**实验报告**

**学号：117060400129** **姓名**： 李娟 **班级：** 应用统计一班  **指导老师：** 林卫中

**实验名称**： 课后习题练习

**实验要求：熟练应用所学编程**

**实验题目：朋友圈爬虫**

**算法实现：**

def get\_var(var,friends):

variable=[]

for i in friends:

value=i[var]

variable.append(value)

return variable

def analyseGender(friends):

male=female=other=0

sexes = get\_var('Sex', friends)

for sex in sexes:

if sex == 1:

male += 1

elif sex == 2:

female += 1

else:

other += 1

total = len(friends[1:])

malecol = round( float(male)/total \* 100, 2)

femalecol = round( float(female)/total \* 100, 2)

othercol = round( float(other)/total \* 100, 2)

print('男性好友：{:.2f}%%'.format( malecol))

print('女性好友：{:.2f}%%'.format( femalecol))

print('不明性别好友：{:.2f}%%'.format( othercol))

mpl.rcParams['font.sans-serif']=['SimHei']

mpl.rcParams['axes.unicode\_minus'] = False

map = {

'Female':(malecol, '#7199cf'),

'Male': (femalecol, '#4fc4aa'),

'other': (othercol, '#e1a7a2')

}

fig = plt.figure( figsize=(5,5))

ax = fig.add\_subplot(111)

ax.set\_title( '朋友圈性别')

xticks = np.arange(3) + 0.15

bar\_width = 0.5

names = map.keys()

values = [ x[0] for x in map.values()]

colors = [ x[1] for x in map.values()]

bars = ax.bar( xticks, values, width=bar\_width, edgecolor='none')

ax.set\_ylabel('比例')

ax.set\_xlabel('性别')

ax.grid()

ax.set\_xticks( xticks)

ax.set\_xticklabels( names)

ax.set\_xlim( [bar\_width/2 - 0.5, 3 - bar\_width/2])

ax.set\_ylim( [0, 100])

for bar, color in zip( bars, colors):

bar.set\_color( color)

height = bar.get\_height()

plt.text( bar.get\_x(), bar.get\_height()/4.+ height, '{:.2f}%'.format( float(height)))

plt.show()

fig1 = plt.figure( figsize=(5,5))

ax = fig1.add\_subplot(111)

ax.set\_title('饼图')

labels = ['{}\n{}%'.format(name, value) for name, value in zip( names, values)]

ax.pie(values, labels=labels, colors=colors)

plt.show()

import itchat

import numpy as np

import matplotlib.pyplot as plt

import matplotlib as mpl

import re

import jieba

import PIL.Image as Image

def login():

itchat.login()

friends=itchat.get\_friends(update=True)[0:]

return friends

def get\_var(var, friends):

variable = []

for i in friends:

value = i[var]

variable.append(value)

return variable

def analyseGender(friends):

male=female=other=0

sexes = get\_var('Sex', friends)

for sex in sexes:

if sex == 1:

male += 1

elif sex == 2:

female += 1

else:

other += 1

total = len(friends[1:])

malecol = round( float(male)/total \* 100, 2)

femalecol = round( float(female)/total \* 100, 2)

othercol = round( float(other)/total \* 100, 2)

print('男性好友：{:.2f}%%'.format( malecol))

print('女性好友：{:.2f}%%'.format( femalecol))

print('不明性别好友：{:.2f}%%'.format( othercol))

mpl.rcParams['font.sans-serif']=['SimHei']

mpl.rcParams['axes.unicode\_minus'] = False

map = {

'Female':(malecol, '#7199cf'),

'Male': (femalecol, '#4fc4aa'),

'other': (othercol, '#e1a7a2')

}

fig = plt.figure( figsize=(5,5))

ax = fig.add\_subplot(111)

ax.set\_title( '朋友圈性别')

xticks = np.arange(3) + 0.15

bar\_width = 0.5

names = map.keys()

values = [ x[0] for x in map.values()]

colors = [ x[1] for x in map.values()]

bars = ax.bar( xticks, values, width=bar\_width, edgecolor='none')

ax.set\_ylabel('比例')

ax.set\_xlabel('性别')

ax.grid()

ax.set\_xticks( xticks)

ax.set\_xticklabels( names)

ax.set\_xlim( [bar\_width/2 - 0.5, 3 - bar\_width/2])

ax.set\_ylim( [0, 100])

for bar, color in zip( bars, colors):

bar.set\_color( color)

height = bar.get\_height()

plt.text( bar.get\_x(), bar.get\_height()/4.+ height, '{:.2f}%'.format( float(height)))

plt.show()

fig1 = plt.figure( figsize=(5,5))

ax = fig1.add\_subplot(111)

ax.set\_title('饼图')

labels = ['{}\n{}%'.format(name, value) for name, value in zip( names, values)]

ax.pie(values, labels=labels, colors=colors)

plt.show()

def analyseProvince(friends):

provlist = get\_var('Province', friends)

provdict = {}

for p in provlist:

provdict[p] = provdict.get(p,0) + 1

provdict = sorted(provdict.items(), key= lambda x : x[1], reverse=True)

figpro = plt.figure(figsize=(10,5))

axpro = figpro.add\_subplot(111)

axpro.set\_title('省份')

xticks = np.linspace(0.5,20,10)

bar\_width = 0.8

pros= []

values = []

count = 0

for d in provdict:

pros.append(d[0])

values.append(d[1])

count += 1

if count >= 10:

break

colors = ['#FFEC88', '#FFE4C4','#FFC125','#FFB6C1','#CDCDB4','#CDC8B1','#CDB79E','#CDAD00','#CD96CD',\

'#CD853F']

bars = axpro.bar( xticks, values, width=bar\_width, edgecolor='none')

axpro.set\_ylabel('人数')

axpro.set\_xlabel('省份')

axpro.grid()

axpro.set\_xticks( xticks)

axpro.set\_xticklabels(pros)

axpro.set\_xlim(0,20)

axpro.set\_ylim([0,100])

for bar, color in zip( bars, colors):

bar.set\_color(color)

height = bar.get\_height()

plt.text( bar.get\_x()+bar.get\_width()/4., height, '{}'.format(height))

plt.show()

friends = login()

analyseGender(friends)

analyseProvince(friends)

import itchat

import numpy as np

import matplotlib.pyplot as plt

import matplotlib as mpl

import re

import jieba

import PIL.Image as Image

from wordcloud import WordCloud, ImageColorGenerator

def login():

itchat.login()

friends=itchat.get\_friends(update=True)[0:]

return friends

def get\_var(var, friends):

variable = []

for i in friends:

value = i[var]

variable.append(value)

return variable

def analyseGender(friends):

male=female=other=0

sexes = get\_var('Sex', friends)

for sex in sexes:

if sex == 1:

male += 1

elif sex == 2:

female += 1

else:

other += 1

total = len(friends[1:])

malecol = round( float(male)/total \* 100, 2)

femalecol = round( float(female)/total \* 100, 2)

othercol = round( float(other)/total \* 100, 2)

print('男性好友：{:.2f}%%'.format( malecol))

print('女性好友：{:.2f}%%'.format( femalecol))

print('不明性别好友：{:.2f}%%'.format( othercol))

mpl.rcParams['font.sans-serif']=['SimHei']

mpl.rcParams['axes.unicode\_minus'] = False

map = {

'Female':(malecol, '#7199cf'),

'Male': (femalecol, '#4fc4aa'),

'other': (othercol, '#e1a7a2')

}

fig = plt.figure( figsize=(5,5))

ax = fig.add\_subplot(111)

ax.set\_title( '朋友圈性别')

xticks = np.arange(3) + 0.15

bar\_width = 0.5

names = map.keys()

values = [ x[0] for x in map.values()]

colors = [ x[1] for x in map.values()]

bars = ax.bar( xticks, values, width=bar\_width, edgecolor='none')

ax.set\_ylabel('比例')

ax.set\_xlabel('性别')

ax.grid()

ax.set\_xticks( xticks)

ax.set\_xticklabels( names)

ax.set\_xlim( [bar\_width/2 - 0.5, 3 - bar\_width/2])

ax.set\_ylim( [0, 100])

for bar, color in zip( bars, colors):

bar.set\_color( color)

height = bar.get\_height()

plt.text( bar.get\_x(), bar.get\_height()/4.+ height, '{:.2f}%'.format( float(height)))

plt.show()

fig1 = plt.figure( figsize=(5,5))

ax = fig1.add\_subplot(111)

ax.set\_title('饼图')

labels = ['{}\n{}%'.format(name, value) for name, value in zip( names, values)]

ax.pie(values, labels=labels, colors=colors)

plt.show()

def analyseProvince(friends):

provlist = get\_var('Province', friends)

provdict = {}

for p in provlist:

provdict[p] = provdict.get(p,0) + 1

provdict = sorted(provdict.items(), key= lambda x : x[1], reverse=True)

figpro = plt.figure(figsize=(10,5))

axpro = figpro.add\_subplot(111)

axpro.set\_title('省份')

xticks = np.linspace(0.5,20,10)

bar\_width = 0.8

pros= []

values = []

count = 0

for d in provdict:

pros.append(d[0])

values.append(d[1])

count += 1

if count >= 10:

break

colors = ['#FFEC88', '#FFE4C4','#FFC125','#FFB6C1','#CDCDB4','#CDC8B1','#CDB79E','#CDAD00','#CD96CD',\

'#CD853F']

bars = axpro.bar( xticks, values, width=bar\_width, edgecolor='none')

axpro.set\_ylabel('人数')

axpro.set\_xlabel('省份')

axpro.grid()

axpro.set\_xticks( xticks)

axpro.set\_xticklabels(pros)

axpro.set\_xlim(0,20)

axpro.set\_ylim([0,100])

for bar, color in zip( bars, colors):

bar.set\_color(color)

height = bar.get\_height()

plt.text( bar.get\_x()+bar.get\_width()/4., height, '{}'.format(height))

plt.show()

def drawWordcloudPlot(counts):

coloring = np.array(Image.open("E:/baidupic/alice\_color.png"))

wc = WordCloud(background\_color="white",

max\_words=2000,

mask=coloring,

max\_font\_size=60,

random\_state=42,

scale=2,

font\_path="c:/Windows/Fonts/SimHei.ttf")

wc.generate\_from\_frequencies(counts)

image\_colors = ImageColorGenerator(coloring)

plt.imshow(wc)

plt.axis("off")

plt.savefig('friendSign.jpg')

plt.show()

def analyseSignature(friends):

signatures = get\_var('Signature', friends)

siglist = []

for sign in signatures:

sign = sign.strip().replace("span", "").replace("class", "").replace("emoji", "")

rep = re.compile("lf\d+\w\*|[<>/=]")

sign = rep.sub("", sign)

siglist.append(sign)

text = "".join(siglist)

wlist = jieba.cut(text, cut\_all=True)

counts = {}

for word in wlist:

if len(word) == 1:

continue

else:

counts[word] = counts.get(word, 0) + 1

wdict = {}

for d in counts.items():

if d[1] > 2:

wdict[d[0]] = d[1]

drawWordcloudPlot(wdict)

def main():

friends = login()

analyseGender(friends)

analyseProvince(friends)

analyseSignature(friends)

main()

from random import random

def printIntro():

print("这个程序模拟两个选手A和B的某种竞技比赛")

print("程序运行需要A和B的能力值（以0到1之间的小数表示）")

def getInputs():

a = eval(input("请输入选手A的能力值(0-1): "))

b = eval(input("请输入选手B的能力值(0-1): "))

n = eval(input("模拟比赛的场次: "))

return a, b, n

def simNGames(n, probA, probB):

winsA, winsB = 0, 0

for i in range(n):

scoreA, scoreB = simOneGame(probA, probB)

if scoreA > scoreB:

winsA += 1

else:

winsB += 1

return winsA, winsB

def gameOver(a,b):

return a==21 or b==11

def simOneGame(probA, probB):

scoreA, scoreB = 0, 0

serving = 0

t=0

while not gameOver(scoreA, scoreB):

if serving == 0:

if random() < probA:

scoreA += 1

else:

scoreB +=1

else:

if random() < probB:

scoreB += 1

else:

scoreA += 1

t+=1

if t%5==0:

serving=(serving+1)%2

return scoreA, scoreB

def printSummary(winsA, winsB):

n = winsA + winsB

print("竞技分析开始，共模拟{}场比赛".format(n))

print("选手A获胜{}场比赛，占比{:0.1%}".format(winsA, winsA/n))

print("选手B获胜{}场比赛，占比{:0.1%}".format(winsB, winsB/n))

def main():

printIntro()

probA, probB, n = getInputs()

winsA, winsB = simNGames(n, probA, probB)

printSummary(winsA, winsB)

main()

from random import random

def printIntro():

print("这个程序模拟两个选手A和B的某种竞技比赛")

print("程序运行需要A和B的能力值（以0到1之间的小数表示）")

def getInputs():

a = eval(input("请输入选手A的能力值(0-1): "))

b = eval(input("请输入选手B的能力值(0-1): "))

n = eval(input("模拟比赛的场次: "))

return a, b, n

def simNGames(n, probA, probB):

winsA, winsB = 0, 0

for i in range(n):

scoreA, scoreB = simOneGame(probA, probB)

if scoreA > scoreB:

winsA += 1

else:

winsB += 1

return winsA, winsB

def gameOver(a,b):

""" if (a<10 and b==11) or (a==11 and b<10):

return True

if(a>=10 and b-a==2) or(b>=10 and a-b==2):

return True

return False"""

if a<b:

a,b=b,a

if a==11 and b<10:

return True

if b>=10 and a-b==2:

return True

return False

def simOneGame(probA, probB):

scoreA, scoreB = 0, 0

serving = 0

t=0

while not gameOver(scoreA, scoreB):

if serving == 0:

if random() < probA:

scoreA += 1

else:

scoreB +=1

else:

if random() < probB:

scoreB += 1

else:

scoreA += 1

t+=1

if t%2==0:

serving=(serving+1)%2

return scoreA, scoreB

def printSummary(winsA, winsB):

n = winsA + winsB

print("竞技分析开始，共模拟{}场比赛".format(n))

print("选手A获胜{}场比赛，占比{:0.1%}".format(winsA, winsA/n))

print("选手B获胜{}场比赛，占比{:0.1%}".format(winsB, winsB/n))

def main():

printIntro()

probA, probB, n = getInputs()

winsA, winsB = simNGames(n, probA, probB)

printSummary(winsA, winsB)

main()

from random import random

from random import randint

def printIntro():

print("这个程序模拟两个选手A和B的某种竞技比赛")

print("程序运行需要A和B的能力值（以0到1之间的小数表示）")

def getInputs():

a = eval(input("请输入选手A的能力值(0-1): "))

b = eval(input("请输入选手B的能力值(0-1): "))

n = eval(input("模拟比赛的场次: "))

g1=eval(input("A投篮能力:"))

g2=eval(input("B投篮能力:"))

return a, b, n,g1,g2

def simNGames(n, goleA, boardA,goleB,boardB):

winsA, winsB = 0, 0

for i in range(n):

scoreA, scoreB = simOneGame(goleA, boardA,goleB,boardB)

if scoreA > scoreB:

winsA += 1

else:

winsB += 1

return winsA, winsB

def gameOver(t):

return t>=12\*60

def simOneGame(goleA, boardA,goleB,boardB):

scoreA, scoreB = 0, 0

serving = 0

totalTime=0

while not gameOver(totalTime):

t=randint(1,24)

totalTime += t

if t == 24:

serving=(serving + 1)%2

else:

if serving == 0:

if random() < goleA:

scoreA += 1

serving=1

else:

if random()<boardA:

serving=0

else:

serving=1

else:

if random() < goleB:

scoreB += 1

serving=0

else:

if random()<boardB:

serving=1

else:

serving=0

return scoreA, scoreB

def printSummary(winsA, winsB):

n = winsA + winsB

print("竞技分析开始，共模拟{}场比赛".format(n))

print("选手A获胜{}场比赛，占比{:0.1%}".format(winsA, winsA/n))

print("选手B获胜{}场比赛，占比{:0.1%}".format(winsB, winsB/n))

def main():

printIntro()

goleA, boardA,goleB,boardB, n = getInputs()

winsA, winsB = simNGames(n,goleA, boardA,goleB,boardB)

printSummary(winsA, winsB)

main()

import jiebaimport matplotlib.pyplot as pltfrom wordcloud import WordCloud, ImageColorGeneratorimport numpy as npimport PIL.Image as Imagedef calWordFrequence(): excludes = {} # {"将军","却说","丞相"} txt = open("三国演义.txt", "r", encoding='utf-8').read() words = jieba.lcut(txt) counts = {} for word in words: if len(word) == 1: continue else: counts[word] = counts.get(word, 0) + 1 for word in excludes: del (counts[word]) return countsdef drawWordCloud(counts): coloring = np.array(Image.open("E:/baidupic/9.png")) wc = WordCloud(background\_color="white", max\_words=2000, mask=coloring, max\_font\_size=60, random\_state=42, scale=2, font\_path="c:/Windows/Fonts/SimHei.ttf") wc.generate\_from\_frequencies(counts) image\_colors = ImageColorGenerator(coloring) plt.imshow(wc) plt.axis("off") plt.show()def main(): counts = calWordFrequence() drawWordCloud(counts)main()

**实验结果：** 经过这一段时间的学习，虽然我对于程序的编程还是很吃力，难的编程我不会，但是简单的我勉强会做一点，这也是一种收获，我会继续努力的。python课看似结束了，其实这才刚刚开始，还有许多未知等着我们自己去学习。