13-健身的时候该怎么吃

目标：

用多协程爬取HI运动11个常见食物分类里的食物信息（包含食物名、热量、食物详情页面链接）

项目分析：

URL：https://food.hiyd.com/

采用爬虫的四步走战略：

获取数据→解析数据→提取数据→存储数据。

1.获取数据：

食物分类的网址规律

http:s://food.hiyd.com/list-数字-html?page=数字

注意：「菜肴」类的listu事按之前的数据，而是123

2.解析和提取数据：

看Elements可以发现：

所有食物信息都放在其对应的
...
标签里。每页食物记录里有20个食物，刚好对应上网页源代码里的20个
...
标签。

也就是说我们可以用find_all/find就能提取出
...
标签下的食物详情链接、名称和热量

3.存触数据：

在csv和openpyxl模块中任意选择使用其中一个模块来存储数据就能完成整个项目了

代码实现

```
# 导入所需的库和模块：
from gevent import monkey
import gevent,requests,bs4,openpyxl,time
from gevent.queue import Queue
from openpyxl import load_workbook,Workbook,worksheet
#让程序变成异步模式
```

```
monkey.patch_all()
# 创建队列对象，并赋值给work
work = Queue()
# 前3个分类的前3页的食物记录的网址：
url_1 = "https://food.hiyd.com/list-{type}-html?page={page}"
for x in range(1, 4):
    for y in range(1, 4):
        real_url = url_1.format(type=x, page=y)
        work.put_nowait(real_url)
# 通过两个for循环，能设置分类的数字和页数的数字
# 然后，把构造好的网址用put_nowait方法添加进队列里
# 第11个分类的前3页的食物记录的网址：
url_2 = "https://food.hiyd.com/list-132-html?page={page}"
for x in range(1, 4):
    real_url = url_2.format(page=x)
    work.put_nowait(real_url)
# 通过for循环，能设置第11个常见食物分类的食物的页数。
# 然后，把构造好的网址用put_nowait方法添加进队列里。
print(work)
# 打印队列
def crawler(job):# 定义crawler函数
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36'}
    # 添加请求头
    while not job.empty():
        # 当队列不是空的时候，就执行下面的程序
        url = job.get_nowait()
        # 用get_nowait()方法从队列里把刚刚放入的网址提取出来
        res = requests.get(url, headers=headers)
        # 用requests.get获取网页源代码
        bs_res = bs4.BeautifulSoup(res.text, 'html.parser')
        # 用BeautifulSoup解析网页源代码
        category = bs_res.find('b').text
        # 用find提取出<b>标签的内容,当前页面所属的分类
        foods = bs_res.find_all('li')
        # 用find_all提取出<li>标签的内容
        result_list = []
        # 创建空的list用来存储结果
        for food in foods:# 遍历foods
            food_name = food.find('a').find_all('div')[1].find('h3').text
            # 用find_all在<li>标签下，提取出第二个<div>标签中<h3>标签中的文本，也就是食物名称
            food_calorie = food.find('a').find_all('div')[1].find('p').text
            # 用find_all在<li>标签下，提取出第二个<div>标签中<p>标签中的文本，也就是食物热量
            food_url = 'http:' + food.find('a')['href']
            # 用find_all在<li>标签下，提取出唯一一个<a>标签中href属性的值，跟'http:'组合在一起，就是食物详情页的链接
```

```
            print([category, food_name, food_calorie, food_url])
            # 打印食物的名称
            result_list.append([category, food_name, food_calorie,
food_url])
        savedata(result_list)
def savedata(li): #定义写入数据的函数
    try:
        wb = load_workbook("result.xlsx")
    except:
        wb = Workbook()
    finally:
        try:
            ws = wb[li[0][0]] #若工作表不存在就创建一个当前类别命名的工作表
        except Exception as e:
            ws = wb.create_sheet(li[0][0],index=0)
            ws.append(["类别","食物名称","热量","链接"])
        finally:
            for x in li:
                ws.append(x) # 按行写入数据
    wb.save("result.xlsx")
    wb.close()
tasks_list = []
# 创建空的任务列表
for x in range(5):
    # 相当于创建了5个爬虫
    task = gevent.spawn(crawler(work))
    # 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)
# 用gevent.joinall方法，启动协程，执行任务列表里的所有任务，让爬虫开始爬取网站
```