[](tencent://AddContact/?fromId=50&fromSubId=1&subcmd=all&uin=3440037784)[](https://jq.qq.com/?_wv=1027&k=50Xp8mW)

扫码获取更多课程资料，开通直播上课权限（点击下方课程直播即可免费观看）

学神IT教育微信公众号 学神IT官方 QQ群

关注学神公众号有机会获得Python官方书籍

[](https://jq.qq.com/?_wv=1027&k=50Xp8mW)

# 

# 第十八章 Python协程

本节所讲内容：

18.1 协程的概念

18.2 Greenlet模块

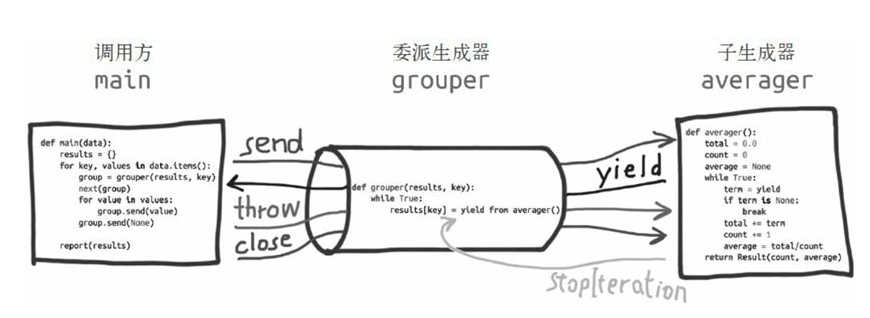
18.3 Gevent的使用

18.4 使用Gevent完成多任务图片下载

## 18.1 协程的概念

协程，又称微线程，纤程,也称为用户级线程，在不开辟线程的基础上完成多任务，也就是在单线程的情况下完成多任务，多个任务按照一定顺序交替执行 通俗理解只要在def里面只看到一个yield关键字，就表示它就是协程。

简单理解：协程是也是实现多任务的一种方式。



协程的优点：

1. 无需线程上下文切换的开销，协程避免了无意义的调度，由此可以提高性能（但也因此，程序员必须自己承担调度的责任，同时，协程也失去了标准线程使用多CPU的能力）；
2. 无需原子操作锁定及同步的开销；
3. 方便切换控制流，简化编程模型；
4. 高并发+高扩展性+低成本：一个CPU支持上万的协程都不是问题。所以很适合用于高并发处理。

线程的缺点：

1. 无法利用多核资源：协程的本质是个单线程,它不能同时将 单个CPU 的多个核用上,协程需要和进程配合才能运行在多CPU上.当然我们日常所编写的绝大部分应用都没有这个必要，除非是cpu密集型应用；

协程yield的代码实现

def work1():  
 while True:  
 print("----work1---")  
 yield  
 time.sleep(0.5)  
def work2():  
 while True:  
 print("----work2---")  
 yield  
 time.sleep(0.5)  
def main():  
 w1 = work1()  
 w2 = work2()  
 while True:  
 next(w1)  
 next(w2)  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

运行结果：

----work1---

----work2---

----work1---

----work2---

----work1---

----work2---

----work1---

----work2---

----work1---

----work2---

----work1---

----work2---

...省略...

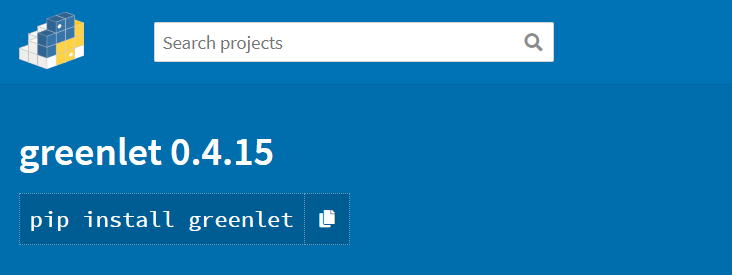
小结:

协程之间执行任务按照一定顺序交替执行。

## 18.2 Greenlet模块

### 18.2.1 greentlet的介绍

我们在工作过程中，为了更好使用协程来完成多任务，python中的greenlet模块对其封装，从而使得切换任务变的更加简单。



使用如下命令安装greenlet模块:pip3 install greenlet

代码展示：

import time  
import greenlet  
def work1():  
 for i in range(5):  
 print('work1...')  
 time.sleep(0.2)  
 *#切换到协程2里面执行对应的任务* g2.switch()  
*#任务2*def work2():  
 for i in range(5):  
 print('work2...')  
 time.sleep(0.2)  
 *#切换到第一个协程执行对应的任务* g1.switch()  
if \_\_name\_\_ == '\_\_main\_\_':  
 *#创建协程指定的对应任务* g1 = greenlet.greenlet(work1)  
 g2 = greenlet.greenlet(work2)  
 *#切换到第一个协程执行对应的任务* g1.switch()

运行结果如下：

work1...

work2...

work1...

work2...

work1...

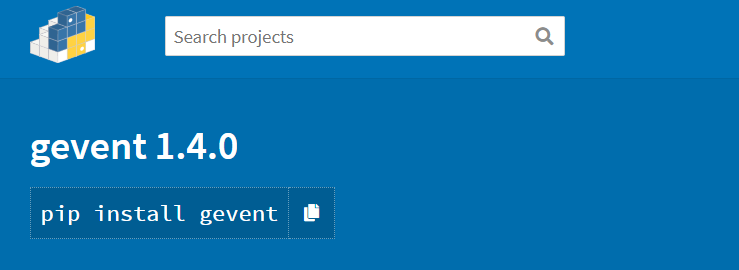
work2...

……

## 18.3 Gevent的使用

### 18.3.1 Gevent的介绍

Greenlet已经实现了协程，但是这个还要人工切换，这里介绍一个比Greenlet更强大而且能够自动切换任务的第三方库，那就是Gevent。



Gevent内部封装的Greenlet，其原理是当一个Greenlet遇到IO(指的是input output 输入输出，比如网络、文件操作等)操作时，就自动切换到其他的Greenlet，等到这个IO操作完成，再在适当的时候切换回来继续执行。

由于IO操作非常耗时，经常使程序处于等待状态，有了Gevent为我们自动切换协程，就保证总有Greenlet在运行，而不是等待IO。

安装：pip3 install gevent

1. gevent的使用

import gevent  
def work(n):  
 for i in range(n):  
 *# 获取当前协程* print(gevent.getcurrent(), i)  
g1 = gevent.spawn(work, 5)  
g2 = gevent.spawn(work, 5)  
g3 = gevent.spawn(work, 5)  
g1.join()  
g2.join()  
g3.join()

运行结果如下：

<Greenlet at 0x10e49f550: f(5)> 0

<Greenlet at 0x10e49f550: f(5)> 1

<Greenlet at 0x10e49f550: f(5)> 2

<Greenlet at 0x10e49f550: f(5)> 3

<Greenlet at 0x10e49f550: f(5)> 4

<Greenlet at 0x10e49f910: f(5)> 0

……

可以看到，3个greenlet是依次运行而不是交替运行

18.3.2 gevent切换执行

import gevent  
def work(n):  
 for i in range(n):  
 *# 获取当前协程* print(gevent.getcurrent().name, i)  
 gevent.sleep(1)  
g1 = gevent.spawn(work, 5)  
g2 = gevent.spawn(work, 5)  
g3 = gevent.spawn(work, 5)  
g1.join()  
g2.join()  
g3.join()

运行结果：

Greenlet-0 0

Greenlet-1 0

Greenlet-2 0

Greenlet-0 1

Greenlet-1 1

Greenlet-2 1

Greenlet-0 2

Greenlet-1 2

Greenlet-2 2

Greenlet-0 3

Greenlet-1 3

Greenlet-2 3

Greenlet-0 4

Greenlet-1 4

Greenlet-2 4

### 18.3.3 给程序打补丁

import gevent  
import time  
from gevent import monkey  
*# 打补丁，让gevent框架识别耗时操作，比如：time.sleep，网络请求延时*monkey.patch\_all()  
*# 任务1*def work1(num):  
 for i in range(num):  
 print("work1....")  
 time.sleep(0.2)  
 *# gevent.sleep(0.2)  
# 任务1*def work2(num):  
 for i in range(num):  
 print("work2....")  
 time.sleep(0.2)  
 *# gevent.sleep(0.2)*if \_\_name\_\_ == '\_\_main\_\_':  
 *# 创建协程指定对应的任务* g1 = gevent.spawn(work1, 3)  
 g2 = gevent.spawn(work2, 3)  
  
 *# 主线程等待协程执行完成以后程序再退出* g1.join()  
 g2.join()

运行结果

work1....

work2....

work1....

work2....

work1....

work2....

### 18.3.4 注意

当前程序是一个死循环并且还能有耗时操作，就不需要加上join方法了,因为程序需要一直运行不会退出

示例代码

import gevent  
import time  
from gevent import monkey  
*# 打补丁，让gevent框架识别耗时操作，比如：time.sleep，网络请求延时*monkey.patch\_all()  
*# 任务1*def work1(num):  
 for i in range(num):  
 print("work1....")  
 time.sleep(0.2)  
 *# gevent.sleep(0.2)  
# 任务1*def work2(num):  
 for i in range(num):  
 print("work2....")  
 time.sleep(0.2)  
 *# gevent.sleep(0.2)*if \_\_name\_\_ == '\_\_main\_\_':  
 *# 创建协程指定对应的任务* g1 = gevent.spawn(work1, 3)  
 g2 = gevent.spawn(work2, 3)  
 while True:  
 print("主线程中执行")  
 time.sleep(0.5)

运行结果如下:

主线程中执行

work1....

work2....

work1....

work2....

work1....

work2....

主线程中执行

主线程中执行

主线程中执行

## 18.3 使用Gevent完成多任务图片下载



多图片下载任务！

import gevent

import urllib.request *# 网络请求模块*

from gevent import monkey

*# 打补丁： 让gevent使用网络请求的耗时操作，让协程自动切换执行对应的下载任务*

monkey.patch\_all()

*# 根据图片地址下载对应的图片*

def download\_img(img\_url, img\_name):

try:

print(img\_url)

*# 根据图片地址打开网络资源数据*

response = urllib.request.urlopen(img\_url)

*# 创建文件把数据写入到指定文件里面*

with open(img\_name, "wb") as img\_file:

while True:

*# 读取网络图片数据*

img\_data = response.read(1024)

if img\_data:

*# 把数据写入到指定文件里面*

img\_file.write(img\_data)

else:

break

except Exception as e:

print("图片下载异常:", e)

else:

print("图片下载成功: %s" % img\_name)

if \_\_name\_\_ == '\_\_main\_\_':

*# 准备图片地址*

img\_url1 = 'http://p0.so.qhmsg.com/bdr/576\_\_/t013ee81b64eb53f6f5.jpg'

img\_url2 = "http://p2.so.qhimgs1.com/bdr/594\_\_/t017ec94ec006189032.jpg"

img\_url3 = "http://p3.so.qhmsg.com/bdr/864\_\_/t01f9daf42a666bb408.jpg"

*# 创建协程指派对应的任务*

g1 = gevent.spawn(download\_img, img\_url1, "1.jpg")

g2 = gevent.spawn(download\_img, img\_url2, "2.jpg")

g3 = gevent.spawn(download\_img, img\_url3, "3.jpg")

*# 主线程等待所有的协程执行完成以后程序再退出*

gevent.joinall([g1, g2, g3])

运行结果如下：

http://p0.so.qhmsg.com/bdr/576\_\_/t013ee81b64eb53f6f5.jpg

http://p2.so.qhimgs1.com/bdr/594\_\_/t017ec94ec006189032.jpg

http://p1.so.qhimgs1.com/bdr/594\_\_/t012c11259e883bbc8d.jpg

图片下载成功: 2.jpg

图片下载成功: 1.jpg

图片下载成功: 3.jpg

分析：

从上面的结果可以看到依次根据图片地址去下载，但是收到数据的先后顺序不一定与发送顺序相同，这也就体现出了异步，即不确定什么时候会收到数据，顺序不一定。



import gevent,requests*#pip install requests*from gevent import monkey  
from urllib import request  
*#打补丁，识别网路延迟*monkey.patch\_all()  
*#下载图片*def download\_img(num):  
 *#开启下载* print('start download')  
 *#图片的url地址* url = 'http://image.so.com/zj?ch=beauty&sn=150&listtype=new&temp=1'  
 *#模拟浏览器* headers={  
 'Referer': 'http://image.so.com/z?ch=beauty',  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.80 Safari/537.36'  
 }  
 *#模拟浏览器访问服务器发送内容* str\_data = '''h: beauty  
 sn: 30  
 listtype: new  
 temp: 1'''  
 *#简单的数据清洗* send\_data = {}  
 *#进行一行分割* for data in str\_data.splitlines():  
 line\_data = data.split(': ')*#返回一个列表* if len(line\_data) == 2:*#如果这个列表当中有两个数据* key,value = line\_data*# a,b = [1,2] #进行序列解包赋值！* if key and value:*#如果两者key和 value都有值 我就进行 搭建send\_data* send\_data[key] = value  
 *#end\_data = {'h':'beauty','sn':'30'}* send\_data['sn'] = eval(str(num)+'\*'+'30') *#eval（'3 \* 4'）#z字典的修改  
 #requests 这个方法进行网略请求，模拟刘燃气访问服务器返回结果* response = requests.get(url,headers=headers,params=send\_data)  
 *#json()方法 转换为python可操作对象{‘a" ;1}* json\_data = response.json()['list']  
 *#利用么酷的方法，序列解包赋值* for index,src in enumerate(json\_data):  
 *#获取图片url地址* image\_url = src['qhimg\_url']  
 try:  
 *#给定本地图片地址* image\_name = './360\_image/'+image\_url[-8:]  
 *#吧网路上的图片下载到本地* request.urlretrieve(url=image\_url,filename=image\_name)  
 except Exception as e:  
 print(e)  
 else:  
 *#format格式化* print('{} is download'.format(image\_name))  
 print('image is download')  
if \_\_name\_\_ == '\_\_main\_\_':  
 num = int(input('你想要爬取的图片的组:'))  
 *#列表推导式完成协程任务分发* gevent.joinall([gevent.spawn(download\_img,i)for i in range(1,num+1)])

总结：

18.1 协程的概念

18.2 greenlet模块

18.3 gevent的使用

18.4 使用gevent完成多任务图片下载