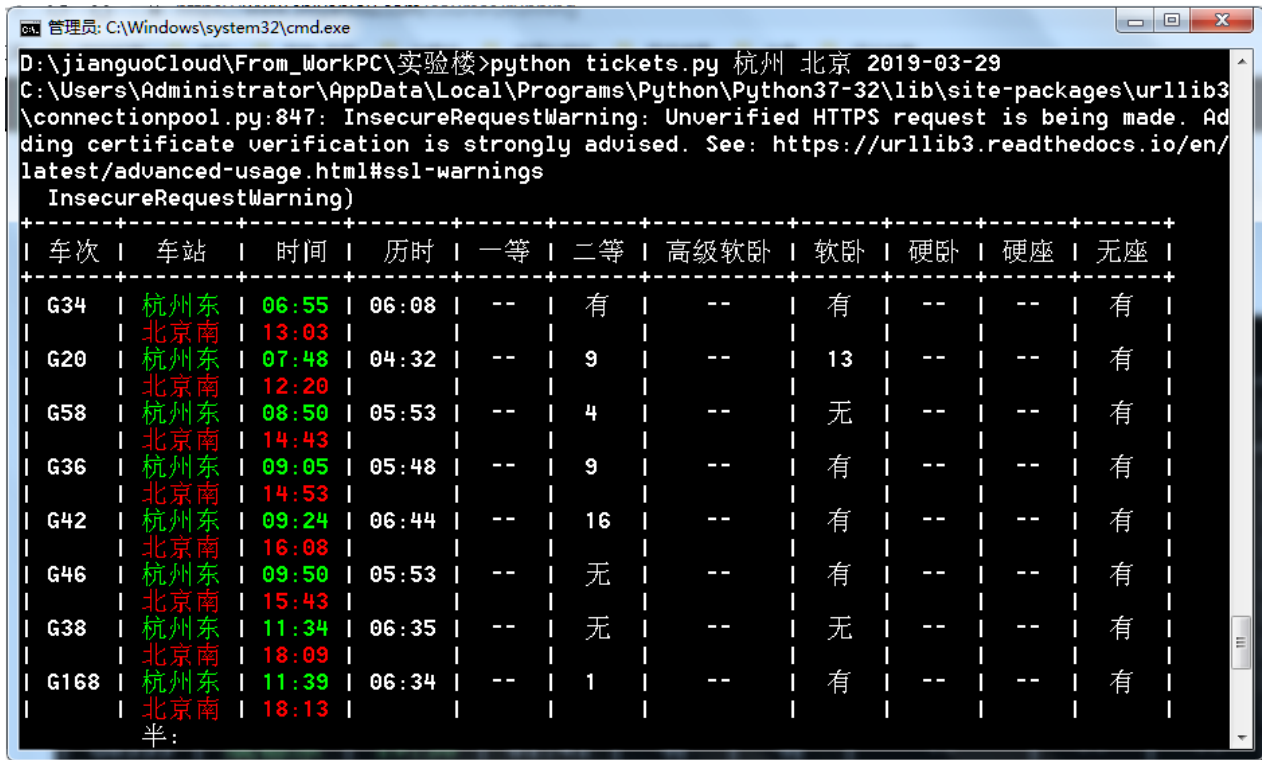


一、引言

学了 Python 基础语法后想锻炼一下解决实际问题的水平，那本文介绍的小项目刚刚好。通过本文中项目，你可以学会不打开12306或者APP就能查询火车票，同时熟悉 Python 库的使用方法，了解 Python 程序设计思路。

这个项目来源于 [此处](#)，本文是在学习后进行的总结，最终效果如图所示。



```
cmd 管理员: C:\Windows\system32\cmd.exe
D:\jianguoCloud\From_WorkPC\实验楼>python tickets.py 杭州 北京 2019-03-29
C:\Users\Administrator\AppData\Local\Programs\Python\Python37-32\lib\site-packages\urllib3\connectionpool.py:847: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
InsecureRequestWarning)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 车次 | 车站 | 时间 | 历时 | 一等 | 二等 | 高级软卧 | 软卧 | 硬卧 | 硬座 | 无座 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| G34 | 杭州东 | 06:55 | 06:08 | -- | 有 | -- | 有 | -- | -- | 有 | |
| G20 | 杭州东 | 13:03 | 04:32 | -- | 9 | -- | 13 | -- | -- | 有 |
| G58 | 杭州东 | 07:48 | 05:53 | -- | 4 | -- | 无 | -- | -- | 有 |
| G36 | 杭州东 | 12:20 | 05:48 | -- | 9 | -- | 有 | -- | -- | 有 |
| G42 | 杭州东 | 09:05 | 06:44 | -- | 16 | -- | 有 | -- | -- | 有 |
| G46 | 杭州东 | 14:53 | 05:53 | -- | 无 | -- | 有 | -- | -- | 有 |
| G38 | 杭州东 | 15:43 | 06:35 | -- | 无 | -- | 无 | -- | -- | 有 |
| G168 | 杭州东 | 18:09 | 06:34 | -- | 1 | -- | 有 | -- | -- | 有 |
| 半: | 北京南 | 11:39 | | | | | | | | | |
| 半: | 北京南 | 18:13 | | | | | | | | | |
```

本项目需要 Python 基础语法知识，在项目中可以学习到如下内容：

- docopt 、 requests 、 colorama 和 prettytable 库的使用
- 命令行安装需要的 Python 库
- 分析网站的请求和响应过程

二、设计思路

为了实现如图所示功能效果，我们需要先进行需求分析。

- 当我们查询火车票时一般来说需要指定条件，例如列车时间、起始站点等信息才能查看列车时刻信息，因此，我们需要明确定义有哪些查询条件；
- 然后需要有命令行界面，用来接受用户输入的查询条件；
- 根据用户输入的查询条件，去寻找列车时刻信息；
- 最后将获得的列车时刻信息经过排版再显示出来。

这样，总结下来可以分为如下几个大的模块

- 命令行界面，用于接受用户输入的查询条件
- 列车信息API，向12306服务器发起查询请求，并获得列车信息
- 列车信息展示，美观的展示出查询到达列车信息

三、知识点

1.项目所需 Python 库

- `docopt` 库：终端命令行参数解析器，[查看详情](#)；
- `requests` 库：简单易用的Python HTTP库，[快速上手文档](#)；
- `prettytable` 库：格式化输出打印内容，[查看详情](#)；
- `colorama` 库：命令行着色工具，[查看详情](#)；

2.命令行一键安装

通过如下命令一键安装以上资源：

```
pip3 install requests prettytable docopt colorama
```

四、分步讲解

下面开始构建属于自己的命令行列车时刻查询工具。

1.命令行界面

```
# coding: utf-8

"""命令行火车票查看器

Usage:
    tickets [-gdtkz] <from> <to> <date>

Options:
    -h, --help    显示帮助菜单
    -g            高铁
    -d            动车
    -t            特快
    -k            快速
    -z            直达

Example:
    tickets 杭州 北京 2019-03-25
    tickets -dg 杭州 北京 2019-03-25
"""

from docopt import docopt

def cli():
```

```
"""command-line interface"""
arguments = docopt(__doc__)
print(arguments)

if __name__ == '__main__':
    cli()
```

根据官方文档说明，`docopt` 需要1个必传参数和4个可选参数：

```
docopt(doc, argv=None, help=True, version=None, options_first=False)
```

- `doc` 是由字符串或者注释构成的命令行模块，格式如以上代码所示（`doc` could be a module docstring (`doc`) or some other string that contains a help message that will be parsed to create the option parser）。
- 其返回值类型为字典，也就是 `arguments`，键为 `Usage` 和 `Options` 中选项，值为实际输入内容。
- 实际运行下如上代码，看到如下所示返回结果内容。

```
管理员: 命令提示符
4 个文件      7,330 字节
11 个目录 17,980,772,352 可用字节

C:\>python tickets.py
Usage:
    tickets [-gdtkz] <from> <to> <date>

C:\>python tickets.py -h
命令行火车票查看器

Usage:
    tickets [-gdtkz] <from> <to> <date>

Options:
    -h, --help    显示帮助菜单
    -g            高铁
    -d            动车
    -t            特快
    -k            快速
    -z            直达

Example:
    tickets 杭州 北京 2019-03-25
    tickets -dg 杭州 北京 2019-03-25

C:\>python tickets.py -gd 北京 杭州 2019-03-25
{'-d': True,
 '-g': True,
 '-k': False,
 '-t': False,
 '-z': False,
 '<date>': '2019-03-25',
 '<from>': '北京',
 '<to>': '杭州'}

C:\>_

半:
```

2. 获取列车信息

查火车票只能去12306，但是铁道部没有提供API，那么我们就不能查询了吗，答案是否定的。我们现在在网站上查询几次列车信息，寻找接口的蛛丝马迹。

2.1 分析页面接口

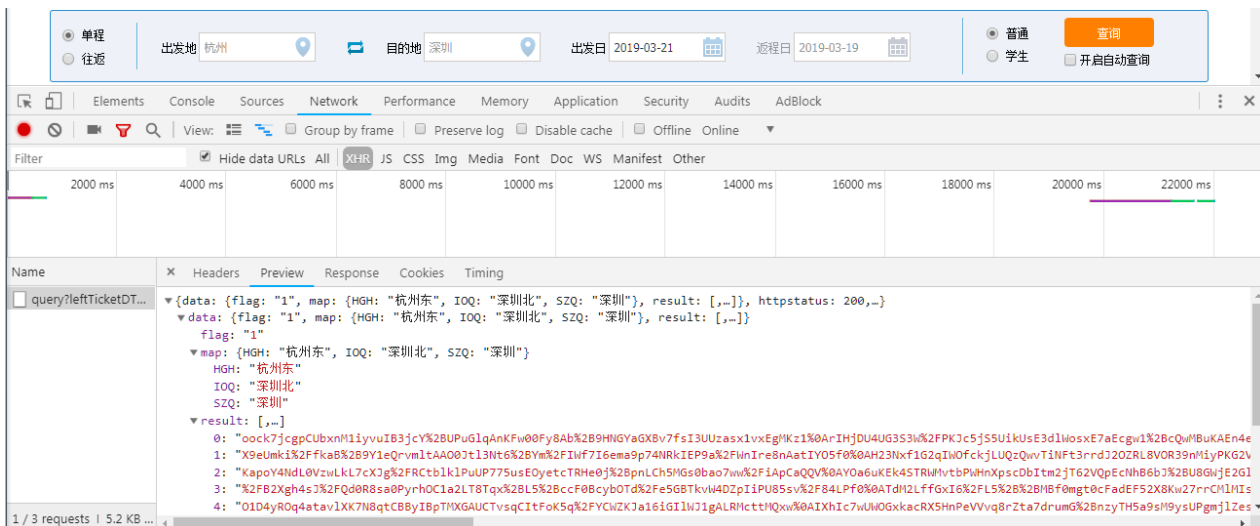
- 在列车查询页面，然后打开浏览器调试模式（Windows下Chrome浏览器快捷键是F12），进入 **Network** 中，查看 **XHR**，然后查询列车信息

The screenshot shows the China Railway 12306 website interface. The search criteria are: Departure: 杭州 (Hangzhou), Destination: 深圳 (Shenzhen), Date: 2019-03-21. The '查询' (Search) button is highlighted with a red box. Below the website, the browser's developer tools are open, showing a network request. The request name is 'query?leftTicketDTO.train_date=2019-03-21&leftTicketDTO.to_station=SZQ'. A red arrow points from the '查询' button to this request.

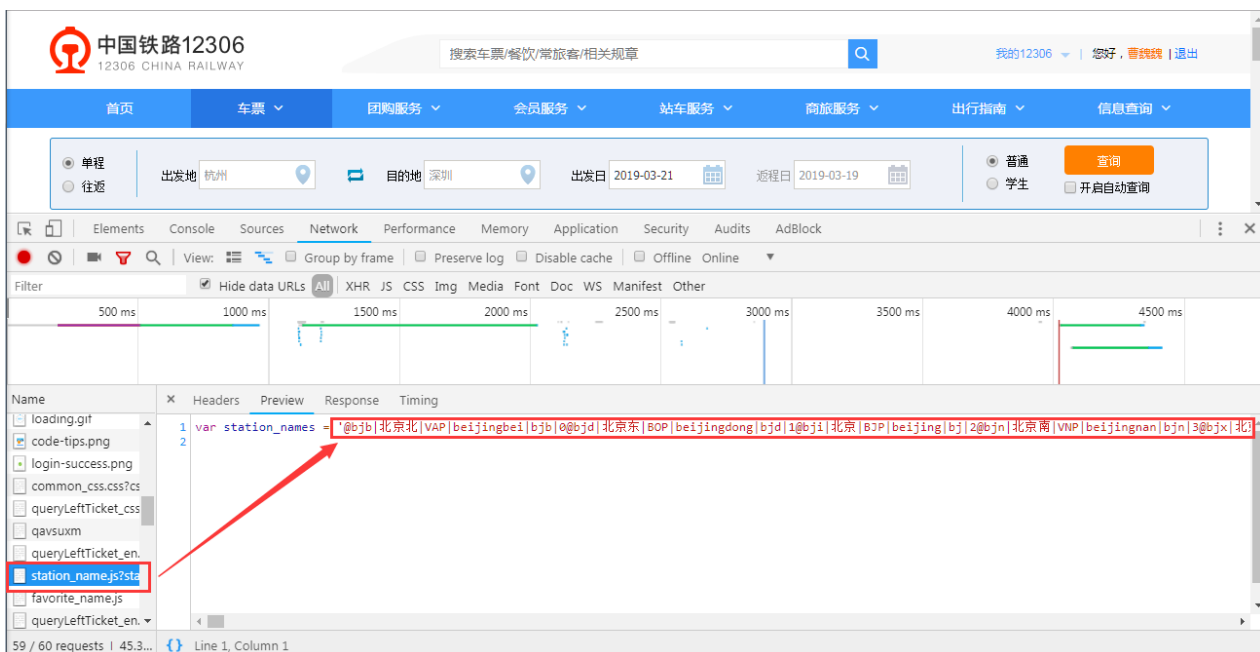
- 点击箭头指向的文件，我们可以看到在 **Headers** 栏目中的 **Request URL** 中包括了查询的所有条件信息：出发地、目的地、出发日

The screenshot shows the same website interface, but now the 'Headers' tab is selected in the developer tools. The 'Request URL' is highlighted, showing the full query string: 'https://kyfw.12306.cn/otn/leftTicket/query?leftTicketDTO.train_date=2019-03-21&leftTicketDTO.from_station=HZH&leftTicketDTO.to_station=SZQ'. Arrows point from the search criteria on the website to the corresponding parameters in the URL.

- 再点击 **Preview** 查看返回结果，这个就是查询到的列车时刻信息了



- 知道如何查询列车车次信息后，还有一个问题，就是我们输入的是中文车站名，接口处理的是英文车站代码，这个映射关系还不知道，需要继续查看页面加载的文件，寻找这个对应关系的文件，结果找到了如图文件



2.2使用 requests 库

接口信息分析完毕，下面用 requests 库进行实战。

- 首先，来获得我们想要的车站英文代码数据，通过向车站中英文关系文件发起请求获得响应，分析结果

```

# --coding: UTF-8--
import re, requests
from pprint import pprint

# 车站中英文文件URL
url = 'https://kyfw.12306.cn/otn/resources/js/framework/station_name.js?station_version=1.9098'

# 使用requests库发起请求，获得响应的json文件
resp = requests.get(url)

```

```
# 对获得的响应文件使用正则表达式，获得车站中文名和英文代码组成的元祖构成的列表
stations = re.findall(r'([\u4e00-\u9fa5]+\)\|([A-Z]+)', resp.text)

# 格式化输出车站
pprint(dict(stations), indent=4)
```

— `stations` 返回的结果如下图，是车站中文名和英文代码构成的元组组成的列表

```
192:实验楼 ➤ python3 parse_stations.py
[('北京北', 'VAP'), ('北京东', 'BOP'), ('北京', 'BJP'), ('北京南', 'VNP'), ('北京西', 'BXP'), ('广州南', 'IZQ'), ('重庆北', 'CUW'), ('重庆', 'CQW'), ('重庆南', 'CRW'), ('重庆西', 'CKW'), ('广州东', 'GGQ'), ('上海', 'SHH'), ('上海南', 'SNH'), ('上海虹桥', 'ADH'), ('上海西', 'SXH'), ('天津北', 'TBP'), ('天津', 'TJP'), ('天津南', 'TIP'), ('天津西', 'TXP'), ('香港西九龙', 'XJA'), ('长春', 'CCT'), ('长春南', 'CET'), ('长春西', 'CRT'), ('成都东', 'ICN'), ('成都南', 'CNW'), ('成都', 'CDW'), ('长沙', 'CSQ'), ('长沙南', 'CNQ'), ('大明湖', 'JAK'), ('福州', 'FZS'), ('福州南', 'FYS'), ('贵阳', 'GIW'), ('广州', 'GZQ'), ('广州西', 'GXQ'), ('哈尔滨', 'HBB'), ('哈尔滨东', 'VBB'), ('哈尔滨西', 'VAB'), ('合肥', 'HFH'), ('合肥西', 'HTH'), ('呼和浩特东', 'NDC'), ('呼和浩特', 'HHC'), ('口东', 'KEQ'), ('海口东', 'HMQ'), ('海口', 'VUQ'), ('杭州东', 'HGH'), ('杭州', 'HZH'), ('杭州南', 'XHH'), ('济南', 'JNK'), ('济南西', 'JGK'), ('昆明', 'KMM'), ('昆明西', 'KXM'), ('拉萨', 'LSQ'), ('兰州东', 'LVJ'), ('兰州', 'LZJ'), ('兰州西', 'LAJ'), ('南昌', 'NCG'), ('南京', 'NJH'), ('南京南', 'NKH'), ('南宁', 'NNZ'), ('石家庄北', 'VVP'), ('石家庄', 'SJP'), ('沈阳', 'SYT'), ('沈阳北', 'SBT'), ('沈阳东', 'SDT'), ('沈阳南', 'SOT'), ('太原北', 'TBV'), ('太原东', 'TDV'), ('太原', 'TYV'), ('武汉', 'WHN'), ('王家湾西', 'KNM'), ('乌鲁木齐', 'WAR'), ('西安北', 'EAY'), ('西安', 'XAY'), ('西安南', 'CAY'), ('西宁', 'XNO'), ('银川', 'YIJ'), ('郑州', 'ZZF'), ('阿尔山', 'ART'), ('安康', 'AKY'), ('阿克苏', 'ASR'), ('阿里河', 'AHK'), ('阿拉山口', 'AKR'), ('安平', 'APT'), ('安庆', 'AQH'), ('安顺', 'ASW'), ('鞍山', 'AST'), ('安阳', 'AYF'), ('北安', 'BAB'), ('蚌埠', 'BBH'), ('白城', 'BCT'), ('北海', 'BHZ'), ('白河', 'BEL'), ('白河', 'BAP'), ('宝鸡', 'BJY'), ('滨江', 'BJB'), ('博克图', 'BKX'), ('百色', 'BIZ'), ('白山市', 'HJL'), ('北台', 'BTT'), ('包头东', 'BDC'), ('包头', 'BTC'), ('北屯市', 'BXR'), ('本溪', 'BXT'), ('白云鄂博', 'BEC'), ('白银西', 'BXJ'), ('亳州', 'BZH'), ('赤壁', 'CBN'), ('常德', 'VGQ'), ('承德', 'CDP'), ('长甸', 'CDT'), ('赤峰', 'CFD'), ('赤峰', 'CDG'), ('苍南', 'CEH'), ('昌平', 'CPP'), ('崇仁', 'CRG'), ('昌图', 'CTT'), ('长汀镇', 'CDB'), ('曹县', 'CXK'), ('楚雄南', 'COM'), ('陈相屯', 'CXT'), ('长治北', 'CBF'), ('池州', 'CZU')]
```

— 使用 `python parse_stations.py > stations.py` 命令并生成 `stations.py` 文件，车站文件内容为中文名和英文代码构成的字典，这样就可以导入 `stations.py` 文件后根据键（车站中文名）从字典中检索对应值（车站中文对应的英文代码）

```
stations = {
    '一间堡': 'YJT',
    '一面坡': 'YPB',
    '一面坡北': 'YXB',
    '一面山': 'YST',
    '七台河': 'QTB',
    '七甸': 'QDM',
    '七营': 'QYJ',
    '七里河': 'QLD',
    '万乐': 'WEB',
    '万发屯': 'WFB',
    '万宁': 'WNQ',
    '万州': 'WYW',
    '万州北': 'WZE',
    ...
}
```

- 回顾一下，在命令行界面输入的起始车站中文名和列车出发日期，这些信息都在 `arguments` 这个字典对象中，只需要根据输入的信息匹配车站英文代码，然后访问车次查询接口，这样就可以查询列车信息了

— 修改命令行函数 `cli` 中代码

```
def cli():
    """command-line interface"""
    arguments = docopt(__doc__)
    # 根据命令行输入的起始车站名称、日期，查询stations中对应的英文代码
    from_station = stations.get(arguments['<from>'])
    to_station = stations.get(arguments['<to>'])
    date = arguments['<date>']

    # 发起请求查询车次信息
    url = 'https://kyfw.12306.cn/otn/leftTicket/query?leftTicketDTO.train_date={}&leftTicketDTO.from_station={}&leftTicketDTO.to_station={}&purpose_codes=ADULT'.format(date, from_station, to_station)
    r = requests.get(url)
    # 将返回的json数据转为字典类型
    # 其中车次信息在返回的字典键为data的值嵌套的字典中，其嵌套的键为result的值中
```

```
available_trains = r.json()['data']['result']
```

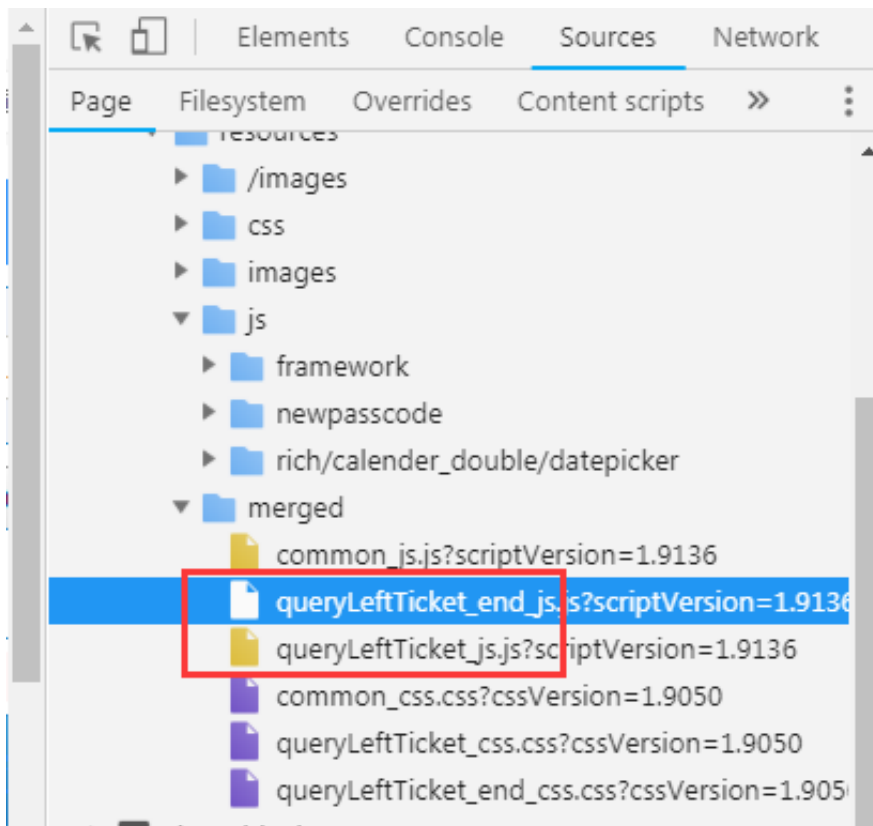
— 返回的车次结果信息如图所示

[illegible]

2.3 解析车次信息

对获取的车次信息一头雾水，那么需要尝试继续在网站中寻找线索。一般来说，网站的js文件中会定义这些数据结构，并标注每个字段的含义，所以现在来寻找这样的js文件。

进入浏览器调试模式，查看源代码 `source`。猜测是图中所示的文件可能包括车票信息，因为查询车次就是查询列车余票信息。



将压缩的js代码调整下格式，然后尝试搜索“硬座”与列车相关的信息关键词，结果找到了如下代码，这下知道坐席代码了：


```

var yxTrainPageSize = 15;
var passengerPageSize = 20;
var timer_time = 3;
var yxTrainChange = "";
var trainListForIE = [];
var queryLeftTicket_times = 0;
var queryLeftTicket_count = 10;
var ifAlertCode = false;
var intervalTime;
var seatTypeForHB = {
  SWZ: "9_商务座",
  TZ: "P_特等座",
  ZY: "M_一等座",
  ZE: "O_二等座",
  GR: "6_高级软卧",
  RW: "4_软卧",
  SRRB: "F_动卧",
  YW: "3_硬卧",
  RZ: "2_软座",
  YZ: "1_硬座",
  WZ: "1_无座",
  QT: "H_其他"
};
var _allowHBHour = 19;
var allowHBMaxNum = 2;

```

继续寻找“硬座”，发现并没有更多有价值内容，然后尝试搜索“硬座”的键值“YZ”，发现关键代码：

```

function cp(c0, cQ) {
    var cN = [];
    for (var cM = 0; cM < c0.length; cM++) {
        var cR = [];
        var cL = c0[cM].split("|");
        cR.secretStr = cL[0];
        cR.buttonTextInfo = cL[1];
        var cP = [];
        cP.train_no = cL[2];
        cP.station_train_code = cL[3];
        cP.start_station_telecode = cL[4];
        cP.end_station_telecode = cL[5];
        cP.from_station_telecode = cL[6];
        cP.to_station_telecode = cL[7];
        cP.start_time = cL[8];
        cP.arrive_time = cL[9];
        cP.lishi = cL[10];
        cP.canWebBuy = cL[11];
        cP.yb_info = cL[12];
        cP.start_train_date = cL[13];
        cP.train_seat_feature = cL[14];
        cP.location_code = cL[15];
        cP.from_station_no = cL[16];
        cP.to_station_no = cL[17];
        cP.is_support_card = cL[18];
        cP.controlled_train_flag = cL[19];
        cP.gg_num = cL[20] ? cL[20] : "--";
        cP.gr_num = cL[21] ? cL[21] : "--";
        cP.qt_num = cL[22] ? cL[22] : "--";
        cP.rw_num = cL[23] ? cL[23] : "--";
        cP.rz_num = cL[24] ? cL[24] : "--";
        cP.tz_num = cL[25] ? cL[25] : "--";
        cP.wz_num = cL[26] ? cL[26] : "--";
        cP.yb_num = cL[27] ? cL[27] : "--";
        cP.yw_num = cL[28] ? cL[28] : "--";
        cP._num = cL[29] ? cL[29] : "--";
        cP.ze_num = cL[30] ? cL[30] : "--";
        cP.zy_num = cL[31] ? cL[31] : "--";
        cP.swz_num = cL[32] ? cL[32] : "--";
        cP.srrb_num = cL[33] ? cL[33] : "--";
        cP.yb_ex = cL[34];
        cP.seat_types = cL[35];
        cP.exchange_train_flag = cL[36];
        cP.houbu_train_flag = cL[37];
        if (cL.length > 38) {
            cP.houbu_seat_limit = cL[38]
        }
        cP.from_station_name = cQ[cL[6]];
        cP.to_station_name = cQ[cL[7]];
        cR.queryLeftNewDTO = cP;
        cN.push(cR)
    }
    return cN
}

```

完美，这段代码十分详细的介绍了车次信息中各个值的含义，我们再拿来和在12306查询到的车次信息比对发现完全正确，这样就可以解析车次信息了。

3. 列车信息展示

3.1 构建列车信息类

接下来使用类来封装获取的列车信息，整体思路是，通过类来处理页面爬虫获取的列车信息，并将数据清洗后展示出来，列车信息类 `TrainsInfo` 的代码如下：

```

class TrainsInfo:
    def __init__(self, available_trains, from_to_stations, options):
        """
        定义类用到的变量：列车信息，列车类型选项
        available_trains: 列车信息

```

```

fom_o_saions: 起止车站，模糊匹配，例如杭州，杭州东等
options: 列车类型选项
"""

self.available_trains = available_trains
self.from_to_stations = from_to_stations
self.options = options

@property
def trains(self):
    for each_train in self.available_trains:
        each_train_split = each_train.split('|')
        train_code = each_train_split[3]#车次

        if not self.options or train_code[0].lower() in self.options):
            """
            显示符合条件的列车信息，即列车类型匹配g/d/z等 或者
            不输入列车等级 也可查询
            """

            trains = [train_code,          #车次
                      each_train_split[6], #起点车站代码
                      each_train_split[7], #终点车站代码

                      each_train_split[8], #发车时间
                      each_train_split[9], #到达时间
                      each_train_split[10], #历时

                      each_train_split[11], #网上购票
                      each_train_split[26] if each_train_split[26] else '--', #无座
                      each_train_split[27] if each_train_split[27] else '--', #硬座
                      each_train_split[24] if each_train_split[24] else '--', #软座
                      each_train_split[28] if each_train_split[28] else '--', #硬卧
                      each_train_split[33] if each_train_split[33] else '--', #动卧
                      each_train_split[23] if each_train_split[23] else '--', #软卧
                      each_train_split[21] if each_train_split[21] else '--', #高级软
卧

                      each_train_split[30] if each_train_split[30] else '--', #二等座
                      each_train_split[31] if each_train_split[31] else '--', #一等座
                      each_train_split[25] if each_train_split[25] else '--', #特等座
                      each_train_split[32] if each_train_split[32] else '--', #商务座
                      each_train_split[22] if each_train_split[22] else '--', #其他
            ]
            yield trains

    def print_train_info(self):
        for train in self.trains:
            print(train)

```

为了调用类，修改后的 cli 代码如下：

```

def cli():
    """command-line interface"""

```


车次	出发站 到达站	出发时间 到达时间	历时	商务座 特等座	一等座	二等座	高级 软卧	软卧 一等卧	动卧	硬卧 二等卧	软座	硬座	无座	其他	备注
G2385	杭州东 绩溪北	08:00 09:33	01:33 当日到达	11	有	有	--	--	--	--	--	--	--	--	预订
G2395	杭州东 绩溪北	08:07 09:53	01:46 当日到达	无	无	2	--	--	--	--	--	--	--	--	预订
G7191	杭州东 绩溪北	12:52 14:24	01:32 当日到达	2	1	无	--	--	--	--	--	--	15	--	预订
G7315	杭州东 绩溪北	15:11 16:50	01:39 当日到达	18	有	有	--	--	--	--	--	--	有	--	预订
D2109	杭州东 绩溪北	16:09 17:35	01:26 当日到达	--	19	有	--	--	--	--	--	--	有	--	预订
D3397	杭州东 绩溪北	16:19 17:52	01:33 当日到达	--	有	有	--	--	--	--	--	--	无	--	预订
G2381	杭州东 绩溪北	16:47 18:15	01:28 当日到达	无	无	有	--	--	--	--	--	--	--	--	预订
G7303	杭州东 绩溪北	17:34 19:06	01:32 当日到达	5	19	有	--	--	--	--	--	--	20	--	预订
G7451	杭州东 绩溪北	18:10 19:43	01:33 当日到达	8	有	有	--	--	--	--	--	--	有	--	预订
G1393	杭州东 绩溪北	18:53 20:13	01:20 当日到达	无	无	有	--	--	--	--	--	--	--	--	预订
G7307	杭州东 绩溪北	19:13 20:39	01:26 当日到达	3	19	有	--	--	--	--	--	--	有	--	预订
D5585	杭州东 绩溪北	19:24 20:51	01:27 当日到达	5	20	有	--	--	--	--	--	--	有	--	预订
D3323	杭州东 绩溪北	20:10 21:42	01:32 当日到达	--	有	有	--	--	--	--	--	--	有	--	预订
G2379	杭州东 绩溪北	-----	-----	--	--	--	--	--	--	--	--	--	--	--	列车停运

3.2美化输出界面

但是工作到这里并没有结束，这个输出界面太丑陋了，需要精致一些。使用prettytable库来将信息用表格形式输出，在调整些颜色即可。

调整列车信息打印函数，通过表格输出信息，在使用前需要引用 prettytable 库 import prettytable

```
def print_train_info(self):
    # 创建表格
    tb = prettytable.PrettyTable()
    # 设置表格标题
    tb.field_names = self.headers
    for train in self.trains:
        # 增加每行数据
        tb.add_row(train)
    print(tb)
```

列车信息类中需要对train信息进行颜色标识，整体代码修改后如下：

```
init()

class TrainsInfo:
    headers = '车次 车站 时间 历时 有票 无座 硬座 软座 硬卧 动卧 软卧 高级软卧 二等座 一等座 特等座 商务座 其他'.split()

    def __init__(self, available_trains, from_to_stations, options):
        """
        定义类用到的变量：列车信息，列车类型选项
        available_trains: 列车信息
        fom_o_saions: 起止车站，模糊匹配，例如杭州，杭州东等
        options: 列车类型选项
        """
```

```

self.available_trains = available_trains
self.from_to_stations = from_to_stations
self.options = options

@property
def trains(self):
    for each_train in self.available_trains:
        each_train_split = each_train.split('|')
        train_code = each_train_split[3]#车次

        if not self.options or train_code[0].lower() in self.options:
            """
            显示符合条件的列车信息，即列车类型匹配g/d/z等 或者
            不输入列车等级 也可查询
            """

            trains = [train_code,          #车次
                      '\n'.join([Fore.GREEN + self.from_to_stations[each_train_split[6]]
                                  + Fore.RESET,
                                  Fore.RED + self.from_to_stations[each_train_split[7]] +
                                  Fore.RESET]),

                      '\n'.join([Fore.GREEN + each_train_split[8] + Fore.RESET,
                                  Fore.RED + each_train_split[9] + Fore.RESET]),

                      each_train_split[10],    #历时

                      each_train_split[11],    #网上购票

                      each_train_split[26] if each_train_split[26] else '--',    #无座
                      each_train_split[27] if each_train_split[27] else '--',    #硬座
                      each_train_split[24] if each_train_split[24] else '--',    #软座
                      each_train_split[28] if each_train_split[28] else '--',    #硬卧
                      each_train_split[33] if each_train_split[33] else '--',    #动卧
                      each_train_split[23] if each_train_split[23] else '--',    #软卧
                      each_train_split[21] if each_train_split[21] else '--',    #高级软
卧

                      each_train_split[30] if each_train_split[30] else '--',    #二等座
                      each_train_split[31] if each_train_split[31] else '--',    #一等座
                      each_train_split[25] if each_train_split[25] else '--',    #特等座
                      each_train_split[32] if each_train_split[32] else '--',    #商务座
                      each_train_split[22] if each_train_split[22] else '--',    #其他

                      ]
            yield trains

```

这里面用到了 `colorama` 命令行着色工具，使用前需要引入库 `from colorama import init, Fore`，然后初始化 `init()` 并设置颜色 `Fore.GREEN` `Fore.RED` `Fore.RESET`

五、总结

本项目最核心知识点在于 `requests` 库的使用，难点在于如何分析js文件寻找车站和英文关系对应以及列车车次信息获取后如何解析。

其他的知识点还有 `yield` 使用，`@property` 装饰器，`dicopt` 命令行界面以及 `prettytable` 表格输出 `colorama` 表格上色。

详细代码请查阅：<https://github.com/elicao/12306query-tickets>