PHP官方手册（中文）：

https://www.php.net/manual/zh/

基本语法：

==========

echo是输出语句，用于输出字符串

PHP标记四种语法风格：

1、标准标记

以 <?php 开始，以 ?> 结束

这是最常用的标记类型，服务器不能禁用这种风格的标记。它可以达到更好的兼容性、可移植性、可复用性，所以PHP推荐使用这种标记。

2、短标记

以 <? 开始，以 ?> 结束

使用短标记，必须在配置文件php.ini中启用short_open_tag选项。另外，因为这种标记在许多环境的默认设置中是不支持的，所以PHP不推荐使用这种标记。

3、ASP标记

以 <% 开始，以 %> 结束

须在配置文件php.ini中启用asp_tags选项，在许多环境的默认设置中是不支持的，因此在PHP中不推荐使用这种标记

4、SCRIPT标记

以 <script language=”php”> 开始，以 </script> 结束

PHP一般不推荐使用这种标记，只需了解即可。

PHP注释

单行注释 / 或 #

多行注释 /* */

PHP变量

定义变量：$变量名

PHP的数据类型

标量类型： boolean（布尔型）Integer（整型）float（浮点型）string（字符串型）

复合类型： array（数组）object（对象）

resource（资源）

NULL（空值）

1、boolean布尔类型

$bool1 = true; //把true值赋给变量$bool1

2、integer整型

前面可加上“+”或“-”号表示正数或负数。 $b = -123; //十进制负数，数值-123
当使用八进制表示时，数字前必须加上0（零），$c = 0123; //八进制数，等于十进制的83
使用十六进制表示时，数字前必须加上0x，$d = 0x123; //十六进制数，等于十进制的291

3、float浮点型

浮点型可以存储整数，也可以存储小数

4、string 字符串

$a='字符串';
$b="字符串"; echo "
"; //会被解析成换行
包含在双引号的字符串会被解析，而包含在单引号中的字符串不会解析，只会输出其字符本身。

PHP中检测数据类型的相关函数

is_bool	检测变量是否属于布尔类型
is_string	检测变量是否属于字符串类型
is_float	检测变量是否属于浮点类型
is_integer	检测变量是否属于整型
is_null	检测变量是否属于空值
is_array	检测变量是否属于数组
is_resource	检测变量是否属于资源
is_object	检测变量是否属于对象类型
is_numeric	检测变量是否属于数字或数字组成的字符串

可变变量：将变量的值作为变量名

实现过程就是在变量的前面加一个$符号

自动类型转换：变量的类型由PHP自动转换，无需做任何操作

1、转换成布尔型

一些值会被转为false，除此之外，其他值会被转为true。具体如下（false）：
整型值0（零）
浮点型值 0.0（零）
空字符串，以及字符串 "0"
不包括任何元素的数组
不包括任何成员变量的对象

2、转换成整型

布尔型转换成整型：布尔值true，转换成整数1；布尔值false，转换成整数0。

强制类型转换： 在变量前加一个小括号，并把目标类型填写在括号中实现

（boolean）	强转为布尔型
（string）	强转为字符串型
（integer）	强转为整型

(float)　　强转为浮点型
(array)　　强转为数
(object)　　强转为对象

赋值运算符

/=　除等于　　$a=3;$b=2;$a/=$b; $a=1.5;$b=2;
%=　模等于　　$a=3;$b=2;$a%=$b;　　$a=1;$b=2;
.=　连接等于　$a='abc';$a .= 'def';　　$a='abcdef'
在PHP语言中可以通过一条赋值语句对多个变量进行赋值
$a = $b = $c = 5;　　//为三个变量同时赋值

比较运算符

==　等于
!=　不等于
<>　不等于
===　　恒等
!==　不恒等

逻辑运算符

&&	与	$a && $b	$a和$b都为true，结果为true，否则为false
||	或	$a || $b	$a和$b中至少有一个为true，则结果为true，否则为false
!	非	!$a	若$a为false，结果为true，否则相反
xor	异或	$a xor $b	$a和$b其一为true，但不同时是，结果为true，否则为false
and	与	$a and $b	与&&相同，但优先级较低
or	或	$a or $b	与||相同，但优先级较低

错误控制运算符　　：使用@符号来表示，把它放在一个PHP表达式之前，将忽略该表达式可能产生的任何错误信息。

使用示例：
$a = @4/0;
注意：
@运算符只对表达式有效，例如可以把它放在变量、函数和include()调用、常量之前，但不能把它放在函数或类的定义之前。

流程控制语句

选择结构语句
if...elseif...else语句：执行语句用{}包含

switch语句:

```
switch (表达式){
        case 目标值1:
                执行语句1
                break;
        case 目标值2:
                执行语句2
                break;
        . . . . . .
        case 目标值n:
                执行语句n
                break;
        default:
                执行语句n+1
                break;
}
```

循环结构语句

1、while循环语句

2、do...while循环语句

3、for循环语句

```
for(初始化表达式; 循环条件; 操作表达式){
        执行语句
        ……
}
```

跳转语句

跳转语句用于实现循环执行过程中程序流程的跳转，在PHP中的跳转语句有

break语句、continue语句和goto语句

1、break语句

在switch条件语句和循环语句中都可以使用break语句。当它出现在switch条件语句中时，作用是终止某个case并跳出switch结构。当它出现在循环语句中，作用是跳出循环语句，执行后面的代码。

注意：break 可以接受一个可选的数字参数来决定跳出几重循环

2、continue语句：终止本次循环，执行下一次循环。

3、goto语句

注意：

goto语句仅在PHP 5.3及以上版本有效。

PHP中的goto语句只能在同一个文件或作用域中跳转，也就是说无法跳出一个函数或类方法，也无法跳入另一个函数。

函数

==========

函数的定义

function 函数名 ([参数1，参数2，……])

{

函数体

}

函数名不区分大小写，如search()和SEARCH()指的是同一个函数，这点与变量的命名不同。

函数的返回值

```
<?php
    function sum($a,$b)  {
      return $a+$b;
    }
     echo "两个数的和等于：";
     echo sum(23,96);
?>
```

函数中变量的作用域

```
<?php
    $var = 100;                          //此处$var是全局变量
    function test(&$i) {
   global $var;
    $i = 100+ $i+$var;
        echo "在函数内部var的值为：".$i;  //在函数内部调用全局变量$var，结果都是300
    }
    test($var);
     echo"<br>";
    echo"$var";
?>
```

上例中显示了：形参和实参、函数作用域、

&引用不是获得变量原本的值，而是指向原值。任何对引用的修改都会影响原变量值

global 关键字用于函数内访问全局变量。

可变函数

PHP 支持可变函数的概念，这意味着如果一个变量名后有圆括号，PHP 将寻找与变量的值同名的函数，并且尝试执行它。

```
<?php
    function calculatePrice($price,$discount){   //定义函数calculatePrice()
        $discount_price = $price * $discount;
          echo "商品的原价为".$price."元";
          echo "<br>";
        echo "商品的折扣为".$discount;
    echo "<br>";
        echo "商品的折扣价为".$discount_price."元";
        echo "<br>";
    }
    $price = 100;
    $discount = 0.7;
    calculatePrice($price,$discount);  //直接调用函数calculatePrice()
    $calculateFunc = "calculatePrice"; //将函数名"calculatePrice"赋值给变量$calculateFunc
    $calculateFunc($price,$discount); //调用与变量值同名的函数
?>
```

结果：

商品的原价为100元
商品的折扣为0.7
商品的折扣价为70元
商品的原价为100元
商品的折扣为0.7
商品的折扣价为70元

函数的嵌套调用

```
<?php
    function sum($subject1,$subject2,$subject3){          //定义计算总分的函数
            return $subject1 + $subject2 + $subject3;       //返回总分
    }
    function avg($subject1,$subject2,$subject3,$number){       //定义计算平均分的函数
            return sum($subject1,$subject2,$subject3) / $number;  //返回平均分
    }
    $chinese = 90;
```

```
    $math = 85;
    $english = 79;
    $number = 3;
    echo "平均分为" . avg($chinese,$math,$english,$number);
?>
```

结果：

平均分为84.666666666667

函数的递归调用

```
<?php
    // 下面的函数使用递归实现 求1~n的和
    function getSum($n) {

        if ($n == 1) {          // 满足条件，递归结束
            return 1;
        }
        $temp = getSum($n - 1);
        echo "<br/>";
        echo "$n";
        return $temp + $n;
    }
    echo "<br/>sum = ".getSum(4); // 调用递归函数，打印出1~4的和
?>
```

结果：

2

3

4

sum = 10

字符串相关函数

explode函数

作用：

使用一个字符串分割另一个字符串。每个元素都是 string 的一个子串

它们被字符串 $delimiter 作为边界点分割出来

函数：

explode (string $delimiter , string $string [, int $limit]) : array

参数：

$delimiter　　边界上的分隔字符。 $string　　输入要进行分割的字符串。

$limit　　如果设置limit参数并且是正数，则返回的数组包含最多limit个元素，而且最后哪个元素将包含string的剩余部分。

如果limit参数是负数，则返回最后的-limit个元素外的所有元素。

如果limit是0，则会当作1

返回值：

此函数返回由字符串组成的 array，每个元素都是 string 的一个子串，它们被字符串 delimiter 作为边界点分割出来。

备注：

如果 $delimiter 为空字符串（""），explode() 将返回 FALSE。

如果 $delimiter 所包含的值在 string 中找不到，并且使用了负数的 limit。

那么会返回空的 array， 否则返回包含 string 单个元素的数组。

例子：

```
<?php
    $str = 'tacks1 tacks2 tacks3 tacks4 tacks5';
var_dump(explode('',$str));//boolean false //空的$delimiter 会返回False并且报错。
var_dump(explode(' ',$str));//按照空格分开每个子串成为数组。
var_dump(explode(' ',$str,3));//数组元素为3个，前连两个按照指定的字符分割，剩下的全部挡在组后一个数组元素
var_dump(explode(' ',$str,-1));//在分割后的数组，删除最后一个元素tacks5
var_dump(explode('AAA',$str,0));//如果字符串中没有$delimiter，那么会全部当成数组的一个单元
?>
```

这里特别注意：**var_dump()** 函数用于输出变量的相关信息。

echo()函数是在屏幕上输出字符串，要输出数据需使用print_r() 函数

implode()函数

作用：

将一个一维数组的值按照特定的字符串$glue转化为字符串。

函数：

implode (string $glue , array $pieces) : string

join (string $glue , array $pieces) : string（join是implode的别名）

implode (array $pieces) : string (最好不用)

参数：

$glue 默认为空的字符串作为粘合数组每个元素 $pieces你想要转化的数组

返回值：

返回一个字符串，其内容为由 glue 分割开的数组的值。

备注：

因为历史原因，implode() 可以接收两种参数顺序，也就是第一个参数$glue也可以不写。

但是最好还是些两个参数向后兼容。而且第二个参数必须是一维数组。

例子：

```
$str = 'tacks1 tacks2 tacks3 tacks4 tacks5';
$arr = ['tacks1','tacks2','tacks3','tacks4','tacks5'];
echo implode(',',$arr),'<br/>';//tacks1,tacks2,tacks3,tacks4,tacks5
echo implode($arr),'<br/>';//tacks1tacks2tacks3tacks4tacks5
echo join('-',$arr),'<br/>';//tacks1-tacks2-tacks3-tacks4-tacks5
```

strcmp()函数

作用：判断两个字符串大小

函数：strcmp(string $str1,string $str2)

效果：返回一个整数

如果字符串$str1和$str2相等，则函数返回0；如果字符串$str1小于$str2，则函数返回值小于0（两个字符串的差值）；如果字符串$str1大于$str2，则函数返回值大于0。

例子：

```
echo strcmp("123","1234");  //-1
```

str_replace()函数

作用：字符串替换

函数：str_replace(string $search,string $replace,string $subject[,int &$count]) ： string

参数:

$search参数表示被替换掉的字符串，$replace参数表示替换后的字符串，$subject参数表示需要被操作的字符串，count()函数是用来统计$search参数被替换的次数，它是一个可选参数，与其他函数参数不同的是，当完成str_replace()函数的调用后，该参数还可以在函数外部直接被调用。

例子：

```
$sss = 'This is a book,That is an apple';
$str = 'apple';
echo str_replace('is',$str,$sss,$count),"<br/>",$count;
```

结果：

Thapple apple a book,That apple an apple

3

substr()函数

作用：截取一个字符串中的某一部分，也就是获取字符串中的某个子串

函数：string substr(string $str,int $start[,int $length])

参数: 函数名前的string表示函数的返回值类型是字符串类型，参数$str用于表示待处理的字符串，参数$start表示，从位置为start的字符处开始进行截取，参数$length表示截取的子串长度为

$length，该参数是可选的，如果$length为空，则默认截取到字符串的末尾。

例子：

```
$sss = 'This is a book,That is an apple';
echo substr($sss,1),"<br/>",substr($sss,0,1),"<br/>",substr($sss,-1);
```

结果：

his is a book,That is an apple

T

e

strlen()函数

作用：统计字符串的长度

函數：int strlen(string $str)

参数：int表示strlen()函数的返回值类型是整数类型，参数$str用于表示待获取长度的字符串。

例子：

```
$sss = 'This is a book,That is an apple';
echo strlen($sss); //31
```

trim()函数

作用：过滤字符串

函數：string trim (string $str [, string $charlist])

参数：函数名前的string表示函数的返回值类型是字符串类型，参数$str 用于表示待处理的字符串，参数$charlist是可选的，在调用函数时，若指定了$charlist，则函数会从字符串末端开始删除$charlist指定的字符，若没有指定$charlist，则函数会从字符串末端开始删除空白字符。

例子：

```
$sss = 'This     is     a       appleapple,That is an appleapple';
echo trim($sss,"apple"),"<br/>",trim($sss);
```

结果：

This is a appleapple,That is an //末尾的apple全部去除了

This is a appleapple,That is an appleapple//去除多余空格

日期和时间管理

获取时间

获取系统当前时间

time函数：

time(void)：int

time()函数没有参数，返回值为int类型。

date()函数

作用：格式化日期时间

参数：string date(string $format [, int $timestamp])

返回值是string类型，$timestamp为可选参数，如果没有指定则使用本地当前时间。format为固定参数，表示给定的格式

例子：

```
<?php
// Prints: July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000)),"<br>";
echo date("Y-m-d H:i:s",time()) ,"<br>";//当前时间
?>
```

关于date（）相关参数请参考：

https://www.runoob.com/php/php-date.html

PHP默认的时区设置是UTC，修改默认的时区设置，通常情况下有两种修改方式：

1.有权限情况下，直接修改php.ini中的date.timezone配置，例如将默认时区设置为PRC（中华人民共和国），具体如下：

date.timezone = PRC

修改完date.timezone配置后，需要重启服务器。

2.在程序中使用**date_default_timezone_set()**函数来设置时区，该函数的声明方式如下所示：

bool date_default_timezone_set(string $timezone_identifier)

例：date_default_timezone_set("PRC");

在上述声明中，返回类型是bool型，timezone_identifier用于指定时区标识符，可以是“PRC”、“Asia/Shanghai”或者“Asia/Chongqing”等

UNIX时间戳

Unix时间戳(Unix timestamp)是一种时间表示方式，定义为从格林威治时间**1970年01月01日00时00分00秒**起至现在的**总秒数。**以32位二进制数来表示，其中1970年1月1日零点也叫Unix纪元。时间戳不能为负数，因此1970年以前的时间戳无法使用

mktime()函数

作用：取得一个日期的 Unix 时间戳

参数：

mktime ([int $hour = date("H") [, int $minute = date("i") [, int $second = date("s") [, int $month = date("n") [, int $day = date("j") [, int $year = date("Y") [, int $is_dst = -1]]]]]]) : int

例子：

1：基本例子

```
<?php
// Set the default timezone to use. Available as of PHP 5.1
date_default_timezone_set('UTC');
// Prints: July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));
// Prints something like: 2006-04-05T01:02:03+00:00
echo date('c', mktime(1, 2, 3, 4, 5, 2006));
?>
```

2：mktime() 在做日期计算和验证方面很有用，它会自动计算超出范围的输入的正确值

```
<?php
echo date("M-d-Y", mktime(0, 0, 0, 12, 32, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 13, 1, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1998));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 98));
?>
```

结果都为：Jan-01-1998

strtotime函数

作用：将时间字符串转换为时间或时间戳，经常用于获取用户提交的时间

参数：int strtotime(string $time [, int $now])

$time用于指定日期时间字符串，$now用于计算相对时间的参考点，如果省略则使用系统当前时间。

例子及用法：

```
<?php
echo strtotime("2016-1-22"),"<br>";//获取指定日期的unix时间戳
 /*获取英文文本日期时间,便于比较，使用date将当时间戳与指定时间戳转换成系统时间*/
echo date("Y-m-d H:i:s",time()) ,"<br>";//当前时间
echo date("Y-m-d H:i:s",strtotime("+1 day")) ,"<br>";/*明天此时的时间,打印明天此时的时间戳strtotime("+1 day")*/
echo date("Y-m-d H:i:s",strtotime("-1 day")),"<br>";/*昨天此时的时间，打印昨天此时的时间戳strtotime("-1 day") */
echo date("Y-m-d H:i:s",strtotime("+1 week")) ,"<br>";/*下个星期此时的时间,打印下个星期此时的时间戳strtotime("+1 week")*/
echo date("Y-m-d H:i:s",strtotime("-1 week")) ,"<br>";/*上个星期此时的时间，打印上个星期此时的时间戳strtotime("-1 week")*/
```

```
echo date("Y-m-d H:i:s",strtotime("next Thursday")) ,"<br>";/*指定下星期几的时间，打印指定下星期几的时间戳strtotime("next Thursday")*/
echo date("Y-m-d H:i:s",strtotime("last Thursday")) ,"<br>";/*指定上星期几的时间，打印指定上星期几的时间戳strtotime("last Thursday") */
?>
```

microtime()函数

作用：获取精确时间，如计算脚本的执行时间

参数：mixed microtime([bool $get_as_float])

该函数当前返回Unix时间戳以及微秒数，参数$get_as_float是可选参数，如果未设置为true将返回一个浮点数，如果省略，则以msec sec格式返回一个字符串，其中msec是微秒部分，sec是秒数，但都是以秒为单位返回的。

数组

数组类型：

索引数组：

索引数组的下标是从0开始，并依次递增

关联数组

关联数组是指下标为字符串的数组。它的键和值之间有一定的业务逻辑关系，因此，通常使用关联数组存储一系列具有逻辑关系的变量。

数组的定义：

1、使用赋值方式定义数组：$arrayName[key] = value

“$”是定义变量开始的标识符，“key”是数组的下标，其类型可以是整型或字符串，“value”可以是任意类型的数据。

2、使用array()函数定义数组：$arrayName = array(key1 => value1, key2 => value2, ...)

如果省略了key部分，则定义的数组默认为索引数组。

注意:

如果在定义数组时没有给某个元素指定下标，PHP就会自动将目前最大的那个整数下标值加1，作为该元素的下标，并依次递增后面元素的下标值。

数组元素的下标只有整型和字符串两种类型，如果是其他类型，则会进行类型转换。

由于合法的整型值的字符串下标会被类型转换为整型下标，所以在创建数组的时候，如果转换后数组存在相同的下标时，后面出现的元素值会覆盖前面的元素值。

数组的使用

$数组名[键名]

访问数组：可以使用方括号（[]）访问数组元素，还可以使用花括号（{}）。例如，$arr[0]和$arr{0}的效果是一样的。

输出数组：

PHP提供了print_r()和var_dump()函数，用于输出数组中的所有元素。

其中，print_r()函数可以按照一定格式显示数组中所有元素的键和值。

删除数组：

unset()函数

作用：删除数组中的元素。

数组指针

函数名	作用
mixed current (array &$array)	获取数组中当前元素的值，如果内部指针超出数组的末端，则返回false
mixed key (array &$array)	获取当前元素的下标，即键名
mixed next (array &$array)	将数组的内部指针向前移动一位
mixed prev (array &$array)	将数组的内部指针倒回一位
mixed end (array &$array)	将数组的内部指针指向最后一个元素
mixed reset (array &$array)	重置指针，即将数组的指针指向第一个元素

数组遍历

格式一：无键名遍历

```
foreach ($arr as $value) {
    循环体
}
```

格式二：键值对遍历

```
foreach ($arr as $key => $value) {
    循环体
}
```

两种语法格式中都是通过foreach语句来实现对数组的遍历，在格式一中，只是将当前元素的值赋给$value。而在格式二中，将当前元素的键名赋值给$key，值赋值给$value，这样可以同时获取当前元素的键名和值。

注意:

使用foreach遍历数组时，$key 和 $value 只不过是一个变量名而已，任何符合语法的变量名均可，如$k和$v。

$key和$value保存的数据是通过值传递的方式赋值的，这意味着对$key 和 $value的修改不影响数组本身。可以使用引用传递，在变量前加上&即可，但要注意这种方式只对 $value有效，$key不会改变。

数组排序

冒泡排序：在冒泡排序的过程中，不断地比较数组中相邻的两个元素，较小者向上浮，较大者往下沉，整个过程和水中气泡上升的原理相似。

数组元素查找

在数组中常用的查找元素的方法有顺序查找和二分法查找。

1、顺序查找法

顺序查找就是按照数组中的元素排列序号，从前往后一个一个查，如果找到则返回当前元素所在的下标。

2、二分查找法

二分法查找就是每次将指定元素和数组中间位置的元素进行比较，从而排除掉其中的一半元素，以此类推，继续进行查找。需要注意的是二分查找法只用于排序后的数组。

数组基本函数：

1、is_array()函数：作用是判断一个变量是否是数组，如果是数组，则返回true，否则返回false。

2、count()函数：作用是计算数组中元素的个数

声明方式：int count(mixed $var [, int $mode]);

在声明中，count()函数接收两个参数，其中$var参数是必需的，它表示传入的数组对象。$mode参数是可选参数，其值为0或1（COUNT_RECURSIVE）。该参数默认值为0，如果将该参数设置为1，则count()函数会递归计算多维数组中每个元素的个数。

3、array_unique()函数：作用是移除数组中的重复元素

array_unique()函数接收一个数组对象，去除重复元素后返回一个新的数组对象。

作用原理：在使用该函数时，首先将数组元素的值作为字符串排序，然后对每个值只保留第一个键名，忽略后面所有的键名。

数组键值对的相关函数：方便操作PHP数组键与值

1、array_search()函数：用于获取数组中元素对应的键名

声明方式：mixed array_search(mixed $needle , array $haystack [, bool $strict]);

$needle参数表示在数组中要查找的值，$haystack参数表示被查询的数组。$strict是可选参数，当值为true时，就会在$haystack数组中检查$needle的类型。

2、array_keys()函数：也是用于获取数组中元素对应的键名。不同的是，array_keys()函数可以返回所有匹配的键名

声明方式如下：

array array_keys(array $input[,mixed $search_value[,bool $strict]]);

$input参数表示被查询的数组。$search_value参数是可选参数，当给$search_value赋值时，该函数返回该值的键名，否则返回$input数组中的所有键名。自PHP 5起，可以用$strict参数来进行全等比较（===），需要传入一个布尔值，默认false，如果传入true值则根据类型返回带有指定值的键名。

数组排序函数：

sort()函数：对数组中的元素按照由小到大的顺序进行排序，

声明方式：bool sort (array &$array [, int $sort_flags = SORT_REGULAR]);

$array参数表示需要排序的数组，$sort_flags是可选参数，sort()函数会根据$sort_flag的值来改变数组的排序方式。

$sort_flag的取值范围以及对应的排序方式：

取值范围	**排序方式**
SORT_REGULAR	默认值，将自动识别数组元素的类型进行排序
SORT_NUMERIC	用于数字元素的排序
SORT_STRING	用于字符串元素的排序
SORT_LOCALE_STRING	根据当前的locale设置来把元素当做字符串比较

数组合并和拆分函数：

array_merge()函数：作用是合并一个或多个数组，

声明方式：array array_merge(array $array1 [, array $...]);

注意：

array_merge()将一个或多个数组的单元合并起来，一个数组中的值附加在另一个数组的后面，返回一个新的数组。如果输入的数组中有相同的字符串键名，则该键名后面的值将覆盖前一个值。如果数组包含数字键名，后面的值将不会覆盖原来的值，而是附加到数组的后面。如果数组是数字索引的，则键名会以连续方式重新编排索引。

array_chunk()函数：作用是将一个数组分割成多个数组

声明方式：array array_chunk(array $input , int $size [, bool $preserve_keys]);

$input表示是要分割的数组，$size是分割后的每个数组中元素的个数。preserve_keys是一个可选参数，默认值为false，如果将该参数设置为true，则分割后的数组中元素保留原来的索引，如果将该参数设置为false，则分割后的数组中元素的索引将从零开始。

操作数组的其他函数：

array_rand()函数：作用是从数组中随机取出一个或多个元素

声明方式：mixed array_rand(array $input [, int $num_req]);

array_rand()函数接收一个input参数和一个可选的参数num_req，其中input参数用于指定接收的数组，num_req参数用于指定取出元素的个数，默认为1。 如果只取出一个元素，array_rand()会返回一个随机元素的键名，否则就返回一个包含随机键名的数组。

array_reverse()函数：作用是返回一个元素顺序相反的数组

声明方式：array array_reverse(array $array [, bool $preserve_keys]);

array_reverse()接收数组array作为输入并返回一个元素为相反顺序的新数组，如果preserve_keys为 true，则保留原来的键名。