第五关 （案例实训）健康的减肥方式

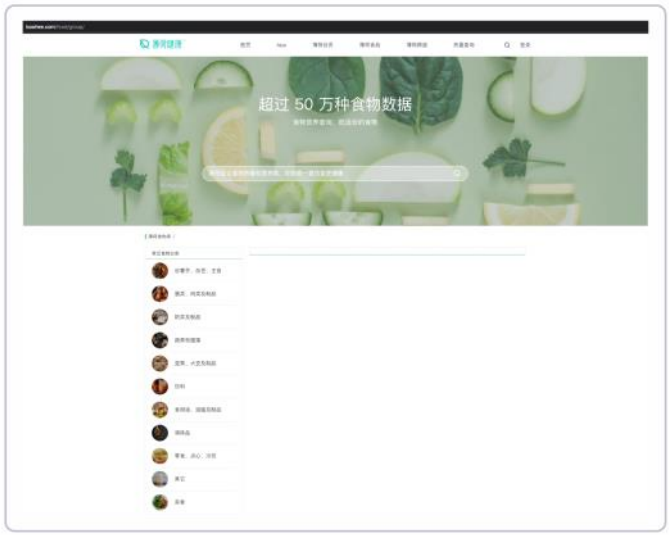
2022年12月6日 13:50

1. 项目代码

在展示代码之前，我们要知道，本节课要爬取网站的哪些信息。

1.1 项目目标

本节课要爬取的目标网站是《薄荷健康》，为了你能更好的学习本节课，我们统一使用谷歌 Chrome 浏览器，打开网站：http://www.boohee.com/food/group/ 。



单选题

爬取数据之前，还记得要先看什么吗？

A.

查看开发者工具的 Elements 选项卡

B.

查看网站的 Robots 协议

回答正确

爬取前还是需要先查询网站的 Robots 协议，否则会带来法律风险。

想要查看某一网站的 Robots 协议，只需要在网站的主机域名后加上/robots.txt。

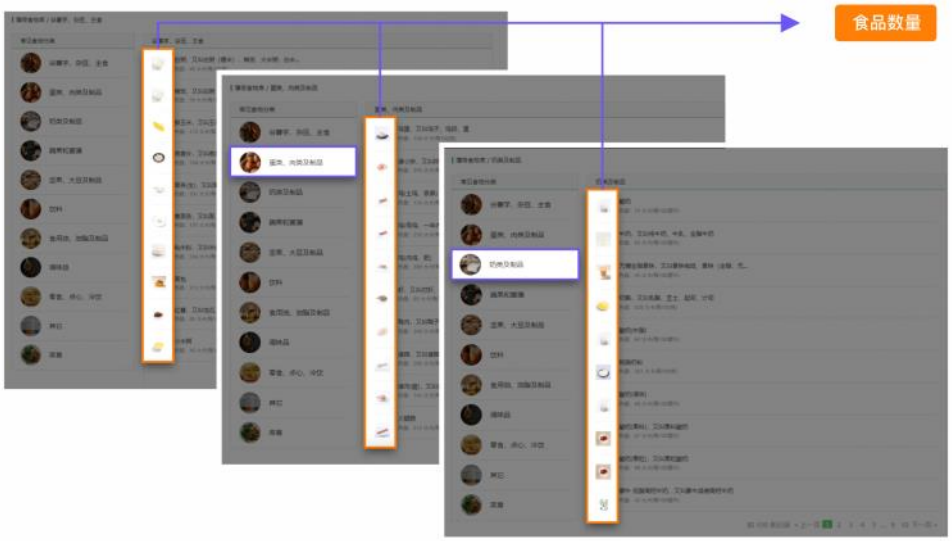
如《薄荷健康》网站的 Robots 协议为： http://www.boohee.com/robots.txt

```
User-agent: *
Disallow: /messages/
Disallow: /space/
Disallow: /travel/
Disallow: /hi/
Disallow: /friend/
Disallow: /album/
Disallow: /profile/
Disallow: /api/
Disallow: /simple_captcha/
Disallow: /can/add_eating
Disallow: /can/add_activity
Disallow: /can/remove_eating
Disallow: /can/remove_activity
Disallow: /can/change_eating
Disallow: /can/change_activity
Disallow: /travel/write_diary
Disallow: /travel/new_diary
Disallow: /posts/new_topic
Disallow: /posts/to_reply
```

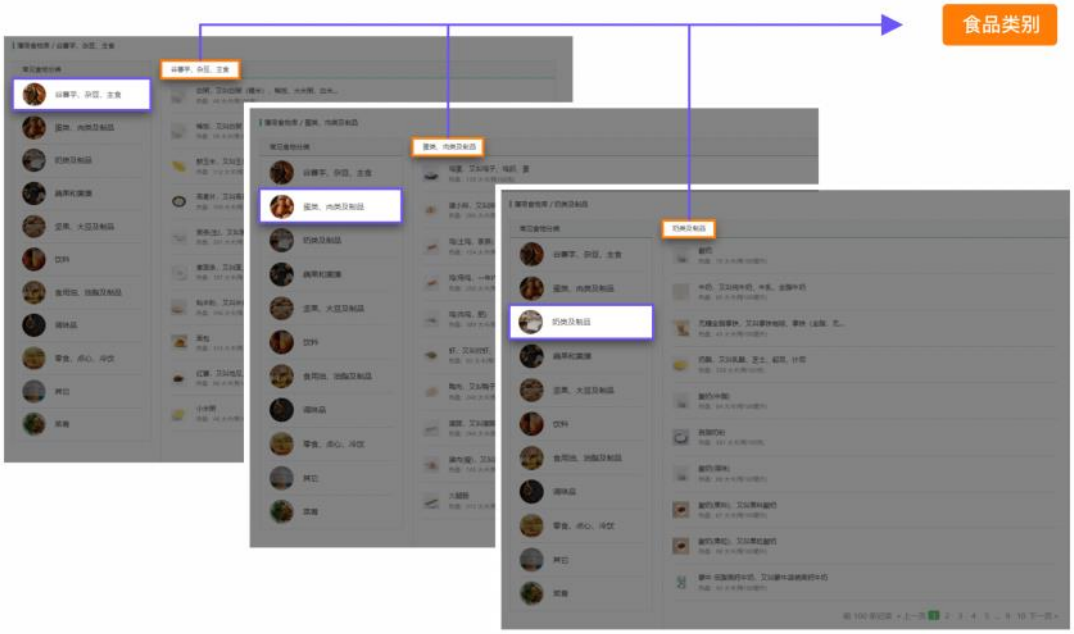

从协议内容中可以得知，我们想要爬取的网页链接，《薄荷健康》网站并没有不允许，那就可以继续对网页进行分析。
本项目爬取的内容是该网站所有食品的信息，包括：食品类别、食品名、食品热量、食品链接。
由于爬取整个网站的所有食品信息会给网站带来很大的负担，而且为了方便教学，这里我先提供实现“爬取网站前三类食物，每类食物各 30 种食品信息”的代码。
在学习代码之前，需要知道数据位于网页的哪些位置。

1.2 分析网页

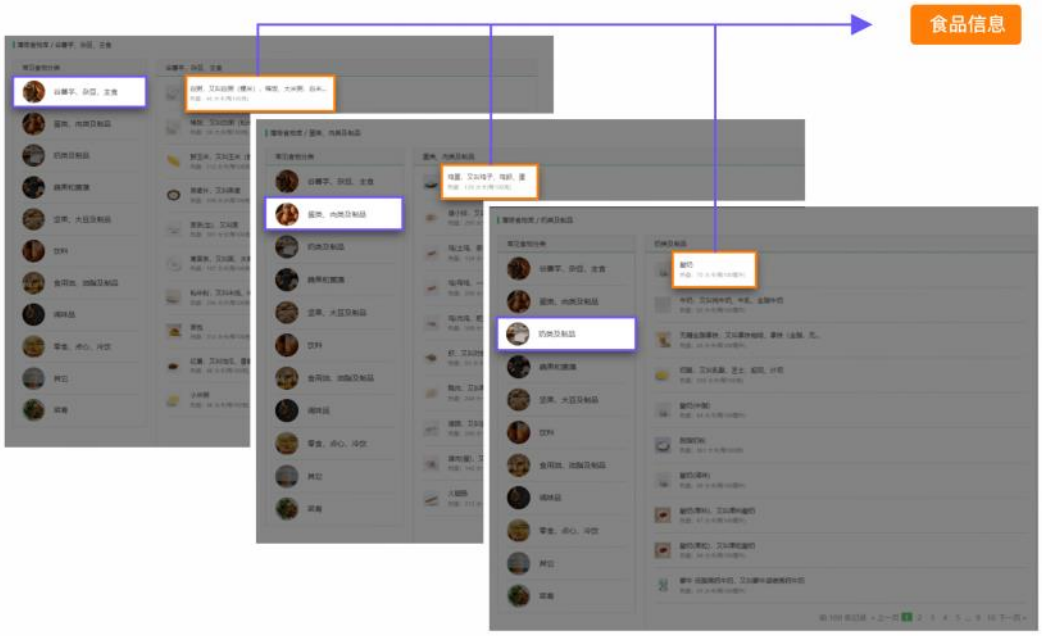
项目目标是网站前三类食物，分别是“谷薯芋、杂豆、主食”、“蛋类、肉类及制品”、“奶类及制品”。
我们依次点击这三类食物，在展开的食品列表页可以看到，每类食物的单个食品列表页都由 10 种食品组成。



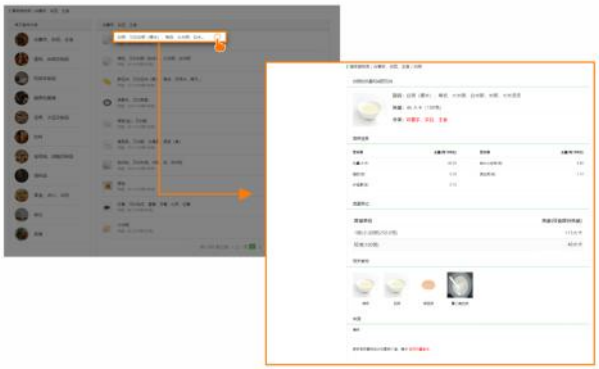
那在本次项目中，我们只需要爬取这前三类食物，每类食物前 3 页食品列表页的食品信息。
在上图中同时可以看到，食物类别在页面上的位置都是一致的：



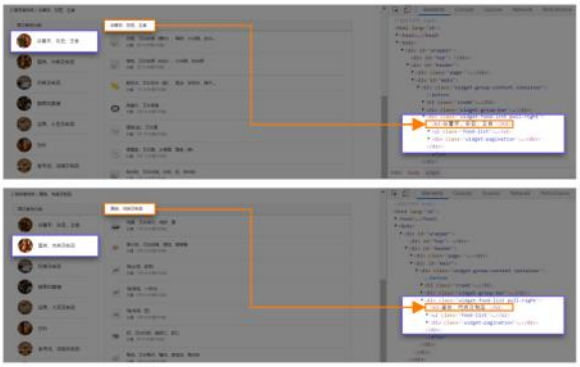
而且页面上有每种食品的食品名、食品热量的信息：



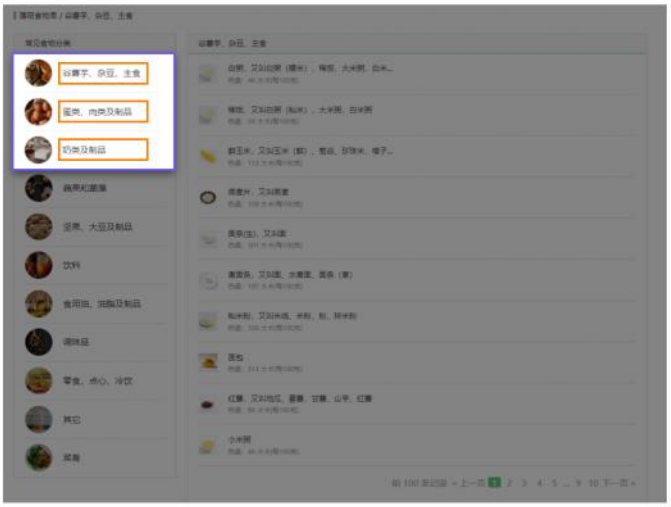
ok，食品列表页上已经可以拿到目标数据的其中三个了，还差食品链接的数据未在这里呈现。
以食物“谷薯芋、杂豆、主食”的食品白粥为例，我们点击它的食品名，发现可以进入它的详细页： http://www.boohee.com/shiwu/jingmizhou 。



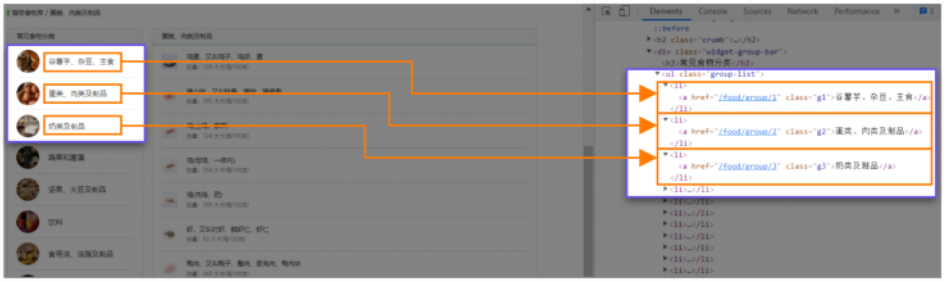
那么食品列表页上应该是有食品链接的存在，后续可以在 Elements 选项卡再来找找食品链接。
确定完目标数据在网页上的位置后，我们需要定位这些数据在 Elements 选项卡上的位置。
请打开浏览器的开发者工具，跟着我一起来看看。
为了检验不同食物、不同食品在 Elements 选项卡的数据位置是否相同，我们需要多对比几种不同的食物、食品。
以食物“谷薯芋、杂豆、主食”和“蛋类、肉类及制品”为例，打开开发者工具，使用【指针工具】分别定位两种食物的类别在 Elements 选项卡上的位置。



可以看到，不同食品的食物类别都在标签 中，元素 h3 的文本内容里。
可能你会问到，下图左侧的位置也能取到食物的类别，为什么不在那里取呢?



那是因为在这个位置取值，并非最优的选择。
还是使用【指针工具】，在这个位置分别点击前三类食物的类别。



它们都在 li 元素内，同在一个类别列表里面。虽说也能提取到食品类别，但我们原先提取食品类别的位置，对单个食品列表页的类别说明会更有针对性，

它可以更直观的让我们知道，某页食品列表页下的食品都属于哪一类食物。

所以我们选择标签为 的元素 h3 ，来提取食品的食物类别。

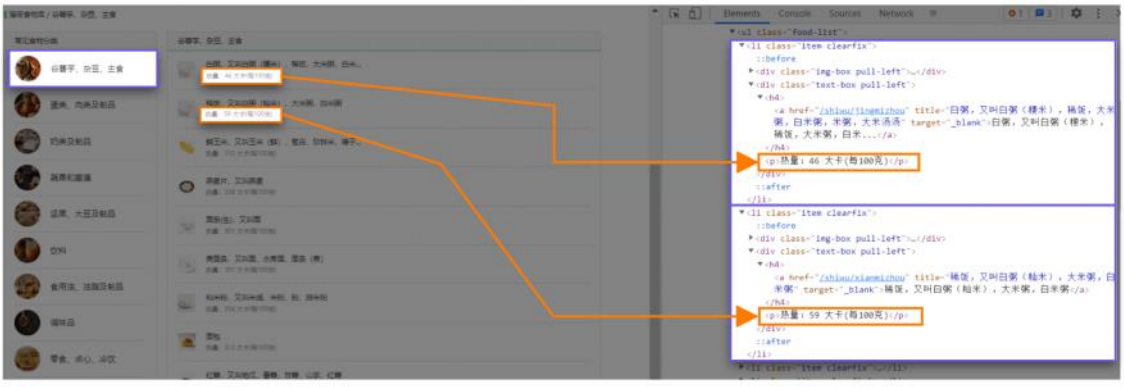
接着来对比一下食品的其它数据，下面我们挑选食物“谷薯芋、杂豆、主食”下的食品白粥、稀饭，以及食物“蛋类、肉类及制品”下的食品鸡蛋，看看它们

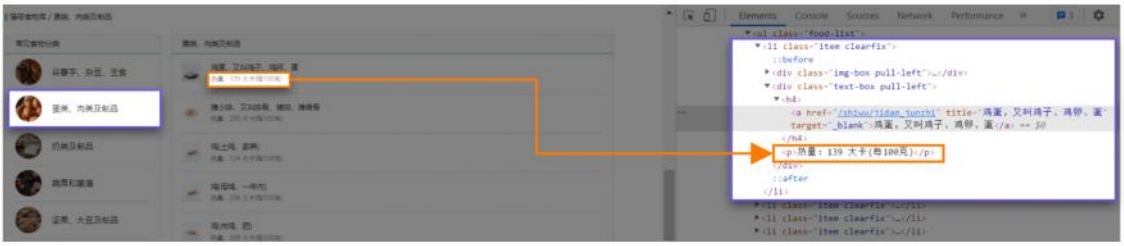
的数据在 Elements 选项卡上的位置。

先通过【指针工具】分别点击它们的食品名。

食品的食品名都在标签 中，元素 h4 下的元素 a 里。

同时可以看到，食品的食品热量也在该标签下，不过是在元素 p 里。





最后需要确定的数据是食品链接。

为了方便你对比，我先把上面几种食品的详情页链接展示给你看看：

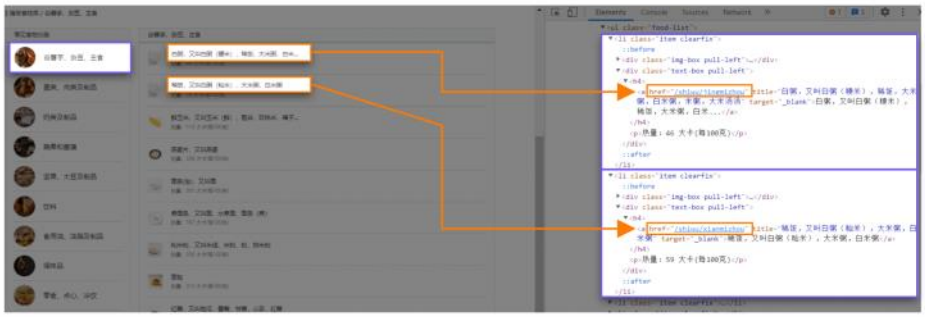
① 白粥：http://www.boohee.com/shiwu/jingmizhou；

② 稀饭：http://www.boohee.com/shiwu/xianmizhou；

③ 鸡蛋：http://www.boohee.com/shiwu/jidan_junzhi。

前面也说过，我们是点击食品名跳转的详情页，所以我们可以在 Elements 选项卡，看看食品名的位置是否有食品链接。

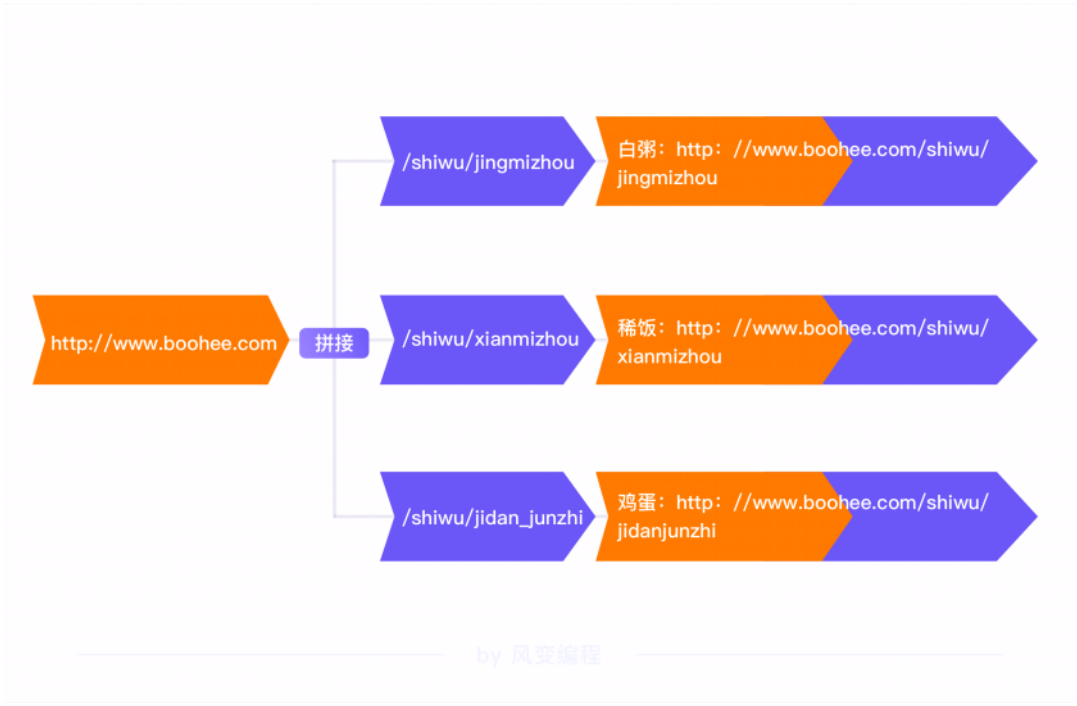
同样使用【指针工具】分别点击它们的食品名。



食品名所在的 a 元素里有一个属性 href，它的属性值是一个类似路径的数据。我们把各个属性值都取出来，与详情页的网址对比看看。

对比过后可以发现，不同食品的食品链接都由两部分组成，第一部分固定为http://www.boohee.com；第二部分都在标签 中，

元素 h4 下元素 a 的属性 href 内。



ok，数据在 Elements 选项卡的位置我们都清楚了，接下来请体验一下我写好的代码吧。

1.3 体验代码

```python
# 体验代码
'''url:http://www.boohee.com/food/group/
实现“爬取网站前三类食物
包括：食品类别、食品名、食品热量、食品链接'''

import csv
import requests
from bs4 import BeautifulSoup

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置列表，用以存储每种食物的信息
data_list = []

# 使用 for 循环遍历取值范围为 1~3 的数据
for type_number in range(1, 4):
    # 使用 for 循环遍历取值范围为 1~3 的数据
    for page_number in range(1, 4):
        # 设置要请求的网页链接
```

```
        menu_url = 'http://www.boohee.com/food/group/{}?page={}'.format(type_number,
        page_number)
        # 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
        food_list_res = requests.get(menu_url, headers = headers)
        # 解析请求到的网页内容
        bs = BeautifulSoup(food_list_res.text, 'html.parser')
        # 提取食物类别
        food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.
        strip()
        # 搜索网页中所有包含食物信息的 Tag
        food_list = bs.find_all('div', class_='text-box pull-left')

        # 使用 for 循环遍历搜索结果
        for food in food_list:
            # 提取食物名
            food_name = food.find('a')['title']
            # 提取食物热量
            food_calorie = food.find('p').text[3:]
            # 提取食物链接
            food_href = 'http://www.boohee.com/{}'.format(food.find('a')['href'])

            # 将信息添加到字典中
            food_dict = {
                '食物类别': food_type,
                '食物名': food_name,
                '食物热量': food_calorie,
                '食物链接': food_href
            }

            # 将每种食物的信息添加至列表中
            data_list.append(food_dict)
            # 打印食物的信息
            print(food_dict)

# 新建 csv 文件，用以存储食物信息
with open('foods.csv', 'w', encoding='utf-8-sig') as f:
    # 将文件对象转换成 DictWriter 对象
    f_csv = csv.DictWriter(f, fieldnames=['食物类别', '食物名', '食物热量', '食物链接'])
    # 写入表头与数据
    f_csv.writeheader()
    f_csv.writerows(data_list)
```

方法2

```
import requests
from bs4 import BeautifulSoup
import csv

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
    like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

food_list = []
```

```python
for page in range(1, 4):
    for number in range(1,4):

        url = f'https://www.boohee.com/food/group/{number}?page={page}'


        res = requests.get(url, headers=headers)
        bs = BeautifulSoup(res.text, 'html.parser')
```

```python
        food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()

        food_item = bs.find_all('div', class_='text-box pull-left')
        for food in food_item:
            food_name = food.find('h4').find('a')['title']
            food_calorie = food.find('p').text[3:]
            food_link = 'https://www.boohee.com'+food.find('h4').find('a')['href']
            print(food_link)

            food_dict = {
                'food_name': food_name,
                'food_calorie': food_calorie,
                'food_type': food_type,
                'food_link': food_link
            }
            food_list.append(food_dict)

            print(food_list)
```

```python
with open(r'D:\PythonTest\风变python学习资料\Python爬虫\food_type.csv', 'w', newLine='', encoding='utf-8-sig') as f:
    csv_write = csv.DictWriter(f, fieldnames=['food_name', 'food_calorie', 'food_type', 'food_link'])
    csv_write.writeheader()
    csv_write.writerows(food_list)
```

```
...  Output exceeds the size limit. Open the full output data in a text editor
https://www.boohee.com/shiwu/mifan_zheng
[{'food_name': '米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭', 'food_calorie': '116 大卡(每100克)', 'food_type': '谷薯芋、杂豆、主食', 'food_link': 'https://www.boohee.com/shiwu/mifan_zheng'}]
https://www.boohee.com/shiwu/yumi_xian
[{'food_name': '米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭', 'food_calorie': '116 大卡(每100克)', 'food_type': '谷薯芋、杂豆、主食', 'food_link': 'https://www.boohee.com/shiwu/mifan_zheng'}, {'food_name': '鲜玉米，又叫玉米（鲜）、苞谷、珍珠米、棒子、玉蜀黍、苞米、六谷、粟米、', 'food_calorie': '112 大卡(每100克)', 'food_type': '谷薯芋、杂豆、主食', 'food_link': 'https://www.boohee.com/shiwu/yumi_xian'}]
https://www.boohee.com/shiwu/maipeimianbao
[{'food_name': '米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭', 'food_calorie': '116 大卡(每100克)', 'food_type': '谷薯芋、杂豆、主食', 'food_link': 'https://www.boohee.com/shiwu/mifan_zheng'}, {'food_name': '鲜玉米，又叫玉米（鲜）、苞谷、珍珠米、棒子、玉蜀黍、苞米、六谷、粟米、', 'food_calorie': '112 大卡(每100克)', 'food_type': '谷薯芋、杂豆、主食', 'food_link': 'https://www.boohee.com/shiwu/yumi_xian'}, {'food_name': '全麦面包，又叫全麦面包、全麦吐司、全麦面包片、全麦土司', 'food_calorie': '254 大卡(每100克)', 'food_type': '谷薯芋、杂豆、主食', 'food_link': 'https://www.boohee.com/shiwu/maipeimianbao'}]
https://www.boohee.com/shiwu/mantou_junzhi
[{'food_name': '米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭', 'food_calorie': '116 大卡(每100克)',
```

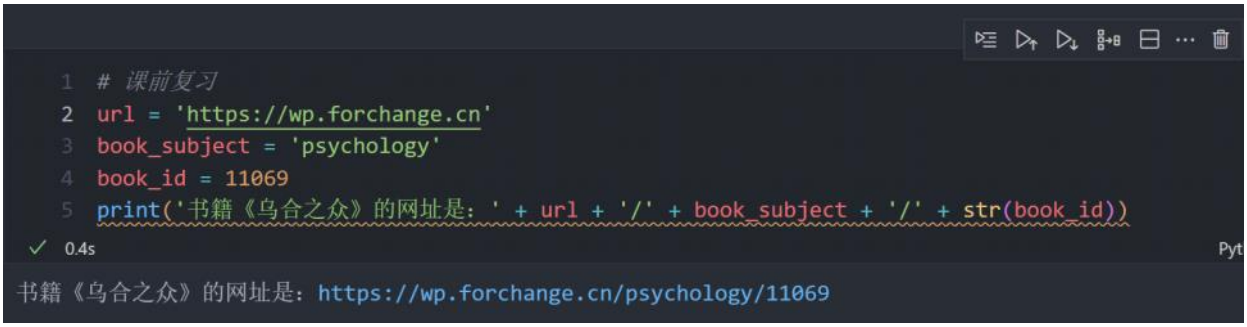
	A	B	C	D
1	food_name	food_calorie	food_type	food_link
2	叫大米饭，饭，蒸米、锅巴饭	116 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/mifan_zheng
3	、苞谷、珍珠米、棒子、玉蜀	112 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/yumi_xian
4	全麦面包、全麦吐司、全麦面	254 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/maipeimianbao
5	又叫手工馒头、大馒头、白i	223 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/mantou_junzhi
6	又叫地瓜、番薯、甘薯、山芋	86 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/ganshu_hongxin
7	燕麦片，又叫燕麦	338 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/yanmaipian
8	条，又叫面、水煮面、面条（	107 大卡(每100克)	谷薯芋、杂豆、主食	s://www.boohee.com/shiwu/miantiao_fuqiangfen_
9	紫薯，又叫甘薯（紫色）	133 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/zishu
10	洋芋、地蛋、山药蛋、洋番薯	81 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/malingshu
11	小米粥	46 大卡(每100克)	谷薯芋、杂豆、主食	https://www.boohee.com/shiwu/xiaomizhou
12	鸡蛋，又叫鸡子、鸡卵、蛋	139 大卡(每100克)	蛋类、肉类及制品	https://www.boohee.com/shiwu/jidan_junzhi
13	又叫鸡柳肉、鸡里脊肉、鸡胸、	118 大卡(每100克)	蛋类、肉类及制品	https://www.boohee.com/shiwu/jixiongfurou
14	)，又叫荷包蛋、煎蛋、煎荷包	195 大卡(每100克)	蛋类、肉类及制品	ttps://www.boohee.com/shiwu/hebaodan_youjian
15	猪肉(瘦)，又叫猪精肉，瘦肉	143 大卡(每100克)	蛋类、肉类及制品	https://www.boohee.com/shiwu/zhurou_shou
16	虾，又叫对虾、鲜虾仁、虾仁	93 大卡(每100克)	蛋类、肉类及制品	https://www.boohee.com/shiwu/fd5d8dea

```
import requests
from bs4 import BeautifulSoup
import csv
headers = {
    'User-Agent':
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}
food_list = []

for page in range(1, 4):
    for number in range(1,4):
        url = f'https://www.boohee.com/food/group/{number}?page={page}'

        res = requests.get(url, headers=headers)
        bs = BeautifulSoup(res.text, 'html.parser')

        food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()

        food_item = bs.find_all('div', class_='text-box pull-left')
        for food in food_item:
            food_name = food.find('h4').find('a')['title']
            food_calorie = food.find('p').text[3:]
            food_link = 'https://www.boohee.com'+food.find('h4').find('a')['href']
            print(food_link)
            food_dict = {
                'food_name': food_name,
                'food_calorie': food_calorie,
                'food_type': food_type,
                'food_link': food_link
            }
            food_list.append(food_dict)

            print(food_list)
with open(r'D:\PythonTest\风变python学习资料\Python爬虫\food_type.csv', 'w', newline='',encoding='utf-8-sig') as f:
    csv_write = csv.DictWriter(f, fieldnames=['food_name', 'food_calorie', 'food_type', 'food_link'])
    csv_write.writeheader()
    csv_write.writerows(food_list)
```

2. 课前复习

【第一题】format() 相关的练习。

请使用 format() 方法改写下方代码中，字符串的拼接部分。



```
# 课前复习
url = 'https://wp.forchange.cn'
book_subject = 'psychology'
book_id = 11069
print('书籍《乌合之众》的网址是：' + url + '/' + book_subject + '/' + str(book_id))
```

0.4s

书籍《乌合之众》的网址是：https://wp.forchange.cn/psychology/11069

答案

```
url = 'https://wp.forchange.cn'
book_subject = 'psychology'
book_id = 11069
print(f'书籍《乌合之众》的网址是：{url}/{book_subject}/{book_id}')
```

0.3s

书籍《乌合之众》的网址是：https://wp.forchange.cn/psychology/11069

+ 代码 + Markdown

【第二题】for 循环嵌套语句的练习。

1）已知潘潘要闯三个关卡，每个关卡需要吃到三种鱼干：“红鱼干”、“蓝鱼干”、“绿鱼干”；

2）请使用 for 循环语句嵌套的知识，使程序的执行结果如下图所示。

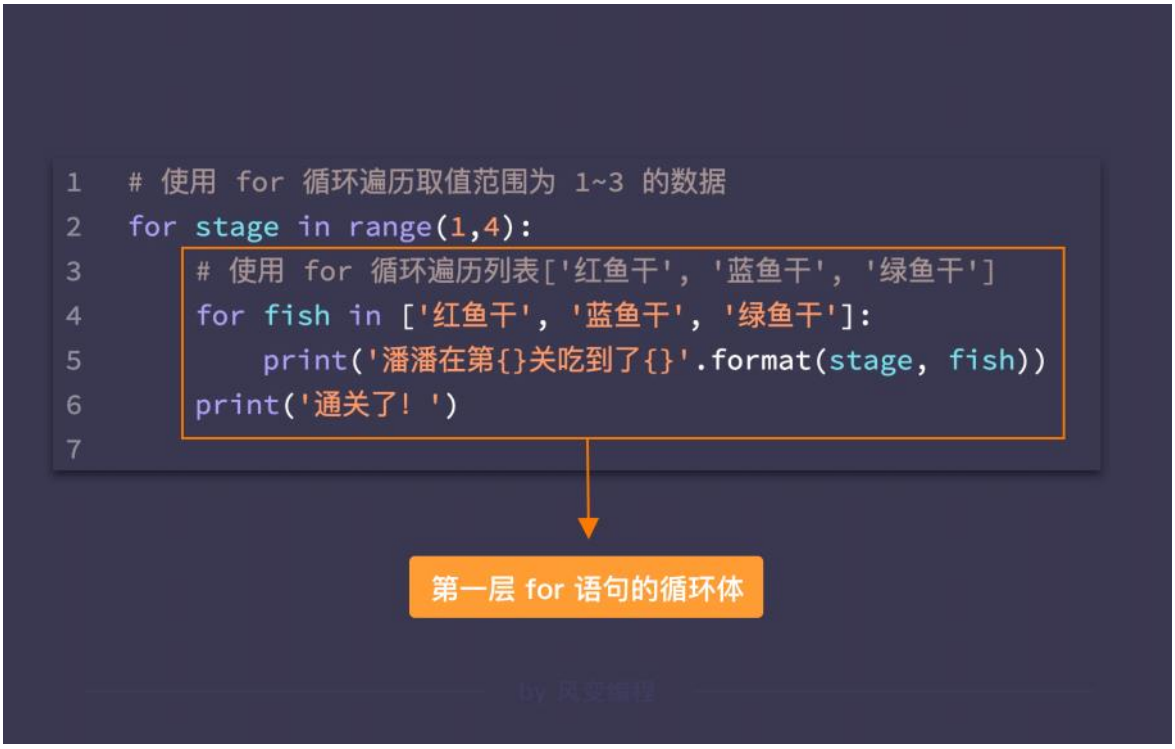

```
# 使用 for 循环遍历取值范围为 1~3 的数据
for ... in ...:
    # 使用 for 循环遍历列表['红鱼干', '蓝鱼干', '绿鱼干']
    for ... in ...:
```

```
'''【第二题】for 循环嵌套语句的练习。
1）已知潘潘要闯三个关卡，每个关卡需要吃到三种鱼干：“红鱼干”、“蓝鱼干”、“绿鱼干”；
2）请使用 for 循环语句嵌套的知识，使程序的执行结果如下图所示。
'''
# 使用 for 循环遍历取值范围为 1~3 的数据
for i in range(1, 4):
    # 使用 for 循环遍历列表['红鱼干', '蓝鱼干', '绿鱼干']
    for n in ['红鱼干', '蓝鱼干', '绿鱼干']:
        print(f'盼盼在第{i}关，吃到{n}')
    print('通关了')
```

✓ 0.3s

```
盼盼在第1关，吃到红鱼干
盼盼在第1关，吃到蓝鱼干
盼盼在第1关，吃到绿鱼干
通关了
盼盼在第2关，吃到红鱼干
盼盼在第2关，吃到蓝鱼干
盼盼在第2关，吃到绿鱼干
通关了
盼盼在第3关，吃到红鱼干
盼盼在第3关，吃到蓝鱼干
盼盼在第3关，吃到绿鱼干
通关了
```

由于 for 循环嵌套的知识点，对于下面获取多页网页链接的思路有关系，这里需要讲解一下。
第一层 for 语句是为了取出关卡数 1、2、3。每取出一个关卡数就执行该 for 循环语句的循环体，即第二层 for 语句。

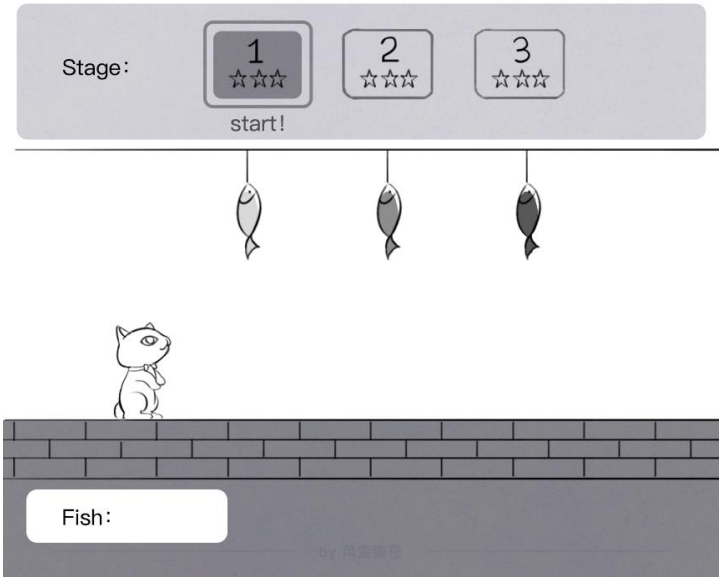


而第二层 for 语句是为了取出三种颜色的鱼干，每取出一种鱼干就执行该 for 循环语句的循环体，即 print() 打印语句：“马里奥在第 x 关吃到了 y 鱼干”。

```python
# 使用 for 循环遍历取值范围为 1~3 的数据
for stage in range(1,4):
    # 使用 for 循环遍历列表['红鱼干', '蓝鱼干', '绿鱼干']
    for fish in ['红鱼干', '蓝鱼干', '绿鱼干']:
        print('潘潘在第{}关吃到了{}'.format(stage, fish))
    print('通关了！')

```

第二层 for 语句的循环体

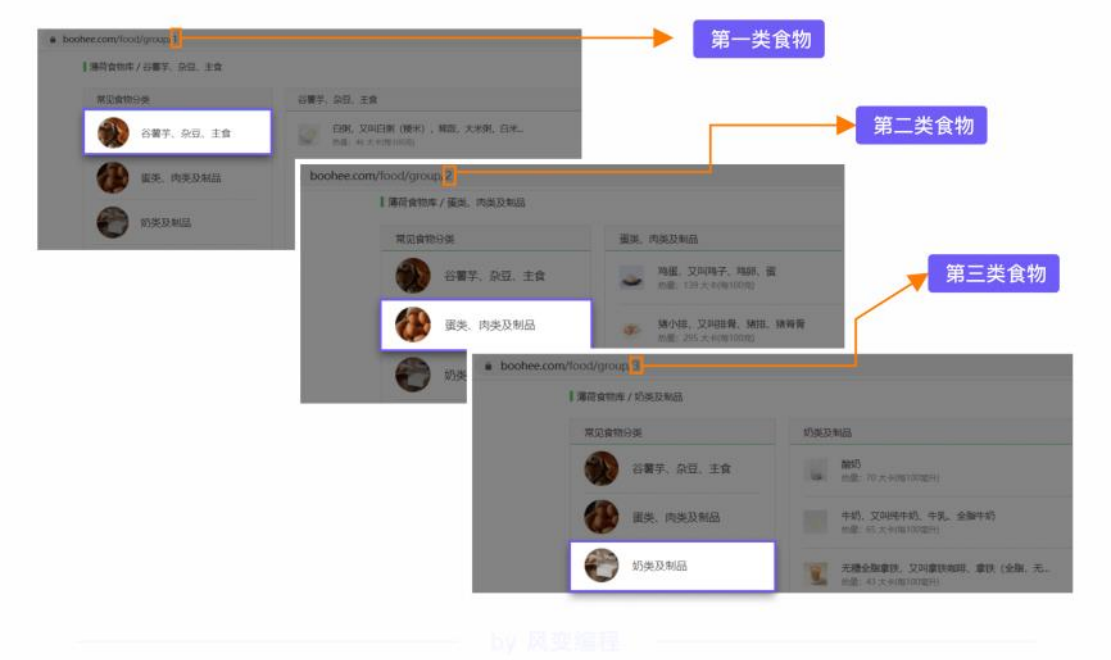
每当第二层 for 循环语句遍历完列表，并执行完循环体的打印语句后，会执行同缩进的 print() 打印语句：“通关了！”。
直到第一层 for 语句取到最后一个值，第二层 for 语句遍历完列表，且执行完代码最后一行的打印语句，该程序才运行完毕。
你也可以看看下图，辅助你理解这道题：



ok，重要的知识点就先复习到这里。
我们开始学习今天的项目代码。依旧是从网络爬虫的第一步：获取网页开始。

3. 获取网页

要获取的网页链接为前三类食物的前三页食品列表页，需要先找到它们之间的规律。
首先看看前三类食物的网页链接，我们分别点击前三类食物，然后观察它们的网页链接：



网页链接的前半部分都是一样的：http://www.boohee.com/food/group/，只是后面的数字在变化，那这个就很好实现了。
请你使用 for 循环语句，以及刚复习过的 format() 方法，打印出这三个网页链接吧。

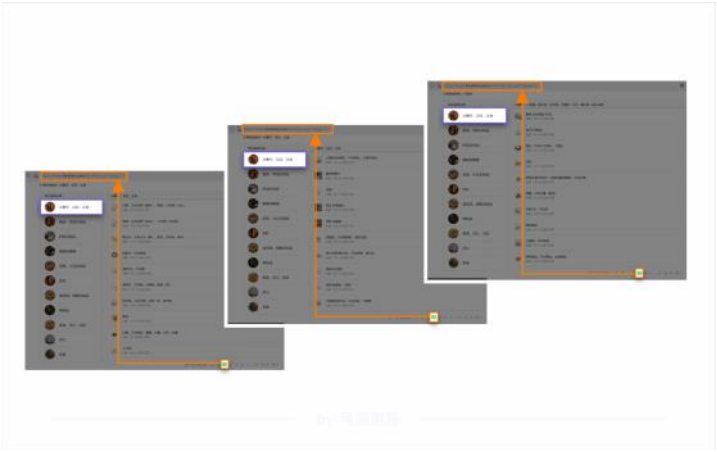
```
# 使用 for 循环遍历取值范围为 1~3 的数据

    # 拼接前三类食物的网页链接，并打印
```

```
# http://www.boohee.com/food/group/

# 使用 for 循环遍历取值范围为 1~3 的数据
for i in range(1, 4):

    # 拼接前三类食物的网页链接，并打印
    url =  f'http://www.boohee.com/food/group/{i}'
    print(url)
```

0.4s

```
http://www.boohee.com/food/group/1
http://www.boohee.com/food/group/2
http://www.boohee.com/food/group/3
```

然后，继续观察每类食物的食品列表页。在开头的网页分析，我们知道每类食物的食品列表页都是一样的，所以我们以某一类食物来找食品列表页的规律即可。
比如我们以第一类食物“谷薯芋、杂豆、主食”为例，分别点击该类食物列表页右下方的页数，观察前三页食品列表页的网页链接：



图中可以看出，后两页食品列表页的链接前半部是一样的：http://www.boohee.com/food/group/1?page=，后半部分只是数字不同，刚好对得上页数。

那第一页的列表页链接是否也是有同样的规律呢？
如果你点击其它列表页面后再点回第一页的话，你可以看到，它也是相同的规律：

第一页食品列表页呈现的两个网页链接，其实都是对应的同一页。按照我们的经验，只是网站将该食品类别的默认页面设置为第一页：`http://www.boohee.com/food/group/1`。
好了，同样使用 `for` 循环语句，食物“谷薯芋、杂豆、主食”的前三个食品列表页的链接，可以这么获取：

```
# 使用 for 循环遍历取值范围为 1~3 的数据
for number in range(1, 4):
    # 拼接第一类食物的前三页食品列表页的网页链接，并打印
    print('http://www.boohee.com/food/group/1?page={}'.format(number))
```

梳理完前三类食物的网页链接，以及单类食物的前三页食品列表页链接的规律后，可以发现存在这样的一个规律：

食品列表页的网页链接中，`/group/` 后面的数字代表着食物类别的序号，而 `?page=` 后面的数字代表着食品列表页的页数。

接下来，请你结合我们复习过的 `for` 循环嵌套语句的知识，打印出前三类食物的前三个食品列表页的网页链接吧。

```python
import itertools
for page, number in itertools.product(range(1, 4), range(1,4)):
    url = f'https://www.boohee.com/food/group/{number}?page={page}'
    print(url)
```

0.4s

```
https://www.boohee.com/food/group/1?page=1
https://www.boohee.com/food/group/2?page=1
https://www.boohee.com/food/group/3?page=1
https://www.boohee.com/food/group/1?page=2
https://www.boohee.com/food/group/2?page=2
https://www.boohee.com/food/group/3?page=2
https://www.boohee.com/food/group/1?page=3
https://www.boohee.com/food/group/2?page=3
https://www.boohee.com/food/group/3?page=3
```

```python
# 使用 for 循环遍历取值范围为 1~3 的数据
for type_number in range(1, 4):
    # 使用 for 循环遍历取值范围为 1~3 的数据
    for page_number in range(1, 4):
        # 设置要请求的网页链接
        menu_url = 'http://www.boohee.com/food/group/{}?page={}'.format(type_number,
        page_number)
        # 打印网页链接
        print(menu_url)
```

0.4s

```
http://www.boohee.com/food/group/1?page=1
http://www.boohee.com/food/group/1?page=2
http://www.boohee.com/food/group/1?page=3
http://www.boohee.com/food/group/2?page=1
http://www.boohee.com/food/group/2?page=2
http://www.boohee.com/food/group/2?page=3
http://www.boohee.com/food/group/3?page=1
http://www.boohee.com/food/group/3?page=2
http://www.boohee.com/food/group/3?page=3
```

代码中的第一层 `for` 语句依次取出食物类别的序号，每取出一个数字，便进入到第二层 `for` 语句。
第二层 `for` 语句依次取出食品列表页的页数，每取出一个数字，便和食物类别的序号及网页链接，组成一个食品列表页的链接。
到这里，已经获取到前三类食物的前三个食品列表页的网页链接。
接下来就要使用 `requests` 库来获取网页信息，你来运行一下我写好的代码吧：

```python
import itertools
import requests

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 使用 for 循环遍历取值范围为 1~3 的数据
for type_number, page_number in itertools.product(range(1, 4), range(1, 4)):
    # 设置要请求的网页链接
    menu_url = f'http://www.boohee.com/food/group/{type_number}?page={page_number}'
    # 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
    food_list_res = requests.get(menu_url, headers = headers)
    # 打印网页响应状态码
    print(f'第{type_number}类食物的第{page_number}个食品列表页的响应状态码为：{food_list_res.status_code}')
```

✓ 1.9s Python

```
第1类食物的第1个食品列表页的响应状态码为：200
第1类食物的第2个食品列表页的响应状态码为：200
第1类食物的第3个食品列表页的响应状态码为：200
第2类食物的第1个食品列表页的响应状态码为：200
```

方法2

```python
import requests

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}
# 使用 for 循环遍历取值范围为 1~3 的数据
for type_number in range(1, 4):
    # 使用 for 循环遍历取值范围为 1~3 的数据
    for page_number in range(1, 4):
        # 设置要请求的网页链接
        menu_url = 'http://www.boohee.com/food/group/{}?page={}'.format(type_number, page_number)
        # 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
        food_list_res = requests.get(menu_url, headers = headers)
        # 打印网页响应状态码
        print('第{}类食物的第{}个食品列表页的响应状态码为：{}'.format(type_number, page_number, food_list_res.status_code))
```

✓ 1.7s Pytho

```
第1类食物的第1个食品列表页的响应状态码为：200
第1类食物的第2个食品列表页的响应状态码为：200
第1类食物的第3个食品列表页的响应状态码为：200
```

4. 解析网页

接下来是网络爬虫的第二步：解析网页。同样使用 BeautifulSoup 库来解析网页。

上面我们也看到，不同食品的信息（食品类别、食品名、食品热量、食品链接）在 Elements 选项卡的位置都一致，所以我们使用其中一种食品来分析即可。

下面我们以食物“谷薯芋、杂豆、主食”的第一个食品列表页为例：

http://www.boohee.com/food/group/1?page=1。

还记得 BeautifulSoup 的用法吗？请补全下方代码的第 14、16 行，实现解析该食品列表页，并打印 BeautifulSoup 对象的操作吧。

```python
import requests
from bs4 import BeautifulSoup
```

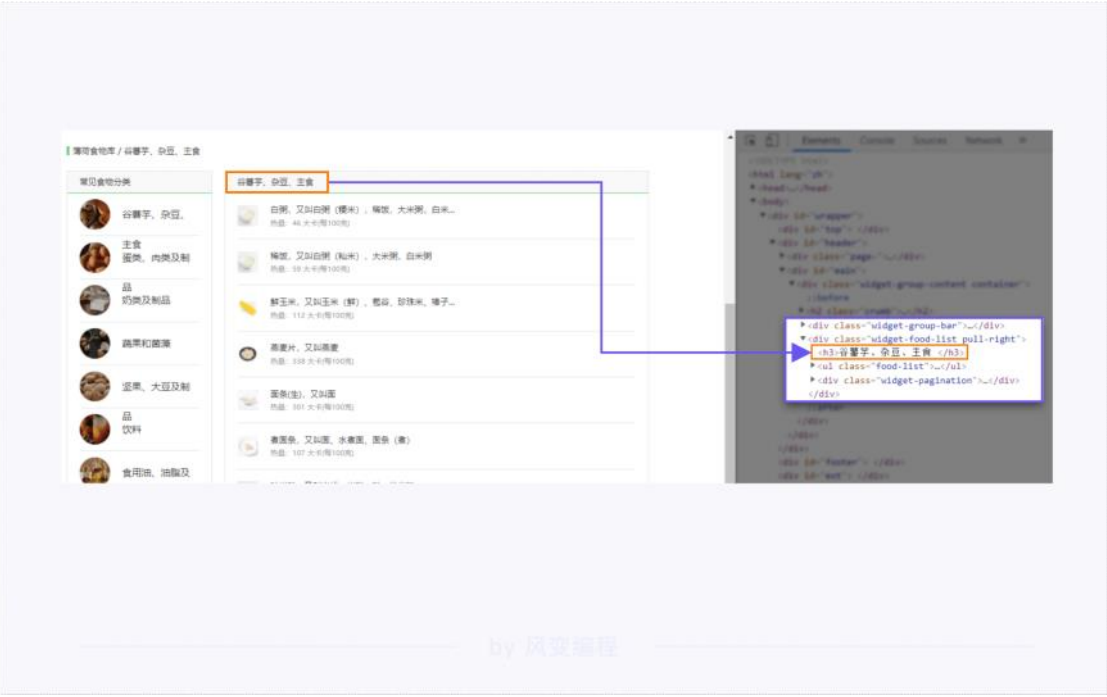
```
# （headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码）
headers = {
    'User-Agent':
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}
# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容

# 打印 BeautifulSoup 对象
```

```
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
    like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析网页内容
menu_res_bs = BeautifulSoup(menu_res.text, 'html.parser')
# 打印 BeautifulSoup 对象
print(menu_res_bs)

✓ 0.2s
```

```
Output exceeds the size limit. Open the full output data in a text editor
<!DOCTYPE html>

<html lang="zh">
<head>
<meta charset="utf-8"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<title>谷薯芋、杂豆、主食食物热量 - 薄荷网 - 薄荷网</title>
<meta content="谷薯芋、杂豆、主食食物热量，食物卡路里" name="keywords"><meta content="食物热量查询，食
物卡路里大全" name="keywords">
<meta content="WPvgwdf4aqsujE8987ZHPpBNvtHguD/yT9AaQdi784c=" name="verify-v1">
<meta content="app-id=457856023" name="apple-itunes-app"/>
<meta content="format=xhtml;url=http://m.boohee.com" name="mobile-agent"/>
<link enablehash="true"
href="https://www.boohee.com/stylesheets/core_v2_0a8b4baac6f2850ce455f7f83db9bc0e.css?1667296200"
media="screen" rel="stylesheet" type="text/css"/>
<link href="//up.boohee.cn/house/u/site/pc/main-v4.css" rel="stylesheet"/>
<link enablehash="true"
href="https://www.boohee.com/stylesheets/food_v2_0a7b4750654e9cbb53b81bf75c08901b.css?1667296200"
media="screen" rel="stylesheet" type="text/css"/>
<style>
```

4.1 提取食物类别

使用【指针工具】点击食物类别。



可以看到食品类别在元素 h3 里，我们在 Elements 选项卡上按【Ctrl+F】键，打开搜索框，输入h3，看看这个元素是否具有唯一性。

往上看可以看到最近的一个父级标签是： 。同样在 Elements 选项卡的搜索框，输入属性值widget-food-list pull-right，看看该 div 元素是否具有唯一性。

从图片上看到，这个元素确实是唯一的。ok，标签找到了，接下来请你补全下方代码的第 16 行来提取食物的类别吧。

（给你几个小提示）

① 先定位到 class 属性值为 widget-food-list pull-right 的 div 标签；

② 再定位到其内部的 h3；

③ 使用 Tag 对象的 text 属性获取食物类别；

④ 因为 find() 返回的结果是一个 Tag 对象，所以 find() 之间可以连用，例如 find('xxx').find('yyy')。

```
import requests
from bs4 import BeautifulSoup
# （headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码）
headers = {
    'User-Agent':
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}
```

```
# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type =
# 打印食物类别
print(food_type)
```

```
import requests
from bs4 import BeautifulSoup

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
    like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type = bs.find('div', class_ = 'widget-food-list pull-right').find('h3').text
# 打印食物类别
print(food_type)
```

✓ 0.2s

```
谷薯芋、杂豆、主食
```

不过你在复制 class 属性的属性值 widget-food-list pull-right 时，是否有个小问号呢？这里的 class 属性的属性值里有空格，跟往常看到的不大一样。

多个属性值的 class 属性

其实一个元素的 class 属性可以包含一到多个属性值，属性值之间用空格隔开。例如这里 class 的属性值，有属性值1：widget-food-list；属性值2：pull-right。

而我们在使用时，可以只匹配其中一个属性值，比如只保留其中一个属性值 widget-food-list。

同样在 Elements 选项卡按【Ctrl+F】键，打开搜索框，再输入这个属性值，可以看到确实只有一个。

你可以运行我下方的代码，看看打印的结果是否依旧为谷薯芋、杂豆、主食。

```python
import requests
from bs4 import BeautifulSoup

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type = bs.find('div', class_='widget-food-list').find('h3').text
# 打印食物类别
print(food_type)
```

```
✓ 0.2s
谷薯芋、杂豆、主食
```

这里是对属性 `class` 的知识补充，出现多属性值的 `class` 属性时，如果想要只使用其中的一个属性值，我们还是得先确认这个属性值是否具有唯一性，否则容易定位错位置。所以一般情况下，还是建议 `class` 属性写上所有属性值。

对了，这里还想和你分享另一个知识点。

strip() 方法

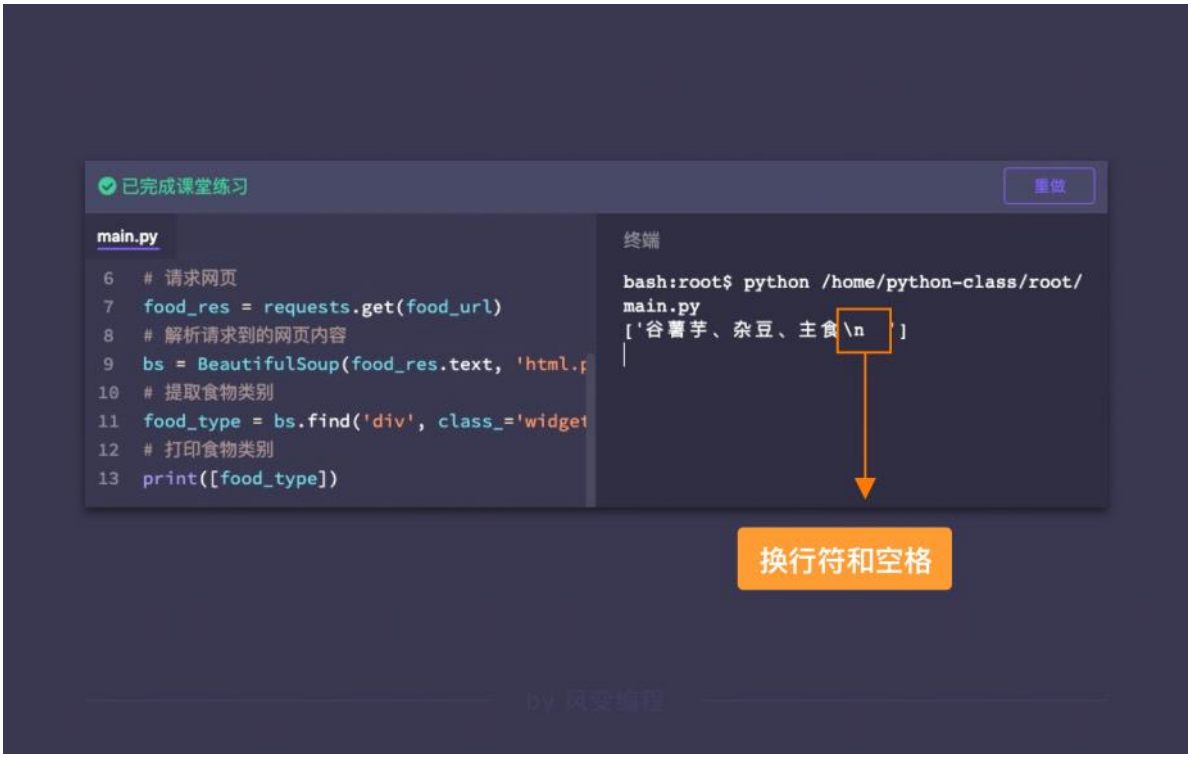
其实我们打印到终端的食物类别，有一些见不到的空格或者换行符，我把最后打印出的食品类别放进列表内，你可以运行看看效果：

```python
import requests
from bs4 import BeautifulSoup

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text
# 打印食物类别
print([food_type])
```

✓ 0.2s

```
['谷薯芋、杂豆、主食\n  ']
```

可以看到，打印出来的内容多了几个空格和一个换行符。



为了优化信息的呈现，我们可以借助字符串的 `strip()` 方法，它可以移除字符串头尾指定的单个或多个指定字符，返回一个新的字符串。默认情况下，是移除所有空白符（比如：空格符或换行符）。
你可以通过运行下方的程序，体验一下 `strip()` 的效果：

```
name = '    陈知枫    \n'
print([name])
print('=====我是分隔线=====')
name = '    陈知枫    \n'.strip()
print([name])
```

```
[16]  ✓ 0.3s
['    陈知枫    \n']
=====我是分隔线=====
['陈知枫']
```

回到项目，我们只需在原来提取食物类别的代码，第 16 行加上这个方法即可，运行我下方的代码看看，是否换行符和空格都被去除了：

```
import requests
from bs4 import BeautifulSoup


headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
     like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 打印食物类别
print([food_type])
```

```
]  ✓ 0.2s                                                    Python
['谷薯芋、杂豆、主食']
```

这个拓展的知识点跟你分享完了，下面我们来总结一下提取食物类别这块的内容。

在提取食物类别的代码中，我们依然是使用 BeautifulSoup 库来解析网页内容，并通过 find() 方法定位元素并提取内容。而在此基础上，可以有两个知识点对代码进行优化：

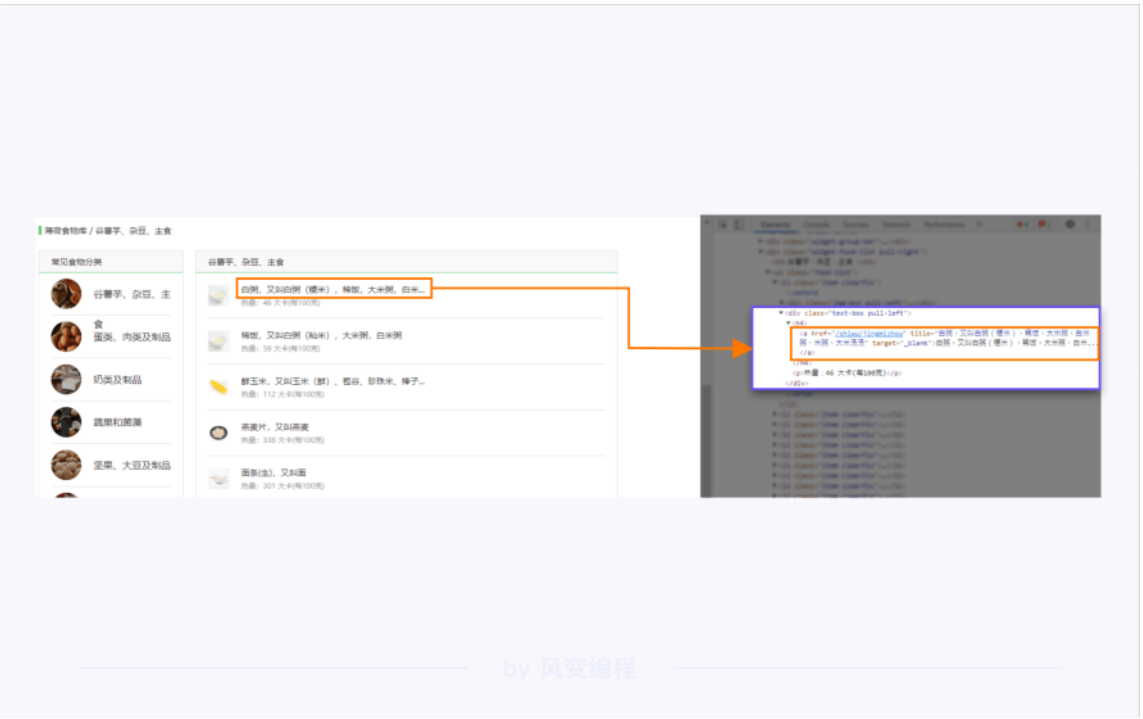
① 定位多属性值的 class 属性时，可以只匹配一个属性值。但前提是要确认该属性值是否唯一，否则需要匹配上多个属性值，以确保定位目标数据的元素具有唯一性；（本次项目依旧写上所有属性值）

② 打印出的字符串头尾带有空白符（比如：空格符或换行符）时，可以使用 strip() 方法去除。

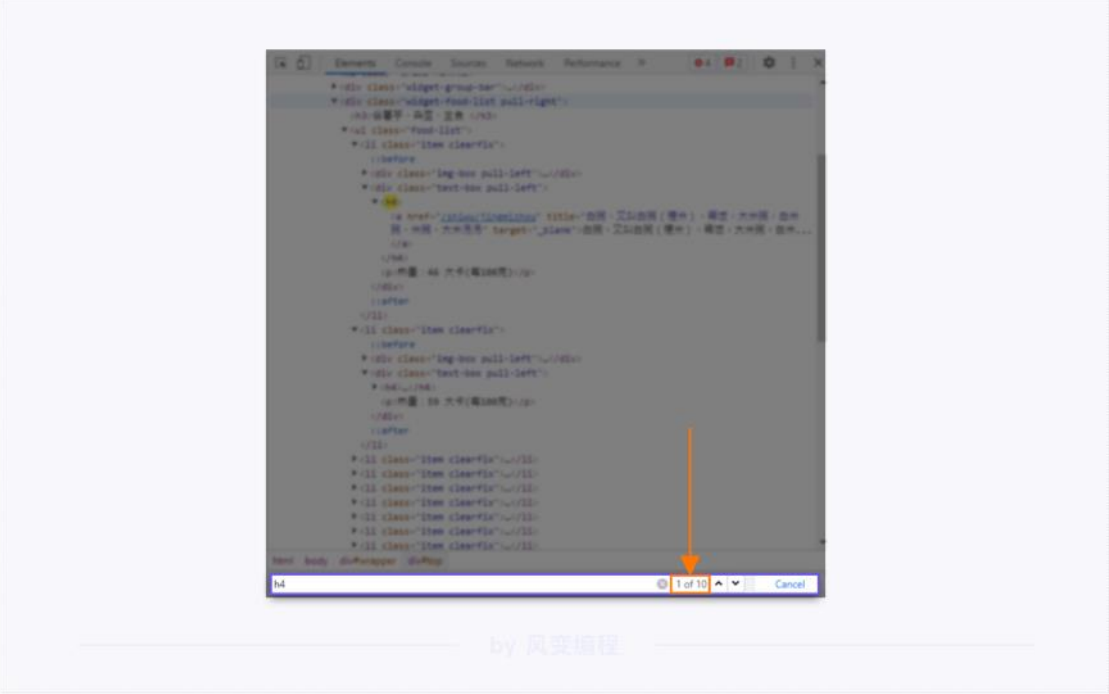
现在，食品类别的提取已完成，接下来要爬取的数据依次是食品的名字、热量、链接。

4.2 提取食品名

先来提取食品名，在 Elements 选项卡使用【指针工具】点击食品白粥的名字。

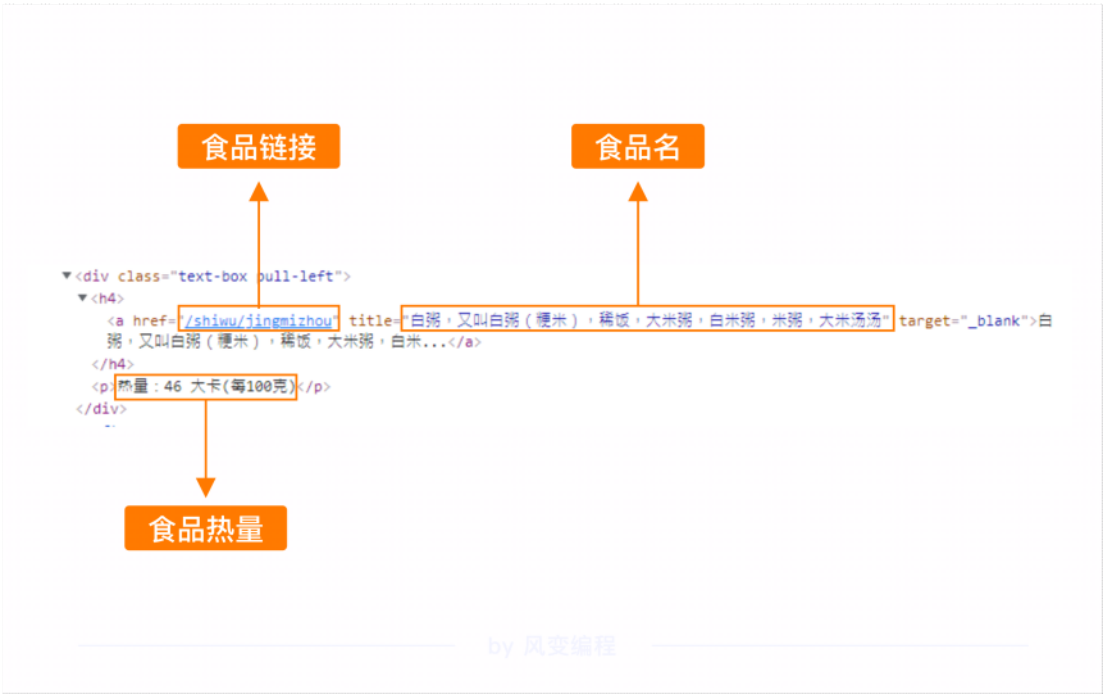


可以看到食品名在元素 a 的属性 tilte 里，虽然元素 a 的文本内容里也存有食品名，但显示内容不全，我们择优取属性 title 里的食品名。
同样，我们需要找到元素 a 最近的父级标签。可以看到，最近的为元素 h4，在 Elements 选项卡打开搜索框，输入h4。



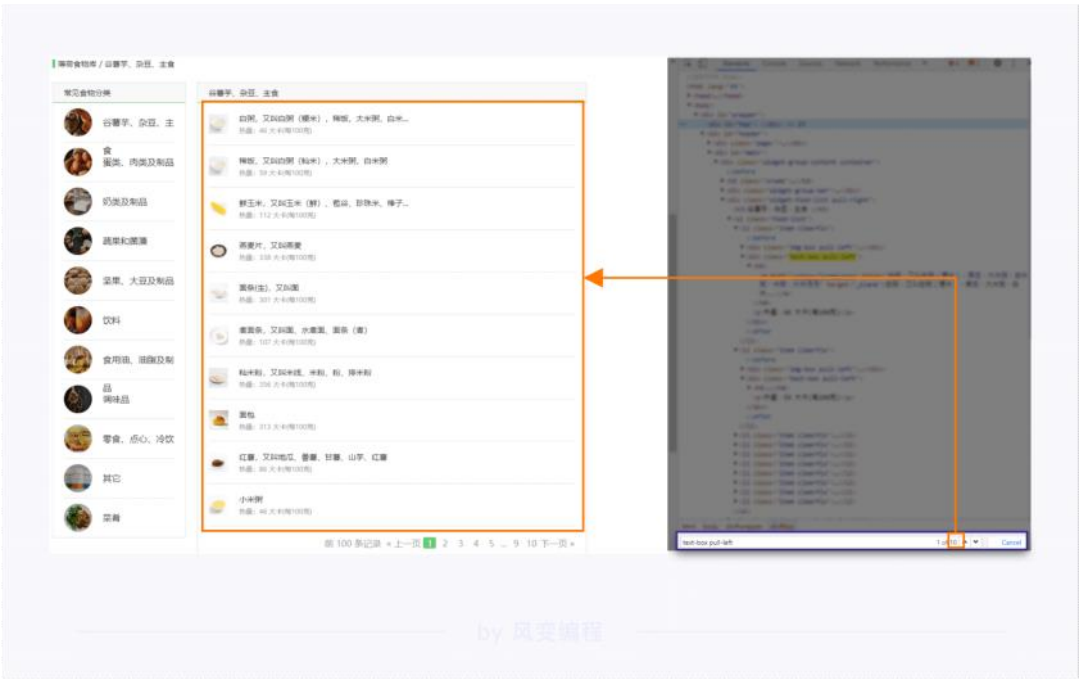
这里 h4 的数量有 10 个，不具有唯一性。所以我们继续往上找，可以找到标签

<div class='text-box pull-left>。
仔细观察这个标签，可以发现食品的名字、热量、链接都存在里面。



此时可以猜想一下：这个标签是不是列表页上，所有食品的共同标签呢？

然后我们在 Elements 选项卡上按【Ctrl+F】键，打开搜索框，再输入 text-box pull-left，可以看到它的数量与食品的个数一致。



那么可以确定，我们可以使用 find_all() 方法来定位所有食品信息的位置。请你运行下方的代码看看吧：

```
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent':
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}
# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
```

```
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 搜索网页中所有包含食物信息的 Tag
food_list = bs.find_all('div', class_='text-box pull-left')
# 打印食物类别
print(food_list)
```

```
import requests
from bs4 import BeautifulSoup


headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
    like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 搜索网页中所有包含食物信息的 Tag
food_list = bs.find_all('div', class_='text-box pull-left')
# 打印食物类别
print(food_list)
```

✓ 0.2s Python

```
Output exceeds the size limit. Open the full output data in a text editor
[<div class="text-box pull-left">
<h4>
<a href="/shiwu/mifan_zheng" target="_blank" title="米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭">米饭，
又叫大米饭，饭，蒸米、锅巴饭、煮米饭</a>
</h4>
<p>热量：116 大卡(每100克)</p>
</div>, <div class="text-box pull-left">
<h4>
<a href="/shiwu/yumi_xian" target="_blank" title="鲜玉米，又叫玉米（鲜）、苞谷、珍珠米、棒子、玉蜀黍、
苞米、六谷、粟米、">鲜玉米，又叫玉米（鲜）、苞谷、珍珠米、棒子...</a>
</h4>
<p>热量：112 大卡(每100克)</p>
</div>, <div class="text-box pull-left">
<h4>
<a href="/shiwu/maipeimianbao" target="_blank" title="全麦面包，又叫全麦面包、全麦吐司、全麦面包片、全
麦土司">全麦面包，又叫全麦面包、全麦吐司、全麦面包...</a>
</h4>
<p>热量：254 大卡(每100克)</p>
```

终端显示的内容确实包含了每种食品的各项信息。

接下来请你补全下方代码的第 21~25 行，提取每种食品的名字吧：

（给你几个小提示）

① 使用 for 语句遍历 find_all() 得到的结果；

② 因为元素 h4 里面的第一个 a 元素有我们所需的数据，所以我们使用 find() 可以越过元素 h4，直接定位到 a 元素；

③ 使用 Tag['属性名'] 提取属性的内容。

```python
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 搜索网页中所有包含食物信息的 Tag
food_list = bs.find_all('div', class_='text-box pull-left')
```

```python

# 使用 for 循环遍历搜索结果
for info in food_list:
    # 提取食品名
    food_name = info.find('h4').find('a')['title']
    # 打印食品名
    print(food_name)
```

```
米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭
鲜玉米，又叫玉米（鲜）、苞谷、珍珠米、棒子、玉蜀黍、苞米、六谷、粟米、
全麦面包，又叫全麦面包、全麦吐司、全麦面包片、全麦土司
馒头，又叫手工馒头、大馒头、白面馒头
红薯，又叫地瓜、番薯、甘薯、山芋、红薯
燕麦片，又叫燕麦
煮面条，又叫面、水煮面、面条（煮）
紫薯，又叫甘薯（紫色）
马铃薯，又叫土豆、洋芋、地蛋、山药蛋、洋番薯、土豆、洋芋、薯仔
```

4.3 提取食品热量

在 Elements 选项卡使用【指针工具】点击食品白粥的热量。

这里只需要取出下图中框住的具体热量信息，而不需要热量信息前面的“热量：”。

从图中可以看到，食品热量在元素 `p` 的文本内容中，且元素 `p` 的最小父级标签也是 `<div class='text-box pull-left>`。

那么只需要在提取食品名的代码上，添加一行提取食品热量的代码即可。请你补充下方代码的第 `25` 行吧：

```
import requests
from bs4 import BeautifulSoup

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物的类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 搜索网页中所有包含食物信息的 Tag
food_list = bs.find_all('div', class_='text-box pull-left')

```

```
# 使用 for 循环遍历搜索结果
for food in food_list:
    # 提取食品名
    food_name = food.find('a')['title']
    # 提取食品热量
    food_calorie = food.find('p').text
    # 打印食品热量
    print(food_calorie)
```

```
热量：116 大卡(每100克)
热量：112 大卡(每100克)
热量：254 大卡(每100克)
热量：223 大卡(每100克)
热量：86 大卡(每100克)
热量：338 大卡(每100克)
热量：107 大卡(每100克)
热量：133 大卡(每100克)
热量：81 大卡(每100克)
```

```
import requests
from bs4 import BeautifulSoup

#(headers的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers={
'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

#设置网页menu_url
menu_url='http://www.boohee.com/food/group/1?page=1'
#请求网页(设置headers部分可先记为固定格式，暂时不用理解)
menu_res=requests.get(menu_url,headers=headers)
#解析请求到的网页内容
bs=BeautifulSoup(menu_res.text,'html.parser')
#提取食物的类别
food_type=bs.find('div',class_='widget-food-listpull-right').find('h3').text.strip()
#搜索网页中所有包含食物信息的Tag
food_list=bs.find_all('div',class_='text-boxpull-left')

#使用for循环遍历搜索结果
for food in food_list:
#提取食品名
food_name=food.find('a')['title']
#提取食品热量
food_calorie=food.find('p').text
#打印食品热量
print(food_calorie)
```

补充的代码 `food.find('p').text[3:]` 中，`[3:]`是为了去掉多余的内容“热量：”，让数据存储时，可以直接将热量信息存到文件中。食品热量的提取也完成了，就差食品链接的爬取啦。

4.4 提取食品链接

在 Elements 选项卡使用【指针工具】点击食品白粥的名字，找到食品链接的半个部分。

前面我们也分析过，食品链接由两部分组成，前部分为固定的值：`http://www.boohee.com`，后者可以通过 `Tag['属性名']` 提取元素 `a` 中，属性 `href` 的内容。

请根据上面的提示，补充下方代码的第 27 行吧。

```
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物的类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 搜索网页中所有包含食物信息的 Tag
food_list = bs.find_all('div', class_='text-box pull-left')
```

```
18
19  # 使用 for 循环遍历搜索结果
20  for food in food_list:
21      # 提取食品名
22      food_name = food.find('a')['title']
23      # 提取食品热量
24      food_calorie = food.find('p').text[3:]
25      # 提取食品链接
26      food_link = 'http://www.boohee.com' + food.find('a')['href']
27      # 打印食品链接
28      print(food_link)
```

```
http://www.boohee.com/shiwu/mifan_zheng
http://www.boohee.com/shiwu/yumi_xian
http://www.boohee.com/shiwu/maipeimianbao
http://www.boohee.com/shiwu/mantou_junzhi
http://www.boohee.com/shiwu/ganshu_hongxin
http://www.boohee.com/shiwu/yanmaipian
http://www.boohee.com/shiwu/miantiao_fuqiangfen_zhu
http://www.boohee.com/shiwu/zishu
http://www.boohee.com/shiwu/malingshu
```

```
import requests
from bs4 import BeautifulSoup

headers={
'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

#设置网页menu_url
menu_url='http://www.boohee.com/food/group/1?page=1'
#请求网页(设置headers部分可先记为固定格式，暂时不用理解)
menu_res=requests.get(menu_url,headers=headers)
#解析请求到的网页内容
bs=BeautifulSoup(menu_res.text,'html.parser')
#提取食物的类别
food_type=bs.find('div',class_='widget-food-listpull-right').find('h3').text.strip()
#搜索网页中所有包含食物信息的Tag
food_list=bs.find_all('div',class_='text-boxpull-left')

#使用for循环遍历搜索结果
for food in food_list:
#提取食品名
food_name=food.find('a')['title']
#提取食品热量
food_calorie=food.find('p').text[3:]
#提取食品链接
food_link='http://www.boohee.com'+food.find('a')['href']
#打印食品链接
print(food_link)
```

由于后面我们会使用 csv 的 DictWriter() 来存储，所以我们会先使用字典来存储每种食品，可以看看下方代码的第 29~35 行，然后运行程序：

```python
import requests
from bs4 import BeautifulSoup

# (headers 的相关知识会在下节课详细讲解，可暂时忽略此处代码)
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

# 设置网页 menu_url
menu_url = 'http://www.boohee.com/food/group/1?page=1'
# 请求网页 (设置 headers 部分可先记为固定格式，暂时不用理解)
menu_res = requests.get(menu_url, headers = headers)
# 解析请求到的网页内容
bs = BeautifulSoup(menu_res.text, 'html.parser')
# 提取食物的类别
food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
# 搜索网页中所有包含食物信息的 Tag
food_list = bs.find_all('div', class_='text-box pull-left')
```

```python

# 使用 for 循环遍历搜索结果
for food in food_list:
    # 提取食品名
    food_name = food.find('a')['title']
    # 提取食品热量
    food_calorie = food.find('p').text[3:]
    # 提取食品链接
    food_href = 'http://www.boohee.com' + food.find('a')['href']

    # 将信息添加到字典中
    food_dict = {
        '食物类别': food_type,
        '食物名': food_name,
        '食物热量': food_calorie,
        '食物链接': food_href
    }

    # 打印食品的信息
    print(food_dict)
```

```
{'食物类别': '谷薯芋、杂豆、主食', '食物名': '米饭，又叫大米饭，饭，蒸米、锅巴饭、煮米饭', '食物热量': '116 大卡(每100克)', '食物链接': 'http://www.boohee.com/shiwu/mifan_zheng'}
{'食物类别': '谷薯芋、杂豆、主食', '食物名': '鲜玉米，又叫玉米（鲜）、苞谷、珍珠米、棒子、玉蜀黍、苞米、六谷、粟米、', '食物热量': '112 大卡(每100克)', '食物链接': 'http://www.boohee.com/shiwu/yumi_xian'}
{'食物类别': '谷薯芋、杂豆、主食', '食物名': '全麦面包，又叫全麦面包、全麦吐司、全麦面包片、全麦土司', '食物热量': '254 大卡(每100克)', '食物链接': 'http://www.boohee.com/shiwu/maipeimianbao'}
{'食物类别': '谷薯芋、杂豆、主食', '食物名': '馒头，又叫手工馒头、大馒头、白面馒头', '食物热量': '223 大卡(每100克)', '食物链接': 'http://www.boohee.com/shiwu/mantou_junzhi'}
{'食物类别': '谷薯芋、杂豆、主食', '食物名': '红薯，又叫地瓜、番薯、甘薯、山芋、红薯', '食物热量': '86 大卡(每100克
```

至此，我们已经完成食物“谷薯芋、杂豆、主食”的第一个食品列表页下，所有食品信息的提取。
接着就要实现项目的目标，获取前三类食物的前三页食品的信息。

4.5 提取项目所需的食品信息

要获取前三类食物的前三页食品的信息，需要加上开头我们写好的代码“获取前三类食物的前三页食品列表页的链接”。

获取前三类食物的前三页食品列表页的链接的代码。
由于我们在每一页爬取的数据都是一样的，所以我们只需要在第二层 for 语句的循环体内，把第 16 ~ 17 行的打印语句替换为解析网页，并提取单页食品列表页的食品信息的代码，就能实现我们的功能。

```python
import requests
from bs4 import BeautifulSoup

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/96.0.4664.45 Safari/537.36'
}

# 使用 for 循环遍历取值范围为 1~3 的数据
for type_number in range(1, 4):
    # 使用 for 循环遍历取值范围为 1~3 的数据
    for page_number in range(1, 4):
        # 设置要请求的网页链接
        menu_url = 'http://www.boohee.com/food/group/{}?page={}'.format(type_number, page_number)
        # 请求网页（设置 headers 部分可先记为固定格式，暂时不用理解）
        food_list_res = requests.get(menu_url, headers = headers)
        # 解析请求到的网页内容
        bs = BeautifulSoup(food_list_res.text, 'html.parser')
        # 提取食物类别
        food_type = bs.find('div', class_='widget-food-list pull-right').find('h3').text.strip()
        # 搜索网页中所有包含食物信息的 Tag
        food_list = bs.find_all('div', class_='text-box pull-left')
```

```
    # 使用 for 循环遍历搜索结果
    for food in food_list:
        # 提取食物名字
        food_name = food.find('a')['title']
        # 提取食物热量
        food_calorie = food.find('p').text[3:]
        # 提取食物链接
        food_href = 'http://www.boohee.com/{}'.format(food.find('a')['href'])

        # 将信息添加到字典中
        food_dict = {
            '食物类别': food_type,
            '食物名': food_name,
            '食物热量': food_calorie,
            '食物链接': food_href
        }
        print(food_dict)
```

```
{'食物类别': '奶类及制品', '食物名': '蒙牛 不添加蔗糖风味酸奶，又叫蒙牛木糖醇酸奶', '食物热量': '65 大卡(每100克)', '
食物链接': 'http://www.boohee.com//shiwu/mengniusuanniunaimutangchun'}
{'食物类别': '奶类及制品', '食物名': '酸奶(调味)', '食物热量': '88 大卡(每100毫升)', '食物链接': 'http://www
.boohee.com//shiwu/suannai_diaowei'}
{'食物类别': '奶类及制品', '食物名': '光明 倍优 高品质鲜牛奶巴氏杀菌乳200ml，又叫光明 优倍高品质鲜牛奶', '食物热量':
'75 大卡(每100毫升)', '食物链接': 'http://www.boohee.com//shiwu/fd962a6c'}
{'食物类别': '奶类及制品', '食物名': '低脂牛奶，又叫低脂奶 脱脂奶', '食物热量': '43 大卡(每100毫升)', '食物链接':
'http://www.boohee.com//shiwu/niuru_bufentuozhi_pamalate'}
{'食物类别': '奶类及制品', '食物名': '脱脂奶粉', '食物热量': '361 大卡(每100克)', '食物链接': 'http://www
```

```
importrequests
frombs4importBeautifulSoup

headers={
'User-Agent':'Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)Chrome/96.0.4664.45Safari/537.36'
}

#使用for循环遍历取值范围为1~3的数据
fortype_numberinrange(1,4):
#使用for循环遍历取值范围为1~3的数据
forpage_numberinrange(1,4):
#设置要请求的网页链接
menu_url='http://www.boohee.com/food/group/{}?page={}'.format(type_number,page_number)
#请求网页(设置headers部分可先记为固定格式，暂时不用理解)
food_list_res=requests.get(menu_url,headers=headers)
#解析请求到的网页内容
bs=BeautifulSoup(food_list_res.text,'html.parser')
#提取食物类别
food_type=bs.find('div',class_='widget-food-listpull-right').find('h3').text.strip()
#搜索网页中所有包含食物信息的Tag
food_list=bs.find_all('div',class_='text-boxpull-left')

#使用for循环遍历搜索结果
forfoodinfood_list:
#提取食物名字
food_name=food.find('a')['title']
#提取食物热量
food_calorie=food.find('p').text[3:]
#提取食物链接
```

```
food_href='http://www.boohee.com/{}'.format(food.find('a')['href'])

#将信息添加到字典中
food_dict={
'食物类别':food_type,
'食物名':food_name,
'食物热量':food_calorie,
'食物链接':food_href
}
print(food_dict)
```

5. 存储数据

网络爬虫的最后一步：存储数据。

我们还是使用 csv 模块的 DictWriter() 类去处理，会比较方便快捷。

```
with open(r'D:\PythonTest\风变python学习资料\Python爬虫\food_type.csv', 'w', newline='',encoding='utf-8-sig')
as f:
    csv_write = csv.DictWriter(f, fieldnames=['food_name', 'food_calorie', 'food_type', 'food_link'])
    csv_write.writeheader()
    csv_write.writerows(food_list)
```

6. 程序实现与总结

我们先回顾一下这个项目是要实现什么功能：

1）通过循环获取网站前三类食物前三页食品列表页的链接；

2）请求网页并解析网页内容，提取食品类别、食品名、食品热量、食品链接；

3）将食品的所有信息写进【foods.csv】文件中。

ok，逻辑也梳理完了，下面请你独立完成这个项目吧。别担心，我会给你提示信息，辅助你完成项目。

6.2 知识归纳与总结

本节课主要以案例练习为主，新增的知识点不多，只有以下几个：

1）多属性值的 class 属性
a.概念：一个元素的 **class** 属性可以包含多个属性值，属性值之间以空格隔开；
b.示例：class='widget-food-list pull-right'。

2）字符串的 strip() 方法
a.功能：移除字符串头尾指定的单个或多个指定字符，返回一个新的字符串。默认情况是移除空白符（比如空格符或换行符）；
b.语法：str1.strip()。

3）
find_all() 方法中，参数 HTML 元素属性的 class 属性
a.作用：可以匹配一到多个属性值
    i.匹配一个属性值
        ① 语法：find_all(HTML 元素名, class_='属性值1')；
        ② 示例：find_all('div', class_='text-box')。
    ii.匹配多个属性值，属性值之间用空格隔开
        ① 语法：find_all(HTML 元素名, class_='属性值1 属性值2')；
        ② 示例：find_all('div', class_='widget-food-list pull-right')。

最后，是我们本节课使用到的知识总结图：


爬取薄荷网
HTML
多个属性值的 class 属性
概念：HTML 的 class 属性可以包含多个属性值，属性值之间以空格隔开
示例：class='widget-food-list pull-right'
字符串
strip() 方法
功能：移除字符串头尾指定的单个或多个指定字符，返回一个新的字符串，默认是移除空格符或换行符
语法：str1.strip()
爬取流程
获取网页
Requests 库
requests.get()
解析网页
BeautifulSoup 库
BeautifulSoup(html, 'html.parser')
参数 html：HTML 文档的字符串格式
参数 html.parser：一种 Python 内置的解析器
Tag 对象
Tag.text：可获取 Tag 中的所有文本内容
Tag['属性名']：提取对应元素中属性的内容
find_all() 方法
参数：HTML 元素名，搜索所有元素名匹配的 Tag 对象
参数：HTML 元素属性，
可以依据 HTML 元素的属性（如 id, class, href）来搜索对应的 Tag 对象
class 属性
作用：可以匹配一到多个属性值
1. 匹配1个属性值
语法：find_all(HTML 元素名, class_='属性值1')；
示例：find_all('div', class_='text-box')
2. 匹配多个属性值（属性值之间用空格隔开）
语法：find_all(HTML 元素名, class_='属性值1 属性值2')
示例：find_all('div', class_='widget-food-list pull-right')
注意：因为 class 是 Python 保留字，class 属性要写作：'class_=xxx'
find() 方法
语法和参数跟 find_all() 的相同
与 find_all() 不同，只返回符合参数条件的第一个 Tag 对象
存储数据
csv 模块
csv.DictWriter(f, fieldnames)
参数 f：使用 open() 函数打开的 csv 文件对象
参数 fieldnames：用来设置文件的表头
DictWriter 对象
writeheader()
可以将 fieldnames 写入 csv 的第一行
writerows(rows)
参数 rows 是可迭代对象，该可迭代对象必须由字典组成