In [4]:

```
from IPython.core.interactiveshell import InteractiveShell  #执行该代码可以使得当前nb支持多输出
InteractiveShell.ast_node_interactivity = "all"
import sys
import pandas as pd
import numpy as np
```

第三章作业答案

作业1:

In [4]:

```
# 输入两个列表，要求统计两个列表相同元素的数量。
from random import shuffle
arr1 = list(range(2000))
shuffle(arr1)
arr2 = list(range(1000, 3000))
shuffle(arr2)
#补充代码
len(set(arr1) & set(arr2))
```

Out[4]:

```
1000
```

作业2

In [4]:

```
#自定义一个用户名列表，输入开头几位字母，程序自动补充用户名
name = ['Adam','Alex','Amy','Bob','Boom','Candy','Chris','David','Jason','Jasonstatham','Bill'];
i_name = input('please input name : ')
#补充代码
wname = [n for n in  name if   n[0:len(i_name)] == i_name]
if len(wname) != 0:
    print('Do you want to find %s?'%(wname))
else:
    print('%s not find'%(i_name))
```

```
please input name : Bo
Do you want to find ['Bob', 'Boom']?
```

作业3

In [1]:

```
all_data = [['Maria', 'Emily', 'Michael', 'Mary', 'Steven'],
            ['John', 'Emily', 'Michael', 'Mary', 'Steven1'],
            ['Maria', 'Juan', 'Javier', 'Natalia', 'Pilar']]

#代码实现只选取含有重复字符的名字，并实现去重功能

set(name for names in all_data  for name in names  if len(set(name))<len(name))
```

Out[1]:

```
{'Maria', 'Natalia', 'Steven', 'Steven1'}
```

第四章作业答案

作业

In [6]:

```
# 三扇门，其中一扇门后有奖品，这扇门只有主持人知道。选手先随机选一扇门，但并不打开，主持人看到后，会打开其余两扇门
#然后，主持人问选手，是否要改变一开始的选择？按照1000次，求选择改变和不改变，下一次选中奖品的比例
# 提示，利用random.randint初始化奖品所在的门，以及每次随机打开的门
# 结论，选择改变的下一次选中奖品的概率约等于2/3，不改变的概率约等于1/3
import numpy.random as random

random.seed(66)

# 做10000次实验
n_tests = 10000

# 生成每次实验的奖品所在的门的编号
# 0表示第一扇门，1表示第二扇门，2表示第三扇门
```

```
winning_doors = random.randint(0, 3, n_tests)

# 记录如果换门的中奖次数
change_mind_wins = 0

# 记录如果坚持的中奖次数
insist_wins = 0

# winning_door就是获胜门的编号
for winning_door in winning_doors:
    #补充代码
    first_try = random.randint(0, 3)
    remaining_choices = [i for i in range(3) if i != first_try]  #剩下可选的门
    if winning_door in remaining_choices:
        remaining_choices = [winning_door]  #如果获胜的门在剩下可选的门里面，那么主持人只能保留获胜的门
#     else:
#         remaining_choices.remove(random.choice(remaining_choices)) #否则随便去掉一扇门，也肯定留下
    change_mind_wins += 1 if remaining_choices[0] == winning_door else 0
    insist_wins += 1 if first_try == winning_door else 0

# 输出10000次测试的最终结果，和推导的结果差不多：
# You win 6616 out of 10000 tests if you changed your mind
# You win 3384 out of 10000 tests if you insist on the iniTIal choice
print(
'You win {1} out of {0} tests if you changed your mind\n'
'You win {2} out of {0} tests if you insist on the initial choice'.format(n_tests, change_mind_w
```

```
You win 6597 out of 10000 tests if you changed your mind
You win 3403 out of 10000 tests if you insist on the initial choice
```

第五章作业

作业

参照numpy，利用pandas实现飞机失事数据分析

In [48]:

```
# 飞机失事数据分析（numpy版）
import numpy as np
import csv
path = "Airplane_Crashes_and_Fatalities_Since_1908_.csv"

# 读入数据
with open(path, "r") as f:
    reader = csv.reader(f)
    content = [row for row in reader]

# 转为numpy
content_array = np.array(content)

print (content_array[:, :2])

content_array_shape = content_array.shape
# 查看总共有多少个样本和变量
print ("此数据有%s个样本，%s个变量"%content_array[1:].shape)

#--------------------------------------------------------

import datetime
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
# plot
def plot_stat(stat):
    fig, ax = plt.subplots(figsize=(32, 16))
    stat_items = stat.items()
#     print(stat_items)
    stat_items = sorted(stat_items, key=lambda x:x[0])
```

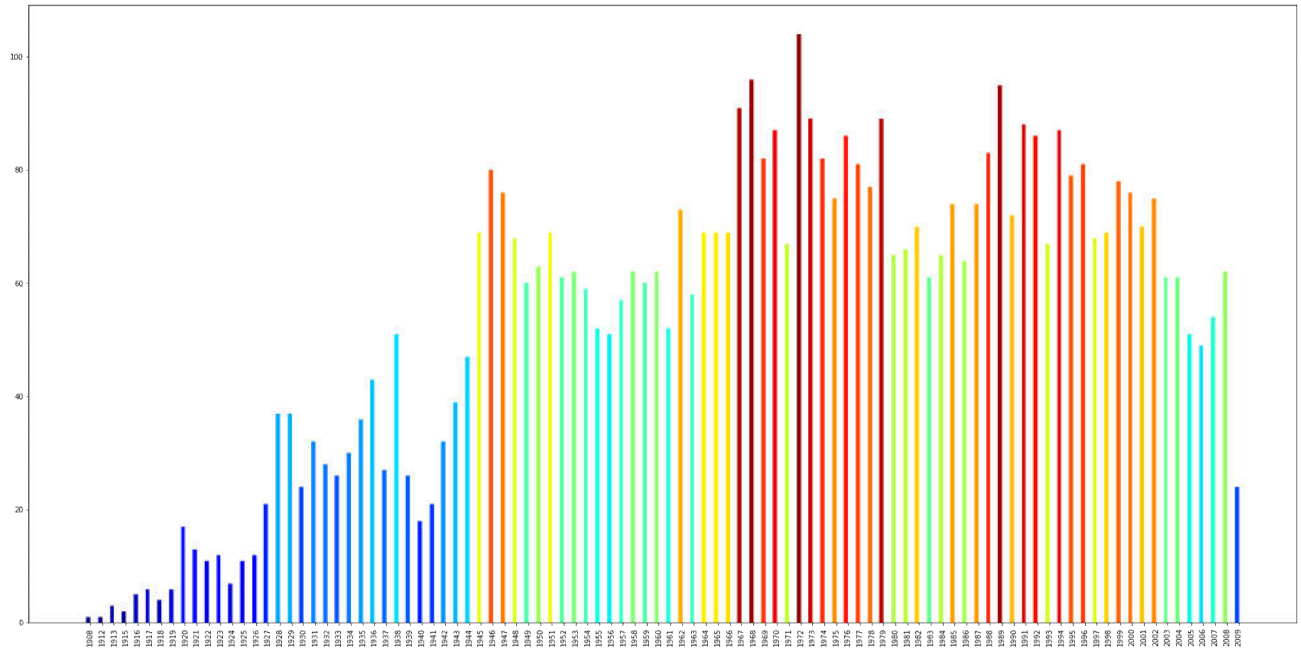
```
    year,accident_count=zip(*stat_items)

    #按照colormap画图，使得最大值颜色最深
    ind = np.arange(1, len(stat_items)+1)
#     print(list(zip(ind,accident_count)))
    width = 0.35
    N = len(stat_items)
    ind = sorted(ind, key=lambda x:accident_count[x-1])
    accident_count = sorted(accident_count)
#     print(list(zip(ind,accident_count)))  #本身年份和事故率依然是匹配的
    for i in range(N):
        ax.bar(ind[i],accident_count[i],width, color=cm.jet(1.*i/N))
#         ax.bar(ind[i],accident_count[i],width)  #无colormap的情况

#     重新按顺序排列颜色柱
    ind = np.array(sorted(ind))
#     print(ind)
    plt.xticks(ind + width/2., year, rotation=90)

    plt.show()
    return


stat_year = {}

# Date日期转为日期格式
# 统计每年发生的次数
for i in range(1,content_array.shape[0]):
    try:
#         print(content_array[i, 0],'***')
        year = datetime.datetime.strptime(content_array[i, 0], '%m/%d/%Y').year
        if year in stat_year.keys():
            stat_year[year] += 1
        else:
            stat_year[year] = 1
    except (Exception) as e:
        print(u"出错的元素: %s" %content_array[i, 0])
        print(e)
print(stat_year)
plot_stat(stat_year)

#-------------------------------------------------------
_content_array = [tuple(row) for row in content[1:]]    #注意不能覆盖前面的content_array，所以要用临时
type_array = [(row,object)  for row in content[0] ]
content_name = np.array(_content_array, dtype=type_array)

content_name["Aboard"][content_name["Aboard"]==""] = np.nan
content_name["Fatalities"][content_name["Fatalities"]==""] = np.nan
content_name["Aboard"] = content_name["Aboard"].astype(float)
content_name["Fatalities"] = content_name["Fatalities"].astype(float)
content_name["Aboard"][content_name["Aboard"] == 0] = np.nan
death_rate = content_name["Fatalities"] / content_name["Aboard"]
zero_death_rate = len(content_name[death_rate==0])

stat_operator = {}
for i in content_name["Operator"]:
    if i not in stat_operator.keys():
        stat_operator[i] = 1
    else:
        stat_operator[i] += 1

sorted_stat_operator = sorted(stat_operator.items(), key=lambda x:x[1], reverse=True)
max_operator = sorted_stat_operator[0][0]
print(max_operator)

#----------------------------
```

```
Aeroflot = content_array[1:][content_name["Operator"]=='Aeroflot']
stat_aeroflot = {}
for i in range(Aeroflot.shape[0]):
    try:
        year = datetime.datetime.strptime(Aeroflot[i, 0], '%m/%d/%Y').year
        if year in stat_aeroflot.keys():
            stat_aeroflot[year] += 1
        else:
            stat_aeroflot[year] = 1
    except (Exception) as e:
        print(u"出错的元素：%s" %Aeroflot[i, 0])
        print(e)
print(stat_aeroflot)
plot_stat(stat_aeroflot)
```

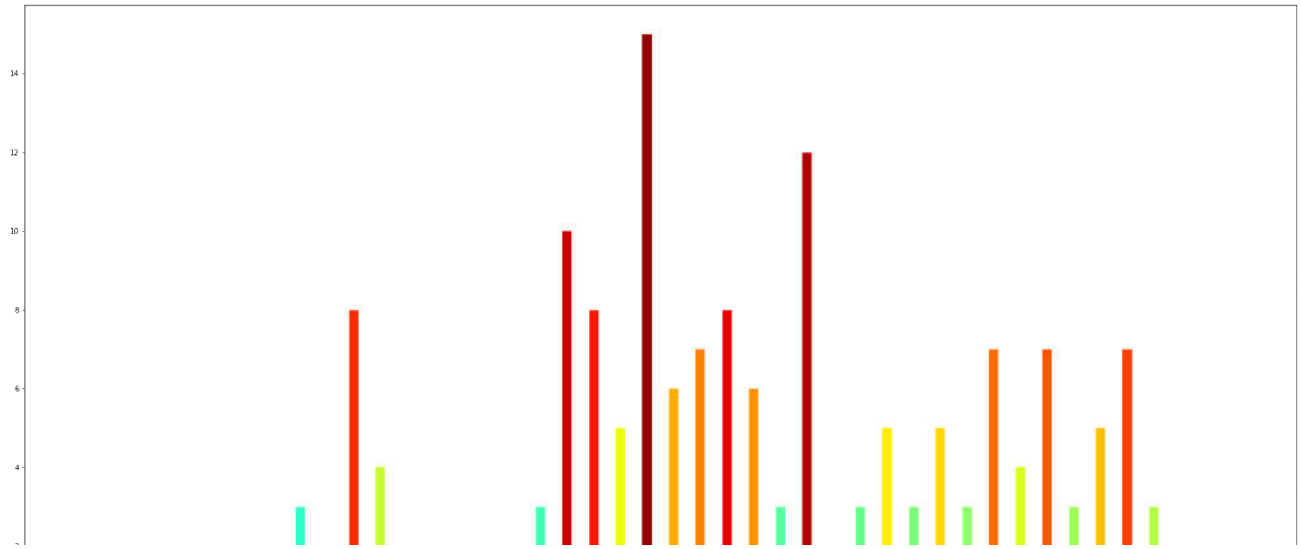
```
[['Date' 'Location']
 ['09/17/1908' 'Fort Myer, Virginia']
 ['07/12/1912' 'AtlantiCity, New Jersey']
 ...
 ['06/01/2009' 'AtlantiOcean, 570 miles northeast of Natal, Brazil']
 ['06/07/2009' 'Near Port Hope Simpson, Newfoundland, Canada']
 ['06/08/2009' 'State of Arunachal Pradesh, India']]
```

此数据有5268个样本，12个变量

```
{1908: 1, 1912: 1, 1913: 3, 1915: 2, 1916: 5, 1917: 6, 1918: 4, 1919: 6, 1920: 17, 1921: 13, 192
2: 11, 1925: 11, 1923: 12, 1924: 7, 1926: 12, 1927: 21, 1928: 37, 1929: 37, 1930: 24, 1931: 32,
1932: 28, 1933: 26, 1934: 30, 1935: 36, 1936: 43, 1937: 27, 1938: 51, 1939: 26, 1940: 18, 1941:
21, 1942: 32, 1943: 39, 1944: 47, 1945: 69, 1946: 80, 1947: 76, 1949: 60, 1948: 68, 1950: 63, 19
51: 69, 1966: 69, 1952: 61, 1953: 62, 1970: 87, 1955: 52, 1954: 59, 1958: 62, 1956: 51, 1957: 5
7, 1962: 73, 1959: 60, 1960: 62, 1973: 89, 1961: 52, 1964: 69, 1963: 58, 1965: 69, 1975: 75, 196
7: 91, 1968: 96, 1969: 82, 1971: 67, 1988: 83, 1991: 88, 1972: 104, 1974: 82, 1976: 86, 1977: 8
1, 1978: 77, 1979: 89, 1980: 65, 1981: 66, 1983: 61, 1982: 70, 1987: 74, 1984: 65, 1986: 64, 198
5: 74, 1989: 95, 1990: 72, 1992: 86, 1993: 67, 1994: 87, 1995: 79, 1996: 81, 1997: 68, 1998: 6
9, 1999: 78, 2001: 70, 2000: 76, 2002: 75, 2003: 61, 2004: 61, 2005: 51, 2006: 49, 2007: 54, 200
8: 62, 2009: 24}
```



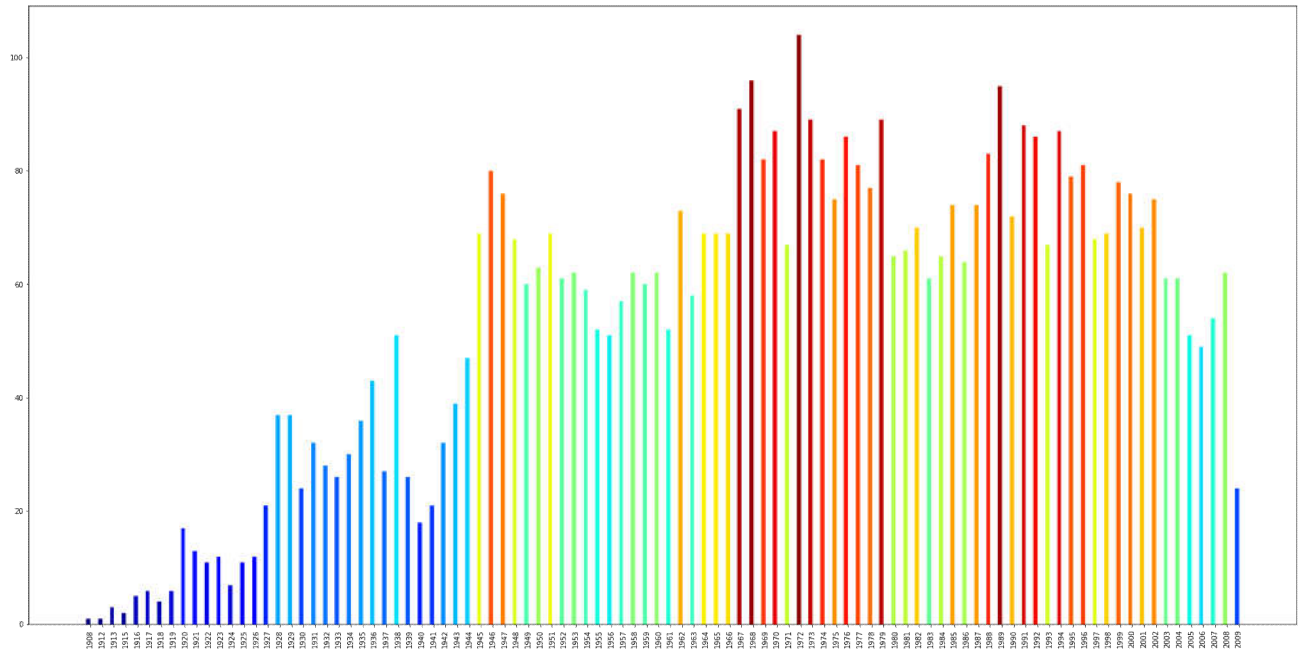
```
Aeroflot
{1946: 1, 1952: 1, 1953: 1, 1954: 2, 1955: 1, 1957: 1, 1958: 2, 1959: 2, 1973: 15, 1960: 3, 196
1: 1, 1962: 8, 1963: 4, 1964: 1, 1965: 2, 1975: 7, 1966: 1, 1967: 2, 1968: 2, 1969: 3, 1970: 1
0, 1971: 8, 1972: 5, 1974: 6, 1976: 8, 1977: 6, 1978: 3, 1979: 12, 1980: 2, 1981: 3, 1982: 5, 19
83: 3, 1987: 4, 1984: 5, 1985: 3, 1986: 7, 1988: 7, 1989: 3, 1990: 5, 1991: 7, 1992: 3, 1995:
2, 1996: 1, 2008: 1}
```



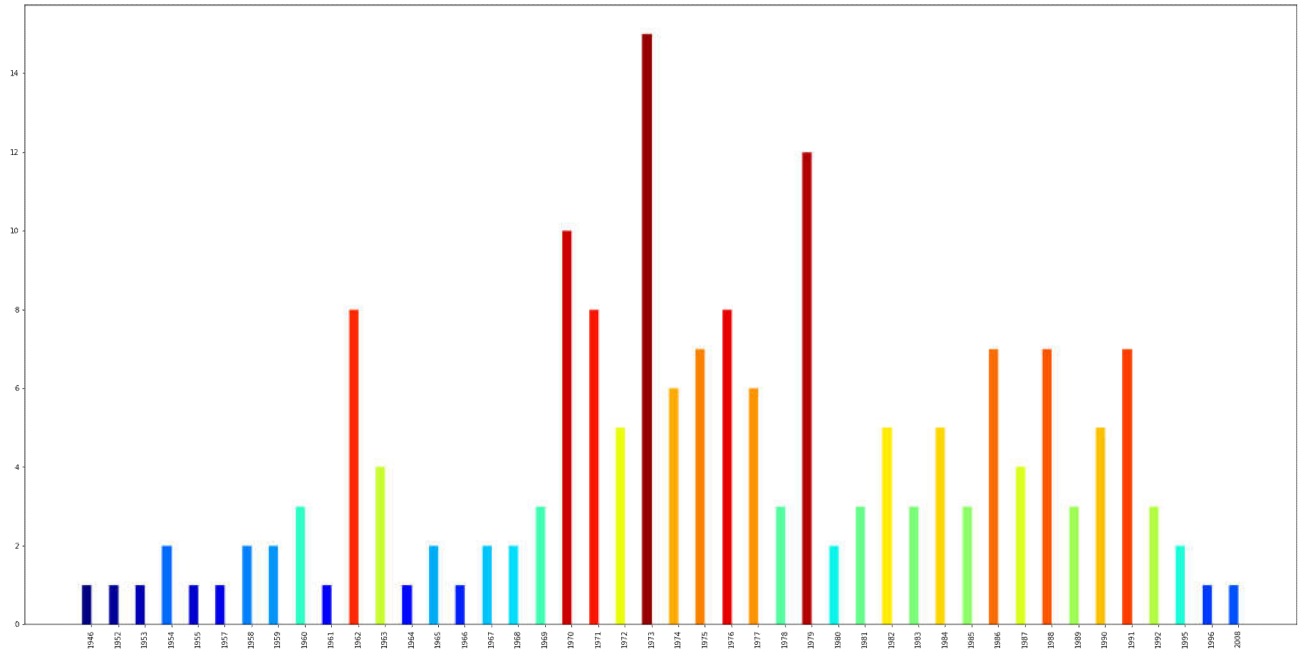
In [51]:

```
# 飞机失事数据分析（pandas版）
import pandas as pd
import csv
import datetime
path = "Airplane_Crashes_and_Fatalities_Since_1908_.csv"

def plot_stat(stat):
    fig, ax = plt.subplots(figsize=(32, 16))
    stat_items = stat.items()
#     print(stat_items)
    stat_items = sorted(stat_items, key=lambda x:x[0])
    year,accident_count=zip(*stat_items)

    #按照colormap画图，使得最大值颜色最深
   ind = np.arange(1, len(stat_items)+1)
#     print(list(zip(ind,accident_count)))
    width = 0.35
    N = len(stat_items)
    ind = sorted(ind, key=lambda x:accident_count[x-1])
    accident_count = sorted(accident_count)
#     print(list(zip(ind,accident_count)))  #本身年份和事故率依然是匹配的
   for i in range(N):
        ax.bar(ind[i],accident_count[i],width, color=cm.jet(1.*i/N))
#         ax.bar(ind[i],accident_count[i],width)  #无colormap的情况

#     重新按顺序排列颜色柱
   ind = np.array(sorted(ind))
#     print(ind)
    plt.xticks(ind + width/2., year, rotation=90)

    plt.show()
    return

#请输入代码完成numpy相同的功能
csv=pd.read_csv("Airplane_Crashes_and_Fatalities_Since_1908_.csv",encoding='gbk',parse_dates=['D
print ("此数据有%s个样本，%s个变量"%csv.shape)

stat_year=csv.Date.dt.year.value_counts().sort_index().to_dict()
plot_stat(stat_year)

max_operator =csv.Operator.value_counts()
print(max_operator.index[0])

stat_aeroflot=csv[csv.Operator==max_operator.index[0]].Date.dt.year.value_counts().sort_index().
plot_stat(stat_aeroflot)
```

此数据有5268个样本，12个变量



Aeroflot



第六章作业

作业1

利用pandas读入csv数据，求化合物的分子式和质量

In [2]:

```
# 元素周期表文件，给出任意一个化学表达式，计算其分子元素和质量。输出结果（不得再用C-O2作为输入）：

import csv
def readTable(fileObj):
    #补充代码
    table=pd.read_csv('periodic-table.csv',usecols=['symbol','name','atomicMass'])
    # table['atomicMass']=table['atomicMass'].str.findall(r'[.\d]+').apply(lambda x:x[0])  #第一
    table['atomicMass']=table['atomicMass'].apply(lambda x:re.findall(r'[.\d]+',x)[0])
```

```
    return table

def parserElement(elementStr,table):
    #补充代码
    tup=[(re.sub( "\d" , "", i),  re.sub( "\D" , "", i)   ) for i in elementStr.split('-') ]
    elnum=pd.DataFrame(tup,columns=['symbol','num']).replace({'':1})
    return pd.merge(elnum,table)

fileHandle = open('periodic-table.csv','r')
periodicTable = readTable(fileHandle)

compountString  = input('Input a chemical compound, hyphenated,eg. C-O2:')
#补充代码
# periodicTable
compountTable=parserElement(compountString,periodicTable)
compountTable
print('The compand is composed of :', compountTable.name.values)
print('The atomic mass of the compound is' ,sum(compountTable.num.astype(int)*compountTable.atom


fileHandle.close()
```

```
Input a chemical compound, hyphenated,eg. C-O2:Ca-C-O3
```

Out[2]:

	symbol	num	name	atomicMass
0	Ca	1	Calcium	40.078
1	C	1	Carbon	12.0107
2	O	3	Oxygen	15.9994

```
The compand is composed of : ['Calcium' 'Carbon' 'Oxygen']
The atomic mass of the compound is 100.0869
```

作业2

利用selenium抓取软科排名数据

In [5]:

```
# https://blog.csdn.net/qq_32897143/article/details/80383502
# https://www.jianshu.com/p/9006ba8478e6
# 参考上述两个教程，实现对页面上的"办学层次"进行定位，以及不需要轮询情况下直接针对item指定的内容进行定位，然后抓取
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("http://www.shanghairanking.cn/rankings/bcur/202011")
item='国际竞争力'

# 请出入代码，建议3行足够
driver.find_element_by_css_selector("input[value='办学层次']").click()
driver.find_element_by_xpath("//li[text()='"+item+"']").click()
pd.read_html(driver.page_source,encoding='utf-8')[1].rename(columns={'Unnamed: 5':item})


driver.quit()
```

Out[5]:

		排名	学校名称*	省市	类型	总分	国际竞争力
0	1		清华大学	北京	综合	852.5	79.9
1	2		北京大学	北京	综合	746.7	61.2
2	3		浙江大学	浙江	综合	649.2	43.0
3	4		上海交通大学	上海	综合	625.9	40.1
4	5		南京大学	江苏	综合	566.1	29.0
5	6		复旦大学	上海	综合	556.7	34.8
6	7		中国科学技术大学	安徽	理工	526.4	42.2
7	8		华中科技大学	湖北	综合	497.7	31.8
8	9		武汉大学	湖北	综合	488.0	25.2
9	10		中山大学	广东	综合	457.2	32.6
10	11		西安交通大学	陕西	综合	452.5	24.5
11	12		哈尔滨工业大学	黑龙江	理工	450.2	27.8
12	13		北京航空航天大学	北京	理工	445.1	24.6
13	14		北京师范大学	北京	师范	440.9	32.0
14	15		同济大学	上海	理工	439.0	21.8
15	16		四川大学	四川	综合	435.7	22.5
16	17		东南大学	江苏	综合	432.7	28.8
17	18		中国人民大学	北京	综合	409.7	15.5
18	19		南开大学	天津	综合	402.1	25.5
19	20		北京理工大学	北京	理工	395.6	20.8
20	21		天津大学	天津	理工	390.3	21.3
21	22		山东大学	山东	综合	387.9	20.8
22	23		厦门大学	福建	综合	383.3	23.0
23	24		吉林大学	吉林	综合	379.5	20.9
24	25		华南理工大学	广东	理工	379.4	27.5
25	26		中南大学	湖南	综合	378.6	23.8
26	27		大连理工大学	辽宁	理工	365.1	18.5
27	28		西北工业大学	陕西	理工	359.6	20.4
28	29		华东师范大学	上海	师范	358.0	18.5
29	30		中国农业大学	北京	农业	351.5	17.3
...	...		...	...	...	...	...
537	538		济宁学院	山东	师范	69.2	NaN
538	539		琼台师范学院	海南	师范	69.0	NaN
539	540		山东女子学院	山东	综合	68.9	NaN
540	540		陕西学前师范学院	陕西	师范	68.9	NaN
541	542		豫章师范学院	江西	师范	68.6	NaN
542	543		昭通学院	云南	师范	68.5	NaN
543	544		河南工学院	河南	理工	68.4	NaN
544	545		新疆理工学院	新疆	综合	68.0	NaN
545	546		汉江师范学院	湖北	师范	67.9	NaN

第七章作业

In [7]:

```
import re
import jieba
from collections import  defaultdict
from collections import Counter
import matplotlib.pyplot as plt
from wordcloud import WordCloud

def data_pre(data):
    note={i:x for i,x in enumerate(data) if re.findall(r'^【注解】',x) } #单独获得注解，以及对应的行数
    author={i:x.strip('\n')[3:] for i,x in enumerate(data) if re.findall(r'^作者',x) }  #单独获得作
    title={i:x.strip('\n') for i,x in enumerate(data) if i in [j-1 for j in author.keys()]} #单独
    return author,title,note   #返回之后再处理

def doubleword (words_lst):
    vc=pd.Series([x for x in words_lst if len(x)==2]).value_counts()
    return list(vc[vc>=2].items())

def poem (tanc_all):
    return [data[line].strip('\n') for tanc in tanc_all for line in tanc[-1]]  #tanc[-1]是诗句所在

data = open('唐诗三百首完整版.txt','rt',encoding = 'utf-8').readlines()
authorAll,titleAll,noteAll  = data_pre(data)  #所有作者的行号、所有题目的行号、所有注解的行号

#求tanc结构 ，即title、author、note、content
tanc = defaultdict(list)
for key, t,a,n in zip(authorAll.values(),titleAll.keys(),authorAll.keys(),noteAll.keys()):  #每个
    tanc[key].append([t,a,n,list(range(a+1,n))])
print(tanc['张九龄']) #用于测试结构

author='李白'
data_lst=poem(tanc[author]) #返回该作者全部诗句列表
# print(data_lst)
jb=jieba.lcut_for_search(''.join(data_lst))  #生成jieba分词
print( '双字词中，词频大于1次的包括：',doubleword(jb))
w = WordCloud(font_path="C:\\Windows\\Fonts\\STFANGSO.ttf")
my_wordcloud=w.generate(' '.join(jb))  #生成词云
plt.imshow(my_wordcloud)
plt.axis("off")
plt.show()
```

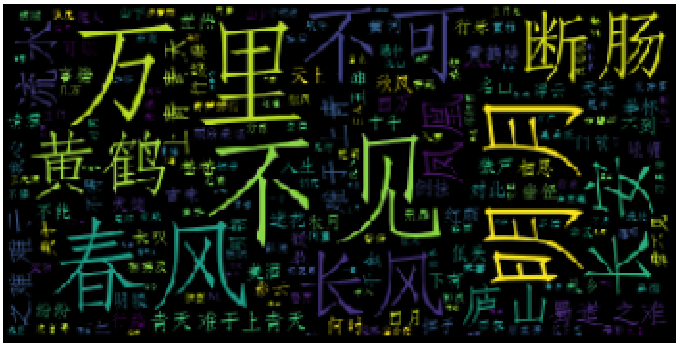
```
[[1, 2, 7, [3, 4, 5, 6]], [21, 22, 28, [23, 24, 25, 26, 27]], [3053, 3054, 3059, [3055, 3056, 3057, 3058]]]
双字词中，词频大于1次的包括： [('青天', 7), ('不见', 5), ('万里', 5), ('春风', 4), ('不可', 4), ('明月', 4), ('长风', 3), ('难于', 3), ('庐山', 3), ('流水', 3), ('断肠', 3), ('上青', 3), ('蜀道', 3), ('黄鹤', 3), ('之难', 3), ('长安', 3), ('凤凰', 3), ('人生', 2), ('名山', 2), ('行乐', 2), ('故乡', 2), ('如麻', 2), ('古来', 2), ('不得', 2), ('挥手', 2), ('幽径', 2), ('苍苍', 2), ('秋月', 2), ('门前', 2), ('我心', 2), ('纷纷', 2), ('五岳', 2), ('日月', 2), ('天上', 2), ('低头', 2), ('高楼', 2), ('峨嵋', 2), ('下有', 2), ('浮云', 2), ('不能', 2), ('秋风', 2), ('彩云', 2), ('相思', 2), ('对此', 2), ('不到', 2), ('可以', 2), ('红颜', 2), ('一生', 2), ('行路', 2), ('倒挂', 2), ('请君', 2), ('猿声', 2), ('何时', 2), ('千金', 2), ('天长', 2), ('茫然', 2), ('美酒', 2), ('十千', 2), ('长叹', 2), ('迷花', 2), ('夫子', 2), ('天姥', 2), ('黄河', 2), ('明镜', 2), ('四万', 2), ('举杯', 2), ('明朝', 2)]
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x8649f60>
```

Out[7]:

```
(-0.5, 399.5, 199.5, -0.5)
```



第八章作业

作业 综合运用stack/unstack，pivot/melt

In [5]:

```
#有可能是版本问题，需要加入.strftime('%Y-%m-%d')
data = pd.read_csv('../examples/macrodata.csv')
data.index = pd.PeriodIndex(year=data.year, quarter=data.quarter,name='date').to_timestamp('D',
data = data.reindex(columns=pd.Index(['realgdp', 'infl', 'unemp'], name='item'))
data
```

Out[5]:

item	realgdp	infl	unemp
date			
1959-03-31	2710.349	0.00	5.8
1959-06-30	2778.801	2.34	5.1
1959-09-30	2775.488	2.74	5.3
1959-12-31	2785.204	0.27	5.6
1960-03-31	2847.699	2.31	5.2
...	...	...	...
2008-09-30	13324.600	-3.16	6.0
2008-12-31	13141.920	-8.79	6.9
2009-03-31	12925.410	0.94	8.1
2009-06-30	12901.504	3.37	9.2
2009-09-30	12990.341	3.56	9.6

203 rows × 3 columns

作业1: data通过stack/melt获得ldata

In [6]:

```
#请输入data使用stack实现的代码（不得使用melt）
```

```
ldata = data.stack().reset_index().rename(columns={0: 'value'})
ldata.head()

# 请输出data使用melt实现的代码（不得使用stack）
ldata = data.reset_index().melt(id_vars=['date']).sort_values(by=['date','item']).reset_index(dr
ldata
```

Out[6]:

	date	item	value
0	1959-03-31	realgdp	2710.349
1	1959-03-31	infl	0.000
2	1959-03-31	unemp	5.800
3	1959-06-30	realgdp	2778.801
4	1959-06-30	infl	2.340

Out[6]:

	date	item	value
0	1959-03-31	infl	0.000
1	1959-03-31	realgdp	2710.349
2	1959-03-31	unemp	5.800
3	1959-06-30	infl	2.340
4	1959-06-30	realgdp	2778.801
...	...	...	...
604	2009-06-30	realgdp	12901.504
605	2009-06-30	unemp	9.200
606	2009-09-30	infl	3.560
607	2009-09-30	realgdp	12990.341
608	2009-09-30	unemp	9.600

609 rows × 3 columns

作业2

ldata通过pivot/unstack恢复data

In [106]:

```
#请输入ldata使用pivot实现的代码（不得使用unstack）
pivoted = ldata.pivot('date', 'item', 'value')
pivoted.head()  #同data.head()

#请输入ldata使用unstack实现的代码（不得使用pivot）
pivoted = ldata.set_index(['date','item']).unstack('item')['value']
pivoted.head()  #同data.head()
```

Out[106]:

	item	infl	realgdp	unemp
date				
1959-03-31	0.00	2710.349	5.8	
1959-06-30	2.34	2778.801	5.1	
1959-09-30	2.74	2775.488	5.3	
1959-12-31	0.27	2785.204	5.6	
1960-03-31	2.31	2847.699	5.2	

Out[106]:

	item	infl	realgdp	unemp
date				
1959-03-31	0.00	2710.349	5.8	
1959-06-30	2.34	2778.801	5.1	
1959-09-30	2.74	2775.488	5.3	
1959-12-31	0.27	2785.204	5.6	
1960-03-31	2.31	2847.699	5.2	

作业3：

参照“经营数据分析.xlsx”，求解下列问题，使得输出与结果相同

In [2]:

```
##   单目标求解
年收益=年销售量*(销售单价-单件成本)-设备投资
求年收益最佳方案？

## 多目标求解 ----
期望值=[min(设备投资),min(单件成本),max(年销售量),max(销售单价),
    max(年收益)];
差值=每个方案中，各项数据与期望值的之差的平方和
求差值最佳方案？

# 不确定性决策分析 ----
## 分析方法
PLm=pd.DataFrame();# 损益矩阵 ProfitLoss matrix
PLm['畅销']= 12000*(销售单价-单件成本)-设备投资;
PLm['一般']= 8000*(销售单价-单件成本)-设备投资;
PLm['滞销']= 1500*(销售单价-单件成本)-设备投资;

## 分析原则----
# 乐观原则
lg=损益矩阵三种情况的最大值

# 悲观原则
bg=损益矩阵三种情况的最小值

# 折中原则
a=0.65  #65%的乐观概率
折中方案= a*lg + (1-a)*bg;
求折中最佳方案？

#  概率性决策分析 ----
## 期望值法 ----
probE=[0.1,0.65,0.25]; #畅销、一般、滞销的概率
期望值=损益矩阵*probE
求期望值最佳方案？
```

Out[2]:

方案	设备投资	单件成本	年销售量	销售单价	年收益
方案1	1500000	1700	8000	2900	8100000
方案2	2000000	1550	8000	2900	8800000
方案3	2500000	1400	8000	2900	9500000

Out[2]:

```
'方案3'
```

Out[2]:

方案	设备投资	单件成本	年销售量	销售单价	年收益	差值
方案1	1500000	1700	8000	2900	8100000	1960000090000
方案2	2000000	1550	8000	2900	8800000	740000022500
方案3	2500000	1400	8000	2900	9500000	1000000000000

Out[2]:

```
'方案2'
```

Out[2]:

Out[2]:

方案	畅销	一般	滞销
方案1	12900000	8100000	300000
方案2	14200000	8800000	25000
方案3	15500000	9500000	-250000

Out[2]:

方案	畅销	一般	滞销	折中
方案1	12900000	8100000	300000	8490000.0
方案2	14200000	8800000	25000	9238750.0
方案3	15500000	9500000	-250000	9987500.0

Out[2]:

```
'方案3'
```

Out[2]:

方案	畅销	一般	滞销	期望
方案1	12900000	8100000	300000	6630000.0
方案2	14200000	8800000	25000	7146250.0
方案3	15500000	9500000	-250000	7662500.0

Out[2]:

```
'方案3'
```

In [5]:

```
# 单目标求解
Tv=pd.read_excel('DaPy_data.xlsx','Target',index_col=0); #Tv  #目标值
Tv['年收益']=Tv.年销售量*(Tv.销售单价-Tv.单件成本)-Tv.设备投资;
Tv
Tv['年收益'].idxmax()  # 最佳方案

##  多目标求解 ----
Ev=[min(Tv.设备投资),min(Tv.单件成本),max(Tv.年销售量),max(Tv.销售单价),
    max(Tv.年收益)]; #理想值
Ev
Tv['差值']=((Tv-Ev)**2).sum(1); #差值
```

```
Tv
Tv['差值'].idxmin()

#  不确定性决策分析 ----
##   分析方法
PLm=pd.DataFrame();#PLm  # 损益矩阵 ProfitLoss matrix
PLm['畅销']= 12000*(Tv.销售单价-Tv.单件成本)-Tv.设备投资;
PLm['一般']= 8000*(Tv.销售单价-Tv.单件成本)-Tv.设备投资;
PLm['滞销']= 1500*(Tv.销售单价-Tv.单件成本)-Tv.设备投资;
PLm

##  分析原则----
# 乐观原则
print('---------------')
lg=PLm.max(1);# pd.concat([PLm,lg],axis=1)
lg
# 悲观原则
bg=PLm.min(1);# pd.concat([PLm,bg],axis=1)
# 折中原则
bg
a=0.65
PLm_zz=PLm.copy()
PLm_zz['折中']= a*lg + (1-a)*bg; # pd.concat([PLm,zz],axis=1)
PLm_zz
PLm_zz['折中'].idxmax()

# # 8.5 概率性决策分析 ----
# ## 8.5.1 期望值法 ----
# PLm  # 损益矩阵
probE=[0.1,0.65,0.25]; #初始概率
PLm_qw=PLm.copy()
PLm_qw['期望']=(probE*PLm).sum(1); #pd.concat([PLm,qw],axis=1)
PLm_qw
PLm_qw['期望'].idxmax()
```

Out[5]:

	设备投资	单件成本	年销售量	销售单价	年收益
方案					
方案1	1500000	1700	8000	2900	8100000
方案2	2000000	1550	8000	2900	8800000
方案3	2500000	1400	8000	2900	9500000

Out[5]:

```
'方案3'
```

Out[5]:

```
[1500000, 1400, 8000, 2900, 9500000]
```

Out[5]:

	设备投资	单件成本	年销售量	销售单价	年收益	差值
方案						
方案1	1500000	1700	8000	2900	8100000	1960000090000
方案2	2000000	1550	8000	2900	8800000	740000022500
方案3	2500000	1400	8000	2900	9500000	1000000000000

Out[5]:

```
'方案2'
```

Out[5]:

	畅销	一般	滞销
方案			
方案1	12900000	8100000	300000
方案2	14200000	8800000	25000
方案3	15500000	9500000	-250000

```
---------------
```

Out[5]:

```
方案
方案1    12900000
方案2    14200000
方案3    15500000
dtype: int64
```

Out[5]:

```
方案
方案1    300000
方案2     25000
方案3   -250000
dtype: int64
```

Out[5]:

	畅销	一般	滞销	折中
方案				
方案1	12900000	8100000	300000	8490000.0
方案2	14200000	8800000	25000	9238750.0
方案3	15500000	9500000	-250000	9987500.0

Out[5]:

```
'方案3'
```

Out[5]:

	畅销	一般	滞销	期望
方案				
方案1	12900000	8100000	300000	6630000.0
方案2	14200000	8800000	25000	7146250.0
方案3	15500000	9500000	-250000	7662500.0

Out[5]:

第九章作业

In [ ]:

知乎数据清洗整理和结论研究

作业要求：
1、数据清洗 - 去除空值
要求：创建函数
提示：fillna方法填充缺失数据，注意inplace参数

2、问题1 知友全国地域分布情况，分析出TOP20
要求：
① 按照地域统计 知友数量、知友密度（知友数量/城市常住人口），不要求创建函数
② 知友数量，知友密度，标准化处理，取值0-100，要求创建函数
③ 通过多系列柱状图，做图表可视化
提示：
① 标准化计算方法 = (X - Xmin) / (Xmax - Xmin)
② 可自行设置图表风格

3、问题2 知友全国地域分布情况，分析出TOP20
要求：
① 按照学校（教育经历字段） 统计粉丝数（‘关注者’）、关注人数（‘关注’），并筛选出粉丝数TOP20的学校，不要求创建函
② 通过散点图 → 横坐标为关注人数，纵坐标为粉丝数，做图表可视化
③ 散点图中，标记出平均关注人数（x参考线），平均粉丝数（y参考线）
提示：
① 可自行设置图表风格

'''

In [3]:

```
data1 = pd.read_csv('知乎数据_201701.csv', engine = 'python')
data2 = pd.read_csv('六普常住人口数.csv', engine = 'python')
```

In [4]:

```
# 数据清洗 - 去除空值
# 文本型字段空值改为“缺失数据”，数字型字段空值改为 0
# 要求：创建函数
# 提示：fillna方法填充缺失数据，注意inplace参数

# 该函数可以将任意数据内空值替换
def data_cleaning(df):
# 请填写代码**********
    return df.transform(lambda x:x.fillna( '缺失数据' if x.dtype ==  'object' else 0))

data1_c = data_cleaning(data1)
print('data_cleaning的结果')
data1_c.head(10)
```

data_cleaning的结果

Out[4]:

	_id	关注的收藏夹	关注	关注者	关注的问题	关注的话题	关注的专栏	职业1	职业2	回答	提问	收藏	个人简介	居住地	所在行业	教育经历	职业经历
0	587598	52	17	1	30	58	2	交通仓储	邮政	0.0	0.0	3.0	缺失数据	缺失数据	邮政	缺失数据	缺失数据
1	587598	27	73	15	87	26	1	高新科技	互联网	56.0	4.0	14.0	缺失数据	重庆	互联网	重庆邮电大学	缺失数据
2	587598	72	94	1	112	20	4	缺失数据	缺失数据	1.0	0.0	21.0	缺失数据	缺失数据	缺失数据	缺失数据	缺失数据
3	587598	174	84	8	895	30	7	金融	财务	0.0	0.0	22.0	缺失数据	缺失数据	财务	缺失数据	缺失数据
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6	587598	13	52	3	47	2	6	缺失数据	缺失数据	0.0	0.0	0.0	大王叫我来巡山。	缺失数据	缺失数据	缺失数据	缺失数据
7	587598	105	104	2	55	46	13	高新科技	电子商务	0.0	0.0	0.0	缺失数据	山东	电子商务	缺失数据	缺失数据
8	587598	795	268	39	49	1	69	高新科技	互联网	0.0	0.0	0.0	缺失数据	缺失数据	互联网	缺失数据	缺失数据
9	587598	8	111	3	31	6	3	缺失数据	缺失数据	0.0	0.0	2.0	缺失数据	缺失数据	缺失数据	缺失数据	缺失数据

10 rows × 17 columns

In [6]:

```
# 问题1 知友全国地域分布情况，分析出TOP20
# 要求：
# ① 按照地域统计 知友数量、知友密度（知友数量/城市常住人口），不要求创建函数
# ② 知友数量，知友密度，标准化处理，取值0-100，要求创建函数
# ③ 通过多系列柱状图，做图表可视化
# 提示：
# ① 标准化计算方法 = (X - Xmin) / (Xmax - Xmin)
# ② 可自行设置图表风格
import matplotlib as mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['font.serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False # 解决保存图像是负号'-'显示为方块的问题,或者转换负号为字符串

# 统计计算知友数量，知友密度
    # 按照居住地统计知友数量
    # 城市信息清洗，去掉城市等级文字
    # 合并数据，求知友密度
# 请填写代码***************
df_city = data1_c.groupby('居住地').count()  # 按照居住地统计知友数量
data2['city'] = data2['地区'].str[:-1]   # 城市信息清洗，去掉城市等级文字
q1data = pd.merge(df_city, data2, left_index = True, right_on = 'city', how = 'inner')[['_id','c
q1data['知友密度'] = q1data['_id']/q1data['常住人口']

print('q1data的结果')
q1data.head()


# 创建函数，结果返回标准化取值，新列列名
def data_nor(df, *cols):
# 请填写代码***************
    return(df,colnames)
resultdata = data_nor(q1data,'_id','知友密度')[0]
```

```
resultcolnames = data_nor(q1data,'_id','知友密度')[1]
print('resultdata的结果')
resultdata.head()

# 标准化取值后得到知友数量，知友密度的TOP20数据
# 请填写代码*************
q1data_top20_sl = resultdata.sort_values(resultcolnames[0], ascending=False)[['city',resultcolna
q1data_top20_md = resultdata.sort_values(resultcolnames[1], ascending=False)[['city',resultcolna

print('q1data_top20_sl数量和q1data_top20_md密度的结果')
q1data_top20_sl.head()
q1data_top20_md.head()


fig1 = plt.figure(num=1,figsize=(12,4))
y1 = q1data_top20_sl[resultcolnames[0]]
plt.bar(range(20),
        y1,
        width = 0.8,
        facecolor = 'yellowgreen',
        edgecolor = 'k',
        tick_label = q1data_top20_sl['city'])
plt.title('知友数量TOP20\n')
plt.grid(True, linestyle = "--",color = "gray", linewidth =0.5, axis = 'y')
for i,j in zip(range(20),y1):
    plt.text(i+0.1,2,'%.1f' % j, color = 'k',fontsize = 9)

fig2 = plt.figure(num=2,figsize=(12,4))
y2 = q1data_top20_sl[resultcolnames[0]]
plt.bar(range(20),
        y2,
        width = 0.8,
        facecolor = 'lightskyblue',
        edgecolor = 'k',
        tick_label = q1data_top20_md['city'])
plt.grid(True, linestyle = "--",color = "gray", linewidth = 0.5, axis = 'y')
plt.title('知友密度TOP20\n')
# 添加注释
for i,j in zip(range(20),y2):
    plt.text(i+0.1,2,'%.1f' % j, color = 'k',fontsize = 9)
```

Out[6]:

	_id	关注的收藏夹	关注	关注者	关注的问题	关注的话题	关注的专栏	职业1	职业2	回答	提问	收藏	个人简介	所在行业	教育经历	职业经历	居住地
*******	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
- -	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
----	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Out[6]:

	省	地区	结尾	常住人口	city
0	安徽省	安徽省	省	59500468.0	安徽
1	安徽省	安庆市	市	5311379.0	安庆
2	安徽省	蚌埠市	市	3164467.0	蚌埠
3	安徽省	亳州市	市	4850657.0	亳州
4	安徽省	巢湖市	市	3873102.0	巢湖

Out[6]:

	_id	city	常住人口	知友密度
565	3417	上海	23019196.0	1.484413e-04
358	1	东台	990306.0	1.009789e-06
642	1	东阳	804398.0	1.243166e-06
656	1	临海	1028813.0	9.719939e-07
357	1	丹阳	1660662.0	6.021695e-07

Out[6]:

```
<BarContainer object of 20 artists>
```

Out[6]:

```
Text(0.5, 1.0, '知友数量TOP20\n')
```

Out[6]:

```
Text(0.1, 2, '100.0')
```

Out[6]:

```
Text(1.1, 2, '94.0')
```

Out[6]:

```
Text(2.1, 2, '73.8')
```

Out[6]:

```
Text(3.1, 2, '47.2')
```

Out[6]:

```
Text(4.1, 2, '41.4')
```

Out[6]:

```
Text(5.1, 2, '27.1')
```

Out[6]:

```
Text(6.1, 2, '26.7')
```

Out[6]:

```
Text(7.1, 2, '24.4')
```

Out[6]:

```
Text(8.1, 2, '19.1')
```

Out[6]:

```
Text(9.1, 2, '19.0')
```

Out[6]:

```
Text(10.1, 2, '15.2')
```

Out[6]:

```
Text(11.1, 2, '12.9')
```

Out[6]:

```
Text(12.1, 2, '11.7')
```

Out[6]:

```
Text(13.1, 2, '10.8')
```

Out[6]:

In [6]:

```
# 问题2 不同高校知友关注和被关注情况
# 要求：
# ① 按照学校（教育经历字段） 统计粉丝数（‘关注者’）、关注人数（‘关注’），并筛选出粉丝数TOP20的学校，不要求创
# ② 通过散点图 → 横坐标为关注人数，纵坐标为粉丝数，做图表可视化
# ③ 散点图中，标记出平均关注人数（x参考线），平均粉丝数（y参考线）
# 提示：
# ① 可自行设置图表风格

# 统计计算学校的粉丝数、被关注量（TOP20）
# 请填写代码***************
q2data = data1_c.groupby('教育经历').sum()[['关注','关注者']].drop(['缺失数据','大学','本科']) #去除不
q2data_c = q2data.sort_values('关注',ascending=False)[:20]
print('q2data_c的结果')
q2data_c.head()

# 创建散点图
plt.figure(figsize=(10,6))
x = q2data_c['关注']
y = q2data_c['关注者']
follow_mean = q2data_c['关注'].mean()
fans_mean = q2data_c['关注者'].mean()
plt.scatter(x,y,marker='.',
            s = y/1000,
            cmap = 'Blues',
            c = y,
            alpha = 0.8,
            label = '学校')

# 添加显示内容
plt.axvline(follow_mean,label="平均关注人数：%i人" % follow_mean,color='r',linestyle="--",alpha=0.8
plt.axhline(fans_mean,label="平均粉丝数：%i人" % fans_mean,color='g',linestyle="--",alpha=0.8)   #
plt.legend(loc = 'upper left')
plt.grid()
# 添加注释
for i,j,n in zip(x,y,q2data_c.index):
    plt.text(i+500,j,n, color = 'k')
```

q2data_c的结果

Out[6]:

教育经历	关注者	关注
浙江大学	604144	45057
北京电影学院	223671	5236
北京大学	197571	43343
中央音乐学院北京师范大学	195175	479
吉林大学	159261	28348

Out[6]:
```
<Figure size 720x432 with 0 Axes>
```
Out[6]:
```
<matplotlib.collections.PathCollection at 0x8d14908>
```
Out[6]:
```
<matplotlib.lines.Line2D at 0x8cd58d0>
```
Out[6]:
```
<matplotlib.lines.Line2D at 0x8cd5668>
```
Out[6]:
```
<matplotlib.legend.Legend at 0x8d1acc0>
```
Out[6]:
```
Text(45557,604144,'浙江大学')
```
Out[6]:
```
Text(5736,223671,'北京电影学院')
```
Out[6]:
```
Text(43843,197571,'北京大学')
```
Out[6]:
```
Text(979,195175,'中央音乐学院北京师范大学')
```
Out[6]:
```
Text(28848,159261,'吉林大学')
```
Out[6]:
```
Text(20103,138058,'上海财经大学')
```
Out[6]:
```
Text(1169,137133,'五道口男子职业技术学院')
```
Out[6]:
```
Text(1769,129251,'医学')
```
Out[6]:
```
Text(776,121484,'为往圣继绝学')
```
Out[6]:
```
Text(852,115992,'四川烹饪高专')
```
Out[6]:
```
Text(26990,103087,'哈尔滨工业大学（HIT）')
```
Out[6]:
```
Text(2004,103009,'我的老師，是山川和大地')
```
Out[6]:
```
Text(25140,100392,'四川大学')
```
Out[6]:
```
Text(10326,96153,'中央财经大学')
```
Out[6]:
```
Text(24340,93728,'厦门大学')
```
Out[6]:
```
Text(666,90480,'重庆第一工程尸培养基地')
```
Out[6]:
```
Text(811,85635,'London School of Economics')
```
Out[6]:
```
Text(15840,79103,'中国人民大学')
```
Out[6]:
```
Text(541,72467,'NERV')
```
Out[6]:
```
Text(1282,71052,'UBC中山大学（SYSU）柏林自由大学亚琛工业大学Uni WürzburgUni Bamberg')
```

第十章作业

作业

求作者h指数:https://baike.baidu.com/item/H%E6%8C%87%E6%95%B0/9951340?fr=aladdin

h指数的定义：一名科学家的h指数是指其发表的Np篇论文中有h篇每篇至少被引h次、而其余Np-h篇论文每篇被引均小于或等于h次

在中国知网上查询某个主题，例如关键词=“金融科技”，按照被引用次数排序，得到前100页的结果，已下载到“金融科技相关文章.xlsx”

In [4]:

```
df.head()
```

Out[4]:

	Unnamed: 0	标题	作者	被引数
0	0	我国互联网金融的特殊风险及防范研究	杨群华	533
1	1	区块链技术与应用前瞻综述	何蒲	370
2	2	普惠金融的国际比较研究——基于银行服务的视角	郭田勇	302
3	3	我国科技金融发展指数实证研究	曹颢	257
4	4	金融科技(FinTech)发展与监管:一个监管者的视角	李文红	201

In [3]:

```
# 请求出金融科技领域h指数最高的10位作者，结果如下

#请输入代码
pd.read_excel('金融科技相关文章.xlsx').groupby('作者') .apply(
    lambda x:[i+1 for i,j in enumerate(x['被引数']) if i+1<=j][-1] ) .nlargest(10)
```

Out[3]:

```
作者
陆岷峰    21
杨东      8
叶纯青     7
吴晓光     7
       ..
伍旭川     6
孙国峰     6
巴曙松     6
张锐      6
Length: 10, dtype: int64
```

第十一章作业

作业

做近期强势股票的板块k线，步骤如下：

选取基于中证800（不考虑成分股历史变动情况）股票数据（直接从通达信导出前复权数据）

统计c>半年线ma(c,120)、c>年线ma(c,250)情况下两种情况下，按照N日涨幅排序前m个股票撮合指数（等权均值）的日线K线序列。只选择2011年1月4日之后的数据。

统计以上2种动量策略情况下，持有m天（m=5天，10天）指数的收益统计

In [27]:

```
import os
import pandas as pd
import numpy as np
from datetime import datetime
N=10    #直接指定N日动量
m1=5
m2=10
df=pd.DataFrame(columns=['c','zf','_zf1','_zf2','ma120','ma250'])  #按ohlc顺序
```

```
column_types = {'c': 'float16'}
#遍历当前目录下所有股票xls文件
for parent,dirnames,filenames  in os.walk('H:\\python\\pydata-notebook-master\\tdx-download\\tdx
    for filename in filenames:
        #请输入代码  #其中zf为N天动量的涨幅，_zf1为持股m1天的涨幅，_zf2为持股m2天的涨幅，ma120为c>ma(c,120)
       df1=pd.read_csv(parent+filename,dtype= column_types,encoding = 'gb18030',sep='\t',parse_da
        df1['s']=filename[-10:-4]
        df1['zf']=df1['c']/df1['c'].shift(N)-1    #N日来的涨幅
       df1['_zf1']=df1['c'].shift(-m1)/df1['c']-1  #m1天以后的涨幅
       df1['_zf2']=df1['c'].shift(-m2)/df1['c']-1  #m1天以后的涨幅
       df1['ma120']=df1['c']>df1['c'].rolling(120).mean()  #是否>120天均线
       df1['ma250']=df1['c']>df1['c'].rolling(250).mean()  #是否>250天均线
       df=pd.concat([df,df1])
df['s']=df['s'].astype('category')
df.index.name='date'
df=df[df.index>'2011-01-04'].reset_index()
df.head()

#求N天以来涨幅最大的，且满足ta条件下的m个股票组合的涨幅均值
def topmean1(df,N=10,m=3,column='zf',ta='ma120'):
    #请输入代码   #缺省升序,ta和zf均升序排列，选最后的-m为正确
    return df.sort_values(by=[ta,column])[-m:][['_zf1','_zf2']].mean()    #缺省升序,ta和zf均升序排列,

df1=df.groupby('date').apply(topmean1)
df2=df.groupby('date').apply(topmean1,ta='ma250')     #验证TA=ma250进行过滤的动量选股

df1.mean()                                  #所有天按照以上两种动量选股策略，持有m1,m2情况下的平均收益
df2.mean()
```

Out[27]:

	date	c	zf	_zf1	_zf2	ma120	ma250	s
0	2011-01-05	4.550781	-0.008301	0.091797	0.012695	False	False	600000
1	2011-01-06	4.531250	0.015625	0.079102	-0.022461	False	False	600000
2	2011-01-07	4.828125	0.060547	-0.009766	-0.053223	False	False	600000
3	2011-01-10	4.750000	0.039062	-0.052734	-0.037598	False	False	600000
4	2011-01-11	4.921875	0.108398	-0.077637	-0.057129	False	False	600000

Out[27]:

```
_zf1    0.027344
_zf2    0.043213
dtype: float16
```

Out[27]:

```
_zf1    0.028412
_zf2    0.045319
dtype: float16
```

第十二章作业

作业

题目：读入棒球比赛数据game_logs.csv为gl对象，分析比赛日的分布情况和比赛时长的分布情况。其中，重点在于体验利用category类型大幅压缩内存占用空间

① gl.info(memory_usage='deep')可以显示对象gl占用的内存大小；

② 用describe()函数查看各列数据的情况。

③ 对freq=全部数量/唯一值数量 > 200 的object列，使用category类型；否则不使用category类型

④ 比较转换列部分，转换前后的内存占用情况，如图所示：

⑤ 在read_csv函数中指定dtype=column_types参数，并查看读入对象的内存占用情况

⑥ 用pivot_table分析比赛日的分布情况(一周七天的比例）的逐年变化

⑦ 用pivot_table分析比赛时长的逐年变化

In [44]:

```
gl = pd.read_csv('game_logs.csv',parse_dates=['date'],infer_datetime_format=True)
```

```
gl.head()
gl.info(memory_usage='deep')
# 对应②③请输入代码
cols=gl.describe(include='all',datetime_is_numeric=True).loc['freq']>200
gl_converted=gl.loc[:,~cols].join(gl.loc[:,cols].astype('category'))
gl_converted.info(memory_usage='deep')

# 对应⑤请输入代码
column_types={x:'category' for x in cols[cols].index.values}
gl_opt=pd.read_csv('game_logs.csv',dtype=column_types,parse_dates=['date'],infer_datetime_format
gl_opt.info(memory_usage='deep')

gl_converted['year'] = gl_converted.date.dt.year
# 对应⑥请输入代码
games_per_day = gl_converted.pivot_table(index='year',columns='day_of_week',values='date',aggfun
games_per_day = games_per_day.divide(games_per_day.sum(axis=1),axis=0)

ax = games_per_day.plot(kind='area',stacked='true')
ax.legend(loc='upper right')
ax.set_ylim(0,1)
plt.show()

# 对应⑦请输入代码
game_lengths = gl_converted.pivot_table(index='year', values='length_minutes')
game_lengths.reset_index().plot.scatter('year','length_minutes')
plt.show()
```

```
F:\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (1
2,13,14,15,19,20,81,83,85,87,93,94,95,96,97,98,99,100,105,106,108,109,111,112,114,115,117,118,12
0,121,123,124,126,127,129,130,132,133,135,136,138,139,141,142,144,145,147,148,150,151,153,154,15
6,157,160) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[44]:

	date	numb	day_c	v_nar	v_lea	v_gar	h_nar	h_lea	h_gar	v_sco	...	h_pla	h_pla	h_pla	h_pla	h_pla	h_pla	h_pla	h_pla	addit	acqui
0	1871-05-04	0	Thu	CL1	na	1	FW1	na	1	0	...	Ed Minch	7.0	mcdej	James McDe	8.0	kellb1	Bill Kelly	9.0	NaN	Y
1	1871-05-05	0	Fri	BS1	na	1	WS3	na	1	20	...	Asa Braina	1.0	burrh	Henry Burro	9.0	berth	Henry Berth	8.0	HTBF	Y
2	1871-05-06	0	Sat	CL1	na	2	RC1	na	1	12	...	Pony Sager	6.0	birdg	Georg Bird	7.0	stirg1	Gat Stires	9.0	NaN	Y
3	1871-05-08	0	Mon	CL1	na	3	CH1	na	1	12	...	Ed Duffy	6.0	pinke	Ed Pinkh	5.0	zettg1	Georg Zettle	1.0	NaN	Y
4	1871-05-09	0	Tue	BS1	na	2	TRO	na	1	9	...	Steve Bellan	5.0	pikel1	Lip Pike	3.0	cravb	Bill Crave	6.0	HTBF	Y

5 rows × 161 columns

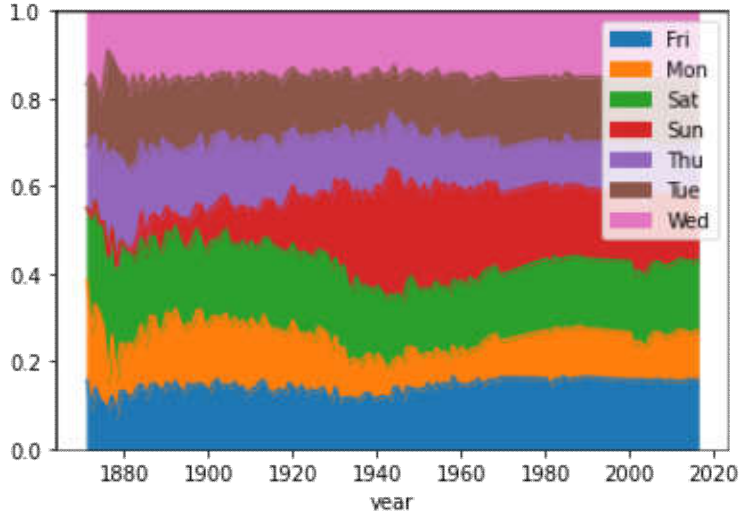
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171907 entries, 0 to 171906
Columns: 161 entries, date to acquisition_info
dtypes: datetime64[ns](1), float64(77), int64(5), object(78)
memory usage: 859.4 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171907 entries, 0 to 171906
Columns: 161 entries, date to acquisition_info
dtypes: category(73), datetime64[ns](1), float64(77), int64(5), object(5)
memory usage: 185.9 MB

F:\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (1
3,14,15,85,87) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171907 entries, 0 to 171906
Columns: 161 entries, date to acquisition_info
dtypes: category(73), datetime64[ns](1), float64(77), int64(5), object(5)
memory usage: 185.9 MB
```

Out[44]:

```
<matplotlib.legend.Legend at 0x5933f160>
```

Out[44]:

```
(0.0, 1.0)
```



第十三章作业

作业

题目：①读入binary.csv文件，其中gpa为在校成绩，gre为分数，rank为本科生母校的声望，admit为入学批准。把rank改名为prestige。查看数据描述性统计，每一列的标准差，prestige与admin的值相应的数量关系,并生成各参数的直方图。

②利用statsmodels.formula,将prestige作为分类变量，利用smf进行预测，并将预测评分存入 predict 列中,设定预测值>0.5时表示预测被录取，计算预测录取数量，实际录取数量和预测命中率；分别用logit和ols进行测试，并比较预测效果。

③利用sklearn，设为虚拟变量。除gre、gpa外，加入了上面常见的虚拟变量（注意，引入的虚拟变量列数应为虚拟变量总列数减1，减去的1列作为基准），利用sklean进行预测，预测值admit为0/1二值，计算计算预测录取数量，实际录取数量和预测命中率。分别用LogisticRegression和LogisticRegressionCv（其中cv=10）进行测试，并比较预测效果。

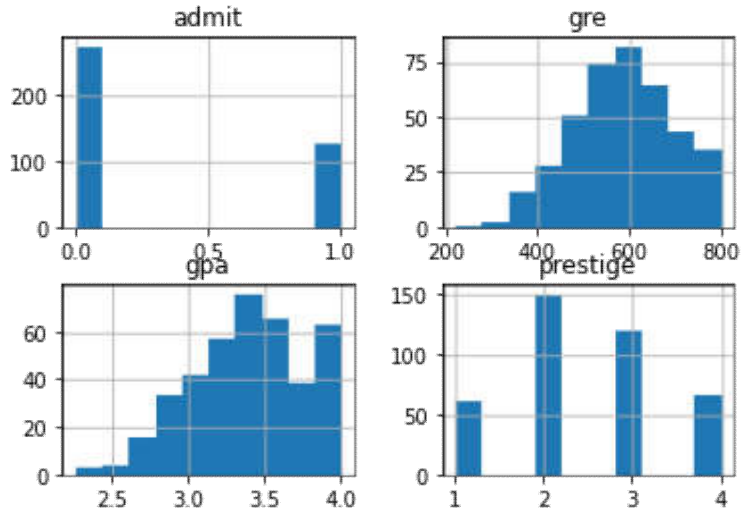
④利用train_test_split把数据随机分为训练集和训练集，其中1/3为测试集，重做logit、ols、LogisticRegression和LogisticRegressionCV，比较其效果

In [1]:

```
# import patsy
import statsmodels.formula.api as smf
import pandas as pd
import numpy as np

# 加载数据
df = pd.read_csv("binary.csv")    #gre分数，rank表示本科生母校的声望
# 重命名'rank'列，因为dataframe中有个方法名也为'rank'
df.columns = ["admit", "gre", "gpa", "prestige"]
# 查看数据描述性统计
df.describe()
# 查看每一列的标准差
df.std()
# 频率表，表示prestige与admin的值相应的数量关系
#请输入代码
pd.crosstab(df['admit'], df['prestige'], rownames=['admit'])
df.hist()
```

Out[1]:

```
array([[<AxesSubplot:title={'center':'admit'}>,
        <AxesSubplot:title={'center':'gre'}>],
       [<AxesSubplot:title={'center':'gpa'}>,
        <AxesSubplot:title={'center':'prestige'}>]], dtype=object)
```



In [4]:

```
#请输入代码
result=smf.logit('admit   ~ gre + gpa + C(prestige)', data=df).fit()
result.summary()
p_admit=result.predict(df)
thresh = df[p_admit>0.5]['admit']    #预测值>0.5的录取率数据
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))
```

```
#请输入代码
result=smf.ols('admit   ~ gre + gpa + C(prestige)', data=df).fit()
# result.summary()
p_admit=result.predict(df)
thresh = df[p_admit>0.5]['admit']    #预测值>0.5的录取率数据
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))
```

```
Optimization terminated successfully.
         Current function value: 0.573147
         Iterations 6
Total: 49, Hit: 30, Precision:61.22%
Total: 47, Hit: 30, Precision:63.83%
```

In [111]:

```
# 引入的虚拟变量列数应为虚拟变量总列数减1，减去的1列作为基准
#请输入代码
data=df.iloc[:,:-1].join(pd.get_dummies(df['prestige'], prefix='prestige').iloc[:,1:])

# 用sklearn实现
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
#请输入代码
model.fit(data.iloc[:,1:].values, df['admit'].values)

y_predict = model.predict(data.iloc[:,1:].values)
thresh=data['admit'][y_predict==1]
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))

# # 用sklearn CV实现
from sklearn.linear_model import LogisticRegressionCV
model_cv = LogisticRegressionCV(Cs=10)
#请输入代码
model_cv.fit(data.iloc[:,1:], data['admit'])
y_predict = model_cv.predict(data.iloc[:,1:])
thresh=data['admit'][y_predict==1]
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))
```

Out[111]:

```
LogisticRegression()

Total: 43, Hit: 26, Precision:60.47%
```

Out[111]:

```
LogisticRegressionCV()

Total: 49, Hit: 30, Precision:61.22%
```

In [110]:

```
# 用sklearn CV实现，区分训练集和测试集
#请输入代码
train_X,test_X,train_y,test_y = train_test_split(data.iloc[:,1:], data['admit'],test_size=1/3,ra

model = LogisticRegression(max_iter=5000)
model.fit(train_X, train_y)
y_predict = model.predict(test_X)
thresh=test_y[y_predict==1]
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))

model_cv = LogisticRegressionCV(Cs=10,max_iter=5000)
model_cv.fit(train_X, train_y)
y_predict = model_cv.predict(test_X)
thresh=test_y[y_predict==1]
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))
```

Out[110]:

```
LogisticRegression(max_iter=5000)

Total: 12, Hit: 6, Precision:50.00%
```

Out[110]:

```
LogisticRegressionCV(max_iter=5000)

Total: 13, Hit: 8, Precision:61.54%
```

In [115]:

```
#请输入代码
train_df,test_df=train_test_split(df,test_size=1/3,random_state=3)
result=smf.logit('admit   ~ gre + gpa + C(prestige)', data=train_df).fit()
# result.summary()
p_admit=result.predict(test_df)
thresh = test_df[p_admit>0.5]['admit']    #预测值>0.5的录取率数据
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))

#请输入代码
result=smf.ols('admit   ~ gre + gpa + C(prestige)', data=train_df).fit()
# result.summary()
p_admit=result.predict(test_df)
thresh = test_df[p_admit>0.5]['admit']    #预测值>0.5的录取率数据
print('Total: %d, Hit: %d, Precision:%.2f%%' % (len(thresh),sum(thresh),thresh.mean()*100))
```

```
Optimization terminated successfully.
         Current function value: 0.577707
         Iterations 6
Total: 17, Hit: 10, Precision:58.82%
Total: 13, Hit: 8, Precision:61.54%
```

补充例子,第十四章

In [2]:

```
#题目：求首字母
li=['abc',None]

# 答案
[x[0] for x in li if type(x)==str]
```

Out[2]:

```
['a']
```

In [ ]:

```
#题目：用One-Line重写相同输出（ch14.1）
by_tz_os = cframe.groupby(['tz', 'os'])
agg_counts = by_tz_os.size().unstack().fillna(0)
agg_counts[:10]

# 答案:
cframe.pivot_table('a',index='tz',columns='os',aggfunc='count',fill_value=0)[:10]
```

In [ ]:

```
#题目：重写相同输出（ch14.2）
mean_ratings = data.pivot_table('rating', index='title',
                                columns='gender', aggfunc='mean')
ratings_by_title = data.groupby('title').size()
active_titles = ratings_by_title.index[ratings_by_title >= 250]
mean_ratings = mean_ratings.loc[active_titles]
mean_ratings  #输出1

rating_std_by_title = data.groupby('title')['rating'].std()
rating_std_by_title = rating_std_by_title.loc[active_titles]
rating_std_by_title.sort_values(ascending=False)[:10]  #输出2

#答案:
Atv=data[data.groupby('title')['movie_id'].transform(len)>=250]
Atv.pivot_table('rating', index='title', columns='gender', aggfunc='mean')   #输出1
Atv.pivot_table('rating', index='title', aggfunc='std')['rating'].nlargest(10) #输出2
```

In [ ]:

```
#题目：用One-Line重写相同输出（ch14.3）
def get_top1000(group):
    return group.sort_values(by='births', ascending=False)[:1000]

grouped = names.groupby(['year', 'sex'])
top1000 = grouped.apply(get_top1000)

# Drop the group index, not needed
top1000.reset_index(inplace=True, drop=True)
top1000 #输出
```

```
# 答案：
names.groupby(['year', 'sex']).apply(lambda x:x.nlargest(1000,'births')).reset_index(drop=True)
```

In [ ]:

```
#题目：用One-Line重写相同输出（ch14.4）//速度提升200倍****
nutrients_all = pd.DataFrame()
for food in db:
    nutrients = pd.DataFrame(food['nutrients'])
    nutrients['id'] = food['id']
    nutrients_all = nutrients_all.append(nutrients, ignore_index=True)
nutrients_all  #输出

#答案
pd.DataFrame([{**y,**{'id':x['id']}}  for  x in db for y in x['nutrients']])
```

In [ ]:

```
#题目：用One-Line重写相同输出（ch14.4）
by_nutrient = ndata.groupby(['nutgroup', 'nutrient'])
get_maximum = lambda x: x.loc[x.value.idxmax()]
max_foods = by_nutrient.apply(get_maximum)[['value', 'food']]
max_foods #输出

#答案
ndata.groupby(['nutgroup', 'nutrient']).apply(lambda x:x.nlargest(1,'value'))[['value', 'food']]
```

In [ ]:

```
#题目：用One-Line重写相同输出（ch14.5）
grouped = fec_mrbo.groupby(['cand_nm', 'contbr_st'])
totals = grouped.contb_receipt_amt.sum().unstack(0).fillna(0)
totals  #输出

# 答案
fec_mrbo.pivot_table('contb_receipt_amt',index='contbr_st',columns='cand_nm',aggfunc='sum')
```

In [ ]:

```
#题目：重写相同输出（ch14.2）
grouped = fec_mrbo.groupby(['cand_nm', 'contbr_st'])
totals = grouped.contb_receipt_amt.sum().unstack(0).fillna(0)
totals = totals[totals.sum(1) > 100000]
totals  #输出

#答案
Atv=fec_mrbo[fec_mrbo.groupby('contbr_st')['contb_receipt_amt'].transform(sum)>100000]
Atv.pivot_table('contb_receipt_amt',index='contbr_st',columns='cand_nm',aggfunc='sum')
```