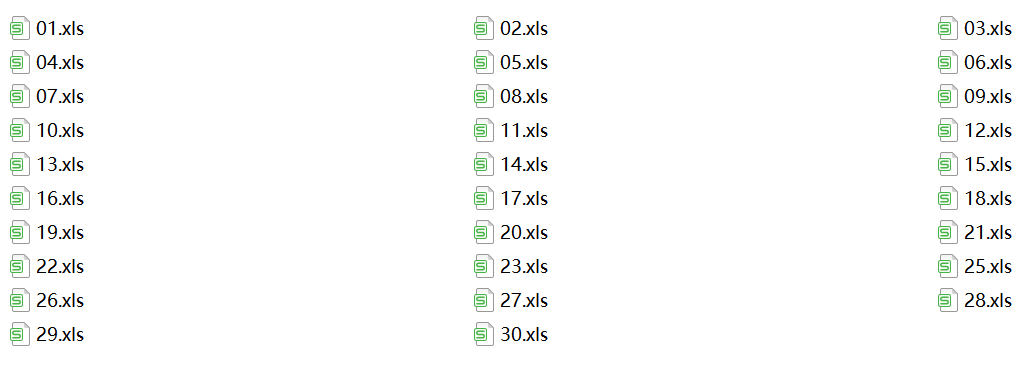
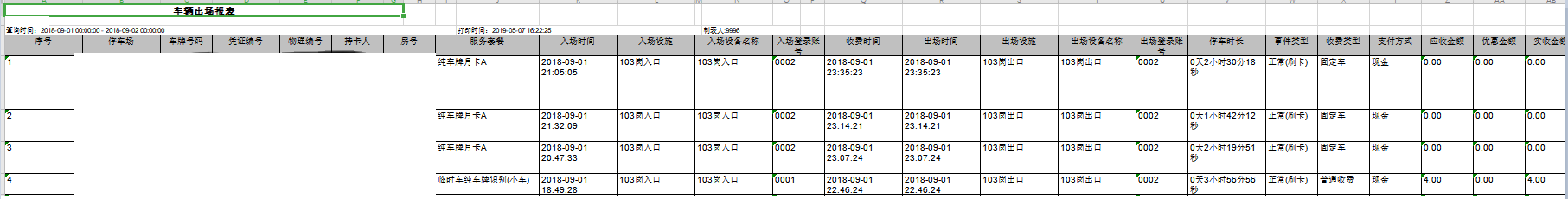
某小区停车数据分析

核心内容：处理大量excel表格数据，数据可视化

1. **问题描述**

现有2018年9月的某小区停车数据，如下图。

01-30共30天的excel表格。后来处理时发现23 和24重复了，于是将24.xls删去。每一个表格内记录了当天出场的车的停车信息，如下图



数据内包含业主的车与临时车。临时车是点对点的共享平台的停车用户。点对点指业主写明可共享车位的时间段，临停用户可在平台上找到适合自己停车需求的车位定点停放。该共享方案的问题是业主参与度不高，白天仍有大量剩余车位，而白天小区周围有停车需求。

现在希望建立一个新的共享停车的方案，不需要业主指明可共享的时间段。根据停车的历史数据进行分析，自动得出适合开放共享停车的时间段和共享的最大停车位数，保证业主有位置停。

1. **详细内容及步骤**
2. 提取数据中有用的信息 get\_data.py文件

使用python 中的pandas库的dataframe类来存储数据。

首先读取一个excel表中的数据，成功后再用类似的方法使用循环读取所有文件。

一些关键步骤展示整行代码，其余仅写出函数名，略去参数

import pandas as pd

1）自动读取excel表，存在一个dataframe中:

df=pd.read\_excel('./data/'+fname,skiprows=3)#跳过前三行，从第四行开始读取 原本不知道有跳过行的功能，还使用了xlrd包先读取再存到dataframe中

1. 数据清理，调整

清除列 df.drop(df.columns[delid], axis=1, inplace=True)

清楚空值所在一整行df.dropna(inplace=True)

重命名列名称 df.rename(columns={'车牌号码':'num',...},inplace=True)

重命名一列中的值df['type']=df['type'].replace(['纯车牌月卡A','临时车纯车牌识别(小车)'],['业主','临时'])

1. 增加数据列

字符串的时间类型转化为[0,24)连续的数，小时为单位df['intimedig']=list(map(time2dig,df['intime']))#数字化的连续时间float64类 time2dig为自编函数

字符串的时间类型转化为datetime64类型，便于以后操作df['indatetime']=pd.to\_datetime(df['intime'])

1. 将处理一个文件扩展到处理多个文件

import pandas as pd

import os

def get\_data():

fnames = os.listdir('./data/')#./相对目录 返回data文件夹下所有文件名称

dfall=pd.DataFrame()#创建一个空的dataframe

for fname in fnames:

df=pd.read\_excel('./data/'+fname,skiprows=3)#跳过前三行

#处理内容

dfall=pd.concat([dfall,df],ignore\_index=True)#竖向拼接 重新分配索引

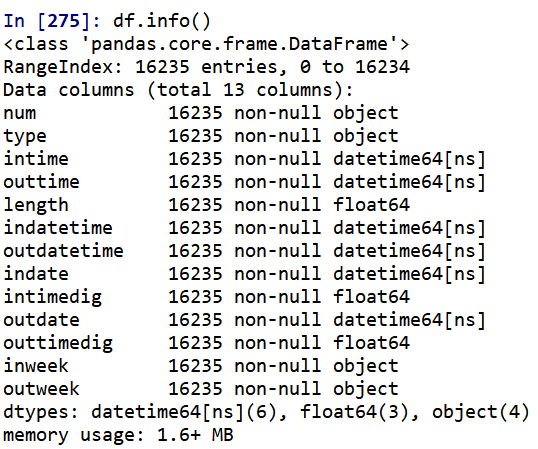
...

return dfall

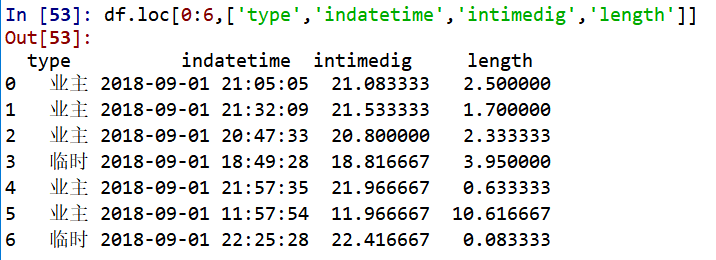
1. 文件内部调用：df=get\_data()

运行文件后，总的数据df就存在了工作区中。以后调用其他的函数文件处理数据，直接在命令行窗口将df作为输入参数，不需要重新生成。这样增加了计算效率（生成df需要几秒钟时间）。

结果：



部分数据展示：(intimedig 表示开始停车时间 length表示停车时长，单位都是小时）



1. 画九月每天的不同车开始停车时刻与停车时长散点图，及不同车离开时刻与停车时长散点图 draw.py文件

在spider编译环境中，控制台(console)中已经有了df变量（运行上面的get\_data文件得到）。运行文件draw.py 在控制台中调用函数draw(df) 得到画出的许多张图存在了文件夹中。

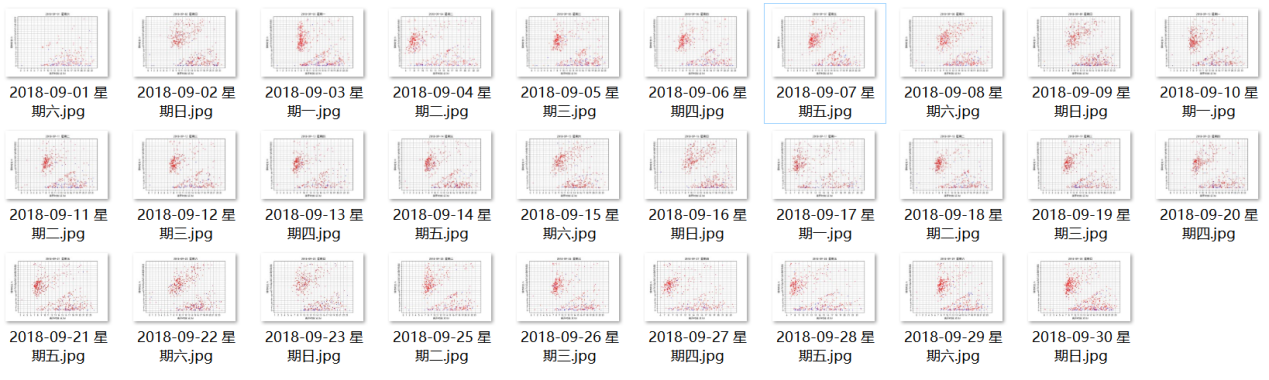
关键代码：

fig1=df.loc[(df['indate']==d)&(df['type']=='业主')&(df['length']<24),:]#忽略停车时长大于一天的

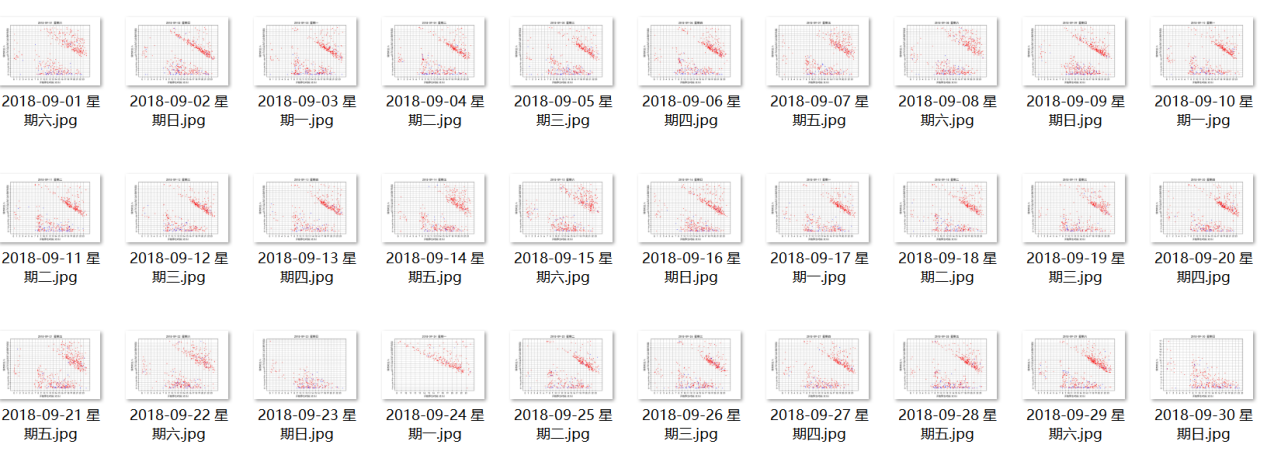
d为循环变量的其中一天。loc方法可以查找想要的值对应的行。

from matplotlib import pyplot as plt 利用matplotlib库画图

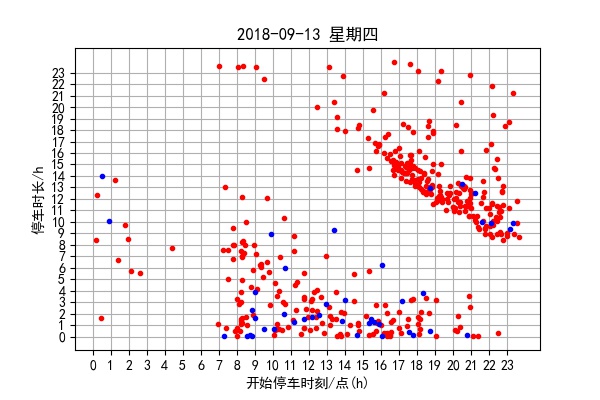
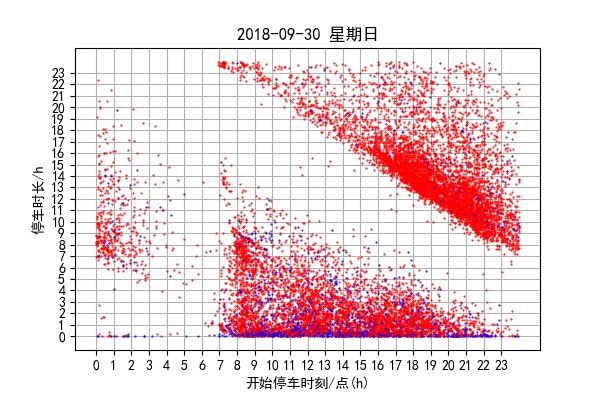
plt.plot(fig1['intimedig'],fig1['length'],'r.')



上图为9月的离开时刻-停车时长散点图。处理时发现excel表格文件23 和24重复了，于是将24.xls删去。而每一个表格内记录了当天离开的车的停车信息，所以没有9月24号的图。9.1,9.30作为边界日期，数据可能不全。



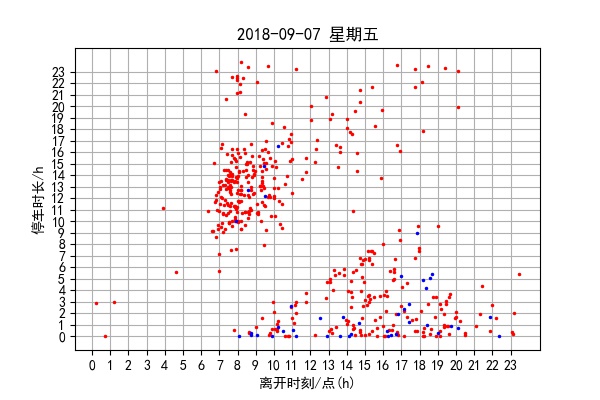
上图为9月的开始停车时刻与停车时长散点图。因为没有24号的离开时刻数据，9.23,9.24的图不要。

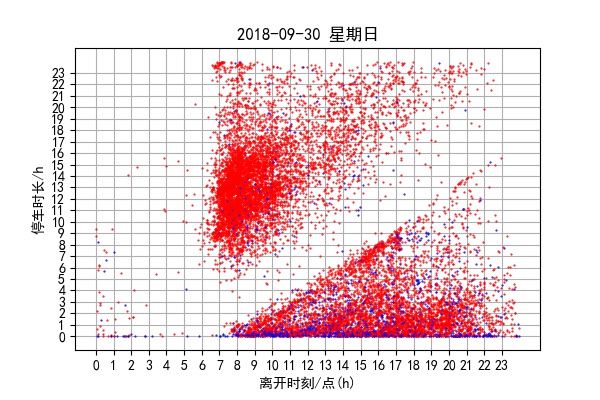


上图左为其中一天开始停车时刻与相应停车时长的散点图，红点为业主车的，蓝点为临时车的。上图右为9月所有天的图的叠加。编程中只要在循环结束不加plt.clf()#清空画布 命令即可。为了减少点的重叠，将点尺寸调小。

分析一个月的叠加图，可以得知开始停车的时刻从凌晨两点多至早上七点多很少。中间有一条斜向的条带，该条带区域内几乎没有点。解释如下。设开始停车时刻为x,停车时长y, 该条带范围大概在直线y=24-x到y=30-x之间，即24<x+y<30范围内几乎没有点。x+y为离开车场的时刻。所以意义是0点到早上6点之间几乎没有车离开车场。符合实际。

再看点分布最密的地方，为开始停车时刻17点至22点之间，开始停车时刻17点最密集的停车时长约14-16小时，开始停车时刻22点密集的停车时长从9-11小时左右，计算得知最密集的离开时刻为早7-9点左右。



上图左为其中一天离开车场时刻与相应停车时长的散点图，右图为一个月内的叠加图。分析一个月的图可以得到与开始停车图的相似的结论。凌晨0-6内几乎没有车离开车场。斜状的条带内点很少，代表凌晨2-7点开始停车（进入车场）的很少。集中离开时刻为早7-9点。

1. 画九月每天的不同时间段在库车数量 柱状图 draw\_bar.py文件

首先运行文件。然后调用get\_cum(df) 返回owner\_sum（不同时间段业主的在库车数量）, temp\_sum（不同时间段临时车的在库车数量）。最后owner\_sum, temp\_sum作为参数调用draw\_bar(owner\_sum, temp\_sum)，画出9月各天的不同时间段在库车数量柱状图

几个关键步骤：

1）切分时间

为了画柱状图，需要将连续的时间离散化，切分成不同的时间段。利用了dataframe 的.cut方法。

f='H'#采样间隔 小时 与f有关的为draw\_bar中name\_list Onum\_list Tnum\_list

timediv=pd.date\_range(start='2018-09-01',end='2018-10-01',freq=f)

#timedivide不同分隔时刻 为DatetimeIndex类型 每个元素datetime64[ns]类

labelnum=timediv.to\_series().size-1

#n个值，切成n-1段，共n-1个标签 eg:[0,1,2] 切分成[0,1),[1,2)

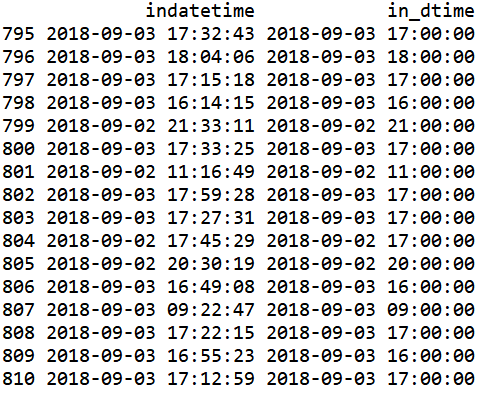
timelabel=pd.date\_range(start='2018-09-01',periods=labelnum,freq=f)

#periods表示生成labelnum个时刻标签 每个标签为时间段的左端点值

df['in\_dtime']=pd.cut(x=df['indatetime'],bins=timediv,right=False,labels=timelabel)

#discrete time 离散时间 right=False 区间左闭右开

切割结果 print(df.loc[795:810,['indatetime','in\_dtime']])



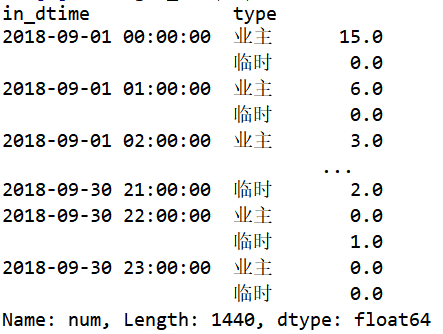
1. 分组，分别统计组内个数

按照时刻标签in\_dtime和停车类别type分组：ingroup=df.groupby([df['in\_dtime'],df['type']])

in\_num=ingroup.count().fillna(0)['num'] #每组计数，计数空值置0，取任意一列

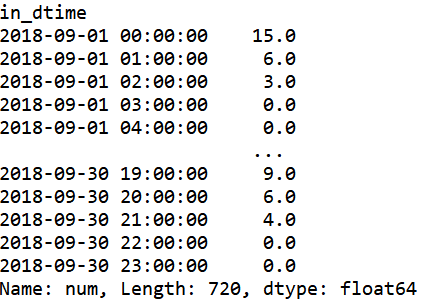
print(in\_num)#双重索引：in\_dtime type 一列值：数量

结果：



inOwner\_num=in\_num[:,'业主']#双重索引 业主的不同时间段的停入车数量

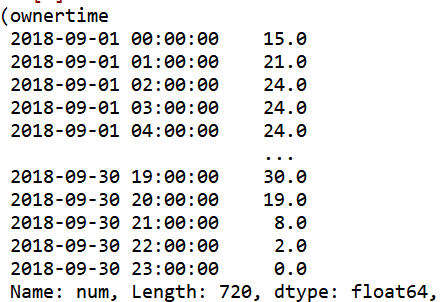
#inOwner\_num单索引：不同时间段的时刻标签in\_dtime 一列值：数量



Out同理in

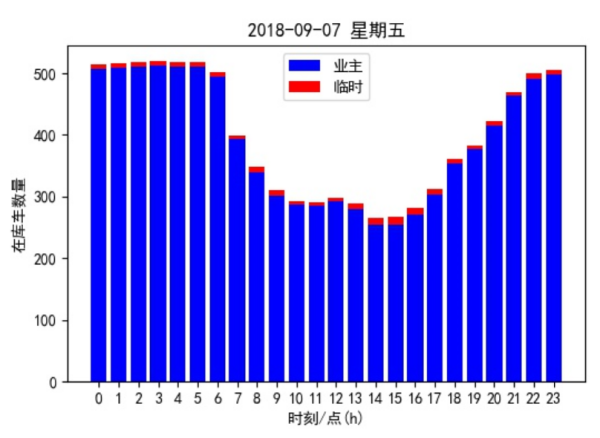
3）由不同时段的停入车数量和离开车数量，计算不同时段的在库车数量

owner\_sum=(inOwner\_num-outOwner\_num).cumsum()#累加 不同时间段业主的在库车数量

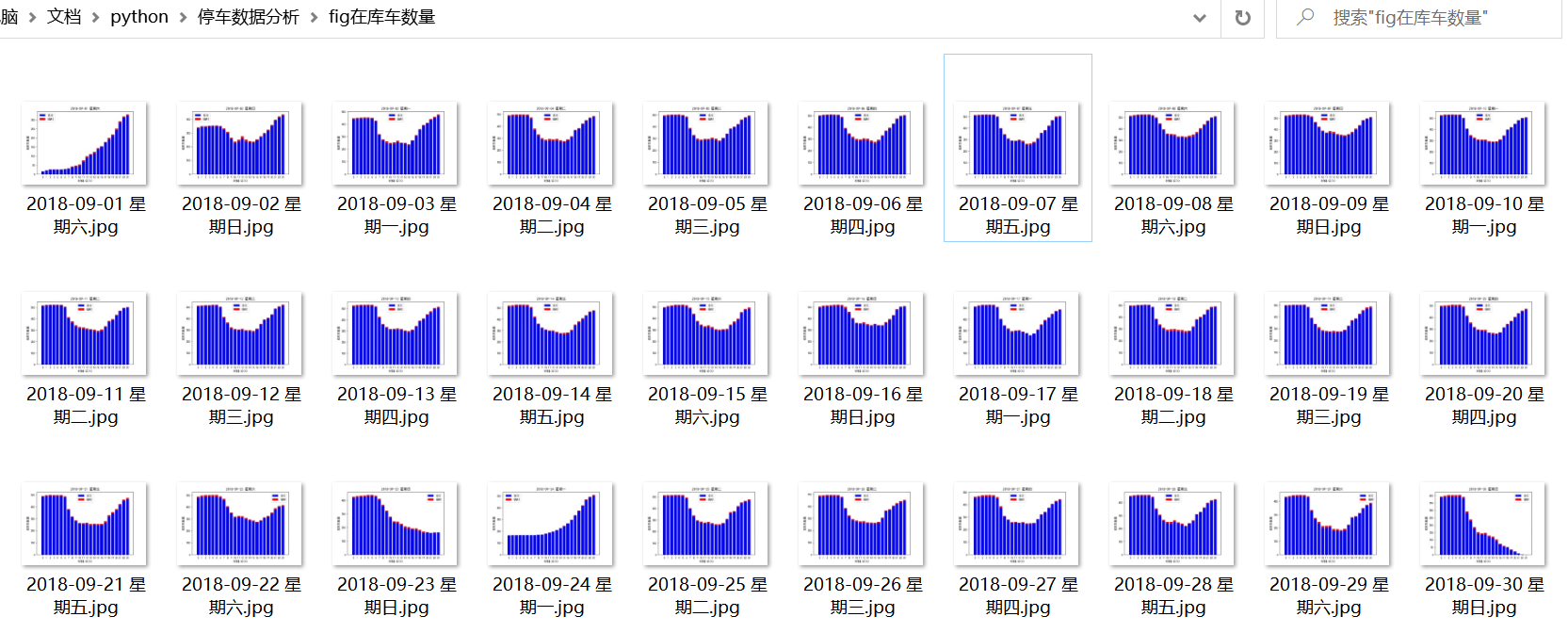


临时车temp同理

4）由不同时段的在库车数量owner\_sum和temp\_sum画出柱状图



上图可见在库车数在白天明显低于晚上，临时停车数占的比重很小，白天的车位没有被临时车充分利用。



1. 画9月在库车数量最大值与平均值 process.py文件

运行文件，在控制台中调用process(owner\_sum),owner\_sum为之前生成的不同时间段业主的在库车数量，画出9月工作日与周末在库车数量最大值与平均值。

关键部分

1. 编写了datetime2week(t)函数，输入Timestamp类型2018-09-01 00:10:00 输出weekday/weekend

import pandas as pd

date=t.date()#Timestamp date方法返回日期

interval=(date-pd.to\_datetime('2018-09-02').date()).days#2018-09-02为星期日 与其的天数差

#.days为Timedelta的属性 天数 int型

r=interval%7#余数0-6 余数0则为星期日 1为星期一

dic=[7,1,2,3,4,5,6]

week=dic[r]

if week in [6,7]:

return 'weekend'

else:

return 'weekday'

2）利用dataframe的分组+每组内调用某种方法（如.max(), .mean()）的基本操作

maxnum=dfn.groupby([dfn['week'],dfn['time']])['num'].max()

#Series类 双索引'week','time' 值'num'为相同索引下在库数量最大值

1. allbar函数

def allbar(s1,s2,S):#画统计好的不同时间段业主在库车数量 柱状图

#s1=mean/max s2=weekday/weekend 分组好处理过的Series S 如maxnum

#S第一个索引week 第二个索引time

在process函数中调用：

allbar('max','weekday',maxnum)

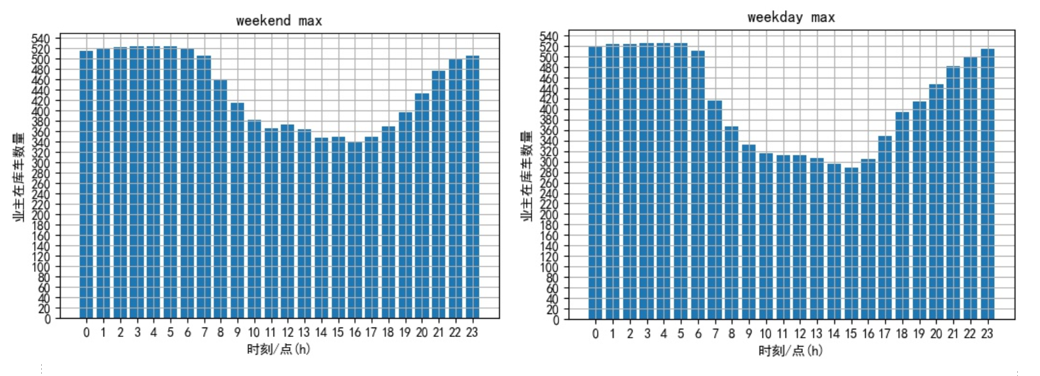
allbar('max','weekend',maxnum)

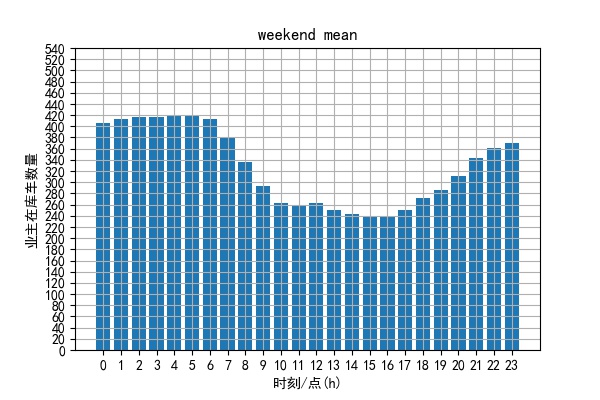
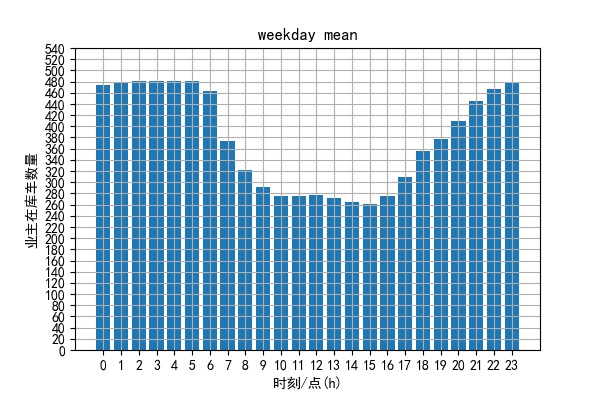
allbar('mean','weekday',meannum)

allbar('mean','weekend',meannum)

减少了代码量，增强了代码的可扩展性。

结果图：





可以根据工作日或周末的最大在库数量确定共享停车的时间窗口与最大共享数量。

1. 画甘特图 Gantt.py文件

运行文件，在控制台中调用Gantt(df) 得到9月每天的甘特图，每天的业主车辆在库时间段

关键：

1. 查找想要的数据

fig=df.loc[(df['indate']==d)&(df['type']=='业主')&(df['length']<24),['length','intimedig']]#忽略停车时长大于一天的

#选取相应日期 type为业主的行，保留'length','intimedig'两列 构成datafram:fig

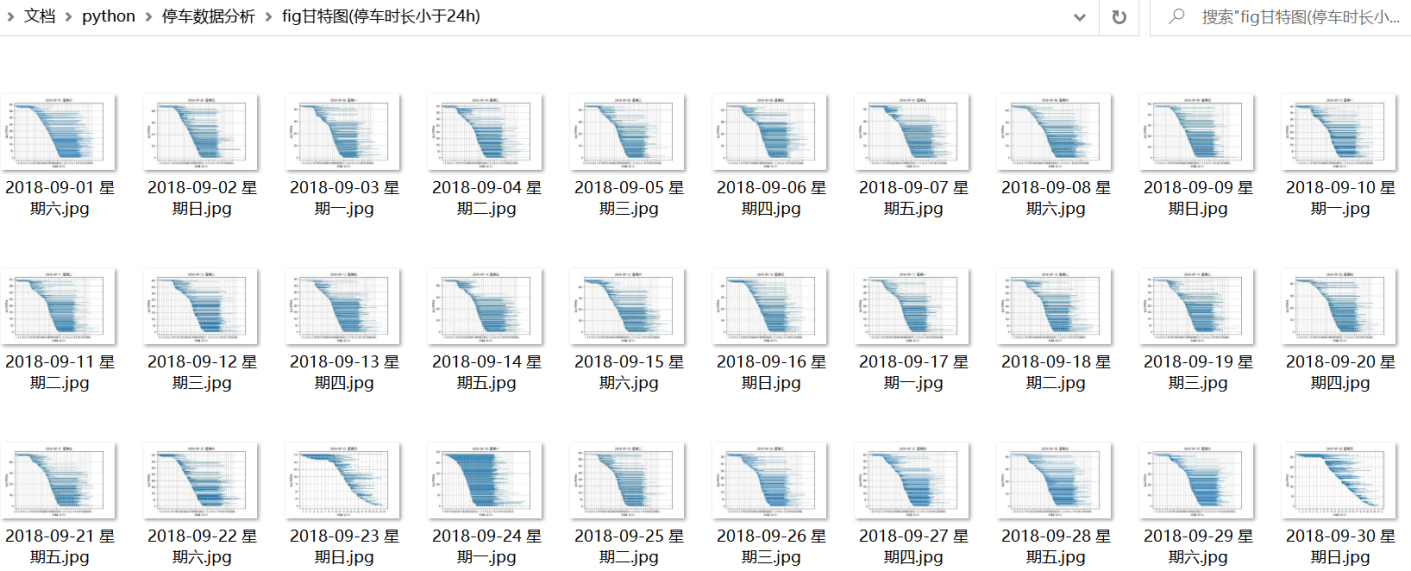
1. 排序

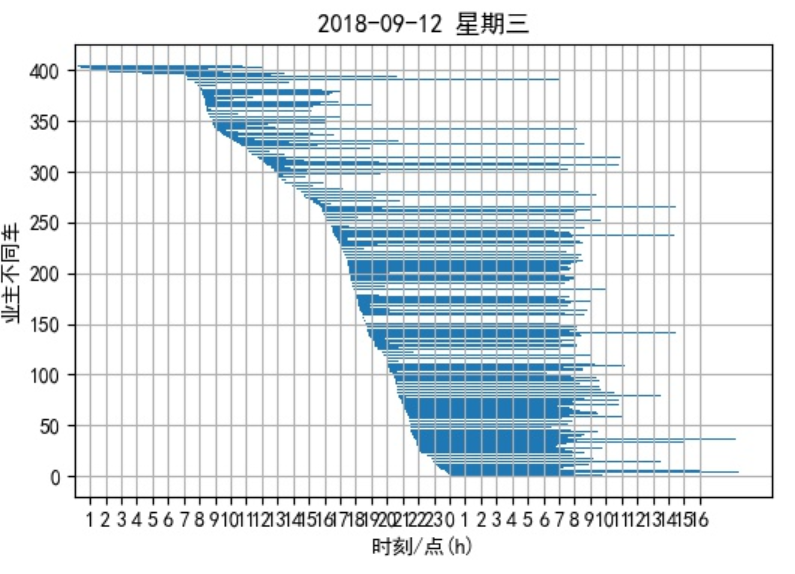
fig=fig.sort\_values('intimedig',ascending=False)#按开始停车时间降序排序

1. 甘特图可理解为横向条形图

plt.barh(y=idlist,width=fig['length'].values,left=fig['intimedig'].values)

结果图：





上图中每一个横条代表一个业主停车记录的在库时间段。起始点为当天入库的时刻（利用的intimedig,之前计算出的，单位为小时的，数字化的连续时刻），终止点为当次停车的离开时刻（限制停车时长小于24小时）。横坐标后半段的0,1,...时刻为下一天的。纵坐标为给每个业主停车记录的编号。开始停车时刻都是当天的（9.12），但许多都是下一天离开。图中没有前一天停车，当天离开的停车记录。

1. **总结及展望**

本次大作业停车数据分析有以下收获：

1. 处理了30个excel表格文件，学会了利用os库得到文件名，利用循环高效处理每个文件
2. 写了一个文件从excel表中读取总的数据（需要几秒），再运行其他文件，将已生成的数据作为输入参数进行处理，画图，不再需要重新从文件中读取数据。第一次尝试用这种方式，提高了程序的编译，调试效率。这是一种模块化的处理方法。
3. 学习了python数据分析必备的pandas库，运用了其中的dataframe类存储数据。学习使用了dataframe的各种方法：查找loc, 合并concat, 删drop，删空dropna, 排序sort\_values, 分组groupby, 切分cut, 类型转换astype... 学习了dataframe的各种数据结构，其中的datatime64类利用的很多。
4. 由提取出的数据画出了3种图：散点图，柱状图，甘特图。学习了利用matplotlib库的pyplot画图。
5. 对三种图进行分析，发现符合实际。利用一个月中车辆最大在库数量柱状图，可以确定共享停车的时间窗口与最大共享数量。

展望：还可对数据进行进一步的分析，如根据数据建立业主停车时间的随机分布函数，得到在一定置信度下的停车位空缺状态，从而确定共享停车方案。