

# Python 实现火车票查询工具

**注意：**由于 12306 的接口经常变化，课程内容可能很快过期，如果遇到接口问题，需要根据最新的接口对代码进行适当修改才可以完成实验。

## 一、实验简介

当你想查询一下火车票信息的时候，你还在上 12306 官网吗？或是打开你手机里的 APP？

下面让我们来用 Python 写一个命令行版的火车票查看器，只要在命令行敲一行命令就能获得你想要的火车票信息！如果你刚掌握了 Python 基础，这将是不错的小练习。

### 1.1 知识点

- Python 基础知识的综合运用
- `docopt`、`requests`、`colorama` 及 `prettytable` 库的使用
- `setuptools` 的使用

### 1.2 效果截图

Terminal 终端 - shiyanlou@c62c36f982e2: ~										
文件(E) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)										
shiyanlou:~/ \$ python3 tickets.py 成都 南京 2016-10-30										
车次	车站	时间	历时	一等	二等	软卧	硬卧	硬座	无座	
D638	成都东	07:12	13小时12分	无	无	--	--	--	83	
D2224	南京南	20:24								
	成都东	07:36	12小时42分	12	354	--	--	--	无	
	南京南	20:18								
D354	成都东	08:01	12小时36分	4	706	--	--	--	170	
	南京南	20:37								
D2208	成都东	08:20	12小时37分	101	372	--	--	--	156	
	南京南	20:57								
D2256	成都东	08:26	12小时47分	无	451	--	--	--	85	
	南京南	21:13								
D2264	成都东	09:05	12小时18分	35	690	--	--	--	164	
	南京南	21:23								
D2374	成都东	09:29	12小时40分	无	322	--	--	--	85	
	南京南	22:09								
K1158	成都	11:03	24小时48分	--	--	无	无	290	339	
	南京	11:51								
K292	成都	17:47	31小时59分	--	--	10	98	587	307	
	南京	01:46								
K284	成都	18:20	33小时20分	--	--	17	95	430	401	
	南京	03:40								

## 二、接口设计

一个应用写出来最终是要给人使用的，哪怕只是给你自己使用。

所以，首先应该想想你希望怎么使用它？让我们先给这个小应用起个名字吧，既然要查询票务信息，那就叫它 `tickets` 好了。

我们希望用户只要输入出发站，到达站以及日期就能获得想要的信息，比如要查看10月30号上海-北京的火车余票，我们只需输入：

```
$ python tickets.py 上海 北京 2016-10-30
```

注意：上面的日期（包括后面的）是笔者写文章时确定的日期，当你在做这个项目的时候可能要根据当前时间做适当调整。

转化为程序语言就是：

```
$ python tickets.py from to date
```

另外，火车有各种类型，高铁、动车、特快、快速和直达，我们希望能提供选项只查询特定的一种或几种的火车，所以，我们应该有下面这些选项：

- -g 高铁
- -d 动车
- -t 特快
- -k 快速
- -z 直达

这几个选项应该能被组合使用，所以，最终我们的接口应该是这个样子的：

```
$ python tickets.py [-gdtkz] from to date
```

接口已经确定好了，剩下的就是实现它了。

## 三、代码实现

首先安装一下实验需要用到的库：

```
$ sudo pip3 install requests prettytable docopt colorama
```

- `requests`，使用 Python 访问 HTTP 资源的必备库。
- `docopt`，Python3 命令行参数解析工具。
- `prettytable`，格式化信息打印工具，能让你像 MySQL 那样打印数据。
- `colorama`，命令行着色工具

### 3.1 解析参数

Python有很多写命令行参数解析工具，如 `argparse`，`docopt`，`click`，这里我们选用的是 `docopt` 这个简单易用的工具。`docopt` 可以按我们在文档字符串中定义的格式来解析参

数，比如我们在 `tickets.py` 中写下下面的内容（实验楼环境下，通过点击右下角的键盘小图标可以选择中文输入法）：

```
# 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 北京 上海 2016-10-10
    tickets -dg 成都 南京 2016-10-10
"""

from docopt import docopt

def cli():

    """command-line interface"""

    arguments = docopt(__doc__)
    print(arguments)

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

上面的程序中，`docopt` 会根据我们在 `docstring` 中的定义的格式自动解析出参数并返回一个字典，也就是 `arguments`，我们打印出了这个字典的内容。下面我们运行一下这个程序，比如查询一下10月30号从成都到南京的动车和高铁：

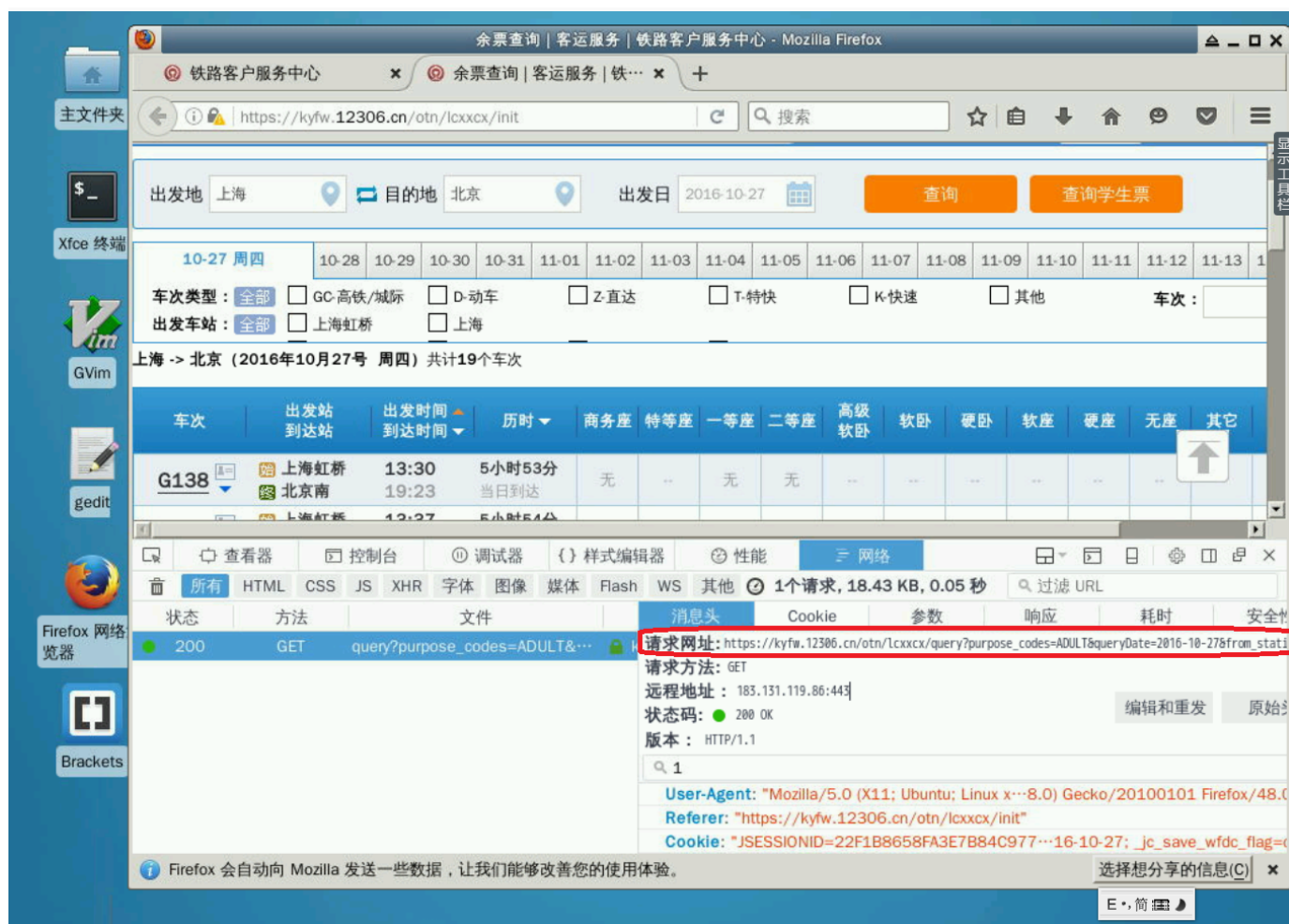
```
$ python tickets.py -dg 成都 南京 2016-10-10
```

我们得到下面的参数解析结果：

```
shiyanolou:~/ $ python3 tickets.py -gd 成都 南京 2016-10-30
{'-d': True,
 '-g': True,
 '-k': False,
 '-t': False,
 '-z': False,
 '<date>': '2016-10-30',
 '<from>': '成都',
 '<to>': '南京'}
```

## 3.2 获取数据

参数已经解析好了，下面就是如何获取数据了，这也是最主要的部分。首先我们用实验楼环境的Firefox浏览打开 [12306](#)，进入余票查询页面，按下 `F12` 打开开发者工具，选中 `Netwo`  
`rk` 一栏，在查询框中随便查询一次，我们在调试工具观察下请求和响应：



注意到上面的请求 URL，它是由基 URL <https://kyfw.12306.cn/otn/lcxxxcx/query> 加四个参数构成的，这四个参数分别代表，查询的类型（成人？学生？），日期，出发车站，到达车站：



再来看看响应：





返回的是 JSON 格式的数据！ 我们打开返回的数据看看：



可以看到一列火车的数据用 Python 的语言说就是一个字典。

接下来问题就简单了，我们只需要利用这个接口， 构建请求URL然后解析返回的JSON数据就可以了。但是我们发现，URL里面参数 `from_station` 和 `to_station` 并不是汉字，而是一个代号，而我们想要输入的是汉字，我们要如何获取代号呢？我们打开网页源码看看有没有什么发现。



果然，这里有个关于 station 的文件，打开看看：



`station_names` 是一个很长的字符串，这里面貌似是包含了所有车站的中文名，拼音，简写和代号等信息。但是这些信息挤在一起，而我们只想要车站的拼音和大写字母的代号信息，怎么办呢？正则表达式！我们写个小脚本来匹配提取出想要的信息吧，在 `parse_station.py` 中：

```
import re
import requests
from pprint import pprint

url = 'https://kyfw.12306.cn/otn/resources/js/framework/station_name.js?station_version=1.8971'
response = requests.get(url, verify=False)
stations = re.findall(u'([\u4e00-\u9fa5]+\)|([A-Z]+)', response.text)
pprint(dict(stations), indent=4)
```

我们运行这个脚本，它将以字典的形式返回所有车站和它的大写字母代号，我们将结果重定向到 `stations.py` 中，

```
$ python3 parse_station.py > stations.py
```

我们为这个字典加名字，`stations`，最终，`stations.py` 文件是这样的：



```
Terminal 终端 - stations.py (~) - \n\n文件(E) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)\n\n1 stations = {\n2     '一间堡': 'YJT',\n3     '一面坡': 'YPB',\n4     '一面山': 'YST',\n5     '七台河': 'QTB',\n6     '七甸': 'QDM',\n7     '七营': 'QYJ',\n8     '七里河': 'QLD',\n9     '万乐': 'WEB',\n10    '万发屯': 'WFB',\n11    '万宁': 'WNQ',\n12    '万州': 'WYW',\n13    '万州北': 'WZE',\n14    '万年': 'WWG',\n15    '万源': 'WYY',\n16    '三义井': 'OYD',\n
```

现在，用户输入车站的中文名，我们就可以直接从这个字典中获取它的字母代码了：

```
...\nfrom stations import stations\n\ndef cli():\n    arguments = docopt(__doc__)\n    from_station = stations.get(arguments['<from>'])\n    to_station = stations.get(arguments['<to>'])\n    date = arguments['<date>']\n\n    # 构建URL\n    url = 'https://kyfw.12306.cn/otn/lcxxcx/query?purpose_codes=ADULT&\nqueryDate={}&from_station={}&to_station={}'.format(\n        date, from_station, to_station\n    )
```

万事俱备，下面我们来请求这个URL获取数据吧！这里我们使用 `requests` 这个库，它提供了非常简单易用的接口，

```
...
import requests

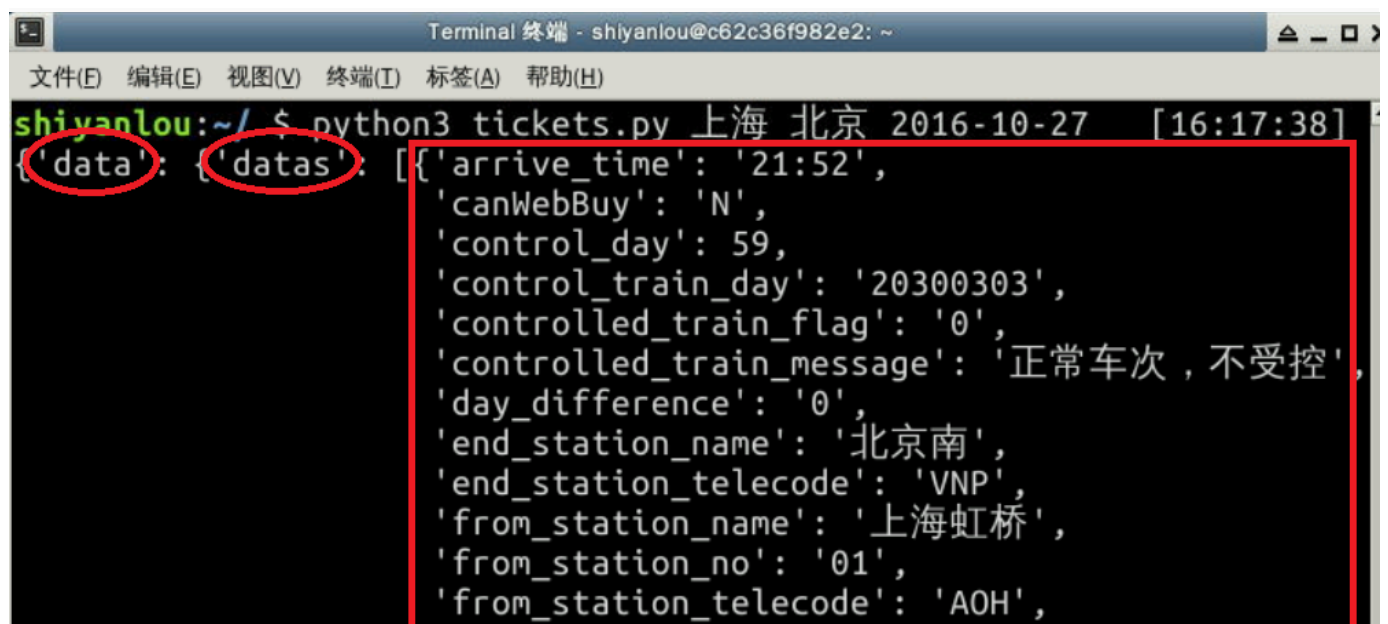
def cli():
    ...
    # 添加verify=False参数不验证证书

    r = requests.get(url, verify=False)
    print(r.json())
```

从结果中，我们可以观察到，与车票有关的信息需要进一步提取：

```
def cli():
    ...
    r = requests.get(url);
    print(r.json())
```

我们已经知道该请求返回的是JSON数据，使用 `requests` 提供的 `r.json()` 可以将JSON数据转化为Python字典，上面我们打印了这个字典，运行程序，我们看到：



```
Terminal 终端 - shiyanlou@c62c36f982e2: ~
文件(E) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)
shiyanlou:~/ $ python3 tickets.py 上海 北京 2016-10-27 [16:17:38]
{'data': {'datas': [{'arrive_time': '21:52',
  'canWebBuy': 'N',
  'control_day': 59,
  'control_train_day': '20300303',
  'controlled_train_flag': '0',
  'controlled_train_message': '正常车次，不受控',
  'day_difference': '0',
  'end_station_name': '北京南',
  'end_station_telecode': 'VNP',
  'from_station_name': '上海虹桥',
  'from_station_no': '01',
  'from_station_telecode': 'AOH',
```

图中方框是一个字典，字典中的数据也就是一班列车的信息，也就是说所有列车的信息都在一个包含多个字典的列表中，而这个列表又嵌套了2个字典，就是图中的2个椭圆，所以，我们需要的信息应该这样来提取：

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

### 3.3 解析数据

我们封装一个简单的类来解析数据：

```
class TrainsCollection:

    header = '车次 车站 时间 历时 一等 二等 软卧 硬卧 硬座 无座'.split()

    def __init__(self, available_trains, options):
        """查询到的火车班次集合

        :param available_trains: 一个列表，包含可获得的火车班次，每个
                                火车班次是一个字典
        :param options: 查询的选项，如高铁，动车，etc...
        """
        self.available_trains = available_trains
        self.options = options

    def _get_duration(self, raw_train):
        duration = raw_train.get('lishi').replace(':', '小时') + '分'

        if duration.startswith('00'):
            return duration[4:]

        if duration.startswith('0'):
            return duration[1:]

        return duration
```

```

@property

def trains(self):
    for raw_train in self.available_trains:
        train_no = raw_train['station_train_code']
        initial = train_no[0].lower()
        if not self.options or initial in self.options:
            train = [
                train_no,
                '\n'.join([raw_train['from_station_name'],
                           raw_train['to_station_name']]),
                '\n'.join([raw_train['start_time'],
                           raw_train['arrive_time']]),
                self._get_duration(raw_train),
                raw_train['zy_num'],
                raw_train['ze_num'],
                raw_train['rw_num'],
                raw_train['yw_num'],
                raw_train['yz_num'],
                raw_train['wz_num'],
            ]
            yield train

def pretty_print(self):
    pt = PrettyTable()
    pt._set_field_names(self.header)
    for train in self.trains:
        pt.add_row(train)
    print(pt)

```

## 3.4 显示结果

最后，我们将上述过程进行汇总并将结果输出到屏幕上：

```
...
```

```

class TrainCollection:
    ...
    ...

def cli():
    """Command-line interface"""
    arguments = docopt(__doc__)
    from_station = stations.get(arguments['<from>'])
    to_station = stations.get(arguments['<to>'])
    date = arguments['<date>']

    url = ('https://kyfw.12306.cn/otn/lcxxcx/query?'
           'purpose_codes=ADULT&queryDate={}&'
           'from_station={}&to_station={}'.format(
               date, from_station, to_station
           ))

    # 获取参数
    options = ''.join([
        key for key, value in arguments.items() if value is True
    ])

    r = requests.get(url, verify=False)

    available_trains = r.json()['data']['datas']
    TrainsCollection(available_trains, options).pretty_print()

```

运行下程序看看效果吧：

The screenshot shows a Linux desktop with a terminal window titled "Terminal 终端 - shiyanlou@c62c36f982e2: ~". The terminal displays the command `python3 tickets.py -gd 上海 北京 2016-10-27` and its output, which is a table of train schedules. The desktop background is blue, and the left sidebar contains icons for the home folder, file manager, terminal, and various applications like Xfce, Vim, and Gedit.

车次	车站	时间	历时	一等	二等	软卧	硬卧
G158	上海虹桥 北京南	17:34 23:18	5小时44分	无	348	--	--
G8	上海虹桥 北京南	19:00 23:48	4小时48分	无	无	--	--
D312	上海 北京南	19:10 07:07	11小时57分	--	无	188	--
D322	上海 北京南	19:53 07:45	11小时52分	--	无	269	--
D314	上海 北京南	21:08 08:55	11小时47分	--	无	104	--

## 3.5 着色

至此，程序的主体已经完成了，但是上面打印出的结果是全是黑白的，很是乏味，我们来给它添加点颜色吧！

这里我们使用 `colorama` 这个命令行着色工具：

```
from colorama import init, Fore

init()
```

修改一下程序，将出发车站与出发时间显示为绿色，到达车站与到达时间显示为红色：

```
...
'\n'.join([Fore.GREEN + raw_train['from_station_name'] + Fore.RESET,
           Fore.RED + raw_train['to_station_name'] + Fore.RESET]),
'\n'.join([Fore.GREEN + raw_train['start_time'] + Fore.RESET,
           Fore.RED + raw_train['arrive_time'] + Fore.RESET]),
...
```

现在再运行程序就可以像文章开始的效果图一样了！

## 3.6 完整代码

```
# coding: utf-8

"""命令行火车票查看器

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

Options:
```



```
-h, --help 查看帮助
-d          动车
-g          高铁
-k          快速
-t          特快
-z          直达
```

#### Examples:

```
tickets 上海 北京 2016-10-10
```

```
tickets -dg 成都 南京 2016-10-10
```

```
"""
```

```
import requests
```

```
from docopt import docopt
```

```
from prettytable import PrettyTable
```

```
from colorama import init, Fore
```

```
from stations import stations
```

```
init()
```

```
class TrainsCollection:
```

```
    header = '车次 车站 时间 历时 一等 二等 软卧 硬卧 硬座 无座'.split()
```

```
    def __init__(self, available_trains, options):
```

```
        """查询到的火车班次集合
```

```
        :param available_trains: 一个列表，包含可获得的火车班次，每个
                                火车班次是一个字典
```

```
        :param options: 查询的选项，如高铁，动车，etc...
```

```
        """
```

```
        self.available_trains = available_trains
```

```
        self.options = options
```

```

def _get_duration(self, raw_train):

    duration = raw_train.get('lishi').replace(':', '小时') + '分'

    if duration.startswith('00'):

        return duration[4:]

    if duration.startswith('0'):

        return duration[1:]

    return duration

```

```

@property

```

```

def trains(self):

```

```

    for raw_train in self.available_trains:

```

```

        train_no = raw_train['station_train_code']

```

```

        initial = train_no[0].lower()

```

```

        if not self.options or initial in self.options:

```

```

            train = [

```

```

                train_no,

```

```

                '\n'.join([Fore.GREEN + raw_train['from_station_name'] + Fo

```

```
re.RESET,

```

```

                        Fore.RED + raw_train['to_station_name'] + Fore.RES

```

```
ET])),

```

```

                '\n'.join([Fore.GREEN + raw_train['start_time'] + Fore.RESET,

```

```

                        Fore.RED + raw_train['arrive_time'] + Fore.RESE

```

```
T])),

```

```

                self._get_duration(raw_train),

```

```

                raw_train['zy_num'],

```

```

                raw_train['ze_num'],

```

```

                raw_train['rw_num'],

```

```

                raw_train['yw_num'],

```

```

                raw_train['yz_num'],

```

```

                raw_train['wz_num'],

```

```

            ]

```

```

            yield train

```

```

def pretty_print(self):

```

```

        pt = PrettyTable()
        pt._set_field_names(self.header)
        for train in self.trains:
            pt.add_row(train)
        print(pt)

def cli():
    """Command-line interface"""
    arguments = docopt(__doc__)
    from_station = stations.get(arguments['<from>'])
    to_station = stations.get(arguments['<to>'])
    date = arguments['<date>']
    url = ('https://kyfw.12306.cn/otn/lcxxcx/query?'
          'purpose_codes=ADULT&queryDate={}&'
          'from_station={}&to_station={}'.format(
              date, from_station, to_station
          ))
    options = ''.join([
        key for key, value in arguments.items() if value is True
    ])
    r = requests.get(url, verify=False)
    available_trains = r.json()['data']['datas']
    TrainsCollection(available_trains, options).pretty_print()

if __name__ == '__main__':
    cli()

```

## 四、Setup

上面的程序中我们运行程序的方式是这样的：

```
python3 tickets.py from to date
```

我们当然可以将脚本改成可执行的，然后这样执行：

```
./tickets.py from to date
```

但这也不是理想的方案，因为我们必须在脚本的目录下才能运行。我们想要的是在命令行的任何地方都可以这样运行：

```
ticktes from to date
```

这是可以实现的，我们需要借助 Python 的 SETUP 工具。写一个简单的 setup 脚本：

```
from setuptools import setup

setup(
    name='tickets',
    py_modules=['tickets', 'stations'],
    install_requires=['requests', 'docopt', 'prettytable', 'colorama'],
    entry_points={
        'console_scripts': ['tickets=tickets:cli']
    }
)
```

在命令行运行一下：

```
python3 setup.py install
```

现在我们可以愉快的查询了：

```
shiyancelou:~/ $ tickets 成都 南京 2016-10-31
```

车次	车站	时间	历时	一等	二等	软卧	硬卧	硬座	无座
D638	成都东 南京南	07:12 20:24	13小时12分	无	256	--	--	--	85
D2224	成都东 南京南	07:36 20:18	12小时42分	14	606	--	--	--	无

## 五、总结

本课程使用 Python3 抓取 12306 网站信息提供一个命令行的火车票查询工具。通过该项目的实现，可以学习并实践 Python3 基础及网络编程，以及 docopt, requests, prettytable, colorama 等库的使用。

## 六、参考资料

本项目详细代码可以从下面链接获取：

- <https://github.com/protream/iquery>  
来源：<https://www.shiyancelou.com/courses/running>