函数名不区分大小写，这点与变量名不同。

例：计算圆的面积案例：

define("PI",3.14); //习惯定义常量为大写。

$r =9;

$s=getCircleArea($r);

echo "圆形的面积为".$s;

function getcirclearea($raduis)

{

$area=PI\*$raduis\*$raduis;

return $area;

}

如果希望在函数内部使用函数外面的变量，需要在函数内部使用关键字global修饰变量。

$var=100;

function test() {

global $var; //在test()范围内，使用函数外面的变量

echo "在函数内部var的值为：".$var;

}

test();

预定义变量**$GLOBALS**：它是一个包含了全部变量的全局组合数组，变量的名称就是数组的键，可以读取全局变量。

$var=100;

function test()

{

//使用$GLOBALS访问全局变量

echo "在函数内部var的值为：".$GLOBALS["var"];

}

test();

function getsum($n)

{

if($n==1) //满足条件，递归结束

return 1;

$temp=getsum($n-1); //调用自身，使n的值递减

return $temp+$n;

}

echo "sum=".getsum(4);

注：getsum()函数被调用了4次，并且每次调用时，n的值都会递减。当n的值为1时，所有递归调用的函数都会以相反的顺序相继结束，所有的返回值会进行累加。

**Explode()函数：分隔字符串**

语法：Array explode( string $separator, string $str[, int $limit] )

Arr表示数组类型，它是函数的返回值类型，参数$separator表示字符串的分隔符，要与定义的字符串分隔符相同，参数$str表示要分隔的字符串，参数$limit是可选的，~~用于表示返回的数组中最多可包含limit个元素。~~(表示分隔的次数,什么鬼解释)

参数$limit有三种取值情况：

1. 如果参数$limit是正数，则返回的数组包含最多limit个，而最后那个元素将包含$str的剩余部分。
2. 如果参数$limit是负数，则返回除了最后的limit个元素外的所有元素。
3. 如果参数$limit是0，则它会被当做1。

$str="apple,pear,banana,orange";

$arr=explode(",",$str); //用逗号对str字符串进行分隔

echo "第一次分隔的结果为：";

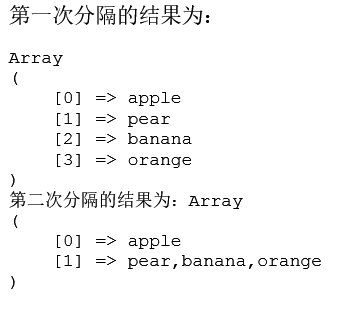
echo "<pre>";

print\_r($arr)."<br>";

echo "第二次分隔的结果为：";

$arr=explode(",",$str,2); //用逗号对str字符串进行分隔，限制返回字符串个数为2

print\_r($arr);



**implode()**函数：将字符数组拼接成一个新的字符串。

声明：string implode( string $glue, array $arr)

函数名前的string表示函数的返回值类型是字符串类型，参数$glue表示连接符，参数$arr表示待合并的数组。

例子：

$color\_arr=array("red","green","blue");

$str="accelworld";

$color\_str=implode($str,$color\_arr);

echo "color\_str=".$color\_str;

返回的值：color\_str=redaccelworldgreenaccelworldblue

**strcmp()函数**：判断两个字符串的大小

声明：int strcmp( string $str1, string $str2 )

Int表示函数的返回值是整数类型，参数$str1和$str2均表示待比较的字符串。

按ASCII码比较，从两个字符串的开头第一个字符开始比较，跳过相同，直到比较到有不相同的字符或字符串的结尾才停止比较，此时返回比较结果。

1．$str1和$str2相等，则函数返回0。

2．$str1小于$str2，则函数返回小于0的值。

3．$str1大于$str2，则函数返回大于0的值。

“==”操作符比较字符串，结果为true(非0)时，表示字符串相等。

注意两者的区别。

例子：

$str1="abcde";

$str2="abcDe";

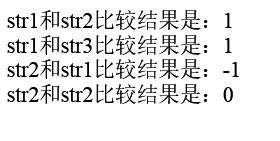
$str3="abcd";

echo "str1和str2比较结果是：".strcmp($str1,$str2)."<br>";

echo "str1和str3比较结果是：".strcmp($str1,$str3)."<br>"; //长度大者大

echo "str2和str1比较结果是：".strcmp($str2,$str1)."<br>";

echo "str2和str2比较结果是：".strcmp($str2,$str2)."<br>"; //可以比较相同字符串



**str\_replace()函数**：对字符进行替换操作

声明：mixed str\_replace( mixed $search, mixed $replace, mixed $subject [,int &$count])

$search表示查找的目标值，

$replace参数表示$search的替换值，

$subject参数表示需要被操作的字符串，

$count参数是用来统计$search参数被替换的次数，是一个可选参数，与其他参数不同的是，在完成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：

$str1="i like play football, and he is also like play football";

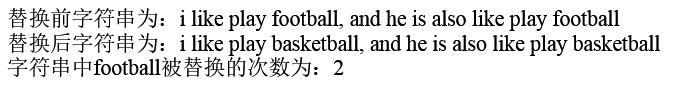
$str2="basketball";

echo "替换前字符串为：".$str1."<br>";

$str=str\_replace("football",$str2,$str1,$count);

echo "替换后字符串为：".$str."<br>";

echo "字符串中football被替换的次数为：".$count."<br>";

**substr()函数**：获取字符串的子串。

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

参数$str表示待处理的字符串，参数$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：

$str="this is a string";

$str1=substr($str,0,4);

$str2=substr($str,0);

$str3=substr($str,0,-4);

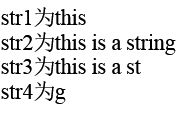
$str4=substr($str,-1);

echo "str1为".$str1."<br>"; //从第一个字符开始，截取4个字符

echo "str2为".$str2."<br>"; //从第一个字符开始截取，直到字符串的末尾

echo "str3为".$str3."<br>"; //从第一个字符开始截取，直到字符串末端第4个字符

echo "str4为".$str4."<br>"; //从字符串末端返回1个字符



**strlen()函数**：获取字符串的长度

声明：int strlen( string $str)

参数$str用于表示待获取字符串的长度

$str1="abcd";

$str2="中文字符串";

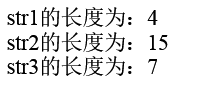
$str3="空 格";

echo "str1的长度为：".strlen($str1)."<br>";

echo "str2的长度为：".strlen($str2)."<br>";

echo "str3的长度为：".strlen($str3)."<br>";

注意：单个英文字符和一个空格的长度均为1，一个中文字符的长度为3。



**trim()函数**：用于去除字符串中首尾两端的空白字符。

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

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$charlist，则函数会删除$charlist指定的字符，若没有指定$charlist，则去除字符串$str首尾处的空白字符。

$str=" hello world ";

echo "未调用trim函数：".$str;

echo "<br>";

echo "调用trim函数：".trim($str);

UNIX时间戳(UNIX timestamp)是一种时间表示方式，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。

由于UNIX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：返回一个指定的时间戳。

声明：int mktime ([int $hour [,int $minute [,int $second [,int $month [,int $day [,int $year] ] ] ] ] ])

参数可以从右向左省略，任何省略的参数会被设置成本地日期和时间的当前值。

echo date("Y-m-d H:i:s",mktime())."<br>";

echo date("Y-m-d H:i:s",mktime(14))."<br>";

echo date("Y-m-d H:i:s",mktime(14,28))."<br>";

echo date("Y-m-d H:i:s",mktime(14,28,56))."<br>";

echo date("Y-m-d H:i:s",mktime(14,28,56,4))."<br>";

echo date("Y-m-d H:i:s",mktime(14,28,56,4,10))."<br>";

echo date("Y-m-d H:i:s",mktime(14,28,56,4,10,2013))."<br>";

Y-m-d H:i:s 代表年，月，日 时，分，秒

注意大小写，新手请按部就班的学习。

由于时间戳是一个整数如1365604136，以32位二进制数来表示，所以使用date()函数用于格式化输出。

**获取系统当前时间**

time()函数

声明：int time( void )

该函数无参数，php默认时区为UTC，根据需要更改时区即可。

echo date("Y-m-d H:i:s",time())."<br>";

**获取用户提交的时间**

strtotime()函数：用于将字符串转化成时间戳。

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，省略则使用系统当前时间。

echo date("Y-m-d H:i:s",strtotime("2016-10-16 17:45:12"))."<br>";

echo date("Y-m-d H:i:s",strtotime("+1 day"))."<br>";

echo date("Y-m-d H:i:s",strtotime("next monday"))."<br>";

先不管那么多，就照上面的用法用着先。。。

**获取精确时间**

microtime()：与time相似，用于处理更精确的时间。

声明：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，设置为true将返回一个浮点数，省略则以msec sec格式返回一个字符串，其中mesc是微秒部分，sec是秒数，但都是以秒为单位返回的。

function microtime\_float()

{

list($usec,$sec)=explode(" ",microtime());

return ((float)$usec+(float)$sec);

}

$time\_start=microtime\_float();

usleep(1000);

$time\_end=microtime\_float();

$time=$time\_end-$time\_start;

echo "执行该脚本花费了($time)秒";

虽然还看不懂，等我全部学完还看不懂就是我自己的问题了~~~

usleep ：以指定的微秒数延缓程序的执行。

**格式化输出**

date()函数：格式化日期时间

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

返回将整数$timestamp按照给定的格式字串而产生的字符串。如果没有给出时间戳则使用本地当前时间。换句话说，$timestamp 是可选的，默认值为 time() 。

例子：echo date("Y年m月d日H时i分s秒",time())."<br>";

echo "Today is ".date("l",time())."<br>"; // l：星期几

echo date("l dS \of F Y h:i:s A"); //非format字符要加上反斜线来转义，否则会按照format字符进行解释

// d 月份中的第几天，有前导零的 2 位数字

// S 每月天数后面的英文后缀，2 个字符

// F月份，完整的文本格式，例如 January 或者 March

// Y 4 位数字完整表示的年份

// h 小时，12 小时格式，有前导零

// I 有前导零的分钟数

// s 秒数，有前导零

// A 大写的上午和下午值

mt\_rand — 生成更好的随机数

它可以产生随机数值的平均速度比 libc 提供的 rand() 快四倍。

range — 建立一个包含指定范围单元的数组

str\_shuffle — 随机打乱一个字符串