

+ 这里需要特别注意的是，实际参数一定要和形式参数的个数相对应，不然程序会出错，
+ （1）读者不对应形式参数调用函数会发生错误。

3.默认参数

- 默认参数就是函数的参数列表中的若干参数指定了值，如果调用函数时候不传入对应的值则函数会接受默认参数的值，这样可以避免调用时候出现没有参数的错误。也可以使一些程序显得更加合理。如果传入对应的参数，就会替换初始值。
- 如下定义的就是默认函数。



3.默认参数

+ （1）我们不传参数就调用它；
+ （2）我们给他传一个参数，他就会替换默认参数的值。

4.2.3 函数参数的传递

- 函数的参数的传递就类似我们前面学习的赋值运算一样，赋值运算有两种赋值方式。而函数传递的方式也类似的有两种方式，一种就是值传递，一种就是引用传递。

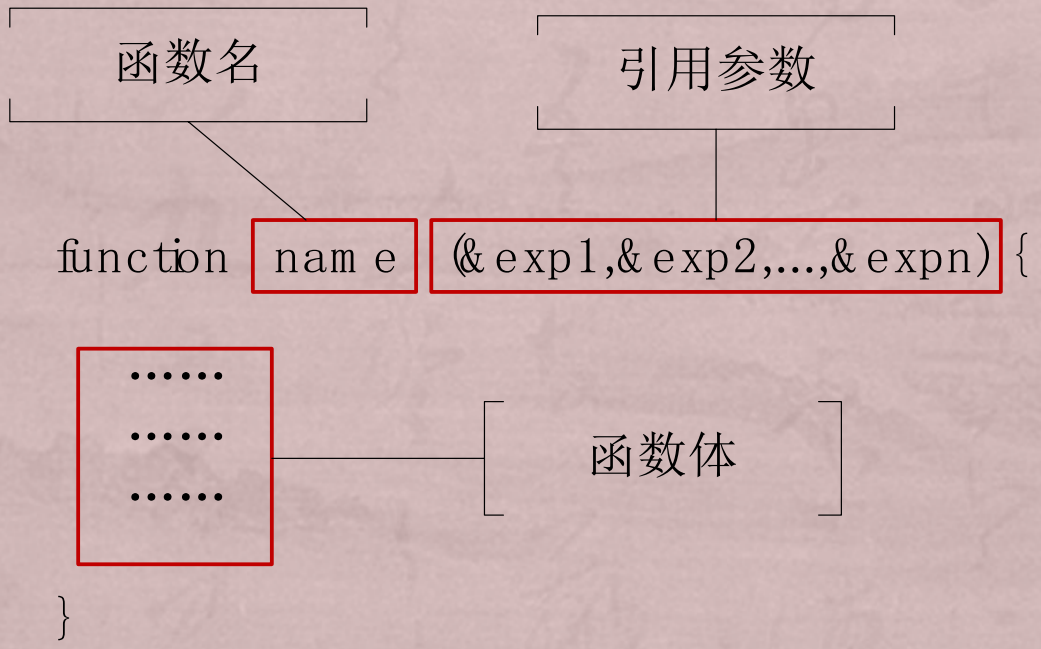
1.值传递

- 默认情况下，函数参数通过值传递的方式传递的，因为值传递的方式是给函数传递了一个参数的副本，所以它并不会改变函数外部的值。因此如果不希望函数修改它的参数值，就通过值传递的范式参数。
- 我们就用一个示例来让读者了解一下值传递。
- （1）通过函数比较两个参数的大小，并且交换两个参数的值。
- 我们可以看到结果是他们的值没有被交换。我们可以多加几条输出语句，来看看这个程序是怎么运行的。
- （2）演示swap的运行情况。
- 我们可以清楚的看到，在函数内，数值确实是交换了，而在函数外部，数值确实是没有变化的。这就再次证实了，函数的值传递只是传递变量的副本。那么要想让函数的操作都保留体现出来，我们就需要使用引用传递的方式了。

2.引用传递

- 参数的引用传递就是把变量自身传给函数，让函数去操作的，因此函数对参数的操作，就会被保留，会在函数外体现出来。
- 上面说了引用传递的的原理，肯定会有同学想到了解决办法了，我们前面学过变量的引用赋值，这里我们就可以使用变量的引用来实现。
- （1）演示采用赋值传递方式给swap函数传入参数。
- 上面的程序虽然实现了输出，但是这个示例只是告诉读者那样确实是可以得到我们想要的结果。它并不是我们真正要讲的知识。PHP函数参数的引用传值是在函数定义的时候使用引用传递的。如图所示。

2.引用传递



+ （1）改写我们程序为如下图所示。
+ 我们可以看到，程序输出了我们最满意的结果，通过上面的学习，读者应该了解这两种传递方式的不同，并且会初步地正确地使用他们。

4.2.4 函数中的变量

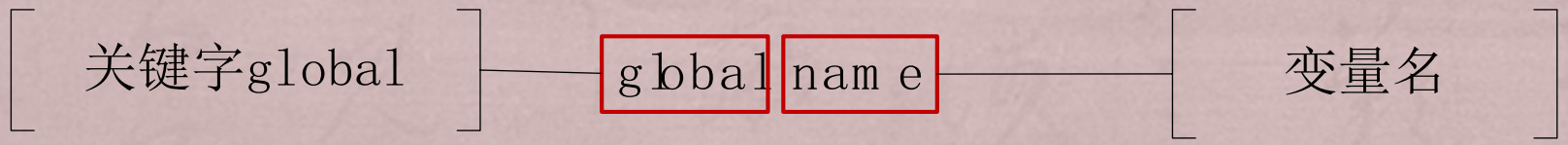
+ 在PHP中，在没有学习函数之前，我们前面已经学习过了变量。在函数中使用变量就不得不提变量的一个属性，那就是变量的作用域。变量的作用域就是变量的有效范围。在函数外部声明的变量叫做全局变量全局变量的作用范围在整个PHP文件，但是在函数中不会被读到，不会影响到函数中的同名变量。也就是说函数内和函数外可以有同名的函数。函数内部声明的变量叫做局部变量，它的作用范围就是函数内，函数调用完毕后，这个变量也就消失了。下面我们就具体来讲解这些知识。

1.全局变量

+ 全局变量就是函数外部定义的变量，它通常不会被函数读到，除非在函数中特别声明后才可以被读到。我们看下面的案例。
+ （1）尝试使用函数输出变量$a.
+ 从运行结果我们可以看出我们尝试调用函数输出全局变量$a的时候，程序会报出变量未定义的错误。也就是说没有$a这个变量。但是很明显我们可以看到程序的开头就定义了$a变量。这就再一次说明了函数内和函数外使用的是两个不同的范围。

1.全局变量

+ 那么我们非想要在函数范围中使用全局变量的话，该怎么办呢？我们就使用global关键字来定义函数中的变量。定义全局变量的语法如图所示。



1.全局变量

+ （1）使用global关键字定义全局变量，并调用函数输出变量的值。

+ 程序正确地输出了我们想要的值。函数中global的意思就是告诉程序函数中要使用全局变量$a。当然使用了全局变量不仅可以输出，我们也可以在函数中改变它的值。

+ （2）在函数内定义全局变量，并且给它赋值后输出。

+ 我们可以看出在函数中的赋值是成功的改变了全局变量的值。

2.局部变量

+ 局部变量就是函数中定义的变量。它的作用范围就是在函数体中。函数之外不会读到它，除非使用global声明为全局变量。这里读者又可以这么想，既然作用范围只在函数内，那就说明多个函数内可以定义一个相同名字的变量，这个是肯定的。
+ （1）在多个函数中定义相同名字的变量。并输出他们的值。
+ 从运行结果我们就可以看出，函数内的$num和函数外的$num不是一个变量。因为我们可以看到经过调用多个函数后，全局变量的值还是保持不变的。
+ 当然函数中的局部变量也不可以在函数外被读到。我们看下面一个简单的示例。
+ （2）局部变量不可在函数外部被使用。
+ 那么如果函数中的变量想要在函数外部使用该怎么办呢？我们也是使用global把局部变量声明为全局变量。我们看下面的例子。
+ （3）局部变量使用global定义后在函数外部使用。

3.静态变量

- 函数在执行时所产生的变量，在函数结束时就消失了，有时因为程序的需要，函数在使用中，当不希望变量在每次执行完函数就消失的话，静态变量就派上用场了。静态变量用在函数内，被调用完后，保留最后值，多用来统计累加。我们看下面的示例。
- （1）在函数中变量使用递增运算并输出结果。
- 从上面的运行结果来看，虽然在函数中执行了递增操作，但是在输出的结果中我们可以看到，三次输出的结果都是相同的。这就说明每次函数执行完毕后其中的变量就被初始化为0了。我们把变量声明为静态变量就可以很好地解决这个问题。下面我们来看定义静态变量的语法。如图所示。

4.2.5 函数的返回值

- 我们在前面的学习中，使用的函数大多是执行一个过程。函数没有明确返回一个值。其实在PHP中函数都是返回值的，只是我们不明确指出返回值的话，函数会返回空（NULL）。在PHP中使用return语句返回值。return语句会立即中止函数的运行，并且将控制权交回调用该函数的代码行。下面我们看return的语法。
- 图中的返回值不只可以是值，也可以是一个表达式。
- （1）演示使函数返回值的案例。
- 上面的代码中因为函数本身没有输出的语句，因此我们需要使用输出语句输出函数的返回值。函数并不是非得返回有相关运算的值，我们也可以返回一个无关运算的值，这种返回值常常用作判断函数是否执行成功。我们看下面的示例。

4.2.5 函数的返回值

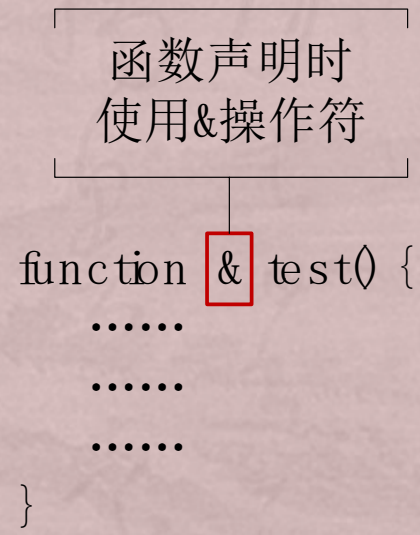
+ （1）使用返回语句返回一个无关运算的值，用于判断函数是否执行成功。
+ 当然上面的代码是我们为了讲解这个知识点才这么写的。在实际中，实现同样的功能可以使用其他更加简明的方法。这里就要求读者举一反三试着写出其他更加简便的方法。这里还要说明一点是，PHP中的return语句不能返回多个值，但是可以返回任意类型的值。因此我们可以使用返回一个数组来间接返回多个值，这个知识我在讲解数组时候会提及。
+ 函数既然有返回值，那么就可以赋值一个变量，我们可以用一个非常简单的例子来说明一下。如下所示。
+ （2）把函数的返回值赋值给一个变量，并输出变量的值。
+ 我们可以看出，程序正确输出了函数的返回值，这种赋值函数赋值个变量的方法也是比较常用的，比如可以做为判断语句的判断条件。

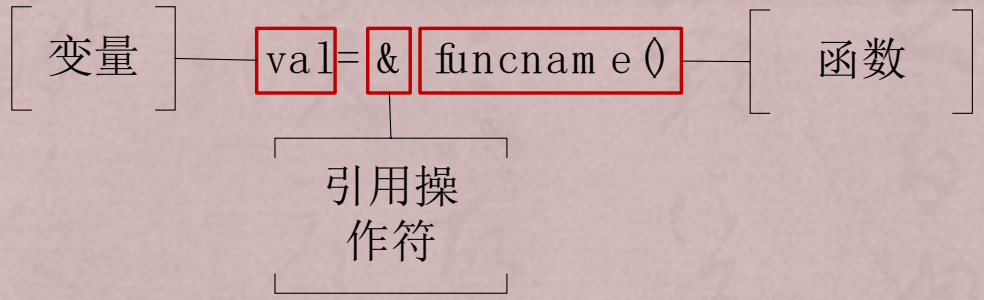
4.3 函数的其他使用方法

+ ①函数的引用返回
+ ②可变函数
+ ③匿名函数
+ ④递归函数

4.3.1 函数的引用返回

- 函数的引用返回就是从函数返回一个引用，PHP的规定是必须在函数声明和返回一个引用值给变量时候使用操作符&。下面我们看来看函数的引用返回声明和赋值给一个变量的语法。
- 在图中，如果把函数的引用赋值写成这样的形式：
- val=funcname()
- 那么函数的返回值会以值的方式传递给变量的。这里是读者特别需要注意的。下面我就来看一个函数引用返回的示例。





4.3.1 函数的引用返回

+ （1）使用函数的引用返回赋值给一个变量后，输出变量的值，然后改变变量的值后再次输出。

4.3.2 可变函数

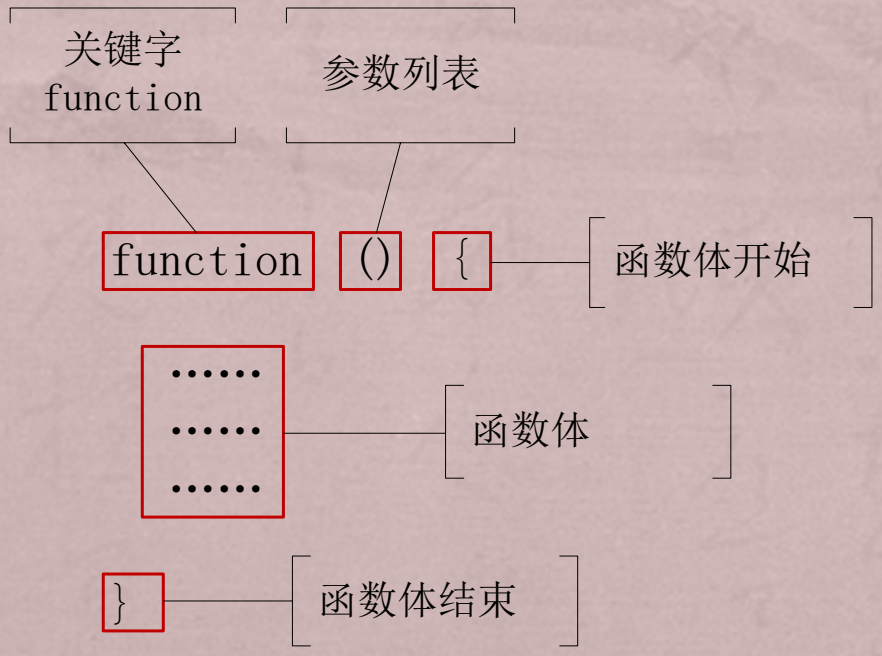
+ PHP支持可变函数的概念。我们可以这样理解，如果一个变量名后有括号，那么PHP就会寻找与变量值同名的函数并执行它。这就意味着给一个变量赋不同的值。程序就会调用不同的函数。可变函数的语法如图所示。

4.3.2 可变函数

+ （1）使用可变函数调用不同的函数输出不同的语句。

4.3.3 匿名函数

+ 匿名函数就是没有函数名的函数。PHP允许临时创建一个没有指定名称的函数。匿名函数通常用在回调函数中。这个我们将会在合适的时机讲解。而匿名函数也通常赋值给一个变量后使用。匿名函数的语法结构如图所示。

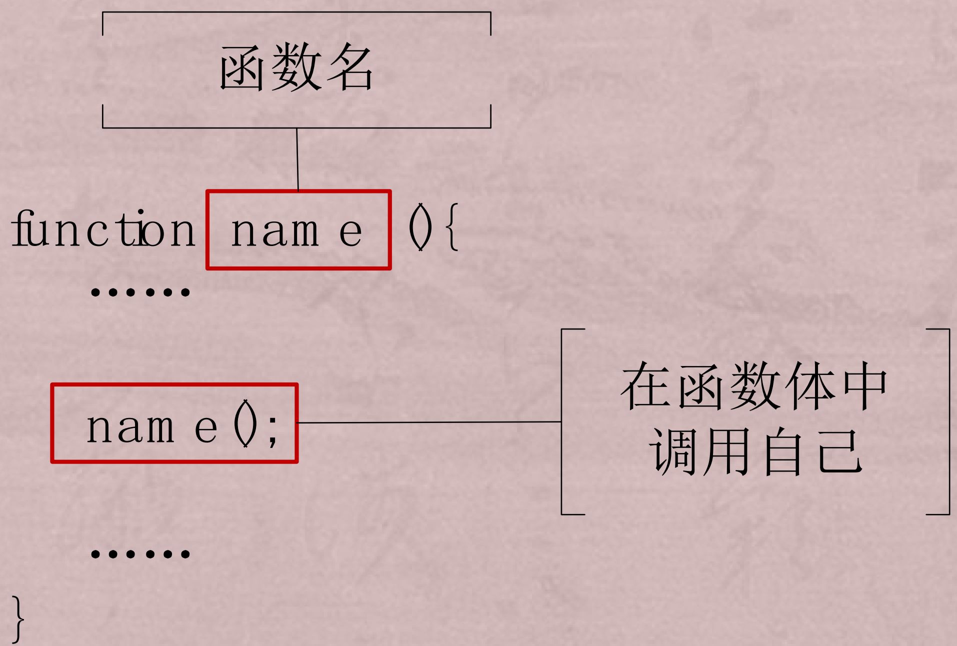


4.3.3 匿名函数

+ （1）使用将匿名函数赋值给变量的方式使用匿名函数。

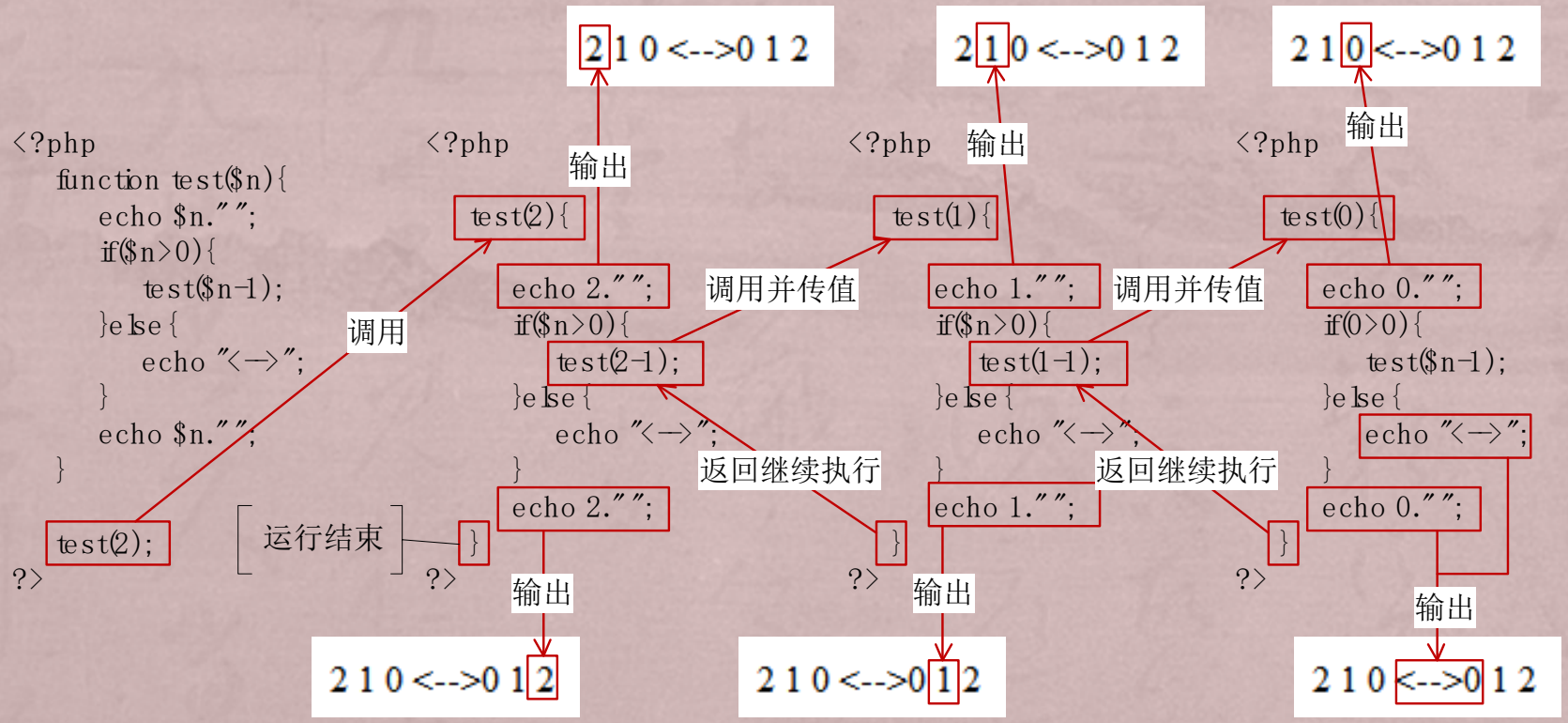
4.3.4 递归函数

- 递归函数即为自调用函数，在函数体内直接或间接自己调用自己。它的语法图4.32所示。
- 通常来说递归函数都被认为是比较难理解的。其实读者只要仔细看懂下面画出的执行过程图，充分理解后，自己再根据程序画一次（当然可以改变一些代码）执行过程图，一般情况下就可以了解了。千万记住，光靠看和脑子分析是很难理解的。下面我们就看一个使用递归的案例。



4.3.4 递归函数

+ （1）使用递归函数输出2 1 0 <--->0 1 2。



4.4 系统函数

- 在PHP中，除了可以自己定义函数之外，系统提供了很多的系统函数。这些函数都不必定义，可以直接拿来使用。他们大多是常常被使用的方法。PHP为了方便我们使用，就把它们直接集成在系统里面了。PHP中包含1000多个系统函数并且在不断增改。系统函数的使用方法和自定义函数的使用方法是一样的。就比如说max函数，它的作用是返回参数列表里最大的数。可以比较无限多个数。它的使用方法如下。
- 当然还有返回最小值的min函数，输出绝对值的abs函数等等。这些函数以及帮助信息都可以在官方的PHP手册里面找到。当然我们在写程序的过程中最好是给使用的系统函数添加备注，以防频繁查询手册带来的时间浪费。

4.4 系统函数

- 在PHP中，除了可以自己定义函数之外，系统提供了很多的系统函数。这些函数都不必定义，可以直接拿来使用。他们大多是常常被使用的方法。PHP为了方便我们使用，就把它们直接集成在系统里面了。PHP中包含1000多个系统函数并且在不断增改。系统函数的使用方法和自定义函数的使用方法是一样的。就比如说max函数，它的作用是返回参数列表里最大的数。可以比较无限多个数。它的使用方法如下。
- max(12,123,54,31,78)
- 当然还有返回最小值的min函数，输出绝对值的abs函数等等。这些函数以及帮助信息都可以在官方的PHP手册里面找到。当然我们在写程序的过程中最好是给使用的系统函数添加备注，以防频繁查询手册带来的时间浪费。

4.5 小结

+ 到这里我们的函数学习就基本完毕了。函数就是将复杂的PHP程序分成若干个功能模块。然后每个模块都编写成一个PHP函数。再通过在程序中调用函数，以及函数中调用函数来解决大型问题。因此读者要对函数的定义，调用和返回值有正确的理解和认识。尤其要注重理解和应用。要在学习和空闲时间多加练习才能逐步走向熟练。