项目名称：列车时间管理大师

——出行列车信息显示

团队名称： 发际线和我作队
队长： 张晋豪 519120910211
队员： 黄云帆 519120910207
舒倩茜 519120910037

项目目标：随着高铁的普及，铁路出行成为了人们日常出行，商务出差，远程旅行的重要选择。我们设计的这款程序，可以帮助周末出行规划好来回的行程，也可以为工作人士提供详尽的火车时刻安排以便管理时间，也能让旅行的人在不同时刻记录下沿途的风景，不错过每一刻的美好。

本小组项目致力于为有需求的用户提供可视化的列车运行情况。（例如列车时刻表、行进路线）

（说明：本组原定项目为仅服务于实时列车信息的爬取工作，后来在实际编程过程中，鉴于中国铁路网 12306.cn 的运行特点及本小组成员的讨论，最终将服务范围改为全体未发车的列车，用户可查询未来一个月内的列车信息，便于做好规划）

项目实现技术报告：

一、张晋豪同学部分：

我在项目中的主要工作是利用爬虫技术爬取数据、处理所得信息以及项目主函数的书写，主要负责了第 15-58,140-218 行的代码书写。通过学习 python 的 requests 库以及 JSON 数据解析，我尽可能的利用并结合所学的代码知识来对需要的信息进行可行的抓取与有效的处理，面向用户的信息展示的前半部分工作，并为后续组员处理时间与可视化做铺垫。

本程序的主要流程是，要求用户在开始输入起始地、目的地与时间三个量，我们将根据这三个量来返回用户可能需要查询的列车班次与对应信息，再让用户输入要详细了解的班次，然后再次爬虫，返回出该列车的时刻表等信息，并接上黄云帆同学负责的可视化部分。

首先，我们需要将用户输入的“起始地”、“目的地”等汉字信息转化为网站自己适用的编码（“例如北京西：BXP”），通过在刷新网页时使用 Chrome 浏览器的 F12 功能，我在网页上经过一番查找找到了

“https://kyfw.12306.cn/otn/resources/js/framework/station_name.js?station_version=1.9143”这一链接，里面记载了所有车站与相应的编码，但是混有大量无用的信息，于是我定义了 get_City_data 函数，利用 requests 库和 split 等方法将信息爬取了下来，并做了有效的处理，返回了一个仅含有车站与编码的字典，待用。

在有了城市对应码后，我们只需找到列车查询 URL，便可得到需要的列车信息。通过观察 12306 查询列车信息时的网址，我找出了

“https://kyfw.12306.cn/otn/leftTicket/init?linktypeid=dc&fs=%E6%88%90%E9%83%BD,{CDW}&ts=%E5%8C%97%E4%BA%AC%E5%8C%97,{VAP}&date={2020-01-01}&flag=N,N,Y”

这一 URL。但是在实际操作中，我发现该网站的源代码中并没有 icon 对应的列车信息，而是一系列脚本，通过 google 我了解到，12306.cn 采用的是 JavaScript 方式书写的，以此来抵抗爬虫对服务器的影响。通过查看 CSDN 上一些大神的建议，我通过 F12 功能，对网页的活动

进行捕捉，最终得到了

“https://kyfw.12306.cn/otn/leftTicket/query?leftTicketDTO.train_date={date}&leftTicketDTO.from_station={from_station}&leftTicketDTO.to_station={to_station}&purpose_codes=ADULT”

这一可以有效查取列车信息的 URL，然后定义了 get_trains_info 函数，用于将查询到的信息返回为一个 json 数据。

在用户输入要详细查询的列车班次后，我们需要根据列车班次去爬取列车的时刻表，同样运用 F12 功能绕开 JavaScript，我得到了

“https://kyfw.12306.cn/otn/czxx/queryByTrainNo?train_no={}&from_station_telecode={}&to_station_telecode={}&depart_date={}”

这一有效链接，并定义了 get_stopover_station 函数以便于主函数爬取时刻表等信息。

接下来是主函数书写，首先调用 get_City_data 返回目标城市对应代码，然后调用 get_trains_info 返回查询到的当天所有班次的列车信息的 json 文件，然后通过 json 类的方法调用出有效信息并呈现给用户。在用户选择了具体班次后调用 get_stop_station，返回并打印列车的列车时刻表信息，并生成字典 location_time、列表 location_list、列表 time_list 供黄云帆同学与舒倩茜同学调用，然后调用黄云帆同学负责的 pyecharts 部分，完成程序的主体。

12306.cn 素有“中国最严反爬虫网站”的美誉，在实际爬虫过程中，我也遇到了很多问题，例如您可以在程序中看到的十分冗长的 headers，以及每个定义的函数第一行的 timesleep，它们的产生是因为在前期爬虫时我每次获得的都不是 json 而是爬虫失败的 html，为此我不断去模拟浏览器的 headers 而产生了这种结果。但是在爬取列车时刻表时依然遇到了失败，为此，我清除了访问 cookies，利用 F12 找到了“https://kyfw.12306.cn/otn/leftTicket/init”链接，每次请求它可以得到一个合法的 cookies，然后将其塞进了我的 headers 中，初步解决了问题。

二、黄云帆同学部分：

我在项目中的主要负责部分是数据的可视化，以及项目中从爬虫数据转化到可用数据的相关处理。通过学习 python 的 pyecharts 类库、阅读文档，我尽可能运用现有的关于面向对象编程的知识，通过调用 pyecharts 附带的函数，调整参数，使呈现的数据更加美观直接。

首先，进行初始化操作，调用 pyecharts 中的 charts,globals,option 类，利用 pyecharts 创建其中的 Geo() class，用 add_schema()方法创建中国地图以及对应的基本坐标和常用坐标。其次，通过 add_coordinate_json()导入 station_geo.json 文件添加全国截止 2019 年 12 月左右的所有火车站坐标。以上是数据可视化的基本工作。

接下来进行的是列车行驶路线的显示。首先是对爬虫在 main()函数生成的 location_list，即指定列车沿途经过的所有车站列表进行处理。由于 pyecharts 只支持一个格式为[(‘str’,’str’),(‘str’,’str’),()…]的 tuple 组成的 list，所以这里定义了一个 wash()函数对原本的车站 list 进行处理，使之返回一个从点到点的符合格式的 list。再将其运用到之后的 show_map()函数，调用 add()方法，调整 is_polyline 参数和 effect_opts 参数，使地图上呈现出一一连接的列车行驶路线。

由于后续小组任务目标的改变，因此增加了在地图上显示各个站点对应到站时间的功能。

对于各个站点到站时间的显示，由于 pyecharts 本身只支持（str,int）的数据对，不支持时间的对应显示，所以我就定义了 info_add()函数，把到站时间和车站名组合成一个 string，把所有的站名+到站时间信息组成一个列表，但是这样做的后果就是原先与车站名一一对应的坐标文件无法与新生成的车站名+到站时间 str 匹配显示，所以我又定义了 get_list_coordinate(station_list)函数，提取原先列车所有经过站点的坐标，然后定义 json_output(station_list,time_list,coordinate_list)函数，把新生成的 str 站名和原

先的站点坐标一一对应，并且在当前工作路径生成一个 station_coordinate.json 文件，方便接下来的批次导入。

之后，定义 show_map()函数，接受始发站、目的地、沿途经过的车站 list、到站时间 list、列车班次参数，利用 Geo()对象和对应的方法创建中国地图、添加全部的火车站坐标，在地图上显示始发站和目的地的位置和列车行驶路线，然后通过添加之前生成的 json 文件坐标显示沿途经过的各个站点坐标和对应的到站时间。

最后，因为 pyecharts 的 render()函数最后会在工作路径生成 html 文件，所以在权衡了一些网络相关包(比如 selenium 等)后，选择 import webbrowser、os，用默认浏览器打开当前工作路径生成的 html 文件。

另外，我也负责了运行文档 README.txt 的书写以及部分程序测试任务和协助小组成员进行各个板块工作的协调和合并串联，以及 github 各类教程(虽然最后没有用到)和团队建立初期一些建议的分享。

三、舒倩茜同学部分：

运用 datetime 和 time 库：运用 datetime 和 time 库，调用当日的时间，时刻。将爬虫爬出并创建的以地点作为 key，时间作为 value 的字典中的 value 转换为时间戳，再通过运算将字典中的时间戳大小进行比较，得出当前时间所对应的两个站点之间的位置。

整体代码的衔接部分。开头运用 datetime 库调用当前日期，再让用户输入自己想要 查询的日期，对当前日期，过去的日期和未来的日期进行比较判断，以确定是否继续接下来的获得火车时刻表的爬虫工作。同时，进行中间部分进行两个主要部分之间的衔接工作。。

查错与纠错：对后期的一些代码进行查错与纠错。例如，开头部分用 datetime 库比较大小的部分始终不能正确比较达到预期效果，后期运用 datetime 进行对时间的二次转换，纠正了错误。

调试：

由于字典里的 values 中的字符串本身不能作为参数直接进行 datetime 的转换，我尝试用字符串的拼接，将 values 字符串转为符合 datetime 的格式，将其转为时间。这其中，格式频频出错，每次报错都提示 string 不能直接转为 datetime，这与我们所查到的方法有出入，经过不断的调整，将 values 转为了 datetime。再将 datetime 转为时间戳。

关于字典中所有 values 的替换问题。一开始我使用的方法是定义一个函数。但是这个方法不能实现 values 的一次性转换。最后我采用了 for 循环的方法，经过格式上的纠正，将字典里字符串形式的“时间”全部替换为已经算好的时间戳。

在开头的部分的时间判断里面，我采用了__le__,__gt__,__ne__,__eq__的方法比较日期的大小，但由于我们 input 里输入的查询日期转换为 datetime 的日期之后，其返回的格式为（2020，6，7，0，0）而非我期待的（2020，6，6），所以我采用了 datetime.date(date_leave.year,date_leave.month,date_leave.day)的方法经过二次调试，转为了我期待的格式。但同时和当日日期如果是同一天的话，仍然会出现输入日期小于等于当日日期的情况，所以我采用了带有__ne__的 if 语句避开了这一问题。

同时我还负责了 demo 的制作。demo 的制作我考虑了我们组设计项目的初衷，然后想要加入一点创意的成分。所以我找到了符合我们项目主题的微电影，在剪辑中使用加速、加入音乐、字幕等将我们的 demo 做成了一个微电影的形式。