《Big Data Analytics》实验报告

年级、专业、班级	2018 级机械 1 班	姓名	易弘睿	学号	20186103
实验题目	简单数据处理				
实验时间	2022 年 3 月 1 日	实验地点	A 理 119		
学年学期	2021-2022(2)	实验性质	□验证性 ■设计性 □综合性		

一、实验目的

1) 掌握 Python 基础语法；
2) 掌握 Python 复杂数据类型；
3) 熟练用 Python 程序解决实际问题；

二、实验项目内容

1.最低难度要求：

实现以《三国演义》为密码本，对输入的中文文本进行加密和解密。

需要用到的知识：

1）文件读写；

2）字典和集合；

提示：

1）为《三国演义》中出现的每个字赋予一个编码，例如：

实：201102

现：301209

利用这个进行加密；

2）为《三国演义》中出现的每一个编码对应相应的字，例如：

201102：实

301209：现

利用这个进行解密；

2.中等难度要求：

对《三国演义》的电子文档进行页的划分，以 400 个字为 1 页，每页 20 行 20 列，那么建立每个字对应的八位密码表示，其中前 1~4 位为页码，5、6 位为行号，7、8 位为这一行的第几列。例如：实：24131209，表示字“实”出现在第 2413 页的 12 行的 09 列。

利用此方法对中文文本进行加密和解密。

3.最高难度要求：(注意下面这个题是对英文而言，不是针对上面的《三国演义了》)

现监听到敌方加密后的密文 100 篇，但是不知道敌方的加密表，但是知道该密码表是由用一个英文字母代替另一个英文字母的方式实现的，现请尝试破译该密码。

提示：

1）对加密后的全部秘文进行统计，统计密文中每个字母出现的概率；

2）从网上采集大量的英文文本（例如从 VOA 采集 10000 篇）构成普通文本集合，统计这些英文文本中每个英文字母出现的概率；

3）然后根据密文和普通文档中的英文字母出现概率值进行猜测加密表中英文字母之间对应关系；

4）如验证猜测的密码表是否正确：利用猜测出的密码表对密文进行解密，看解密所得到的文本中的英文单词是否正确（测试这些单词在所采集的 10000 篇英文文本中是否存在）。

三、实验过程和结果

本次实验我实现了中等难度要求和最高难度要求的加密以及解密。

1. 中等难度要求的实现——《三国演义》的加密及解密

1.1 实验过程

首先，读取电子版三国演义，然后删除其中的标点符号以及空格，只保留文字，再将其储存为列表，使单个文字作为单个字符串储存在列表中。

```
import unicodedata as ucd

def de_duplication(str):
    dedup_str = ''
    for char in str:
        if not char in dedup_str:
            dedup_str += char
    return dedup_str

#读取文件
with open('三国演义.txt ', 'r', encoding="utf-8") as f:
    data = f.read()
word = ucd.normalize('NFKC',data).replace(',', '').replace('.', '').replace('。', '').replace('\', '').replace('"', '')\
    .replace('"','').replace('《', '').replace('》', '').replace('’', '').replace(';', "").replace('\n', '')\
    .replace('? ', '').replace(':', '').replace('；', '').replace('-', '').replace('!', '')\
    .replace('、', '').replace('(', '').replace(')', '').replace('—', '').replace(' ', '')
word = list(word)
word_list = de_duplication(word)
```

然后，创建从 00010101 到 99992020 所有的密码。

```
#创建密码
code = []
for i in range(1,10000):
    code1 = []
    p = "{:0>4d}".format(i)
    code1.extend(p)
    for j in range(1, 21):
        code2 = []
        code2.extend(code1)
        r = "{:0>2d}".format(j)
        code2.extend(r)
        for k in range(1,21):
            code3 = []
            code3.extend(code2)
            c = "{:0>2d}".format(k)
            code3.extend(c)
            code.append("".join(code3))
```

接着，利用 zip()函数将 code 和 word 中对应的元素打包成一个个元组，然后返回由这些元组组成的列表，再用 dict()函数将其建立成解密字典 decrypt。将 decrypt 字典中的 key,value 交换，建立加密字典 encrypt。

```
#解密
decrypt = dict(zip(code, word))

#加密
vs = decrypt.values()
ks = decrypt.keys()
encrypt = dict(zip(vs,ks))
```

最后，完成交互部分，将用户输入的语句转换为列表形式保存至 x 中，利用 for 循环找到加密字典中对应的密码；输入密码，利用 split()函数根据空格分开输入密码，找到解密字典中对应汉字。

```
#交互
x = list(input('请输入语句：'))
for key in x:
    print(encrypt.get(key),'\t',end="")
print('\n')

y = input('请输入代码：  ')
y = y.split(' ')
for key in y:
    print(decrypt.get(key), '\t', end="")
```

1.2 实验结果

```
"E:\Program Files\Anaconda3\python.exe" "D:/OneDrive - University of Cincinnati/2022 Spring/大数据/pythonProject/Lab2_proj/Lab2_Medium.py"
请输入语句：大数据分析
12220819    12230504    12221302    12230513    00600407

请输入代码：  04151314 04161819 03231019 03230415 11231019
马  引  逢  却  在
Process finished with exit code 0
```

2. 最高难度要求的实现——VOA 文本加密及解密

2.1 实验过程

首先，读取 utf-8 编码的英文文本文件。

```
# 读取文件
def load_file(pathinp):
    file = open(pathinp, 'r', encoding='utf-8')
    allcharc = list(file.read())
    file.close()
    return allcharc
```

然后，去除英文文本中字符串的标点符号，并统计字母出现的次数。

```
# 去除字符串的标点符号并统计字母出现次数
def statistics(inptchar):
    dictstastis={}
    for i in inptchar:
        if (97 <= ord(i) < 123) or (65 <= ord(i) < 91):
            if i.lower() in dictstastis:
                dictstastis[i.lower()] += 1
            else:
                dictstastis[i.lower()] = 1
    dictstastis = dict(sorted(dictstastis.items(), key=lambda item:item[1], reverse=True))
    return dictstastis
```

接着，通过字母整体移动的方式产生加密表。

```python
# 产生加密表
def cipher(num):
    cipher_dict = {}
    chri = [chr(i) for i in range(97, 123)]                    #产生26字母

    for i in range(26 - num):
        cipher_dict[chri[i]] = chr(ord(chri[i]) + num)         #字母整体移动
    for i in range(num):
        cipher_dict[chri[26 - num + i]] = chr(ord(chri[i]))
    return cipher_dict
```

完成加密和解密 function。

```python
# 加密
def encode(inpfile,cipher_dict):
    ciphertxt = []
    for m in inpfile:
        if (97 <= ord(m) < 123) or (65 <= ord(m) < 91):
            ciphertxt.append(cipher_dict[m.lower()])
        else:
            ciphertxt.append(m)
    return ciphertxt
```

```python
# 解密
def decode(cipher,sta):
    deco_dictstastis={}
    for n in cipher:
        if (97 <= ord(n) < 123) or (65 <= ord(n) < 91):
            if n in deco_dictstastis:
                deco_dictstastis[n] += 1
            else:
                deco_dictstastis[n] = 1
    deco_dictstastis = dict(sorted(deco_dictstastis.items(), key=lambda item:item[1], reverse=True))
    print('统计密文字母出现次数： ')
    print(deco_dictstastis)
    decode_dict=dict(list(zip(deco_dictstastis.keys(),sta.keys())))
    return decode_dict
```

最后，完成 main 函数部分，根据所布置好的功能函数，依次对 VOA 语料文本进行文件读取、统计字母出现频率、定义密码表、加密和解密操作。

```python
def main():
    voatxt:list = []
    for i in range(60):
        path = 'VOA语料\\'+str(i)+'.txt'
        readtxt = load_file(path)
        voatxt = voatxt + readtxt
    occurrencessta = statistics(voatxt)
    print('统计明文文章中字母出现次数： ')
    print(occurrencessta)
    cipher_dict = cipher(6)
    print('定义的密码表为（字母向右移动六位）： ')
    print(cipher_dict)
    # 加密
    svoatxt: list = []
    for i in range(60):
        path2 = 'VOA语料\\'+str(i)+'.txt'
        sreadtxt = load_file(path2)
        svoatxt = svoatxt + sreadtxt
    endoce_txt = encode(svoatxt,cipher_dict)
    # print(''.join(endoce_txt))                    #如果需要输出密文可以取消注释
    # 解密
    decode_dict = decode(endoce_txt,occurrencessta)
    decode_table={}
```

```python
cipvalue = cipher_dict.values()
for k in cipvalue:
    decode_table[k] = decode_dict[k]
print('解码产生的密码表为： ')
print(decode_table)
```

2.2 实验结果

```
"E:\Program Files\Anaconda3\python.exe" "D:/OneDrive - University of Cincinnati/2022 Spring/大数据/pythonProject/Lab2_proj/Lab2_High.py"
统计明文文章中字母出现次数：
{'e': 18316, 't': 13317, 'a': 12291, 'o': 11816, 'n': 10774, 's': 10679, 'i': 10562, 'r': 9824, 'h': 6974, 'l': 5671, 'd': 5439, 'c': 4753,
定义的密码表为（字母向右移动六位）：
{'a': 'g', 'b': 'h', 'c': 'i', 'd': 'j', 'e': 'k', 'f': 'l', 'g': 'm', 'h': 'n', 'i': 'o', 'j': 'p', 'k': 'q', 'l': 'r', 'm': 's', 'n': 't'
统计密文字母出现次数：
{'k': 18316, 'z': 13317, 'g': 12291, 'u': 11816, 't': 10774, 'y': 10679, 'o': 10562, 'x': 9824, 'n': 6974, 'r': 5671, 'j': 5439, 'i': 4753,
解码产生的密码表为：
{'g': 'a', 'h': 'b', 'i': 'c', 'j': 'd', 'k': 'e', 'l': 'f', 'm': 'g', 'n': 'h', 'o': 'i', 'p': 'j', 'q': 'k', 'r': 'l', 's': 'm', 't': 'n'

Process finished with exit code 0
```

四、实验总结与体会

1. 问题及解决

在实验过程中，最开始导入文件时，我发现其中的标点符号有一些在导入之后变成了英文符号，而且出现了\u3000，\n 这类字符。最后用了大量 replace 函数去删掉所有符号，也百度了 normalize 函数去标准化空格。也是暂时没有想到如何将代码简单化。

在创建密码的过程中，运用了多个 extend 和 append，最初对这两个函数运用不熟练，一直报错，花费了许多时间调整。让我对这两个函数有了更深刻的理解。

在思考如何将文字保存为单个字符串的过程中，也是费了一点时间，最后发现将其转化为 list 就可以解决。

2. 总结

这次实验课让我更加熟练地掌握 Python 基础语法、掌握 Python 复杂数据类型、熟练用 Python 程序解决实际问题，在写代码的过程中对课上地理论知识有了进一步地加深，体会到只有自己动手写代码才会有进步。

实验报告填写说明：

1、第一、二部分由老师提供；

2、第三部分填写主要操作步骤（文字）与结果 3（截图）；

3、第四部分主要填写遇到的问题与解决问题的方法、总结和体会等；

4、报告规范：包含报告页眉、报告的排版、内容是否填写，命名是否规范等内容；

5、实验文件与实验报告命名：学号姓名序号.*（应用对应的扩展名），学号姓名(en).docx，

例如学号 20161234 的张三同学，**他的第一次实验命名为：20161234 张三(e1)1.docx**