# PHP 简介（11.14）

* PHP 文件可包含文本、HTML、JavaScript代码和 PHP 代码
* PHP 代码在服务器上执行，结果以纯 HTML 形式返回给浏览器
* PHP 文件的默认文件扩展名是 **.php**。

[网站]->选择demo.io, 点击[管理],在下拉菜单中选择[打开网站]

可以在打开的默认浏览器中看到网站创建成功的欢迎页

也可直接在浏览器地址栏输入: demo.io访问

打开demo目录, 清空内容,创建一个index.php

phpinfo();

访问demo.io/index.php, 可以看到当前php运行环境信息

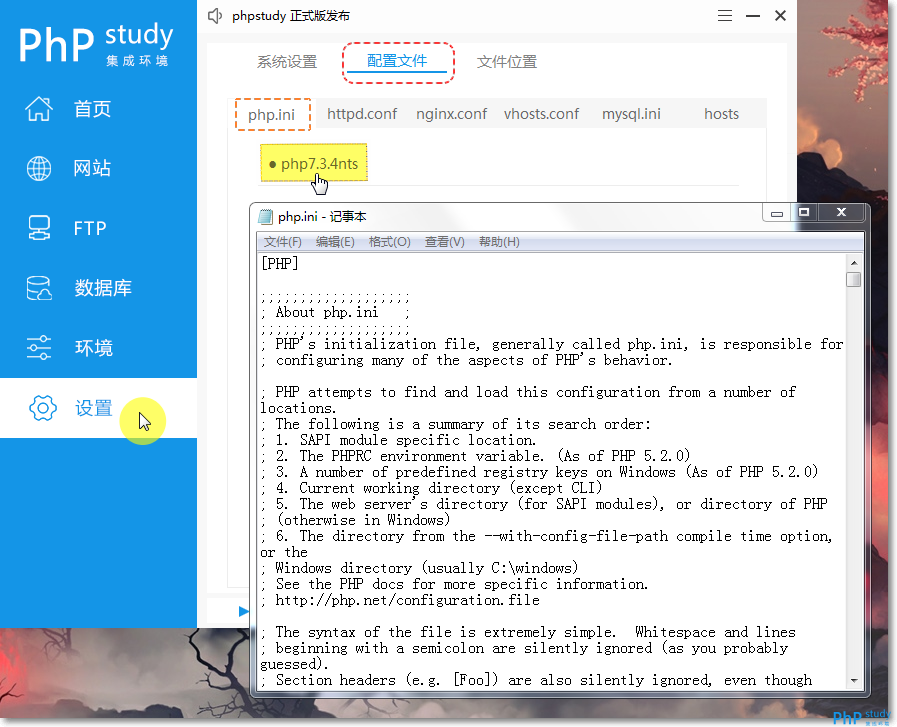
## ****php配置文件****

◆ php配置文件是: php.ini

◆ 为了安全, php.ini所有配置项并未全部开启

◆ php以扩展以模块的方式,在环境初始化时自动加载

◆ 允许用户根据项目需求, 开启对应的php扩展



## ****扩展开启与关闭****

下面以pdo-mysql扩展为例,演示php扩展的开启与关闭

◆ 打开php5.5.9配置文件, 搜索: **pdo\_mysql**

 回到页面中, 在phpinfo()返回的结果中搜索,发现扩展是启用的

◆ php.ini中, 将该扩展前面添加一个分号(注释) 并重启Apache

;extension=php\_pdo\_mysql.dll

◆ 打开demo.io,在phpinfo中查询pdo\_mysql已经找不到了

◆ 再将php.ini,删除php\_mysql扩展模块前的分号, 并重启Apache

◆ 此时刷新页面, 再次查看,会发了该项扩展已经开启了

FTP

FTP协议包括两个组成部分，其一为FTP服务器，其二为FTP客户端。其中FTP服务器用来存储文件，用户可以使用FTP客户端通过FTP协议访问位于FTP服务器上的资源。

FTP协议使用TCP端口中的 20和21这两个端口，其中20用于传输数据，21用于传输控制信息。

[FTP]->选择左上角[创建FTP]

◆ 创建FTP帐号非常简单, 只需要设置用户名, 密码,以及管理的目录即可

◆ 以demo.io项目为例,为它的目录WWW/demo创建FTP帐号

◆ demo.io: 用户名demo,密码123456

## ****测试FTP****

◆ 开启[FTP]

◆ 打开任何一款FTP软件, 如FileZilla

◆ 在快速连接文本框中依次输出主机名, 用户名,密码,端口号等信息

◆ 点击连接, 在服务器浏览窗口中, 如果看到demo目录下内容,则连接成功

## **phpMyAdmin是什么**

phpMyAdmin是使用php语言开发的,Web版的MySQL数据库管理软件

PHP 文件通常包含 HTML 标签和一些 PHP 脚本代码。

# PHP 变量

<?php $x=5;

$y=6;

$z=$x+$y;

echo $z; ?>

* 变量以 $ 符号开始，后面跟着变量的名称
* 变量名必须以字母或者下划线字符开始
* 变量名只能包含字母、数字以及下划线（A-z、0-9 和 \_ ）
* 变量名不能包含空格
* 变量名是区分大小写的（$y 和 $Y 是两个不同的变量）

全局变量可以被脚本中的任何部分访问，要在一个函数中访问一个全局变量，需要使用 global 关键字。

在 PHP 函数内部声明的变量是局部变量，仅能在函数内部访问：

Global用法

global $x;

$x=5; // 全局变量

function myTest()

{ global $x;

需要在外部用global声明，同时在函数内部进行声明，

在外部不用global则外部的就是不是全局变量，无法在函数内部使用：

PHP 将所有全局变量存储在一个名为 $GLOBALS[*index*] 的数组中。 index 保存变量的名称。这个数组可以在函数内部访问，也可以直接用来更新全局变量。

function myTest() { $GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y']; }

## Static 作用域

当一个函数完成时，它的所有变量通常都会被删除.然而，有时候您希望某个局部变量不要被删除。

要做到这一点，请在您第一次声明变量时使用 **static** 关键字：

static $x=0; echo $x; $x++; echo PHP\_EOL; // 换行符

## PHP echo 和 print 语句

echo 和 print 区别:

* echo - 可以输出一个或多个字符串
* print - 只允许输出一个字符串，返回值总为 1

**提示：**echo 输出的速度比 print 快， echo 没有返回值，print有返回值1。

# PHP EOF(定义一个字符串的方法。) 使用说明

位于开始标记和结束标记之间的变量可以被正常解析，但是函数则不可以,变量不需要用连接符 **.** 或 **,** 来拼接

* 1.PHP 定界符 **EOF** 的作用就是按照原样，包括换行格式什么的，输出在其内部的东西；
* 2.在 PHP 定界符 **EOF** 中的任何特殊字符都不需要转义；
* EOF 中是会解析 html 格式内容的，并且在双引号内的内容也有转义效果(会运行转义符)

<?php  
$name="runoob";  
$a= *<<<EOF  
        "abc"$name  
        "123"  
EOF*;  
*// 结束需要独立一行且前后不能空格*  
echo $a;  
?>

## PHP 字符串

一个字符串是一串字符的序列，就像 "Hello world!"。

## PHP 布尔型(布尔型通常用于条件判断)

布尔型可以是 TRUE 或 FALSE。

$x=true;  
$y=false;

## PHP 对象

你必须使用class关键字声明类对象

## PHP NULL 值

NULL 值表示变量没有值

可以通过设置变量值为 NULL 来清空变量数据：

## PHP 资源类型(常见资源数据类型有打开文件、数据库连接、图形画布区域等。)

使用 **get\_resource\_type()** 函数可以返回资源（resource）类型：

get\_resource\_type(resource $handle): string

**var\_dump**

此函数显示关于一个或多个表达式的结构信息，包括表达式的类型与值。数组将递归展开值，通过缩进显示其结构。

**5.var\_dump 和 print\_r 的区别**

var\_dump 返回表达式的类型与值而 print\_r 仅返回结果，相比调试代码使用 var\_dump 更便于阅读。

# PHP 类型比较

* 松散比较：使用两个等号 **==** 比较，只比较值，不比较类型。
* 严格比较：用三个等号 **===** 比较，除了比较值，也比较类型。

"42" 是一个字符串而 42 是一个整数。**FALSE** 是一个布尔值而 **"FALSE"** 是一个字符串。

## PHP 常量(常量名不需要加 $ 修饰符)

1. **不变性**: 常量一旦定义，其值不能改变。
2. **全局作用域**: 常量在定义后，可以在整个脚本的任何地方使用，无需使用 global 关键字。
3. **数据类型**: 常量的值可以是标量数据类型（如布尔值、整数、浮点数、字符串）或数组（PHP 7 及以上版本）。
4. **区分大小写**: 常量名称默认是区分大小写的。如果需要定义大小写不敏感的常量，可以在 define() 函数的第三个参数设置为 true。

bool define ( string $name , mixed $value [, bool $case\_insensitive = false ] )

**自 PHP 7.3.0 开始，定义不区分大小写的常量已被弃用。从 PHP 8.0.0 开始，只有 false 是可接受的值，传递 true 将产生一个警告。**

define("GREETING", "欢迎访问 Runoob.com");

echo GREETING; // 输出 "欢迎访问 Runoob.com"

echo '<br>';

echo greeting; // 输出 "greeting"，但是有警告

## 使用 const 关键字

const CONSTANT\_NAME = "value";

## 预定义常量

PHP 提供了一些预定义常量，可以在脚本中直接使用。这些常量通常用于获取 PHP 的配置信息、版本信息等。常见的预定义常量有：

* PHP\_VERSION：当前 PHP 解析器的版本。
* PHP\_OS：服务器的操作系统。
* PHP\_INT\_MAX：最大的整数值。
* E\_ERROR、E\_WARNING、E\_PARSE 等：错误报告级别。

# PHP 字符串变量

## PHP 并置运算符

在 PHP 中，只有一个字符串运算符。

并置运算符 (.) 用于把两个字符串值连接起来。

## PHP strlen() 函数

有时知道字符串值的长度是很有用的。(在循环中，我们需要在字符串中的最后一个字符之后结束循环)

strlen() 函数返回字符串的长度（字节数）

## PHP strpos() 函数

strpos() 函数用于在字符串内查找一个字符或一段指定的文本。

如果在字符串中找到匹配，该函数会返回第一个匹配的字符位置。如果未找到匹配，则返回 FALSE。

echo strpos("Hello world!","world");//字符串中第一个字符的位置是 0，而不是 1

# PHP 运算符

PHP7+ 版本新增整除运算符 **intdiv()**，该函数返回值为第一个参数除于第二个参数的值并取整（向下取整）

~var\_dump(intdiv(10, 3));

|  |  |  |
| --- | --- | --- |
| ++ x | 预递增 | x 加 1，然后返回 x |
| x ++ | 后递增 | 返回 x，然后 x 加 1 |
| -- x | 预递减 | x 减 1，然后返回 x |
| x -- | 后递减 | 返回 x，然后 x 减 1 |

## PHP 赋值运算符

|  |  |  |
| --- | --- | --- |
| **运算符** | **等同于** | **描述** |
| x = y | x = y | 左操作数被设置为右侧表达式的值 |
| x += y | x = x + y | 加 |
| x -= y | x = x - y | 减 |
| x \*= y | x = x \* y | 乘 |
| x /= y | x = x / y | 除 |
| x %= y | x = x % y | 模（除法的余数） |
| a .= b | a = a . b | 连接两个字符串 |

## 三元运算符

(expr1) ? (expr2) : (expr3)

对 expr1 求值为 TRUE 时的值为 expr2，在 expr1 求值为 FALSE 时的值为 expr3。

自 PHP 5.3 起，可以省略三元运算符中间那部分。表达式 expr1 ?: expr3 在 expr1 求值为 TRUE 时返回 expr1，否则返回 expr3。

$username = $\_GET['user'] ?: 'nobody';

// 如果 $\_GET['user'] 不存在返回 'nobody'，否则返回 $\_GET['user'] 的值 $username = $\_GET['user'] ?? 'nobody';

## 组合比较符(PHP7+)

$c = $a <=> $b;

解析如下：

* 如果**$a > $b**, 则 **$c** 的值为 **1**。
* 如果 **$a == $b**, 则 **$c** 的值为 **0**。
* 如果 **$a < $b**, 则 **$c** 的值为 **-1**。

“=”的表示

一个等号 **=** 用于赋值，如：

两个等号 **==** 用于比较，值相等，类型不一定相等，如：

5==5 // 返回 true5=="5" // 返回 true

三个等号 **===** 用于绝对比较，即类型与值都要相等，如：

5===5 // 返回 true5==="5" // 返回 false

运算符的优先级口诀：括（号）、单（操作数）、算（术）、移（位）、关（系）；位（运算符）、逻（辑）、条（件）、赋（值）、逗（号）。

$c = &$a;

c 是 a 的引用,也就是说 c 和 a 是同一个变量了。

& 是引用的意思，php 没有指针的概念。

自 PHP8 开始，三元运算符不再具有顺序。

意思是如果不使用括号包裹，程序会报错:

$var = true ? 1 : false ? 2 : 3;//错误//

// $var = (true ? 1 : false) ? 2 : 3;可以执行

$t=date("H");

用PHP获取系统时间时，时间比当前时间少 8 个小时。原因是 PHP.ini 中没有设置 timezone 时，PHP 是使用的 UTC 时间，所以在中国时间要少 8 小时。因此你的时间都要 -8，所以代码是没问题的。

解决办法：

1.在PHP.ini文件中修改设置。

[Date]; Defines the default timezone used by the date functions

date.timezone = Asia/Shanghai

简化写法

[$name, $age] = ['Jack', null]; // 解构赋值

// 普通写法if ($name == 'Jack') {

$age = 20;}

// 免 if 写法

$name == 'Jack' and $age = 20;

[$a, $b] = [null, null];

$name == 'Jack' and ([$a, $b] = ['value', 'value']); // 多变量赋值写法

# PHP 数组\

自 5.4 起可以使用短数组定义语法，用 **[]** 替代 **array()** 。

$cars=array("Volvo","BMW","Toyota");

$cars=["Volvo","BMW","Toyota"];

人工分配 ID 键：

$cars[0]="Volvo";  
$cars[1]="BMW";  
$cars[2]="Toyota";

## 获取数组的长度 - count() 函数echo count($cars);

## 遍历数值数组

遍历并打印数值数组中的所有值，您可以使用 for 循环，如下所示：

$cars=array("Volvo","BMW","Toyota");

$arrlength=count($cars); //重要//

for($x=0;$x<$arrlength;$x++)

{

echo $cars[$x];

echo "<br>"; }

## PHP 关联数组

关联数组是使用您分配给数组的指定的键的数组。对**关联数组**进行升序和降序排列应使用

* [asort()](https://www.runoob.com/php/func-array-asort.html" \t "https://www.runoob.com/php/_blank) - 根据关联数组的值，对数组进行升序排列
* [ksort()](https://www.runoob.com/php/func-array-ksort.html" \t "https://www.runoob.com/php/_blank) - 根据关联数组的键，对数组进行升序排列

这里有两种创建关联数组的方法：

$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

## 遍历关联数组

遍历并打印关联数组中的所有值，您可以使用 foreach (foreach 仅能够应用于数组和对象)循环，如下所示：

foreach($age as $x=>$x\_value)

//将$age数组中每一对键值赋值给$x(键),$x-value(值);

# PHP 数组排序

在本章中，我们将一一介绍下列 PHP 数组排序函数：

* sort() - 对数组进行升序排列(遇到字母按字母排序)
* rsort() - 对数组进行降序排列

# PHP 超级全局变量

## PHP $GLOBALS

$GLOBALS 是PHP的一个超级全局变量组，在一个PHP脚本的全部作用域中都可以访问。

$x = 75; $y = 25;

function addition() { $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];

addition();//调用函数,计算//

echo $z;

## PHP $\_SERVER

## $\_SERVER 是一个包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations) \

## PHP $\_REQUEST

PHP $\_REQUEST 用于收集HTML表单提交的数据。

## 当用户通过点击 "Submit" 按钮提交表单数据时, 表单数据将发送至<form>标签中 action 属性中指定的脚本文件。 在这个实例中，我们指定文件来处理表单数据。如果你希望其他的PHP文件来处理该数据，你可以修改该指定的脚本文件名

<html>

<body>

<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">

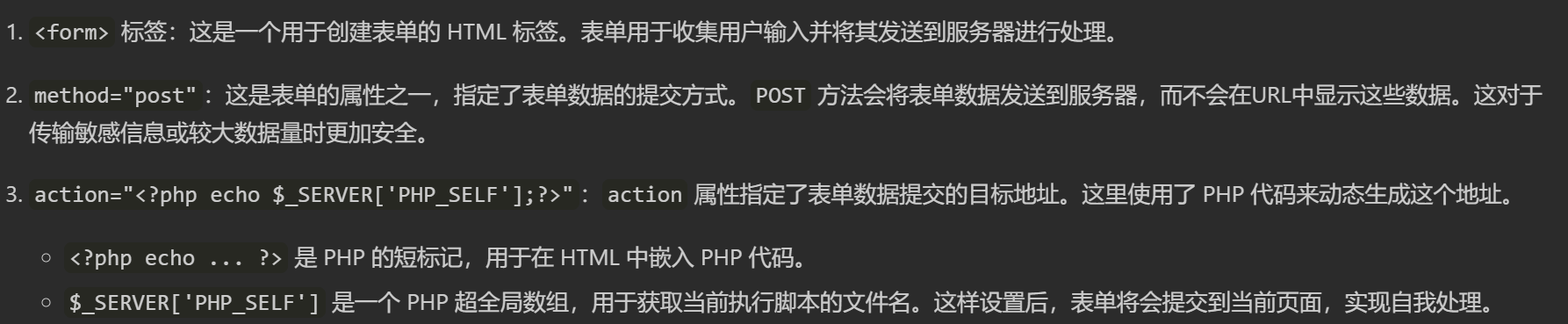
Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php $name = $\_REQUEST['fname']; echo $name; ?>

</body> </html>



## PHP $\_GET

PHP $\_GET 同样被广泛应用于收集表单数据，在HTML form标签的指定该属性："method="get"。

$\_GET 也可以收集URL中发送的数据。

<a href="test\_get.php?subject=PHP&web=runoob.com">Test $GET</a>

当用户点击链接 "Test $GET", 参数 "subject" 和 "web" 将发送至"test\_get.php",你可以在 "test\_get.php" 文件中使用 $\_GET 变量来获取这些数据。

下面显示了 "test\_get.php" 文件的代码:

echo "Study " . $\_GET['subject'] . " @ " . $\_GET['web'];

获取到的数据为:studyPHP@runoob.com

# PHP 循环 - While 循环

## do...while 语句do...while 语句会至少执行一次代码，然后检查条件，只要条件成立，就会重复进行循环。

do{要执行的代码;}while (条件);

$i=1;do{

$i++;

echo "The number is " . $i . "<br>";}while ($i<=5);?>

首先设置变量 *i* 的值为 1 ($i=1;)。

然后，开始 do...while 循环。循环将变量 *i* 的值递增 1，然后输出。先检查条件（*i* 小于或者等于 5），只要 *i* 小于或者等于 5，循环将继续运行：

foreach 循环foreach 循环用于遍历数组。foreach ($array as $key => $value)

使用for循环，实现冒泡排序：

<?php

$arr = array(5,3,6,2,8,10);for($i = count($arr)-1;$i>=0;$i--){

for($j = 0 ; $j < $i ; $j++){

if($arr[$j+1] > $arr[$j] ){

$aa = $arr[$j+1];

$arr[$j+1] = $arr[$j];

$arr[$j] = $aa;

}

}}

print\_r($arr);?>

# PHP 函数

函数是通过调用函数来执行的。

PHP 函数准则：

* 函数的名称应该提示出它的功能
* 函数名称以字母或下划线开头（不能以数字开头）

**含参函数**

function writeName($fname,$punctuation)

{ echo $fname . " Refsnes" . $punctuation . "<br>"; }

echo "My name is ";

writeName("Kai Jim",".");

输出:My name is /Kai Jim/ Refsnes/./

运行含返回值函数

function add($x,$y)

{ $total=$x+$y; return $total; }

echo "1 + 16 = " . add(1,16);

PHP动态调用

<?php  
class Foo//定义’类’  
{function Variable()  
 { $name = 'Bar';  
    $this->$name(); *// $this为动态关键字,调用 Bar() 方法* }  
  function Bar()  {*//定义bar()方法*  
        echo "This is Bar";}  
}$foo = new Foo();*//定义新的类”foo”,使用new关键字定义*  
$funcname = "Variable";  
$foo->$funcname();  *// 动态调用 $foo->Variable()方法*

# PHP 魔术常量

PHP 向它运行的任何脚本提供了大量的预定义常量。

## \_\_LINE\_\_(echo '这是第 " ' . \_\_LINE\_\_ . ' " 行';)文件中的当前行号。

\_\_FILE\_\_(echo '该文件位于 " ' . \_\_FILE\_\_ . ' " ';)文件的完整路径和文件名。如果用在被包含文件中，则返回被包含的文件名。

\_\_DIR\_\_(echo '该文件位于 " ' . \_\_DIR\_\_ . ' " ';)文件所在的目录。如果用在被包括文件中，则返回被包括的文件所在的目录。

## \_\_FUNCTION\_\_(echo '函数名为：' . \_\_FUNCTION\_\_ ;)

## \_\_CLASS\_\_(echo '类名为：' . \_\_CLASS\_\_ . "<br>";)

## \_\_TRAIT\_\_(代码复用)???

## \_\_METHOD\_\_(echo '函数名为：' . \_\_METHOD\_\_ ;)类的方法名

\_\_NAMESPACE\_\_(namespace MyProject; echo '命名空间为："', \_\_NAMESPACE\_\_, '"'; // 输出 "MyProject")当前命名空间的名称

## **[方法（method）和函数（function）的区别](https://blog.csdn.net/ScarletMeCarzy/article/details/108048152" \t "https://cn.bing.com/_blank)**

****1.函数****是独立的代码块，可以接收参数并返回结果，所有数据都是显式传递的。

def add(a, b):

return a + b

在这个例子中，add函数接收两个参数a和b，并返回它们的和。

****2.方法****是与对象关联的函数，可以隐式访问和操作类的属性和方法。

class Calculator:

def add(self, a, b):

return a + b

calc = Calculator()

result = calc.add(2, 3)

在这个例子中，add方法属于Calculator类，并且可以访问类的属性和方法。

# PHP 命名空间(namespace)

PHP 命名空间可以解决以下两类问题：

1. 用户编写的代码与PHP内部的类/函数/常量或第三方类/函数/常量之间的名字冲突。
2. 为很长的标识符名称(通常是为了缓解第一类问题而定义的)创建一个别名（或简短）的名称，提高源代码的可读性。

命名空间通过关键字namespace 来声明。如果一个文件中包含命名空间，它必须在其它所有代码之前声明命名空间。

<?phpnamespace MyProject {

const CONNECT\_OK = 1;

class Connection { /\* ... \*/ }

function connect() { /\* ... \*/ }}

全局命名空间代码必须用一个不带名称的 namespace 语句加上大括号括起来

namespace { // 全局代码

session\_start();

$a = MyProject\connect();

echo MyProject\Connection::start();}

在声明命名空间之前唯一合法的代码是用于定义源文件编码方式的 declare 语句declare(encoding='UTF-8'); //定义多个命名空间和不包含在命名空间中的代码

# PHP 面向对象

它将程序中的数据和操作数据的方法封装在一起，形成"对象"，并通过对象之间的交互和消息传递来完成程序的功能。面向对象编程强调数据的封装、继承、多态和动态绑定等特性，使得程序具有更好的可扩展性、可维护性和可重用性。

**对象的主要三个特性：**

1.对象的行为：对象可以执行的操作，比如：开灯，关灯就是行为。

2.对象的形态：对对象不同的行为是如何响应的，比如：颜色，尺寸，外型。

3.对象的表示：对象的表示就相当于身份证，具体区分在相同的行为与状态下有什么不同。

**面向对象编程的三个主要特性：**

1.封装（Encapsulation）：指将对象的属性和方法封装在一起，使得外部无法直接访问和修改对象的内部状态。通过使用访问控制修饰符（public、private、protected）来限制属性和方法的访问权限，从而实现封装。

2.继承（Inheritance）：指可以创建一个新的类，该类继承了父类的属性和方法，并且可以添加自己的属性和方法。通过继承，可以避免重复编写相似的代码，并且可以实现代码的重用。

3.多态（Polymorphism）：指可以使用一个父类类型的变量来引用不同子类类型的对象，从而实现对不同对象的统一操作。多态可以使得代码更加灵活，具有更好的可扩展性和可维护性。在 PHP 中，多态可以通过实现接口（interface）和使用抽象类（abstract class）来实现。