SY-023

实 验 报 告

课程名称： Python程序设计 \_

系部名称： 计算机科学与技术 \_

专业班级： 计科22-3班 \_\_

学生姓名： 邹晨骏 \_\_

学 号： 20222163 \_

指导教师： 曲天伟 \_\_ \_

**黑龙江工程学院教务处制**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验项目 | 实验八 数据可视化综合实验 | | | | | 实验日期 | 2024.11.11 |
| 实验地点 | A607 | 同组人数 | 1 | 分工 |  | 实验台号 |  |
| 实验类型 | □验证性 □ 综合性 □ 设计性 □ 其 他 | | | | | | |
| 一、实验目的  掌握Python生态库的构建与发布方法，能综合运用本课程所学的知识和第三方库解决文本数据统计和可视化问题。 | | | | | | | |
| 二、实验仪器设备实验工具软件  笔记本电脑，python3.8.2解释器，pycharm2021编辑器 | | | | | | | |
| 三、实验要求  1、基础练习（建立并发布自己的python库）  Python模块本质上是一个包含Python代码片段的.py文件，模块名就是文件名。  1）练习自己建立一个py文件，可以作为一个模块被其他文件调用。  在文件夹下建立jisuan.py 具体内容如下：  def add(a,b):  return a+b *#测试代码 #判断模块的内置变量name的值来解决模块调用时不在打印测试代码的问题。 #模块内置变量name在脚本文件或在命令行里中独立执行时为\_\_name\_\_,被其他文件以模块的形式引用时值为模块文件名。 #可以通过这个结合if语句，使模块中的测试代码只能在模块中执行，不能在其他导入该模块的文件中执行。达到避免测试代码被执行的效果。* if \_\_name\_\_=='\_\_main\_\_':  result=add(12,23)  print(result)  再建立一个test.py 具体内容如下：  import jisuan *# print(jisuan.\_\_name\_\_)* result=jisuan.add(11,22) print(result)  2）练习Python生态库的构建与发布方法。  在工程文件根目录下创建一个文件夹取名package，创建一个要做为模块的文件如：module.py。并在该文件夹中创建\_\_init\_\_.py文件，就构建了一个Python包。  1586250134(1)  按照上图，在package文件夹下创建python package ，起名package\_a ,这样package\_a自动产生了一个\_\_inti\_\_.py文件，在里面再创建个moudle\_a.py。按照同样的方法创建package\_b, 在里面再创建个moudle\_b.py。  在moudle\_a.py简单写个函数。  def say\_hello():  print("hello world")  在moudle\_b.py 写入调用moudle\_a.py模块，然后调用该模块的say\_hello()函数。  import package.package\_a.module\_a module\_a.say\_hello()  Python中的第三方库是由Python使用者自行编写与发布的模块或包，同样的，我们也可以将自己编写的模块与包作为库发布。具体步骤如下：   1. 在与待发布的包同级的目录中创建setup.py文件。 2. 编辑setup.py文件，在该文件中设置包中包含的模块。   from distutils.core import setup setup(  name="package",*#包的名称* version='1.0',*#包的版本号* description='我的python包',*#包的描述信息* author='qtw',*#包的作者* py\_modules=['package.package\_a.module\_a','package.package\_b.module\_b'] )   1. 如果在pycharm中找不到Terminal窗口可以进行如下设置（setting）。      1. 在setup.py文件所在目录下打开命令行，使用python setup.py build命令构建Python库。      1. 在setup.py文件所在目录下打开命令行，使用python setup.py sdist命令创建库的安装包。     这样就生成了一个dist文件夹，里面是一个压缩包package-1.0.gz，我们可以把这个安装包分享出去，其他人获取这个安装包后，解压后，可以使用 python setup.py install进行安装使用。    在桌面建立一个新文件夹，把压缩包拷贝到新文件夹下，解压后可以看到里面有setup.py  在这里按住shift键+鼠标右键，选择在此处打开命令窗口。    在命令窗口中执行python setup.py install       1. **解决问题**   读取存储《哈姆雷特》英文剧本的文件hamlet.txt(已经给出)，分析统计其中单词出现的频率，找出其中使用频率最高的10个词，使用turtle模块绘制词频统计结果，以柱状图的形式进行可视化展示统计结果。    参考代码如下：  import turtle '''全局变量''' amount = 10 *#词频排列显示个数* words = [] *#单词集合-x轴数据* wCounts = [] *#单词频数集合-y轴数据* xPoint = -360 yPoint = -200 '''turtle start''' *#绘制从点(x1,y1)到(x2,y2)的线段* def drawLine(t,x1,y1,x2,y2):  t.penup()  t.goto(x1,y1)  t.pendown()  t.goto(x2,y2) *#在坐标(x,y)处写文字* def drawText(t,x,y,text,fontSize=10):  t.penup()  t.goto(x,y)  t.pendown()  t.write(text,font=('微软雅黑',fontSize,),align='center') *#绘制矩形* def drawRectangle(t,x,y,rWidth):  drawLine(t,x-rWidth,yPoint,x-rWidth,y)  drawLine(t,x-rWidth,y,x+rWidth,y)  drawLine(t,x+rWidth,y,x+rWidth,yPoint)  drawLine(t,x+rWidth,yPoint,x-rWidth,yPoint) *#绘制柱状图* def drawBarchart(t):  drawText(t,0,-yPoint-40,"词频统计结果",15) *#绘制标题* drawRectangle(t,0,-yPoint,-xPoint) *#绘制边框* rWidth = -xPoint/(2\*amount) *#控制条形柱宽度（2rWidth）* xScale = -xPoint\*2/(amount+1) *#x轴显示放大倍数-可根据amount进行调节* yScale = -yPoint/wCounts[0] *#y轴显示放大倍数-可根据频数进行调节* for i in range(amount):  i=i+1 *#右移以避免与原点重合* x=i\*xScale+xPoint  y=wCounts[i-1]\*yScale+yPoint  drawText(t,x,yPoint-20,words[i-1]) *#打印单词* drawText(t,x,y+10,wCounts[i-1]) *#打印频率* t.begin\_fill()  drawRectangle(t,x,y,rWidth)  t.end\_fill() *#初始化窗口* def init():  turtle.title('词频结果柱状图')  turtle.screensize(900,750,"#272727")  t=turtle.Turtle()  t.hideturtle() *#隐藏箭头* t.width(1) *#线条粗细* t.color("#EBEBD0","#006030")  *#t.color("#006030","#FFF68C")* drawBarchart(t) *#绘制* turtle.exitonclick() '''data Processing''' *#对文本的每一行计算词频的函数* def processLine(line,wordamounts):  line = replacePunctuations(line) *#用空格替换标点符号* words = line.split() *#从每一行获取每个词* for word in words:  if word in wordamounts:  wordamounts[word] += 1  else:  wordamounts[word] = 1 *#空格替换标点* def replacePunctuations(line):  for ch in line:  if ch in "~!@#$%^&\*()-\_+=<>?/,.:;{}[]|\'\"":  line = line.replace(ch,' ')  return line *#数据处理，得到文件中出现频率最高的单词及其频数* def dataProcess(filename):  infile=open(filename,'r',encoding='UTF-8')  wordamounts={} *#建立用于记录词频的空字典* for line in infile:  processLine(line.lower(),wordamounts)  pairs = list(wordamounts.items()) *#从字典中获取数据对* items = [[x,y]for (y,x) in pairs] *#列表中的数据对交换位置，使频数成为元素的键值* items.sort() *#列表排序。默认根据键值排序  #输出amount个词频结果* for i in range(len(items)-1,len(items)-amount-1,-1):  print(items[i][1]+"\t"+str(items[i][0]))  wCounts.append(items[i][0])  words.append(items[i][1])  infile.close() def main():  *#用户输入一个文件名* filename= input("enter a filename:").strip()  dataProcess(filename)  init() *#调用main()函数* if \_\_name\_\_ == '\_\_main\_\_':  main()  思考题：  1.请仔细阅读代码，读懂参考代码，在阅读代码中遇到不会的函数请查阅资料，并把函数的解释，写入注释中。  2.读懂代码后请自行编写复现。  3.如果统计高频词的数量不是10个而是用户自己输入的个数，应该如何修改程序。  4.如果给定的文件是汉字文件，测试程序是否有效。请描述你遇到的问题，结合遇到的问题分析，为什么英文文本可以统计高频词出现的次数，而中文文本不能？请查阅资料给出要实现中文文本高频词统计，并可视化，要对中文文本进行怎样的处理。 | | | | | | | |
| 四、实验步骤  1、基础练习 1）练习自己建立一个py文件，可以作为一个模块被其他文件调用。  import hash # print(hash.\_\_name\_\_) result = hash.add(11,22) print(result)  2、解决问题  import turtle '''全局变量''' amount = 10 # 显示的词频个数 words = [] # 单词集合，作为x轴数据 wCounts = [] # 单词频数集合，作为y轴数据 xPoint = -360 # x轴起点 yPoint = -200 # y轴起点  '''turtle 绘图部分'''  # 绘制从点 (x1, y1) 到 (x2, y2) 的线段 def drawLine(t, x1, y1, x2, y2):  t.penup() # 提起画笔，不进行绘图，移动到起点  t.goto(x1, y1) # 移动到起点  t.pendown() # 放下画笔，准备绘图  t.goto(x2, y2) # 绘制到终点  # 在坐标 (x, y) 位置显示文字 def drawText(t, x, y, text, fontSize=10):  t.penup() # 提起画笔，不绘图  t.goto(x, y) # 移动到文字显示位置  t.pendown() # 放下画笔，准备写文字  t.write(text, font=('微软雅黑', fontSize), align='center') # 绘制文字  # 绘制矩形，以 (x, y) 为中心，宽度为 2\*rWidth def drawRectangle(t, x, y, rWidth):  drawLine(t, x - rWidth, yPoint, x - rWidth, y) # 左边线  drawLine(t, x - rWidth, y, x + rWidth, y) # 顶边线  drawLine(t, x + rWidth, y, x + rWidth, yPoint) # 右边线  drawLine(t, x + rWidth, yPoint, x - rWidth, yPoint) # 底边线  # 绘制柱状图 def drawBarchart(t):  drawText(t, 0, -yPoint - 40, "词频统计结果", 15) # 绘制标题  drawRectangle(t, 0, -yPoint, -xPoint) # 绘制边框  rWidth = -xPoint / (2 \* amount) # 控制每个条形柱的宽度  xScale = -xPoint \* 2 / (amount + 1) # x轴的缩放倍数，根据amount调节  yScale = -yPoint / wCounts[0] # y轴的缩放倍数，根据频数调节  for i in range(amount):  i = i + 1 # 右移以避免与原点重合  x = i \* xScale + xPoint  y = wCounts[i - 1] \* yScale + yPoint  drawText(t, x, yPoint - 20, words[i - 1]) # 打印单词  drawText(t, x, y + 10, wCounts[i - 1]) # 打印频率  t.begin\_fill()  drawRectangle(t, x, y, rWidth)  t.end\_fill()  # 初始化窗口 def init():  turtle.title('词频结果柱状图')  turtle.screensize(900, 750, "#272727") # 设置屏幕大小和背景颜色  t = turtle.Turtle()  t.hideturtle() # 隐藏画笔箭头  t.width(1) # 设置线条粗细  t.color("#EBEBD0", "#006030") # 设置画笔颜色和填充颜色  drawBarchart(t) # 开始绘制  turtle.exitonclick() # 点击关闭窗口  '''数据处理部分'''  # 处理每行文本以计算词频 def processLine(line, wordamounts):  line = replacePunctuations(line) # 用空格替换标点符号  words = line.split() # 分割每行文本，获取单词  for word in words:  if word in wordamounts:  wordamounts[word] += 1 # 如果单词已存在，则词频加一  else:  wordamounts[word] = 1 # 如果单词不存在，设置词频为1  # 替换标点符号为空格 def replacePunctuations(line):  for ch in line:  if ch in "~!@#$%^&\*()-\_+=<>?/,.:;{}[]|\'\"": # 列举要替换的标点  line = line.replace(ch, ' ')  return line  # 处理数据，得到出现频率最高的单词及其频数 def dataProcess(filename):  infile = open(filename, 'r', encoding='UTF-8') # 打开文件并指定编码  wordamounts = {} # 初始化一个空字典记录词频  for line in infile:  processLine(line.lower(), wordamounts) # 转小写并处理每行文本  pairs = list(wordamounts.items()) # 将词频字典转换为列表  items = [[x, y] for (y, x) in pairs] # 调换列表内键值顺序，方便按频数排序  items.sort() # 按频数排序  # 输出前amount个词频结果  for i in range(len(items) - 1, len(items) - amount - 1, -1):  print(items[i][1] + "\t" + str(items[i][0]))  wCounts.append(items[i][0])  words.append(items[i][1])  infile.close()  # 主函数，程序入口 def main():  filename = input("请输入文件名:").strip() # 输入文件名  dataProcess(filename) # 处理数据  init() # 初始化绘图  # 调用主函数 if \_\_name\_\_ == '\_\_main\_\_':  main()  思考题：  **3.如果统计高频词的数量不是10个而是用户自己输入的个数，应该如何修改程序。**  只需要修改全局变量amount，将其改为input 输入即可。  import turtle '''全局变量''' words = [] # 单词集合，作为x轴数据 wCounts = [] # 单词频数集合，作为y轴数据 xPoint = -360 # x轴起点 yPoint = -200 # y轴起点 amount = 10 # 默认显示的词频个数  '''turtle 绘图部分'''  # 绘制从点 (x1, y1) 到 (x2, y2) 的线段 def drawLine(t, x1, y1, x2, y2):  t.penup() # 提起画笔，不进行绘图，移动到起点  t.goto(x1, y1) # 移动到起点  t.pendown() # 放下画笔，准备绘图  t.goto(x2, y2) # 绘制到终点  # 在坐标 (x, y) 位置显示文字 def drawText(t, x, y, text, fontSize=10):  t.penup() # 提起画笔，不绘图  t.goto(x, y) # 移动到文字显示位置  t.pendown() # 放下画笔，准备写文字  t.write(text, font=('微软雅黑', fontSize), align='center') # 绘制文字  # 绘制矩形，以 (x, y) 为中心，宽度为 2\*rWidth def drawRectangle(t, x, y, rWidth):  drawLine(t, x - rWidth, yPoint, x - rWidth, y) # 左边线  drawLine(t, x - rWidth, y, x + rWidth, y) # 顶边线  drawLine(t, x + rWidth, y, x + rWidth, yPoint) # 右边线  drawLine(t, x + rWidth, yPoint, x - rWidth, yPoint) # 底边线  # 绘制柱状图 def drawBarchart(t):  drawText(t, 0, -yPoint - 40, "词频统计结果", 15) # 绘制标题  drawRectangle(t, 0, -yPoint, -xPoint) # 绘制边框  rWidth = -xPoint / (2 \* amount) # 控制每个条形柱的宽度  xScale = -xPoint \* 2 / (amount + 1) # x轴的缩放倍数，根据amount调节  yScale = -yPoint / wCounts[0] # y轴的缩放倍数，根据频数调节  for i in range(amount):  i = i + 1 # 右移以避免与原点重合  x = i \* xScale + xPoint  y = wCounts[i - 1] \* yScale + yPoint  drawText(t, x, yPoint - 20, words[i - 1]) # 打印单词  drawText(t, x, y + 10, wCounts[i - 1]) # 打印频率  t.begin\_fill()  drawRectangle(t, x, y, rWidth)  t.end\_fill()  # 初始化窗口 def init():  turtle.title('词频结果柱状图')  turtle.screensize(900, 750, "#272727") # 设置屏幕大小和背景颜色  t = turtle.Turtle()  t.hideturtle() # 隐藏画笔箭头  t.width(1) # 设置线条粗细  t.color("#EBEBD0", "#006030") # 设置画笔颜色和填充颜色  drawBarchart(t) # 开始绘制  turtle.exitonclick() # 点击关闭窗口  '''数据处理部分'''  # 处理每行文本以计算词频 def processLine(line, wordamounts):  line = replacePunctuations(line) # 用空格替换标点符号  words = line.split() # 分割每行文本，获取单词  for word in words:  if word in wordamounts:  wordamounts[word] += 1 # 如果单词已存在，则词频加一  else:  wordamounts[word] = 1 # 如果单词不存在，设置词频为1  # 替换标点符号为空格 def replacePunctuations(line):  for ch in line:  if ch in "~!@#$%^&\*()-\_+=<>?/,.:;{}[]|\'\"": # 列举要替换的标点  line = line.replace(ch, ' ')  return line  # 处理数据，得到出现频率最高的单词及其频数 def dataProcess(filename):  infile = open(filename, 'r', encoding='UTF-8') # 打开文件并指定编码  wordamounts = {} # 初始化一个空字典记录词频  for line in infile:  processLine(line.lower(), wordamounts) # 转小写并处理每行文本  pairs = list(wordamounts.items()) # 将词频字典转换为列表  items = [[x, y] for (y, x) in pairs] # 调换列表内键值顺序，方便按频数排序  items.sort() # 按频数排序  # 输出前amount个词频结果  for i in range(len(items) - 1, len(items) - amount - 1, -1):  print(items[i][1] + "\t" + str(items[i][0]))  wCounts.append(items[i][0])  words.append(items[i][1])  infile.close()  # 主函数，程序入口 def main():  global amount # 声明全局变量 amount  filename = input("请输入文件名:").strip() # 输入文件名  amount = int(input("请输入要显示的高频词数量:")) # 用户输入高频词数量  dataProcess(filename) # 处理数据  init() # 初始化绘图  # 调用主函数 if \_\_name\_\_ == '\_\_main\_\_':  main()  **4.如果给定的文件是汉字文件，测试程序是否有效。请描述你遇到的问题，结合遇到的问题分析，为什么英文文本可以统计高频词出现的次数，而中文文本不能？请查阅资料给出要实现中文文本高频词统计，并可视化，要对中文文本进行怎样的处理。**  代码中没有考虑中文分词，直接按空格进行分词。在英文文本中，单词之间有空格，而中文文本中没有，因此无法直接使用空格来分隔单词。 要解决这个问题，可以使用 jieba 分词库，它是一个适用于中文的分词工具。使用 jieba 分词，可以将每个句子分成单独的词汇，然后计算词频。  import turtle import jieba # 导入 jieba 中文分词库  '''全局变量''' words = [] # 单词集合，作为x轴数据 wCounts = [] # 单词频数集合，作为y轴数据 xPoint = -360 # x轴起点 yPoint = -200 # y轴起点 amount = 10 # 默认显示的词频个数  '''turtle 绘图部分'''  # 绘制从点 (x1, y1) 到 (x2, y2) 的线段 def drawLine(t, x1, y1, x2, y2):  t.penup() # 提起画笔，不进行绘图，移动到起点  t.goto(x1, y1) # 移动到起点  t.pendown() # 放下画笔，准备绘图  t.goto(x2, y2) # 绘制到终点  # 在坐标 (x, y) 位置显示文字 def drawText(t, x, y, text, fontSize=10):  t.penup() # 提起画笔，不绘图  t.goto(x, y) # 移动到文字显示位置  t.pendown() # 放下画笔，准备写文字  t.write(text, font=('微软雅黑', fontSize), align='center') # 绘制文字  # 绘制矩形，以 (x, y) 为中心，宽度为 2\*rWidth def drawRectangle(t, x, y, rWidth):  drawLine(t, x - rWidth, yPoint, x - rWidth, y) # 左边线  drawLine(t, x - rWidth, y, x + rWidth, y) # 顶边线  drawLine(t, x + rWidth, y, x + rWidth, yPoint) # 右边线  drawLine(t, x + rWidth, yPoint, x - rWidth, yPoint) # 底边线  # 绘制柱状图 def drawBarchart(t):  drawText(t, 0, -yPoint - 40, "词频统计结果", 15) # 绘制标题  drawRectangle(t, 0, -yPoint, -xPoint) # 绘制边框  rWidth = -xPoint / (2 \* amount) # 控制每个条形柱的宽度  xScale = -xPoint \* 2 / (amount + 1) # x轴的缩放倍数，根据amount调节  yScale = -yPoint / wCounts[0] # y轴的缩放倍数，根据频数调节  for i in range(amount):  i = i + 1 # 右移以避免与原点重合  x = i \* xScale + xPoint  y = wCounts[i - 1] \* yScale + yPoint  drawText(t, x, yPoint - 20, words[i - 1]) # 打印单词  drawText(t, x, y + 10, wCounts[i - 1]) # 打印频率  t.begin\_fill()  drawRectangle(t, x, y, rWidth)  t.end\_fill()  # 初始化窗口 def init():  turtle.title('词频结果柱状图')  turtle.screensize(900, 750, "#272727") # 设置屏幕大小和背景颜色  t = turtle.Turtle()  t.hideturtle() # 隐藏画笔箭头  t.width(1) # 设置线条粗细  t.color("#EBEBD0", "#006030") # 设置画笔颜色和填充颜色  drawBarchart(t) # 开始绘制  turtle.exitonclick() # 点击关闭窗口  '''数据处理部分'''  # 处理每行文本以计算词频 def processLine(line, wordamounts):  line = replacePunctuations(line) # 用空格替换标点符号  words = jieba.lcut(line) # 使用 jieba 对中文进行分词  for word in words:  if word.strip(): # 去除空白字符  if word in wordamounts:  wordamounts[word] += 1 # 如果单词已存在，则词频加一  else:  wordamounts[word] = 1 # 如果单词不存在，设置词频为1  # 替换标点符号为空格 def replacePunctuations(line):  for ch in line:  if ch in "~!@#$%^&\*()-\_+=<>?/,.:;{}[]|\'\"，。！？；：“”‘’（）【】《》「」": # 列举要替换的标点  line = line.replace(ch, ' ')  return line  # 处理数据，得到出现频率最高的单词及其频数 def dataProcess(filename):  infile = open(filename, 'r', encoding='UTF-8') # 打开文件并指定编码  wordamounts = {} # 初始化一个空字典记录词频  for line in infile:  processLine(line.lower(), wordamounts) # 转小写并处理每行文本  pairs = list(wordamounts.items()) # 将词频字典转换为列表  items = [[x, y] for (y, x) in pairs] # 调换列表内键值顺序，方便按频数排序  items.sort() # 按频数排序  # 输出前amount个词频结果  for i in range(len(items) - 1, len(items) - amount - 1, -1):  print(items[i][1] + "\t" + str(items[i][0]))  wCounts.append(items[i][0])  words.append(items[i][1])  infile.close()  # 主函数，程序入口 def main():  global amount # 声明全局变量 amount  filename = input("请输入文件名:").strip() # 输入文件名  amount = int(input("请输入要显示的高频词数量:")) # 用户输入高频词数量  dataProcess(filename) # 处理数据  init() # 初始化绘图  # 调用主函数 if \_\_name\_\_ == '\_\_main\_\_':  main() | | | | | | | |
| 五．实验结果  1、基础练习   2、解决问题  思考题：  3.如果统计高频词的数量不是10个而是用户自己输入的个数，应该如何修改程序。  4.如果给定的文件是汉字文件，测试程序是否有效。请描述你遇到的问题，结合遇到的问题分析，为什么英文文本可以统计高频词出现的次数，而中文文本不能？请查阅资料给出要实现中文文本高频词统计，并可视化，要对中文文本进行怎样的处理。 | | | | | | | |
| 六、实验中遇到的问题及解决方法 | | | | | | | |