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

函数：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

函数：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