过程分析、设计、实现，面向对象的分析、设计、实现。

1. 过程分析、设计、实现
2. 过程分析
3. 、程序需要读取一个文本里面的单词，可以是任意英文文本
4. 、至少统计100个词以上
5. 、用户可以同时查询多个单词出现的次数
6. 、单词出现的次数进行降序显示
7. 、统计文本单词及词频，并输出结果
8. 、在图形界面显示文件，通过点击文件显示文件具体统计结果
9. 设计

设计一个图形化窗口，包含文件选择框，选择文件按钮，统计按钮，结果显示文本框，对存储在文本文件下的英语文本进行读取，形成列表，遍历列表中的单词，统计单词出现的次数，并在文本框中显示最后的统计结果

1. 设计实现
2. 、
3. 面向对象的设计

实验步骤：

1. 首先读取指定文件.txt，读取文件中的单词；若读取失败，直接结束；
2. 文件读取完后，形成列表，遍历列表重的单词；
3. 统计列表里的单词；
4. 输出运行结果.

实验结果：

the 27

of 10

she 8

to 7

and 7

her 6

our 5

for 5

that 5

as 5

部分代码：

# 获得单个文档词频统计的计数器对象Counter

def getFileCounter(self, filepath):

# 拿到文本

file = open(filepath, "r", encoding="gbk")

text = file.read()

# 英文单词的正则表达式

pattern = "[A-Za-z]+"

# 从text中找出所有英文单词形成列表

wordlist = re.findall(pattern, text)

# wordlist的基础上创建Counter对象

counter = Counter(wordlist)

# 返回单个文件的计数器对象

return counter

# 统计一个文件夹下所有文档的一揽子词频

def countDocWordsInDir(self, dirpath):

# 罗列一个文件下的所有文件(含文件夹)

flist = os.listdir(dirpath)

# 创建空的计数器对象，统计对象通过counter对象的加法操作逐步扩大

mcounter = Counter()

# 遍历文件夹下的所有文件

for name in flist:

# 对文档文件进行统计操作

if name.endswith(".txt"):

docpath = dirpath + name # 获得文档文件的路径

# 获得文档的计数器对象

counter = self.getFileCounter(docpath)

# 计数器对象进行加法操作，扩大统计样本

mcounter += counter

# 得到结果

resList = mcounter.most\_common()

#print(resList)

