函数内访问全局变量需要 global 关键字或者使用 $GLOBALS[index] 数组， index 保存变量的名称。

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

PHP var\_dump() 函数返回变量的数据类型和值

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。

设置常量，使用 define() 函数，函数语法如下：

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

该函数有三个参数:

name：必选参数，常量名称，即标志符。

value：必选参数，常量的值。

case\_insensitive ：可选参数，如果设置为 TRUE，该常量则大小写不敏感。默认是大小写敏感的。

并置运算符 (.) 用于把两个字符串值连接起来。

strlen() 函数返回字符串的长度（字节数）。

strpos() 函数用于在字符串内查找一个字符或一段指定的文本。字符串中第一个字符的位置是 0，而不是 1

* sort() - 对数组进行升序排列
* rsort() - 对数组进行降序排列
* asort() - 根据关联数组的值，对数组进行升序排列
* ksort() - 根据关联数组的键，对数组进行升序排列
* arsort() - 根据关联数组的值，对数组进行降序排列
* krsort() - 根据关联数组的键，对数组进行降序排列

**面向对象内容**

* **类** − 定义了一件事物的抽象特点。类的定义包含了数据的形式以及对数据的操作。
* **对象** − 是类的实例。
* **成员变量** − 定义在类内部的变量。该变量的值对外是不可见的，但是可以通过成员函数访问，在类被实例化为对象后，该变量即可称为对象的属性。
* **成员函数** − 定义在类的内部，可用于访问对象的数据。
* **继承** − 继承性是子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。
* **父类** − 一个类被其他类继承，可将该类称为父类，或基类，或超类。
* **子类** − 一个类继承其他类称为子类，也可称为派生类。
* **多态** − 多态性是指相同的函数或方法可作用于多种类型的对象上并获得不同的结果。不同的对象，收到同一消息可以产生不同的结果，这种现象称为多态性。
* **重载** − 简单说，就是函数或者方法有同样的名称，但是参数列表不相同的情形，这样的同名不同参数的函数或者方法之间，互相称之为重载函数或者方法。
* **抽象性** − 抽象性是指将具有一致的数据结构（属性）和行为（操作）的对象抽象成类。一个类就是这样一种抽象，它反映了与应用有关的重要性质，而忽略其他一些无关内容。任何类的划分都是主观的，但必须与具体的应用有关。
* **封装** − 封装是指将现实世界中存在的某个客体的属性与行为绑定在一起，并放置在一个逻辑单元内。
* **构造函数** − 主要用来在创建对象时初始化对象， 即为对象成员变量赋初始值，总与new运算符一起使用在创建对象的语句中。
* **析构函数** − 析构函数(destructor) 与构造函数相反，当对象结束其生命周期时（例如对象所在的函数已调用完毕），系统自动执行析构函数。析构函数往往用来做"清理善后" 的工作（例如在建立对象时用new开辟了一片内存空间，应在退出前在析构函数中用delete释放）。

表单

**$\_GET** 变量接受所有以 **get** 方式发送的请求，及浏览器地址栏中的 **?** 之后的内容。

**$\_POS**T 变量接受所有以 post 方式发送的请求，例如，一个 form 以 **method=post** 提交，提交后 php 会处理 post 过来的全部变量。

**$\_REQUEST** 支持两种方式发送过来的请求，即 **post** 和 **get** 它都可以接受，显示不显示要看传递方法，get 会显示在 url 中（有字符数限制），post 不会在 url 中显示，可以传递任意多的数据（只要服务器支持）。

PHP date() 函数可把时间戳格式化为可读性更好的日期和时间。

**include 和 require 的区别**

* require 一般放在 PHP 文件的最前面，程序在执行前就会先导入要引用的文件；
* include 一般放在程序的流程控制中，当程序执行时碰到才会引用，简化程序的执行流程。
* require 引入的文件有错误时，执行会中断，并返回一个致命错误；
* include 引入的文件有错误时，会继续执行，并返回一个警告。

setcookie() 函数用于设置 cookie。

setcookie() 函数必须位于 <html> 标签之前。

Session

在您把用户信息存储到 PHP session 中之前，首先必须启动会话。

session\_start() 函数必须位于 <html> 标签之前：

存储和取回 session 变量的正确方法是使用 PHP $\_SESSION 变量

删除某些 session 数据，可以使用 unset() 或 session\_destroy() 函数。

unset() 函数用于释放指定的 session 变量

PHP mail() 函数用于从脚本中发送电子邮件

适当的处理异常代码应该包括：

1. Try - 使用异常的函数应该位于 "try" 代码块内。如果没有触发异常，则代码将照常继续执行。但是如果异常被触发，会抛出一个异常。
2. Throw - 里规定如何触发异常。每一个 "throw" 必须对应至少一个 "catch"。
3. Catch - "catch" 代码块会捕获异常，并创建一个包含异常信息的对象。

**异常的规则**

* 需要进行异常处理的代码应该放入 try 代码块内，以便捕获潜在的异常。
* 每个 try 或 throw 代码块必须至少拥有一个对应的 catch 代码块。
* 使用多个 catch 代码块可以捕获不同种类的异常。
* 可以在 try 代码块内的 catch 代码块中抛出（再次抛出）异常。

简而言之：如果抛出了异常，就必须捕获它。