使用return关键字可以使函数返回值，省略return则默认返回值为NULL。

<?php

function add($a){

return $a+1;

echo “函数add末尾内容输出中……”;

}

$b=add(1);

echo $b; //$b = 2

?>

返回语句会立即中断函数的执行，并且将控制权交回调用该函数的代码行

<?php

function add($a){

return $a+1;

$a=10;

return $a+20;

}

$b=add(1);

echo $b; //$b = 2

?>

可变函数：即通过变量的值来调用函数。因为变量的值是可变的，所以可以通过改变一个变量的值来是实现调用不同的函数，常用在回调函数、函数列表、或者根据动态参数来调用不同的函数。可变函数的调用方法为变量名加括号。

<?php

class book

{

function getName()

{

return 'bookname';

}

}

$func = 'getName';

$book = new book();

echo $book->$func();

//"->"可以理解为它的意思，类似于$book.$func()

?>

创建一个对象

<?php

class Car

{ //定义属性

public $name = '汽车';

//定义方法

public function getName()

{

//方法内部可以使用$this伪变量调用对象的属性或者方法

return $this->name;

}

}

$car = new Car();

echo $car->getName();

?>

在类中定义的变量称为属性，属性声明是由关键字public(公共的)，proteced(受保护的)，private(私有的)开头，后面跟一个普通的变量声明来组成。属性的变量可以设置初始化的默认值，默认值为常量。

默认为public，外部可以访问，一般通过 -> 对象操作符来访问对象的属性或者方法，对于静态属性则使用 :: 双冒号进行访问。

proteced和private不允许外部调用，只能在类的成员方法内部调用。

使用关键字static修饰的，称之为静态方法。静态方法不需要实例化对象，可以通过类名直接调用，操作符为双冒号 :: 。

<?php

class Car

{

public static function getName()

{

return '汽车';

}

}

echo Car::getName();

?>

**PHP类和对象之构造函数和析构函数**

PHP5可以在类中使用**\_\_construct()**定义一个构造函数，具有构造函数的类，会在每次对象创建的时候调用该函数，因此常用来在对象创建的时候进行一些初始化工作。

class Car {

function **\_\_construct()** {

print "构造函数被调用\n";

}

}

$car = new Car(); //实例化的时候 会自动调用构造函数\_\_construct，这里会输出一个字符串

在子类中如果定义了\_\_construct则不会调用父类的\_\_construct，如果需要同时调用父类的构造函数，需要使用parent::\_\_construct()显式的调用。

<?php

class Car {

function \_\_construct() {

print "父类构造函数被调用\n";

}

}

class Truck extends Car {

   function \_\_construct() {

       print "子类构造函数被调用\n";

  parent::\_\_construct();

   }

}

$car = new Truck();

?>

同样，PHP5支持析构函数，使用**\_\_destruct()**进行定义，析构函数指的是当某个对象的所有引用被删除，或者对象被显式的销毁时会执行的函数。

class Car {

function \_\_construct() {

print "构造函数被调用 \n";

}

function **\_\_destruct()** {

print "析构函数被调用 \n";

}

}

$car = new Car(); //实例化时会调用构造函数

echo '使用后，准备销毁car对象 \n';

unset($car); //销毁时会调用析构函数

当PHP代码执行完毕以后，会自动回收与销毁对象，因此一般情况下不需要显式的去销毁对象。

注意\_\_construct是两个\_

<?php

//获取类中的多个属性

class Person

{

private $name;

private $age;

public function \_\_get($a)

{

echo "自动调用\_\_get()方法获取属性值<br>";

if(isset($this->$a))

{

return ($this->$a);

}

else

{

return (NULL);

}

}

public function \_\_set($a,$v)

{

echo "自动调用\_\_set()方法为属性赋值<br>";

$this->$a = $v;

}

}

$p1 = new Person();

$p1->name = "张华";

$p1->age = 10;

echo "姓名：".$p1->name."<br>";

echo "年龄：".$p1->age."<br>";

?>

常量属于类本身而非对象的，所以需要使用范围解析操作符( :: )来连接类名和类常量来访问。

<?php

class MathTool

{

const PI = 3.1415926;

public function show()

{

echo MathTool::PI."<br>"; //通过类名访问

}

public function display()

{

echo self::PI."<br>"; //通过self关键字访问,等同于类名

}

}

echo MathTool::PI."<br>"; //在类外部直接访问

$obj = new MathTool();

$obj->show();

$obj->display();

?>

如果构造函数定义成了私有方法，不允许直接实例化，一般用静态方法来实例化。

<?php

class Car

{

private function \_\_construct()

{

echo 'object create';

}

private static $\_object = null;

public static function getInstance()

{

if(empty(self::$\_object))

{

self::$\_object = new Car(); //内部方法可以调用私有方法，因此此处可以创建对象

}

return self::$\_object;

}

}

//$car = new Car(); //这里不允许直接实例化

$car = Car::getInstance(); //通过静态方法获得一个实例

?>