# 简介

生成器提供了一种更容易的方法来实现简单的[对象迭代](mk:@MSITStore:C:\\Users\\张卫航\\Desktop\\php_enhanced_zh.chm::/res/language.oop5.iterations.html)，相比较定义类实现 **Iterator** 接口的方式，性能开销和复杂性大大降低。

生成器函数时会返回一个内部的 **Generator** 类的对象。 该对象实现了 **Iterator** 接口，基本上和仅向前的迭代器一样， 它提供的方法可以操控生成器的状态，包括发送值、返回值。

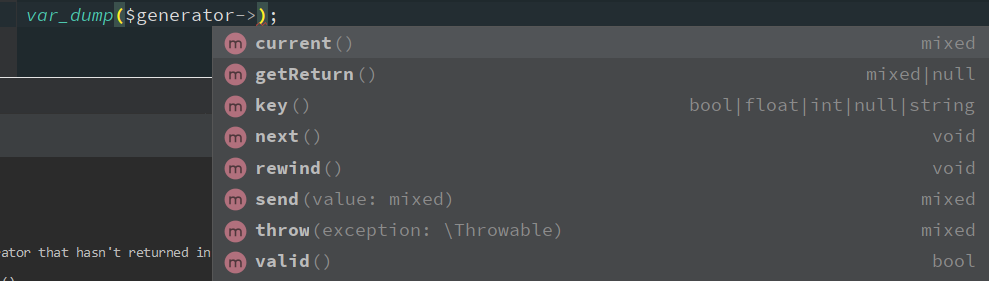
# 语法

生成器函数看起来像普通函数——不同的是普通函数返回一个值，而生成器可以 [yield](mk:@MSITStore:C:\\Users\\张卫航\\Desktop\\php_enhanced_zh.chm::/res/language.generators.syntax.html" \l "control-structures.yield) 生成多个想要的值。 任何包含 [yield](mk:@MSITStore:C:\\Users\\张卫航\\Desktop\\php_enhanced_zh.chm::/res/language.generators.syntax.html" \l "control-structures.yield) 的函数都是一个生成器函数。

当一个生成器被调用的时候，它返回一个可以被遍历的对象.当你遍历这个对象的时候(例如通过一个[foreach](mk:@MSITStore:C:\\Users\\张卫航\\Desktop\\php_enhanced_zh.chm::/res/control-structures.foreach.html)循环)，PHP 将会在每次需要值的时候调用对象的遍历方法，并在产生一个值之后保存生成器的状态，这样它就可以在需要产生下一个值的时候恢复调用状态。

一旦不再需要产生更多的值，生成器可以简单退出，而调用生成器的代码还可以继续执行，就像一个数组已经被遍历完了。

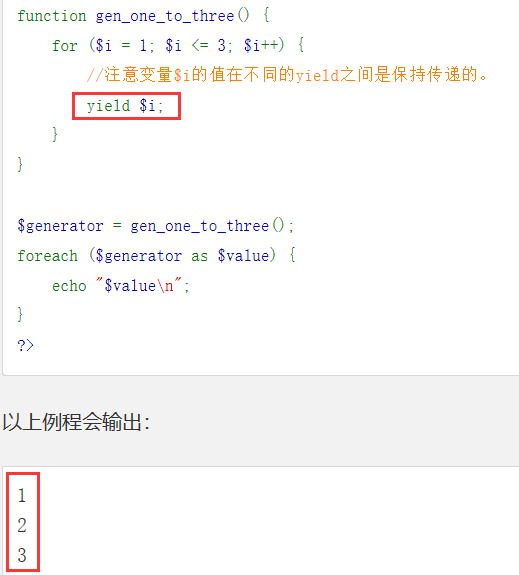
## Generator对象的方法



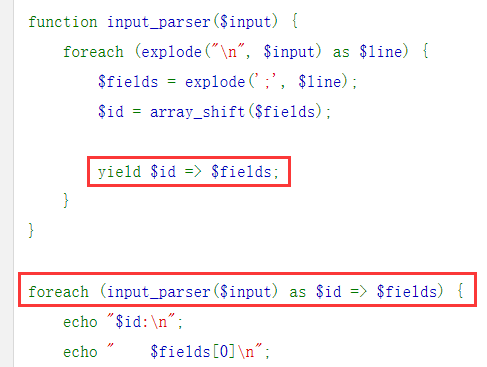
## yield关键字

生成器函数的核心是**yield**关键字。它最简单的调用形式看起来像一个return申明，不同之处在于普通return会返回值并终止函数的执行，而yield会返回一个值给循环调用此生成器的代码并且只是暂停执行生成器函数。

### 一个简单的生成值的例子



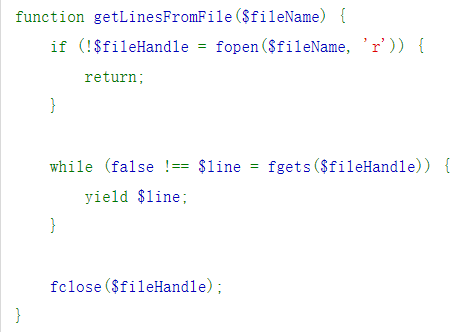
### 生成一个键值对



# 生成器与 Iterator 对象的比较

生成器最主要的优点是简洁。和实现一个 **Iterator** 类相较而言， 同样的功能，用生成器可以编写更少的代码，可读性也更强。

举例，下面的类和函数是相等的：



比较下...



