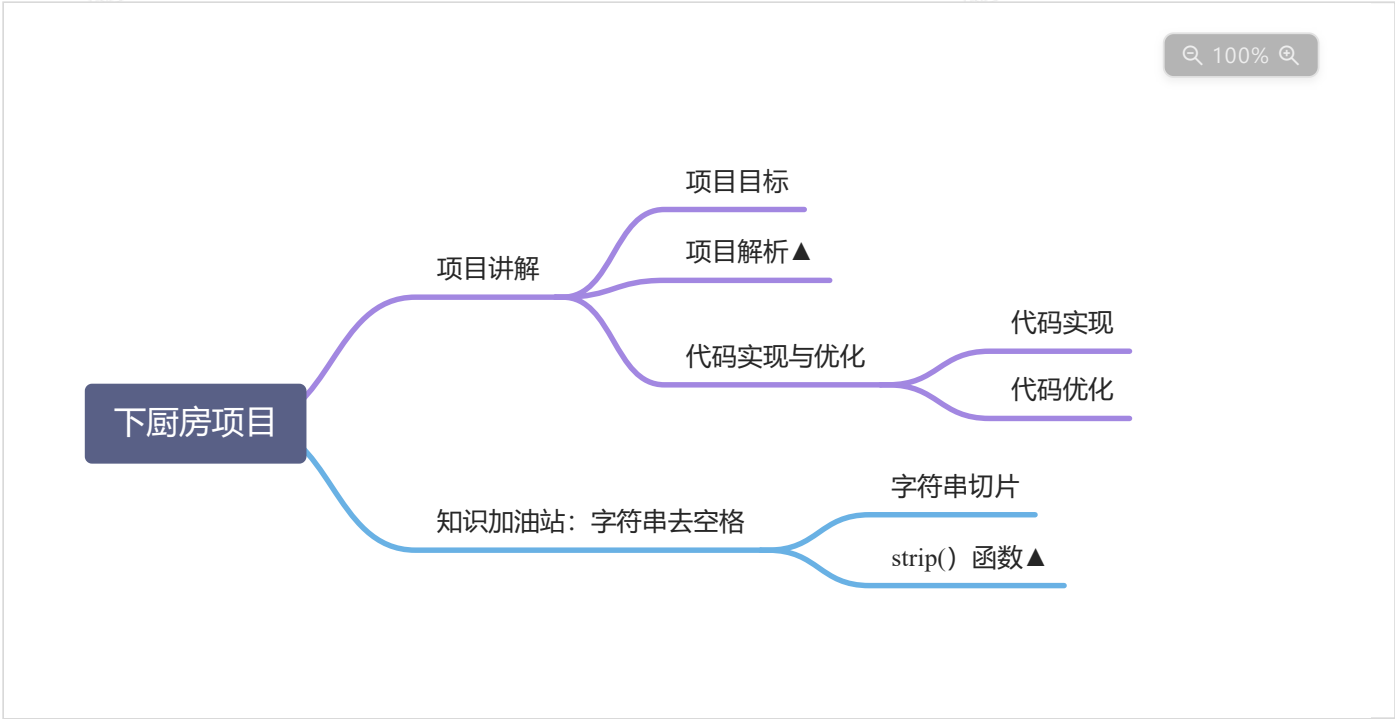


第3课 项目实操：下厨房项目

一、课程结构导图



注：▲为重点知识点。

二、项目讲解

2.1 项目目标

爬取目标：爬取热门菜谱清单，内含：菜名、原材料、详细烹饪流程的URL。

目标网址：<http://www.xiachufang.com/explore/> <<http://www.xiachufang.com/explore/>>

2.2 项目解析

- 查看网页的robots协议
- 1. 先明确需要爬取的数据所在的网页，然后查看该网页的robots协议，确认哪一些数据可以爬取。

2. 在根目录加上robots.txt查看协议，确认 /explore/ 不在禁止爬取的列表内。

目标网址: <http://www.xiachufang.com/explore/>

根目录: <http://www.xiachufang.com/> + [robots.txt](#)

查看协议: <http://www.xiachufang.com/robots.txt>

• 获取数据

使用requests.get()发起请求，获取网页数据。使用status_code属性查看是否请求成功。请求成功之后，使用text属性把相应的网页内容传给BeautifulSoup4库进行解析。

• 解析网页

打开检查工具，通过小箭头进行定位，查看数据所在的标签（多查找几个找到共同规律）。

菜名和详细烹饪流程的URL:

The screenshot shows the Xiaochufang website interface on the left and its HTML structure in a browser's developer tools on the right. Red arrows and boxes highlight the relationship between the recipe names on the website and the HTML tags in the source code.

Website Interface (Left):

- Header: 下厨房 (Xiaochufang), 搜索菜谱、食材 (Search recipes, ingredients), 搜索菜谱 (Search recipes), 首页 (Home), 菜谱分类 (Recipe categories)
- Left Sidebar: 全部分类 (All categories), 本周最受欢迎 (Most popular this week), 新秀菜谱 (New recipes), 往期头条 (Past headlines), 厨房101 (Kitchen 101), 月度最佳 (Monthly best), 最新创建 (Latest creations), 流行菜单 (Popular menus), 流行搜索 (Popular searches), 茄子 (Eggplant), 土豆 (Potato), 家常菜 (Home-style dishes), 金针菇 (Enoki mushrooms), 白菜 (Bok choy)
- Main Content: 本周最受欢迎 (Most popular this week). It lists several recipes with images and titles: 手撕炼乳吐司 (Hand-torn milk bread), 低脂低卡!!鲜嫩好吃不长胖的西兰花炒菌菇 (Low fat, low calorie!! Tender and delicious broccoli stir-fry with mushrooms), 10分钟搞定, 好吃到舔手指的油焖大虾 (10 minutes to get it done, delicious enough to lick your fingers, braised large shrimp).

Developer Tools (Right):

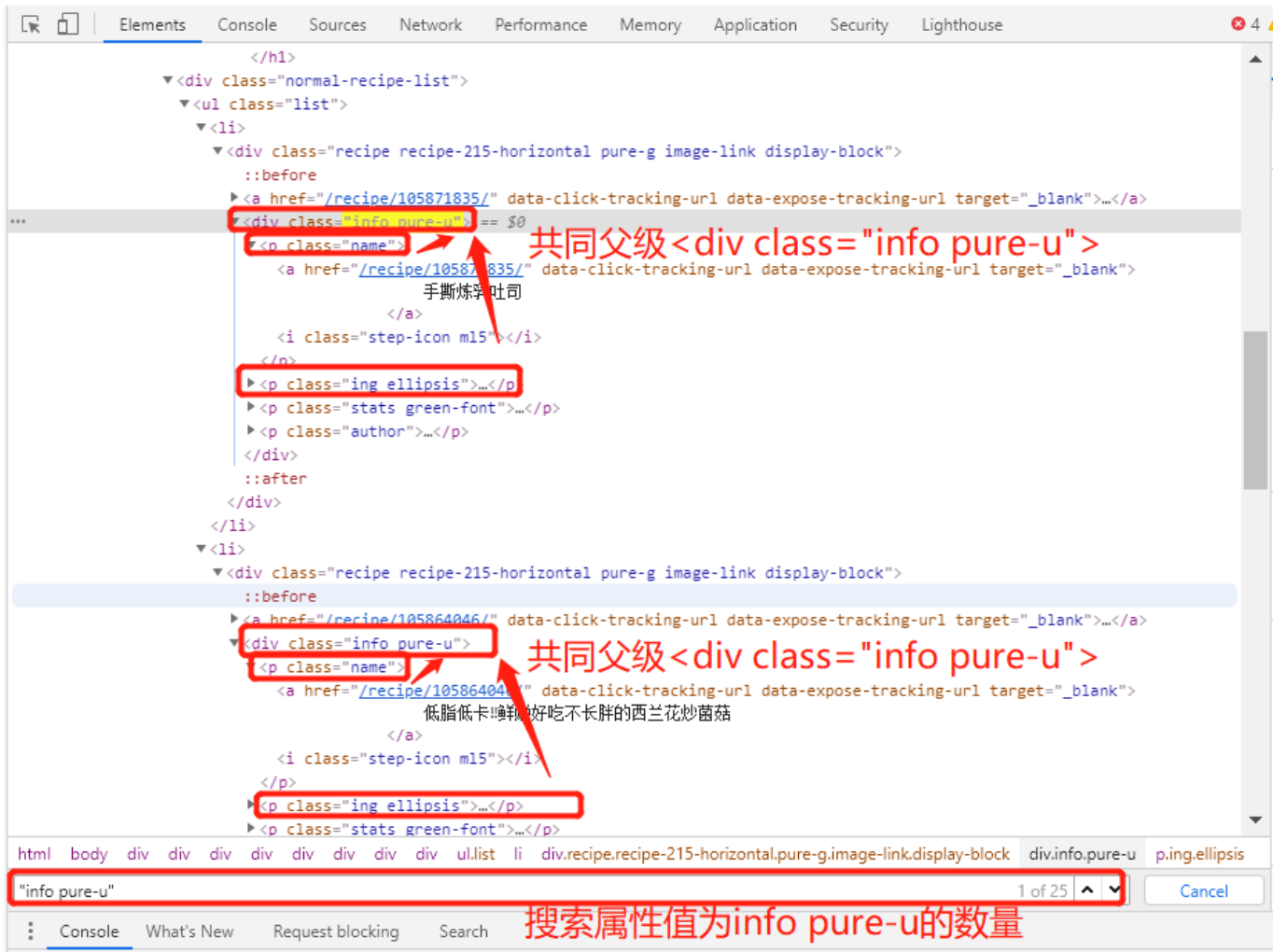
- The 'Elements' panel shows the HTML structure of the recipe listings.
- Red boxes and arrows highlight the following patterns:
 - For the recipe '手撕炼乳吐司' (Hand-torn milk bread), the HTML shows a `<p class="name">` tag containing an `` tag. The text '父级是<p class="name">' (Parent is <p class="name">) points to the `<p class="name">` tag.
 - For the recipe '低脂低卡!!鲜嫩好吃不长胖的西兰花炒菌菇' (Low fat, low calorie!! Tender and delicious broccoli stir-fry with mushrooms), the HTML shows a `<p class="name">` tag containing an `` tag. The text '父级是<p class="name">' (Parent is <p class="name">) points to the `<p class="name">` tag.
 - For the recipe '10分钟搞定, 好吃到舔手指的油焖大虾' (10 minutes to get it done, delicious enough to lick your fingers, braised large shrimp), the HTML shows a `<p class="name">` tag containing an `` tag. The text '父级是<p class="name">' (Parent is <p class="name">) points to the `<p class="name">` tag.
- A red box at the bottom of the developer tools highlights the search bar, with the text '调出搜索框, 查看该属性值的个数' (Open the search bar, check the number of values for this attribute).

通过定位可以发现，这些数据都在<p class="name">标签下的<a>标签中，菜名是<a>标签内的文本，URL是<a>标签里属性href的值，不过这里的url只是一部分网址，需要复制前半部分网址做拼接才能得到完整的网址。而且<p class="name">的个数刚好和左边菜名的个数一致，都是25个。

原材料:

通过定位可以发现，原材料的数据都在<p class="ing ellipsis">标签下的所有标签中，而且该标签的数量和菜名的数量相同（和URL查找方法一样）。

共同的父级标签：



在藏有需要爬取的数据中的两个标签：<p class="name">和<p class="ing ellipsis">中，我们往上一级查看，发现数据有一个共同的父级标签<div class="info pure-u">，而且该标签的数量和菜名的数量相同。

• 提取数据

找到了数据的藏身之处之后，我们可以开始爬取试试。基于刚刚的分析，我们产生了两种写爬虫的思路：

思路一：我们先去爬取所有的最小父级标签<div class="info pure-u">，然后针对每一个父级标签，提取里面的菜名、URL、食材；

思路二：我们分别提取所有的菜名、所有的URL、所有的食材。然后让菜名、URL、食材给一一对应起来（这并不复杂，第0个菜名，对应第0个URL，对应第0组食材，按顺序走即可）。

• 存放数据

写一个for循环，将爬取到的数据都存放到列表中。

2.3 代码实现与优化

2.3.1 代码实现

- 解法一：提取最小父级标签，取出每一组的数据。

- 获取和解析网页数据；

```
1 # 引用requests库
2 import requests
3 # 引用BeautifulSoup库
4 from bs4 import BeautifulSoup
5
6 # 获取数据
7 res_foods = requests.get('http://www.xiachufang.com/explore/')
8 # 解析数据
9 bs_foods = BeautifulSoup(res_foods.text, 'html.parser')
```

- 提取数据（该部分代码和上面代码组合才可正常运行）；

```
1 # 查找最小父级标签
2 list_foods = bs_foods.find_all('div', class_='info pure-u')
3
4 # 提取第0个父级标签中的<a>标签
5 tag_a = list_foods[0].find('a')
6 name = tag_a.text # 菜名
7 URL = 'http://www.xiachufang.com'+tag_a['href'] # 获取URL
8
9 # 提取第0个父级标签中的<p>标签
10 tag_p = list_foods[0].find('p', class_='ing ellipsis')
11 ingredients = tag_p.text # 食材
12
13 print([name, URL, ingredients]) # 打印菜名、URL、食材
```

- 利用循环遍历提取所有数据，将每一组数据取出，作为一个小列表存放到大列表中。

```
1 # 创建一个空列表，用于存储信息
2 list_all = []
3
4 for food in list_foods:
5     tag_a = food.find('a') # 提取第0个<a>标签
6     name = tag_a.text # 菜名
7     URL = 'http://www.xiachufang.com'+tag_a['href'] # 获取URL
8     tag_p = food.find('p', class_='ing ellipsis') # 提取第0个父级标签中的<p>标签
```

```

9     ingredients = tag_p.text    # 食材
10    list_all.append([name,URL,ingredients]) # 将菜名、URL、食材，封装为列表，添加进list_all
11
12    print(list_all) # 打印所有信息

```

- **解法二：分别提取所有的菜名、URL和食材的数据，再一一对应拼接。**

- 获取和解析网页数据（同解法一）；
- 分别提取所有数据；

```

1 # 查找包含菜名和URL的<p>标签
2 tag_name = bs_foods.find_all('p',class_='name')
3 # 查找包含食材的<p>标签
4 tag_ingredients = bs_foods.find_all('p',class_='ing ellipsis')

```

- 利用循环遍历提取所有数据，将每一组数据进行拼接，作为一个小列表存放到大列表中。

```

1 list_all = []          # 创建一个空列表，用于存储信息
2
3 for x in range(len(tag_name)): # 启动一个循环，次数等于菜名的数量
4     name = tag_name[x].text    # 菜名
5     URL = tag_name[x].find('a')['href']    # 链接
6     ingredients = tag_ingredients[x].text    # 食材
7     list_all.append(list_food) # 将菜名、URL、食材，封装为列表，添加进list_all
8
9 print(list_all)        # 打印所有信息

```

- **两种解法对比：**

- 如果我们需要爬取的数据，在网站上每一组数据中的每一个数据都只有一个，不多出也不缺失，那么，这两种方法都可以使用，没有差别。
- 但是在实际的应用中，因为网站的数据本身可能存在一些缺陷，可能会有一些数据确实或者某一个数据不止一个，会使得匹配的数据出现错位。找出每一类数据然后一一对应匹配的错误率会比较高，而找出每一个父级标签，可以比较好的控制这样的错误。所以实战中更推荐使用通过最小父级标签来爬取数据。

2.3.2 代码优化

代码打印的结果存在很多空格和换行符，需要对这些内容进行处理，下面利用两种方法来分别处理。

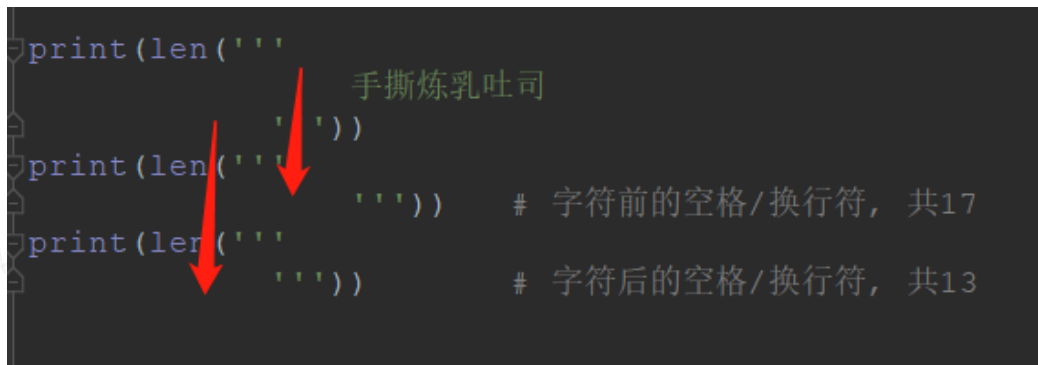
- **使用字符串切片**

- 菜名

- 方法一：通过在网页端双击属性值计算。



- 方法二：复制字符串，使用len()函数+三引号来计算。



- 代码做以下修改（注：解法二是在父级<p>往下取子级<a>的内容，所以需要多切去两边的一个空格。）

解法一：

解法二：

name = tag_a.text

tag_name[x].text



name = tag_a.text[17:-13]

tag_name[x].text[18:-14]

- 食材：在父级往下取子级的内容时，会在两端各多出一个空行，需要把它切去。

解法一：

解法二：

ingredients = tag_p.text

tag_ingredients[x].text



ingredients = tag_p.text[1:-1]

tag_ingredients[x].text[1:-1]

- 使用strip()函数

菜名

解法一：

```
name = tag_a.text
```



```
name = tag_a.text.strip()
```

解法二：

```
tag_name[x].text
```



```
tag_name[x].text.strip()
```

食材

解法一：

```
ingredients = tag_p.text
```



```
ingredients = tag_p.text.strip()
```

解法二：

```
tag_ingredients[x].text
```



```
tag_ingredients[x].text.strip()
```

优化后代码如下：

- 解法一：查找最小父级标签，再将每一组数据提取出来。

```
1 # 引入模块
2 import requests
3 from bs4 import BeautifulSoup
4 # 获取数据
5 url = 'http://www.xiachufang.com/explore'
6 # 定义请求头，伪装为浏览器，放在get()前面即可。
7 headers={'user-agent':'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36'}
8 # 把headers传递给get()函数的headers参数，获取数据
9 res_foods = requests.get(url,headers = headers)
10
11 # 解析数据
12 bs_foods = BeautifulSoup(res_foods.text,'html.parser')
13 # 提取数据
14 list_foods = bs_foods.find_all('div',class_='info pure-u') # 获取最小父级标签
15 list_all = []
16 for food in list_foods:
17     tag_a = food.find('a')
18     name = tag_a.text[17:-13]
19     URL = 'http://www.xiachufang.com'+tag_a['href']
20     tag_p = food.find('p',class_='ing ellipsis')
21     ingredients = tag_p.text[1:-1]
```

```
22 # 存储数据
23 list_all.append([name,URL,ingredients])
24
25 print(list_all) # 打印所有信息
```

- 解法二：分别提取所有的菜名、URL和食材的数据，再一一对应拼接。

```
1 import requests
2 from bs4 import BeautifulSoup
3 # 获取数据
4 url = 'http://www.xiachufang.com/explore'
5 # 定义请求头，伪装为浏览器，放在get()前面即可。
6 headers={'user-agent':'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, li
7 # 把headers传递给get()函数的headers参数，获取数据
8 res_foods = requests.get(url,headers = headers)
9 # 解析数据
10 bs_foods = BeautifulSoup(res_foods.text,'html.parser')
11 # 提取数据，分别提取所有数据
12 tag_name = bs_foods.find_all('p',class_='name')
13 tag_ingredients = bs_foods.find_all('p',class_='ing ellipsis')
14 list_all = []
15 for x in range(len(tag_name)):
16     name = tag_name[x].text[18:-14]
17     URL = 'http://www.xiachufang.com'+tag_name[x].find('a')['href']
18     ingredients = tag_ingredients[x].text[1:-1]
19     # 存储数据
20     list_all.append([name,URL,ingredients])
21
22 print(list_all) # 打印所有信息
```

三、知识加油站：字符串去空格

3.1 字符串切片

概念：取出某个字符串中特定的部分。

语法：字符串[]；字符串切片和列表的切片原理是一样的。有正向取值和负向取值，都遵循左取右不取的规则。

示例：


```
1 content = '今晚吴枫下厨房，给大家做美食啦~'
2 print(content[2:4])      # 结果：吴枫
3 print(content[-4:-2])    # 结果：美食
4 print(content[2:-1])     # 结果：吴枫下厨房，给大家做美食啦
```

3.2 strip()函数

概念：是一个可以把字符串两端的空格和换行符去掉的函数。

风变科技

语法：字符串.strip()。

示例：

```
1 content = ' \n学霸的笔记 '
2 print(content.strip()) # 结果：学霸的笔记
```