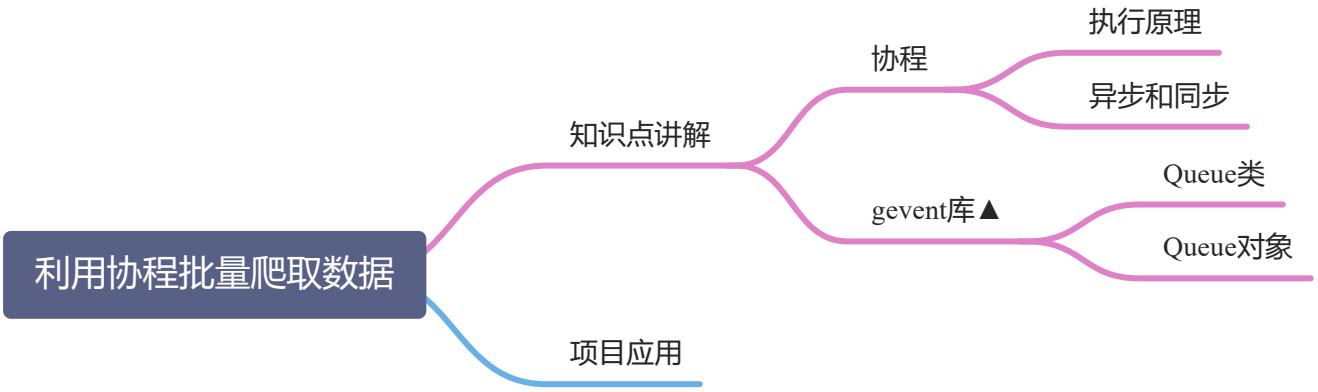
第11课 gevent库

一、课程结构导图



注：▲为重点知识点。

二、知识点讲解

2.1 协程

2.1.1 执行原理

计算机在执行一个任务的过程中，如果遇到等待，就先去执行其他的任务，当等待结束，再回来继续之前的任务。协程在任务间来回切的速度非常快速，就像多个任务被同时执行一样。协程可以通过gevent 库实现。

2.1.2 异步和同步

异步：在一个任务未完成时，就可以执行其他多个任务，彼此不受影响。

同步：一个任务结束才能启动下一个。

异步执行任务会比同步更加节省时间，因为它能减少不必要的等待。如果你需要对时间做优化，异步是一个很值得考虑的方案。

2.2 gevent 库

应用场景： gevent 库是用于实现多协程的库，实现程序异步进行，提高效率。例如：可以创建多个爬虫发起爬取，在等待网站响应的时候，进行其他网站的爬取，以节省时间。gevent 库是第三方库，本地使用前需要先安装，命令如下：

- window系统：pip install gevent
- Mac系统：pip3 install gevent

常用方法1： monkey.patch_all()

- 使用场景：程序需要以协作式运行，实现异步执行时，可以通过调用monkey.patch_all() 打补丁来实现。
- 注意：先给程序打上补丁，即在导入其他库和模块前，先把monkey模块导入进来，并运行monkey.patch_all()。

示例1：

```
from gevent import monkey # 从gevent库里导入monkey模块
monkey.patch_all()        # 调用patch_all() 实现异步执行
```

常用方法2： spawn()

- 功能：创建任务。
- 常用参数：spawn() 的第一个参数是执行任务的函数的函数名；当函数有参数时，第二个参数开始传入函数的参数，spawn() 会在后台自行将参数传给函数。
  - 注意第一个参数传函数名时不需要加上括号，加上括号是调用函数，传给spawn() 方法的是函数的返回值，而不是函数本身。
- 返回值：调用spawn() 返回一个Greenlet 对象（Greenlet是gevent库的greenlet模块中的一个类）。

示例2：

```
from gevent import monkey
monkey.patch_all()
import gevent
```

```
4
5 def crawler(param):         # 定义一个crawler()函数。
6     print(param)
7
8 params_list = ['a', 'b', 'c']
9 tasks_list = [ ]             # 创建空列表，存放任务
10
11 for param in params_list:   # 遍历params_list
12     task = gevent.spawn(crawler, param) # 调用spawn()函数创建任务，如果函数没有参数，传入函数名
13     tasks_list.append(task) # 添加任务到列表
```

常用方法3：joinall()

- 功能：执行任务。
- 常用参数：joinall() 的参数是任务列表。
- 返回值：调用spawn() 返回一个Greenlet 对象（Greenlet是gevent库的greenlet模块中的一个类）

示例3：

```
1 from gevent import monkey
2 monkey.patch_all()
3 import gevent
4
5 def crawler(param):
6     print(param)
7
8 params_list = ['a', 'b', 'c']
9 tasks_list = [ ]
10
11 for param in params_list:
12     task = gevent.spawn(crawler, param)
13     tasks_list.append(task)     # 添加任务到列表
14 gevent.joinall(tasks_list)     # 执行任务，调用函数并分别打印了a、b、c
```

2.2.1 Queue类

应用场景： 当需要执行的任务比较多的时候，使用queue模块中的Queue类把任务排成队列，依次执行，完成一个任务再进行下一个任务，保证多个爬虫几乎能够同时完成任务，而不用因为一些网站响应慢使得

等待时间变长。

调用语法：from gevent.queue import Queue

2.2.2 Queue对象

基本介绍：Queue对象是通过实例化调用Queue类之后创建的实例对象。

常用方法1：put_nowait()

- 功能：把值添加到队列中。
- 常用参数：put_nowait() 的参数是用于任务的值。

示例1：

```
from gevent import monkey
monkey.patch_all()
from gevent.queue import Queue

work = Queue()              # 创建Queue类的实例对象
values_list = ['小明','小红','小芳','小强'] # 需要添加到队列中的值所在的列表
for value in values_list:
    work.put_nowait(value)  # 把值添加到队列中

print(work)
# 结果：<Queue queue=deque(['小明', '小红', '小芳', '小强'])>
```

常用方法2：empty()

- 功能：判断队列中的任务是否为空。
- 返回值：调用empty()返回布尔值。当队列为空时，返回True；当队列不为空时，返回False。

示例2：

```
from gevent import monkey
monkey.patch_all()
from gevent.queue import Queue

work = Queue()
print(work.empty())     # 结果：True
values_list = ['小明','小红','小芳','小强']
```

```
for value in values_list:
    work.put_nowait(value)

print(work.empty())     # 结果：False
```

常用方法3：get_nowait()

- 功能：把值从队列中取出。
- 返回值：每次返回偏移量为0的值。

示例3：

```
from gevent import monkey
monkey.patch_all()
from gevent.queue import Queue

work = Queue()
values_list = ['小明','小红','小芳','小强']
for value in values_list:
    work.put_nowait(value)

value_0 = work.get_nowait() # 从队列中取出第0个索引值的值
print(value_0)              # 结果：小明
```

常用方法4：qsize()

- 功能：查看队列的长度。
- 常用参数：put_nowait() 的参数是用于任务的值。

示例4：

```
from gevent import monkey
monkey.patch_all()
from gevent.queue import Queue

work = Queue()
values_list = ['小明','小红','小芳','小强']
for value in values_list:
    work.put_nowait(value)

size = work.qsize()     # 查看队列中有多少个值
```

```
print(size)             # 结果：4
```

三、项目应用

项目代码：

```
from gevent import monkey
monkey.patch_all()      # 打补丁，调用patch_all() 实现异步执行
import gevent,time,requests
from gevent.queue import Queue

start = time.time()

url_list = ['https://www.baidu.com/','https://www.sina.com.cn/',
            'http://www.sohu.com/','https://www.qq.com/',
            'https://www.163.com/','http://www.iqiyi.com/',
            'https://www.tmall.com/','http://www.ifeng.com/']

work = Queue()      # 创建队列对象，并赋值给work。
for url in url_list:# 遍历url_list
    work.put_nowait(url)        # 用put_nowait()函数可以把网址都放进队列里。

def crawler():
    while not work.empty():     # 当队列不是空的时候，就执行下面的程序。
        url = work.get_nowait() # 用get_nowait()函数可以把队列里的网址都取出。
        r = requests.get(url)   # 用requests.get()函数抓取网址。
        print(url,work.qsize(),r.status_code)   # 打印网址、队列长度、抓取请求的状态码。

tasks_list  = [ ]   # 创建空的任务列表
for x in range(3):  # 相当于创建了3个爬虫
    task = gevent.spawn(crawler)    # 用gevent.spawn()函数创建执行crawler()函数的任务。
    tasks_list.append(task)         # 往任务列表添加任务。
gevent.joinall(tasks_list)          # 执行任务列表里的所有任务，即让爬虫开始爬取网站。
end = time.time()
print(end-start)    # 计算代码执行时间
```

代码解析：

- 第1、3、4行代码：调用相关的模块。

- 第6、28行代码：记录程序运行前后的时间，相减之后得出运行程序的总时长（即第29行）。
- 第13~15行代码：创建一个队列，并且往队列中添加值。
- 第17~21行代码：定义crawler函数，只要调用这个函数，它就会执行【从队列中取出网址】、【用requests.get()爬取网站】和【打印网址、队列长度、状态码】这三个任务。
- 第24行代码：创建3个协程，相当于创建3个爬虫。

注意：在2.2 gevent库的示例2和示例3中的第11行代码，和该代码的第24行本质上是一样的，都是创建3个协程。