协程，又称微线程和纤程等

协程是用户空间线程，所以需要用户自己去做调度，操作系统其存在一无所知。

其实用协程来做的东西，用线程或进程通常也是一样可以做的，

减少锁使用，简化通信的操作。

提高并发：主要应用在IO密集型应用中。

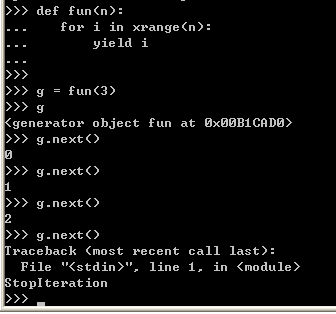
Python的协程实现 yield 和greenlet。

yield 自己主动退出，被人来调用他

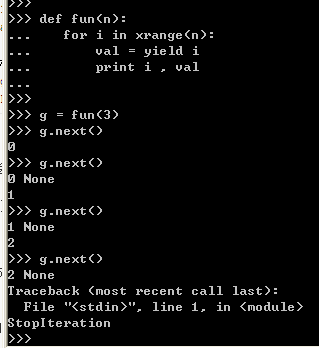
greenlet 他主动退出，调用其他，

在Python中，yield，生成器（generator），含有yield语句的函数都是可重入的函数，它记住上一次返回时在函数体中的位置。

next()——直到下一个yield表达式处停止，结束时抛出异常

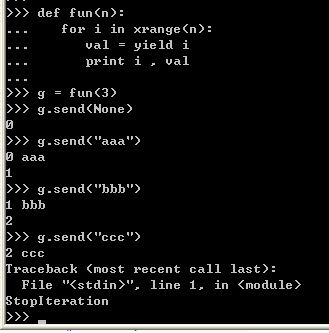


yield 返回值



**send(msg)**

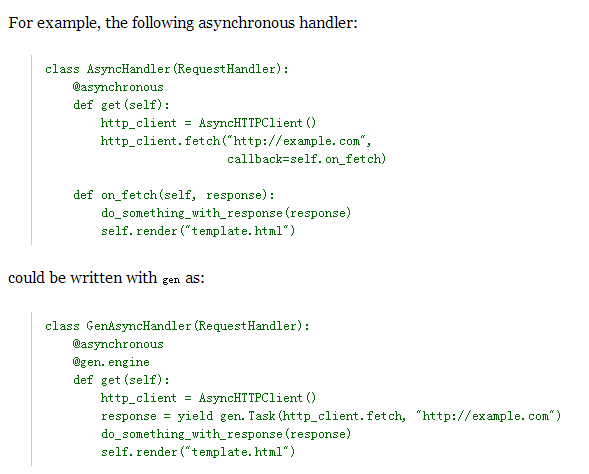
其实next()和send()在一定意义上作用是相似的，区别是send()可以传递yield表达式的值进去，而next()只能传递None进去。



Tornado 对yield的封装

[**tornado.gen** — Simplify asynchronous code](http://www.tornadoweb.org/documentation/gen.html)

* [**tornado.ioloop** — Main event loop](http://www.tornadoweb.org/documentation/ioloop.html)
* [**tornado.iostream** — Convenient wrappers for non-blocking sockets](http://www.tornadoweb.org/documentation/iostream.html)



原理

@asynchronous 简单设置一下变量表示是异步的

@ engine 有个循环run（）

不断调用 yielded = gen.send(next) yielded.start()..

那个异步函数http\_client.fetch 结束时会回调 engine里面的循环函数。

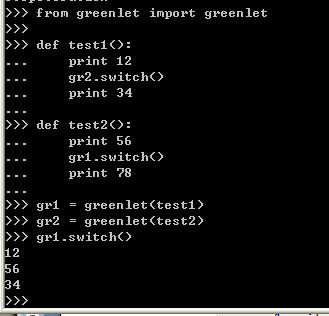
获取Gmdb数据的简单例子

底层封装，代码写起来还是异步代码（回调回调。。。）

Greenlet

用c语言封装可以左右切换的python运行是函数堆栈信息

自己调用其他函数，主动退出



### Gevent——[A coroutine-based network library for Python](http://www.baidu.com/link?url=iHcXGJqjJ4zBBpC8yDF8xDh8vibi1lt8EWoXr9kDLhu)

神器

from gevent import monkey

monkey.patch\_all()

写同步代码，实际上运行是异步的，

几个简单例子