数据提取之json

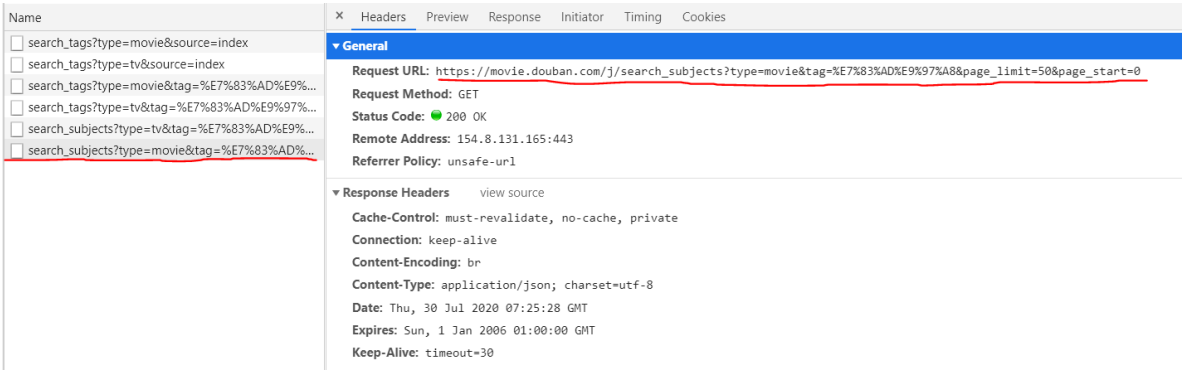
> **目标**：理解json的概念
>
> 了解爬虫中，json出现的位置
>
> 掌握json相关的方法

1. 什么是json，在哪里能找到json

json是一种轻量级的数据交换格式，它使地人们很容易的进行阅读和编写，同事也方便了机器进行解析和生成，适用于进行数据交互的场景，比如web的前台和后台之间的数据交互

至于在哪里能找到能返回json数据的url，以豆瓣电影为例，下面这条url就是返回json数据的url

https://movie.douban.com/j/search_subjects?type=movie&tag=%E7%83%AD%E9%97%A8&page_limit=50&page_start=0



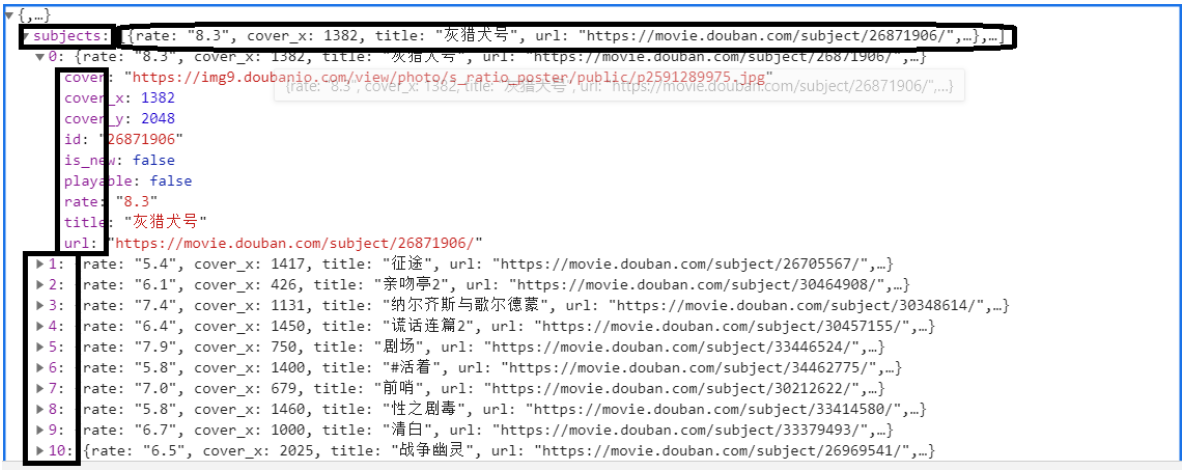
在url地址对应的响应中搜索关键字即可

但是注意:url地址对应的响应中，中文往往是被编码之后的内容，所以更推荐大家去搜索英文和数字；另外一个方法就是在perview中搜索，其中的类容都是转码之后的

还有一个方法，就是将浏览器切换到手机版寻找json

2. json数据格式化

在preview中观察



其中:

方形方框表示json中的健

椭圆框表示健所对应的值，是一个列表，在列表展开之后，下面的数字表示列表中所对应的值

在线解析工具进行解析

json.cn(在线json数据解析，让数据直观易读)

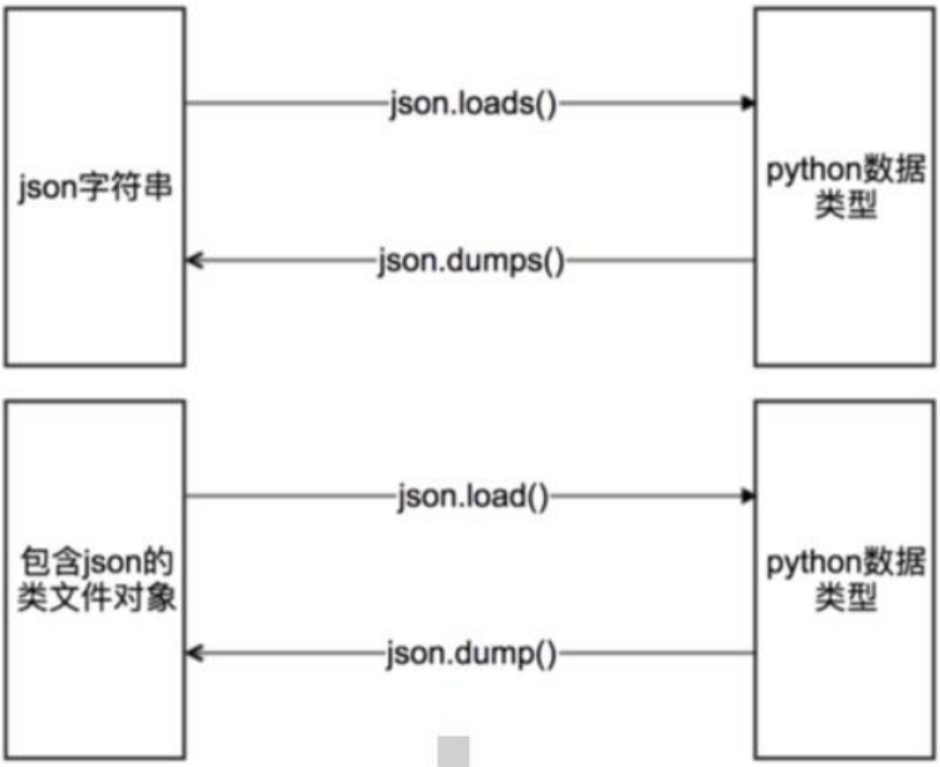
pycharm进行reformat code

在pycharm中新建一个json文件，把数据存入后，点击code下面的reformat code，但是中文往往显示的是unicode格式

json数据的其他来源：

抓包app，app的抓包方式会在后面学习，但是很多时候app中的数据是被加密的，但仍然值得尝试

3. json模块中方法的学习



json.dumps

dump的功能就是把Python对象encode为json对象，一个编码过程。注意json模块提供了json.dumps和json.dump方法，区别是dump直接到文件，而dumps到一个字符串，这里的s可以理解为string。

```
import json
data = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]
print('DATA:', repr(data))

data_string = json.dumps(data)
print('JSON:', data_string)
#结果如下
DATA: [{'a': 'A', 'c': 3.0, 'b': (2, 4)}]
JSON: [{"a": "A", "c": 3.0, "b": [2, 4]}]
print type(data)
print type(data_string)
<type 'list'>
<type 'str'>
```

json.dump

不仅可以把Python对象编码为string，还可以写入文件。因为我们不能把Python对象直接写入文件，这样会报错TypeError: expected a string or other character buffer object，我们需要将其序列化之后才可以

```
import json
data = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]
with open('output.json','w') as fp:
    json.dump(data,fp)
# 结果
[{“a”: “A”, “c”: 3.0, “b”: [2, 4]}]
```

json.loads

从Python内置对象dump为json对象我们知道如何操作了，那如何从json对象decode解码为Python可以识别的对象呢？是的用json.loads方法，当然这个是基于string的，如果是文件，我们可以用json.load方法。

```
decoded_json = json.loads(data_string)
# 和之前一样，还是list
print type(decoded_json)
<type 'list'>
# 像访问 data = [ { 'a':'A', 'b':(2, 4),
'c':3.0 } ]一样
print decoded_json[0]['a']
#结果如下
A
```

json.load

可以直接load文件

```
with open('output.json') as fp:
    print type(fp)
    loaded_json = json.load(fp)

<type 'file'>
# 和之前一样，还是list
print type(decoded_json)
<type 'list'>
# 像访问 data = [ { 'a':'A', 'b':(2, 4),
'c':3.0 } ]一样
print decoded_json[0]['a']

#结果如下
A
```

4. json.dumps常用参数

一些参数，可以让我们更好地控制输出。常见的比如sort_keys，indent，separators，skipkeys等。

sort_keys名字就很清楚了，输出时字典的是按键值排序的，而不是随机的。

```
import json
data = [ { 'a':'A',  'c':3.0 ,'b':(2, 4)} ]
print('DATA:', data)

unsorted = json.dumps(data)
print('JSON:', json.dumps(data))
print('SORT:', json.dumps(data,
sort_keys=True))
#结果如下
DATA: [{'a': 'A', 'c': 3.0, 'b': (2, 4)}]
JSON: [{"a": "A", "c": 3.0, "b": [2, 4]}]
SORT: [{"a": "A", "b": [2, 4], "c":
3.0}]1234567891011
```

indent就是更个缩进，让我们更好地看清结构。

```
import json
data = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]
print('DATA:', repr(data))

print('NORMAL:', json.dumps(data,
sort_keys=True))
print('INDENT:', json.dumps(data,
sort_keys=True, indent=2))

#结果如下
DATA: [{'a': 'A', 'b': (2, 4), 'c': 3.0}]
NORMAL: [{"a": "A", "b": [2, 4], "c": 3.0}]
INDENT: [
  {
```

```
    "a": "A",
    "b": [
      2,
      4
    ],
    "c": 3.0
  }
12345678910111213141516171819 20
```

separators是提供分隔符，可以出去白空格，输出更紧凑，数据更小。默认的分隔符是(', ', ': ')，有白空格的。不同的dumps参数，对应文件大小一目了然。

```
import json
data = [ { 'a':'A', 'b':(2, 4), 'c':3.0 } ]
print('DATA:', repr(data))
print('repr(data)                :',
len(repr(data)))
print('dumps(data)               :',
len(json.dumps(data)))
print('dumps(data, indent=2)   :',
len(json.dumps(data, indent=2)))
print('dumps(data, separators):',
len(json.dumps(data, separators=
(',',':'))))

#结果如下
DATA: [{'a': 'A', 'c': 3.0, 'b': (2, 4)}]
repr(data)                : 35
dumps(data)               : 35
```

```
dumps(data, indent=2)  : 76
dumps(data, separators):
291234567891011121314
```

json需要字典的的键是字符串，否则会抛出 ValueError。

```
import json
data = [ { 'a':'A', 'b':(2, 4), 'c':3.0,
('d',):'D tuple' } ]

print('First attempt')
try:
    print(json.dumps(data))
except (TypeError, ValueError) as err:
    print('ERROR:', err)

print()
print('Second attempt')
print(json.dumps(data, skipkeys=True))

#结果如下
First attempt
ERROR: keys must be a string

Second attempt
[{"a": "A", "c": 3.0, "b": [2, 4]}]
```

5. 案例

目标网站：http://www.whggzy.com

需求：获取前五页的标题跟时间

> 先把第一页的数据获取下来
>
> 目标url：http://www.whggzy.com/portal/category
>
> 请求方式：post请求
>
> 获取的数据：json格式的
>
> 后面几页的数据通过循环重复获取就可以

```
import requests
import time


url =
'http://www.whggzy.com/front/search/category'

# 我们要获取前5页的数据，每一页数据的获取方式是差不多，我是选择每一页的数据获取获取一块
# 代码，还是怎样处理呢--循环
for i in range(1, 5):  # 那这个循环里面的内容是填啥，除了url不变化，其他代码都放循环里面
    # pageNo跟i变量值是对应的
```

```
    data = {"utm":
"sites_group_front.2ef5001f.0.0.f1fc8e604de
a11ed833cefd6ff5c60af",
            "categoryCode":
"GovernmentProcurement", "pageSize": 15,
"pageNo": i}
    res = requests.post(url, json=data)
    result = res.json()['hits']['hits']
    for r in result:
        source = r['_source']
        title = source['title']
        publishDate = source['publishDate']
        tre_timeArray =
time.localtime(publishDate / 1000)
        tre_otherStyleTime =
time.strftime("%Y-%m-%d %H:%M:%S",
tre_timeArray)
        print(title, tre_otherStyleTime)
```

数据提取之jsonpath模块

> 第三方模块：pip install jsonpath -i 清华源

使用场景：多层嵌套的复杂的字典，直接提取数据

方法:

ret = jsonpath(a, 'jsonpath语法规则字符串')　#a是需要提取数据的目标字典

常用节点：

$ 根节点（最外层的大括号）

. 字节点（）

.. 内部任意位置，子孙节点

XPath	JSONPath	描述
/	$	根节点
.	@	现行节点
/	.or[]	取子节点
..	n/a	取父节点，Jsonpath未支持
//	..	就是不管位置，选择所有符合条件的条件
*	*	匹配所有元素节点
@	n/a	根据属性访问，Json不支持，因为Json是个Key-value递归结构，不需要属性访问。
[]	[]	迭代器标示（可以在里边做简单的迭代操作，如数组下标，根据内容选值等）
|	[,]	支持迭代器中做多选。
[]	?()	支持过滤操作.
n/a	()	支持表达式计算
()	n/a	分组，JsonPath不支持

示例1：

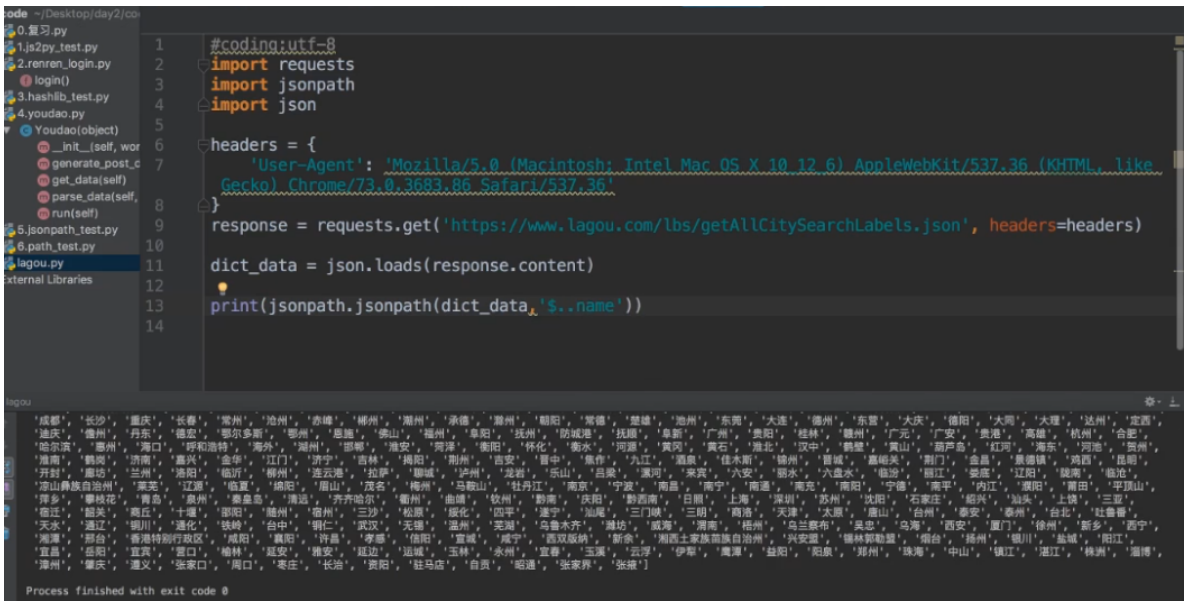
```
#coding:utf-8
from jsonpath import jsonpath

data = {'key1':{'key2':{'key3':{'key4':{'key5':{'key6':'python'}}}}}}

# print(data['key1']['key2']['key3']['key4']['key5']['key6'])

# jsonpath的结果为列表，获取数据需要索引
#print(jsonpath(data, '$.key1.key2.key3.key4.key5.key6')[0])
print(jsonpath(data, '$..key6')[0])
```

示例2：



数据保存——CSV

> csv模块：内置模块
>
> 对表格文件进行读取/写入
>
> 写入
>
> [(),(),()]
>
> [{},{},{}]

utf-8与utf-8-sig的区别

> utf-8以字节为编码单元，他的字节顺序在所有系统都是一样的，没有字节序问题，也因此他实际也不需要BOM
>
> utf-8-sig中的sig为signature,即带签名的utf-8(utf-8 with BOM)
>
> BOM全称ByteOrder Mark,字节顺序标记，出现在文本文件头部，Unicode编码标准中用于标识文件是采用那种格式的编码

为什么写入csv文件要用utf-8-sig编码

> Excel 在读取csv文件的时候是通过读取文件头上的BOM来识别编码的，如果文件头无BOM信息，则默认按照Unicode编码读取
>
> 当外面使用utf-8编码生成csv文件的时候，并没有生成BOM信息，Excel就会自动按照Uni