**人工智能实验报告**

班 级： 1803

学 号： 19170333

姓 名： 樊世雄

实验日期： 2020-9-22

南京师范大学 计算机科学与技术学院制

数据爬取和数据分析

# 一、实验内容

Python一个重要的应用就是数据爬取，可以从互联网上轻松获取大量的数据用于数据分析。本案例使用Python爬取城市的历史天气，然后对天气进行分析。

# 二、实验步骤

1、检查实验中用到的库是否齐全，如未安装请先安装。

2、分析网站<http://www.tianqihoubao.com>中历史数据的网址规律，得到需要爬取的网址。

3、向服务器请求数据，将服务器返回的数据解析，并保存到文件中。

4、对数据进行分析（可考虑使用pandas库）。如某天的历史最高温和最低温是多少？该天晴天和下雨的概率是多少？哪天下雨的概率最高？哪天晴天的概率最高？最高温度和最低温度出现在哪天？

爬取数据代码：

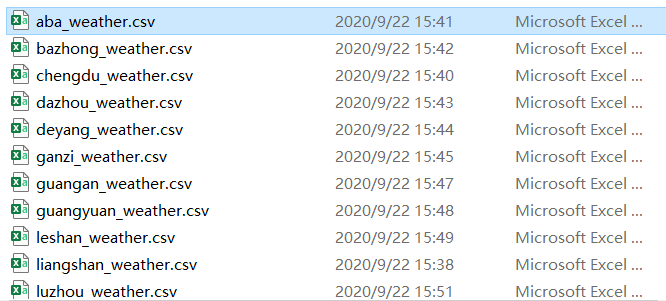
# https://blog.csdn.net/weixin\_43327576/article/details/86514093  
**import** requests  
**from** requests.exceptions **import** RequestException  
**from** bs4 **import** BeautifulSoup  
**import** csv  
**import** time  
  
  
**def** get\_one\_page(*url*):  
 '''  
 获取网页  
 '''  
 print(**'正在加载'** + *url*)  
 headers = {**'User-Agent'**: **'User-Agent:Mozilla/5.0'**}  
 **try**:  
 response = requests.get(*url*, headers=headers)  
 **if** response.status\_code == **200**:  
 **return** response.content  
 **return None  
 except** RequestException:  
 **return None  
  
  
def** parse\_one\_page(*html*):  
 '''  
 对网页内容进行解析  
 '''  
 soup = BeautifulSoup(*html*, **"lxml"**)  
 info = soup.find(**'div'**, class\_=**'wdetail'**)  
 rows = []  
 tr\_list = info.find\_all(**'tr'**)[**1**:] # 使用从第二个tr开始取  
 **for** index, tr **in** enumerate(tr\_list): # enumerate可以返回元素的位置及内容  
 td\_list = tr.find\_all(**'td'**)  
 date = td\_list[**0**].text.strip().replace(**"\n"**, **""**) # 取每个标签的text信息，并使用replace()函数将换行符删除  
 #weather = td\_list[1].text.strip().replace("\n", "").split("/")[0].strip()  
 weather = td\_list[**1**].text.strip().replace(**"\n"**, **""**).split(**"/"**)[**0**].strip() + **"/"** + td\_list[**1**].text.strip().replace(**"\n"**, **""**).split(**"/"**)[**1**].strip()  
 temperature\_high = td\_list[**2**].text.strip().replace(**"\n"**, **""**).split(**"/"**)[**0**].strip()  
 temperature\_low = td\_list[**2**].text.strip().replace(**"\n"**, **""**).split(**"/"**)[**1**].strip()  
  
 # print(td\_list[0].text + " " + td\_list[1].text + " " + td\_list[2].text + "\n")  
 # print(td\_list[0].text.strip().replace("\n", "") + " " + td\_list[1].text.strip().replace("\n", "") + " " + td\_list[2].text.strip().replace("\n", "") + "\n")  
 # print(date + " " + weather + " " + temperature\_high + " " + temperature\_low + "\n")  
  
 rows.append((date, weather, temperature\_high, temperature\_low))  
 **return** rows  
  
  
# 爬取的城市拼音名称，实验中可只选取一个城市  
cities = [**'chengdu'**,**'aba'**,**'bazhong'**,**'dazhou'**,**'deyang'**,**'ganzi'**,**'guangan'**,  
 **'guangyuan'**,**'leshan'**,**'luzhou'**,**'meishan'**,**'mianyang'**,**'neijiang'**,**'nanchong'**,  
 **'panzhihua'**,**'scsuining'**,**'yaan'**,**'yibin'**,**'ziyang'**,**'zigong'**,**'liangshan'**]  
  
years = [**'2018'**, **'2019'**]  
months = [**'01'**, **'02'**, **'03'**, **'04'**, **'05'**, **'06'**, **'07'**, **'08'**, **'09'**, **'10'**, **'11'**, **'12'**]  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 **'''  
 with open(cities[2] + '\_weather.csv', 'a', newline='') as f:  
 writer = csv.writer(f)  
 writer.writerow(['date', 'weather', 'temperature\_high', 'temperature\_low'])  
 url = 'http://www.tianqihoubao.com/lishi/' + cities[2] + '/month/' + years[0] + months[0] + '.html'  
 html = get\_one\_page(url)  
 content = parse\_one\_page(html)  
 writer.writerows(content)  
 print(cities[2] + years[0] + months[0] + ' is OK!')  
 '''** # os.chdir() # 设置工作路径  
  
 **for** city **in** cities:  
 **with** open(city + **'\_weather.csv'**, **'a'**, newline=**''**) **as** f:  
 writer = csv.writer(f)  
 writer.writerow([**'date'**, **'weather'**, **'temperature\_high'**, **'temperature\_low'**])  
 **for** year **in** years:  
 **for** month **in** months:  
 url = **'http://www.tianqihoubao.com/lishi/'** + city + **'/month/'** + year + month + **'.html'** html = get\_one\_page(url)  
 content = parse\_one\_page(html)  
 writer.writerows(content)  
 print(city + year + month + **' is OK!'**)  
 time.sleep(**0.01**)

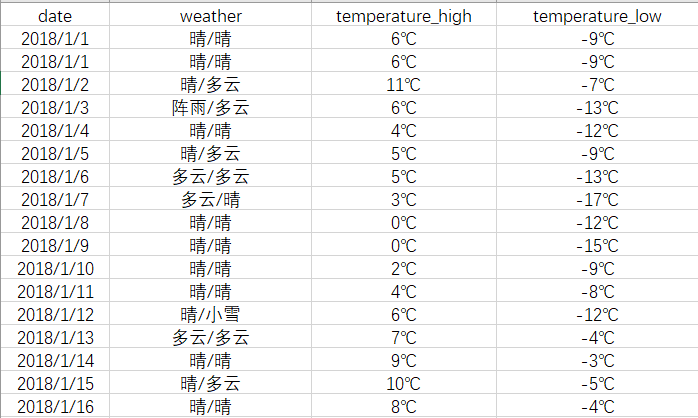
数据分析代码：

# -\*- codeing = utf-8 -\*-  
  
**import** pandas  
**from** matplotlib **import** pyplot **as** plt  
**from** pyecharts.charts **import** Bar  
**from** pyecharts.charts **import** Line  
**from** pyecharts.globals **import** ThemeType  
**from** pyecharts **import** options **as** opts  
  
# 处理中文  
**from** pylab **import** \*  
  
mpl.rcParams[**'font.sans-serif'**] = [**'SimHei'**]  
  
cities = [**'chengdu'**, **'aba'**, **'bazhong'**, **'dazhou'**, **'deyang'**, **'ganzi'**, **'guangan'**,  
 **'guangyuan'**, **'leshan'**, **'luzhou'**, **'meishan'**, **'mianyang'**, **'neijiang'**, **'nanchong'**,  
 **'panzhihua'**, **'scsuining'**, **'yaan'**, **'yibin'**, **'ziyang'**, **'zigong'**, **'liangshan'**]  
  
all\_min\_weather = [**35**, **28**, **38**, **41**, **36**, **28**, **39**, **36**, **37**, **39**, **37**, **36**, **38**, **39**, **42**, **39**, **35**, **38**, **38**, **38**, **36**]  
all\_max\_weather = [-**2**, -**19**, -**2**, -**1**, -**2**, -**17**, -**2**, -**4**, **0**, **0**, -**1**, -**2**, -**1**, -**1**, -**8**, -**2**, -**2**, -**1**, -**2**, **0**, -**4**]  
  
  
**def** draw\_img\_plt(*city*, *dates*, *highs*, *lows*):  
  
 # 分析数据  
 min\_weather = *lows*.idxmin()  
 max\_weather = *highs*.idxmax()  
 # all\_min\_weather.append(lows[min\_weather])  
 # all\_max\_weather.append(highs[max\_weather])  
 text\_max = **"最高温度："** + str(*highs*[max\_weather]) + **"℃ -- "** + str(*dates*[max\_weather])  
 text\_min = **"最低温度："** + str(*lows*[min\_weather]) + **"℃ -- "** + str(*dates*[min\_weather])  
 # print(lows[min\_weather])  
 # print(highs[max\_weather])  
  
 # 画图  
  
 fig = plt.figure(dpi=**128**, figsize=(**10**, **6**))  
  
 plt.plot(*dates*, *highs*, c=**'red'**, alpha=**0.5**)  
 plt.plot(*dates*, *lows*, c=**'blue'**, alpha=**0.5**)  
  
 plt.fill\_between(*dates*, *highs*, *lows*, facecolor=**'blue'**, alpha=**0.2**)  
 # 图表格式  
 # 设置图标的图形格式  
 plt.title(**'2 years weather'**, fontsize=**24**)  
 plt.xlabel(**''**, fontsize=**6**)  
 fig.autofmt\_xdate()  
 plt.ylabel(**'Temperature(℃)'**, fontsize=**12**)  
 plt.tick\_params(axis=**'both'**, which=**'major'**, labelsize=**10**)  
 # 修改刻度  
 plt.xticks(*dates*[::**60**])  
 # 显示最高和最低温度  
 plt.text(**570**, **0**, text\_max, size=**8**, color=**"r"**, alpha=**0.8**)  
 plt.text(**570**, -**2**, text\_min, size=**8**, color=**"b"**, alpha=**0.8**)  
 # 保存  
  
 img = *city* + **'.png'** plt.savefig(img, dpi=**100**)  
 # 显示  
 # plt.show()  
  
  
**def** draw\_img\_bar():  
 # print(cities)  
 print(all\_max\_weather)  
 print(all\_min\_weather)  
 cities = [**'cd'**, **'aba'**, **'bz'**, **'dz'**, **'dy'**, **'gz'**, **'gg'**,  
 **'gy'**, **'ls'**, **'lz'**, **'ms'**, **'my'**, **'nj'**, **'nc'**,  
 **'pzh'**, **'ss'**, **'y'**, **'yb'**, **'zy'**, **'zg'**, **'ls'**]  
 bar = (  
 # init\_opts = opts.InitOpts(width="1500px") # 设置可视化图的长度，1500像素  
 Bar()  
 .add\_xaxis(xaxis\_data=cities)  
 .add\_yaxis(**"Maximum Temperature"**, yaxis\_data=all\_max\_weather)  
 .add\_yaxis(**"Minimum Temperature"**, yaxis\_data=all\_min\_weather)  
 .set\_global\_opts(tooltip\_opts=opts.TooltipOpts(is\_show=**False**),  
 xaxis\_opts=opts.AxisOpts(  
 axislabel\_opts={**"interval"**: **"0"**}  
 ),  
 yaxis\_opts=opts.AxisOpts(  
 # 分割线配置，显示 y 轴每个刻度的分割线  
 splitline\_opts=opts.SplitLineOpts(is\_show=**True**),  
 ))  
  
 .render(**"weather.html"**)  
 )  
  
  
**def** draw\_img\_line():  
 cities = [**'cd'**, **'aba'**, **'bz'**, **'dz'**, **'dy'**, **'gz'**, **'gg'**,  
 **'gy'**, **'ls'**, **'lz'**, **'ms'**, **'my'**, **'nj'**, **'nc'**,  
 **'pzh'**, **'ss'**, **'y'**, **'yb'**, **'zy'**, **'zg'**, **'ls'**]  
 line = (Line()  
 .add\_xaxis(cities)  
 .add\_yaxis(**"Maximum Temperature"**, all\_max\_weather)  
 .add\_yaxis(**"Minimum Temperature"**, all\_min\_weather)  
 )  
 line.render(**"render.html"**)  
  
#计算历史上某一天里的晴雨的概率  
**def** calc\_weather(*month*, *day*, *dates*, *weather*, *is\_sun*):  
 # print("2020年5月2日".find(month + "月" + day + "日") != -1)  
 dates\_idx = list(filter(**lambda** *x* : x.find(*month* + **"月"** + *day* + **"日"**) != -**1**, *dates*))  
  
 # print(dates\_idx)  
  
 count1 = **0** # 下雨  
 **if** *is\_sun* == **0**:  
 **for** i **in** dates\_idx:  
 **if** *weather*[list(*dates*).index(i)].find(**"雨"**) != -**1** :  
 count1 += **1** print(count1 / np.size(dates\_idx))  
 #return np.size(list(filter(lambda x: ((dates[list(weather).index(x)] in dates\_idx) and weather[list(dates).index(x)].find("雨") != -1), weather))) / np.size(dates\_idx)  
 **return** count1 / np.size(dates\_idx)  
 # 晴  
 **else**:  
 **for** i **in** dates\_idx:  
 **if** *weather*[list(*dates*).index(i)].find(**"晴"**) != -**1** :  
 count1 += **1** print(count1 / np.size(dates\_idx))  
 #return np.size(list(filter(lambda x: ((dates[list(weather).index(x)] in dates\_idx) and weather[list(dates).index(x)].find("雨") != -1), weather))) / np.size(dates\_idx)  
 **return** count1 / np.size(dates\_idx)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
  
 **for** city **in** cities:  
 file = **'datas/'** + city + **'\_weather.csv'** datalsit = pandas.read\_csv(file, encoding=**'gbk'**)  
  
 # 数据处理  
 # datalsit['temperature\_high'] = datalsit['气温'].str.split('/', expand=True)[0]  
 # datalsit['temperature\_low'] = datalsit['气温'].str.split('/', expand=True)[1]  
  
 # print(datalsit['temperature\_high'])  
 datalsit[**'temperature\_high'**] = datalsit[**'temperature\_high'**].map(**lambda** x: int(x.replace(**'℃'**, **''**)))  
 datalsit[**'temperature\_low'**] = datalsit[**'temperature\_low'**].map(**lambda** x: int(x.replace(**'℃'**, **''**)))  
  
 dates = datalsit[**'date'**]  
 highs = datalsit[**'temperature\_high'**]  
 lows = datalsit[**'temperature\_low'**]  
 weather = datalsit[**'weather'**]  
  
 day1 = **"2019年05月04日"** # 拿到月份和日期  
 month = day1.strip().split(**"月"**)[**0**].split(**"年"**)[**1**]  
 day = day1.strip().split(**"月"**)[**1**].split(**"日"**)[**0**]  
 # print(month + " " + day)  
 prob = calc\_weather(month, day, dates, weather, **0**)  
 print(str(prob) + **"\n"**)  
 draw\_img\_plt(city, dates, highs, lows)  
  
 draw\_img\_bar()  
 draw\_img\_line()

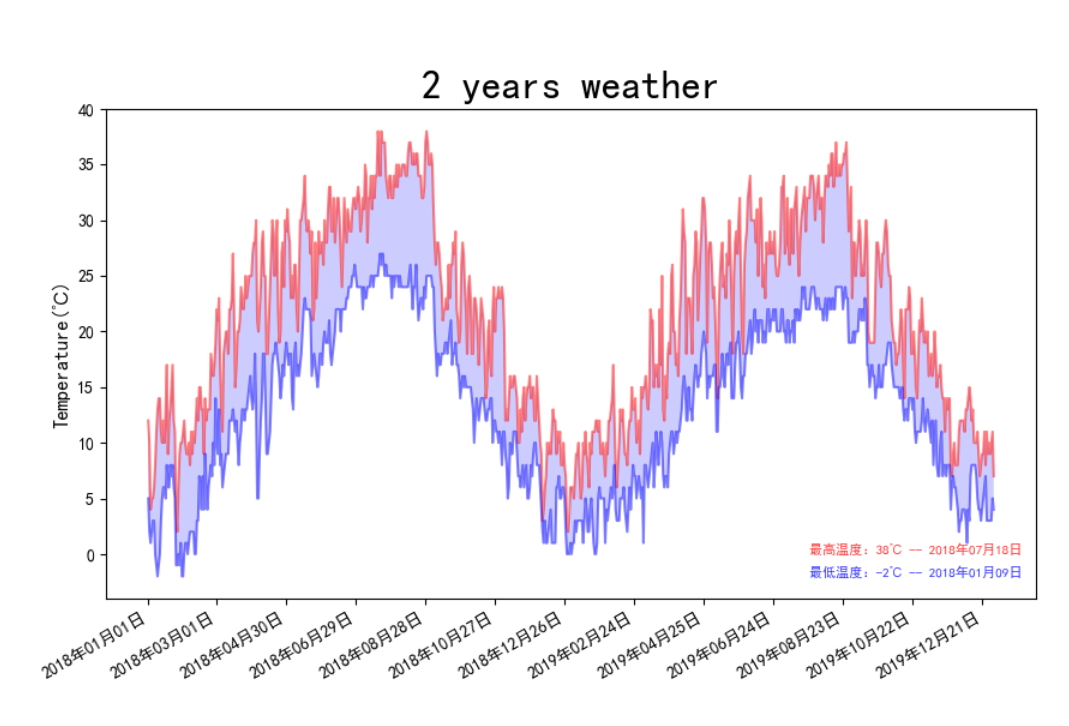
# 三、实验结果

1. 拿到了各个城市两年以内的天气数据：



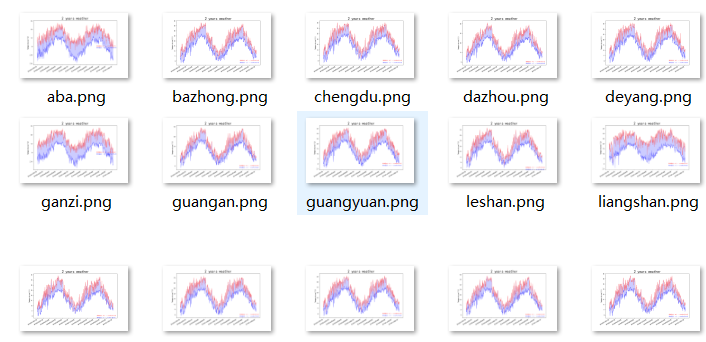


2. 通过分析数据，画出了各个城市2年以内的天气变化情况：

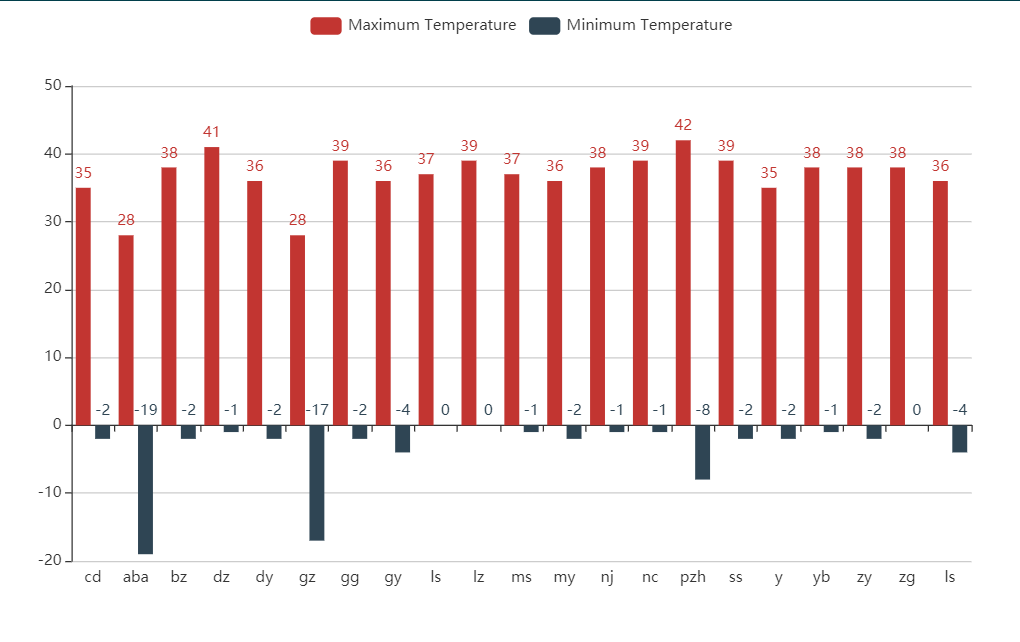


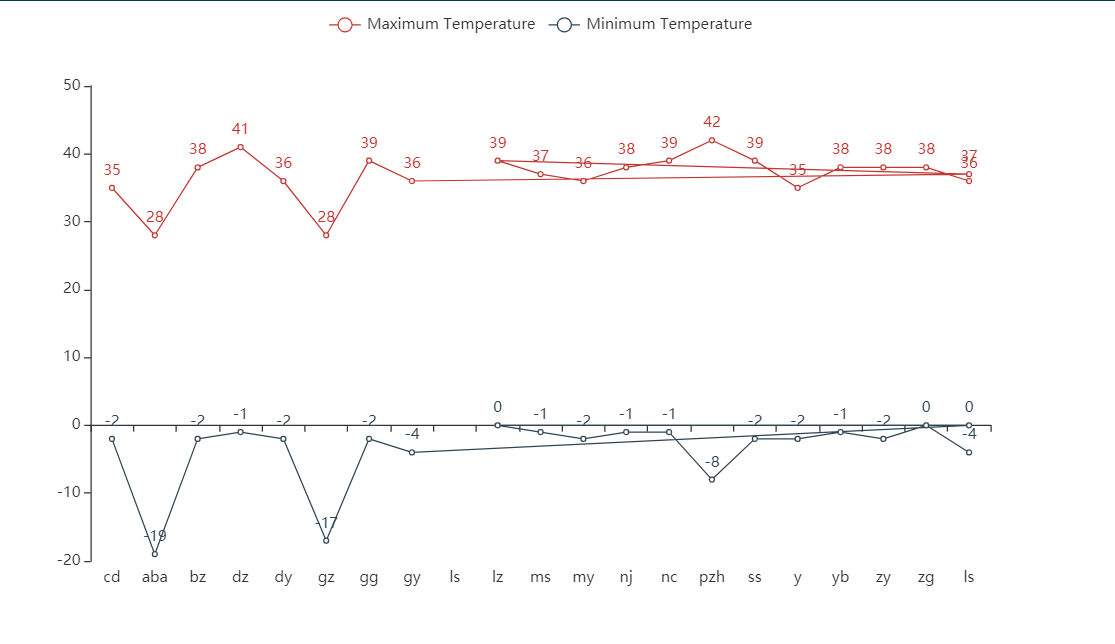
红色线条为最高温度，蓝色为最低温度。

图中右下角红色字体为2年内的最高温度，蓝色字体为2年以内的最低温度。



3. 根据各个城市的最低温度和最高温度，画出了各个城市的温度对比图：





4. 概率：

计算某一天下雨或者晴天的概率：

#通过计算历史上这天里的晴雨来计算当天的晴雨的概率  
**def** calc\_weather(*month*, *day*, *dates*, *weather*, *is\_sun*):  
 # print("2020年5月2日".find(month + "月" + day + "日") != -1)  
 dates\_idx = list(filter(**lambda** *x* : x.find(*month* + **"月"** + *day* + **"日"**) != -**1**, *dates*))  
  
 # print(dates\_idx)  
  
 count1 = **0** # 下雨  
 **if** *is\_sun* == **0**:  
 **for** i **in** dates\_idx:  
 **if** *weather*[list(*dates*).index(i)].find(**"雨"**) != -**1** :  
 count1 += **1** print(count1 / np.size(dates\_idx))  
 #return np.size(list(filter(lambda x: ((dates[list(weather).index(x)] in dates\_idx) and weather[list(dates).index(x)].find("雨") != -1), weather))) / np.size(dates\_idx)  
 **return** count1 / np.size(dates\_idx)  
 # 晴  
 **else**:  
 **for** i **in** dates\_idx:  
 **if** *weather*[list(*dates*).index(i)].find(**"晴"**) != -**1** :  
 count1 += **1** print(count1 / np.size(dates\_idx))  
 #return np.size(list(filter(lambda x: ((dates[list(weather).index(x)] in dates\_idx) and weather[list(dates).index(x)].find("雨") != -1), weather))) / np.size(dates\_idx)  
 **return** count1 / np.size(dates\_idx)

# 四、收获

通过本次实验，我对于python的使用更加熟练，尤其是beautifulsoup库，pyecharts 库以及matplotlib库。Python对于处理大数据有很大的优点。

虽然这次实验还不能称大数据，但是通过实验，我认识到了大数据的好处，通过分析数据来使得生活更加便利。