深 圳 大 学 实 验 报 告

课程名称：Python 程序设计

实验项目名称：作业 1

学院：计算机与软件学院

专业：软件工程

指导教师：潘浩源

报告人：郑彦薇 学号：2020151022

实验时间：2022/4/25~2022/4/27

实验报告提交时间：2022/4/27

教务部制

一、实验目的：

利用所学基础知识与数据结构，解决生活实际问题。

二、实验方法步骤

1、读题，对每个问题提出解决问题的思路

2、对需要编程解决的题目按照得到的思路编写源代码

三、实验过程及内容：

（一）解题思路

1. 信用卡号码：

1) 利用 python 中将输入默认为字符串，将信用卡号码视为一串字符串进行处理。

2) 获得信用卡号码的长度，从倒数第二位开始，获取每隔一位数字。因为 0*2=0，任何数加 0 仍为本身，因此可以只需记录不等于 0 的字符。

3) 将（2）中获得的字符强制转化为整型，如果字符大于或等于 10，则需获得各个数位，利用对 10 取余、原数除以 10 的操作即可实现。其它小于 10 的数直接相加即可，将结果存放在 sum1 中。

4) 从倒数第一位开始，每隔一位获取没有乘以 2 的字符并强制转化为整型，相加，将结果存放在 sum2 中。

5) sum1 与 sum2 相加，将结果赋值给 sum3，sum3 对 10 取余，如果结果为 0，说明卡号有效。进一步判断该信用卡由哪家公司发行。

6) 根据不同公司发行的信用卡卡号的特点，设置选择语句，找到卡号对应的发行公司，进行输出即可。

代码及细节解释如下：

```
def IssueCmp(cardNum):
    i = 0
    Len = len(cardNum)
    if Len == 15 and cardNum[i] == '3':
        if cardNum[i+1] == '4' or cardNum[i+1] == '7':
            print("American Express")
    if Len == 16 and cardNum[i] == '5':
        if int(cardNum[i+1]) >= 1 and int(cardNum[i+1]) <= 5:
            print("MasterCard")
    if Len == 13 or Len == 16:
        if cardNum[i] == '4':
            print("Visa")
```

根据条件判断有效信用卡卡号对应的发行公司

```python
def validCredit(cardNum):
    longs = len(cardNum)
    i = longs-2
    temp = []
    while i >= 0:
        if cardNum[i] != '0':
            temp.append(int(cardNum[i])*2)
        i -= 2
    sum1 = 0
    for j in temp:
        num = int(j)
        if num >= 10:
            while num > 0:
                sum1 += num % 10
                num //= 10
        else:
            sum1 += num
    i = longs-1
    sum2 = 0
    while i>=0:
        sum2 += int(cardNum[i])
        i -= 2
    sum3 = sum1 + sum2
    if sum3 % 10 == 0:
        print("Valid,", end=' ')
        IssueCmp(cardNum)
    else:
        print("Invalid")

cardNum = input()
validCredit(cardNum)
```

获得需要进行乘 2 的数字，将每个数位相加，并将结果存放在 sum1 中。

获得不需要进行乘 2 的数字，进行相加，并将结果存放在 sum2 中。

根据 sum3 对 10 取余的结果，判断卡号是否有效。

2. 分析名著：

1) 存放数据：将三本名著存放在与运行代码同个文件夹里，并在 python 中打开。
2) 对文本文档中的标点符号、换行符及数字转换为空格，根据空格将文本文档中的单词进行分割。进一步处理单词列表，如果列表为空或者列表为换行列，丢弃这些行，并将得到的单词列表中的大写转换为小写。
3) 读取文件的每一行，按照（2）的做法获得单词列表，设置字典存放单词以及出现的次数。如果单词在字典中已经存在，则将次数加 1；若单词在字典中不存在，则将该单词作为 key 加入字典中，然后将次数设置为 1。
4) 设置元组，将字典中的键值信息复制。对元组进行降序排序，并将前 20 个单词以及出现次数进行输出。

代码及细节解释如下：

```python
import string
def DealTxt(word):
    for ch in word:
        if ch in string.punctuation or ch == '\n' or '0' <= ch <= '9':
            word = word.replace(ch, " ")
    temp = word.split(' ')
    for i in range(len(temp)-1, -1, -1):
        if temp[i] == '\n' or temp[i] == ' ':
            del temp[i]
            continue
        temp[i] = temp[i].lower()
    return temp

dictword = {}
def getWordNum(filename):
    inputFile=  open(filename, "r")
    while True:
        line = inputFile.readline()
        if not line:
            break
        if line == '\n':
            continue
        wordList = DealTxt(line)
        for temp in wordList:
            if temp in dictword.keys():
                dictword[temp] += 1
            else:
                dictword[temp] = 1
    inputFile.close()

#getWordNum('treasure.txt')
#getWordNum('hyde.txt')
getWordNum('war.txt')
res = []
for temp in dictword:
    tuple_temp = (dictword[temp], temp)
    res.append(tuple_temp)
res.sort()
res.reverse()
print("Leo Tolstoy's Top 20 Words:")
#print("Robert Louis Stevenson's Top 20 Words:")
for i in range(1,21):
    print("Frequency: "+str(res[i][0])+" Word: "+str(res[i][1]))
```

处理不是单词的字符，去除文件中空行或换行列

对单词出现次数进行统计，结果用字典存放

用元组存放结果，方便对结果进行降序排序。

3. 字符串“邻居”：

offByOne(str1, str2)函数：

1) 根据这种“邻居”字符串的规定，首先获取两个字符串的长度，进行比较。如果长度相同，进一步判断；如果长度不相同，返回 False。
2) 对于长度相同的字符串，从第一位开始依次对每一位进行比较，用一个临时变量 count 统计两个字符串中不同的字符数。
3) 字符串比较到最后一位后，对 count 值进行判断：如果 count 值为 1 即只在一个位置不同，则返回 True；否则返回 False。

代码及细节解释如下：

```python
def offByOne(str1, str2):
    l1 = len(str1)
    l2 = len(str2)
    if l1 != l2:
        return False
    count = 0
    for i in range(l1):
        if str1[i] != str2[i]:
            count += 1
    if count != 1:
        return False
    return True

str1 = input()
str2 = input()
if offByOne(str1, str2):
    print("True")
else:
    print("False")
```

判断两个字符串的长度是否相同

统计两个字符串有几个位置不同

offBySwap(str1, str2)函数：

1) 因为字符串类型不能进行元素的赋值，因此在传入函数之前，首先将输入的字符串强制转换为列表结构，以便于函数中对字符串进行元素互换操作。
2) 对于传入函数的两个字符串，首先判断长度是否相等以及两个字符串是否已经相等，若两个字符串长度不等或字符串已经相等，则返回 False，否则进行进一步判断。
3) 利用切片功能复制其中的一个字符串，根据该字符串的长度设置循环，进行相邻元素的互换，若在其中一轮互换中可以得到互换的结果与另一个字符串相等，则满足条件，返回 True；若循环结束仍不出现相等的情况，说明不符合条件，返回 False。

代码及细节解释如下：

```python
def offBySwap(str1, str2):
    if len(str1) == len(str2) and str1 != str2:
        for i in range(len(str1)-1):
            temp = str1[:]
            temp[i] = str1[i+1]
            temp[i+1] = str1[i]
            if temp == str2:
                return True
    return False
```

利用切片功能复制其中一个字符串

```
str1 = input()
str2 = input()
s1 = list(str1)
s2 = list(str2)
if offBySwap(s1, s2):
    print("True")
else:
    print("False")
```

将输入的字符串强制转换为列表，方便函数内赋值给 temp

offByExtra(str1, str2)函数：

1) 与 offBySwap 类似，为了能够实现对字符串单个元素的操作，将输入的字符串强制转换为列表
2) 在函数中，判断一个字符串的长度是否比另一个字符串的长度大 1，如果是，继续接下来的判断，如果不是，返回 False。
3) 根据字符串的长度设置循环，首先利用切片功能赋值字符串，然后依次删去长度较长的字符串中的一个元素，判断删除当前元素后的字符串是否与另一个相同，如果是，则返回 True；如果不是，则返回 False。

代码及细节解释如下：

```
def offByExtra(str1, str2):
    l1 = len(str1)
    l2 = len(str2)
    if l1 == l2+1:
        for i in range(l1):
            temp1 = str1[:]
            temp1.pop(i)
            if temp1 == str2:
                return True
    if l2 == l1+1:
        for i in range(l2):
            temp2 = str2[:]
            temp2.pop(i)
            if temp2 == str1:
                return True
    return False

str1 = input()
str2 = input()
s1 = list(str1)
s2 = list(str2)
#if offBySwap(s1, s2):
if offByExtra(s1, s2):
    print("True")
else:
    print("False")
```

判断其中的一个字符串长度是否是另一个字符串长度加 1

利用函数 pop 删除指定位置的元素

ListOfNeighbors(str, L)函数：

1) 首先是根据 EnglishWords 文本文件获得一个从存放了这些单词的单词列表，读取文件，按照换行符对单词仅从分割，获取单词列表。
2) 根据单词列表的长度设置循环，获得列表中的每个单词，将其强制转换为列表，与输入的字符串做上述 3 种“邻居”单词的判断，若为其中一种情况，将当前的单词添加到列表 res 中。
3) 查找结束后对 res 列表进行输出，可以得到输入单词的所有邻居。

代码及细节解释如下：

```
41  def getWords(filepath):
42      file = open(filepath)
43      wordOne = []
44      while file:
45          line = file.readline()
46          word = line.split(' ')
47          wordOne.extend(word)
48          if not line:
49              break
50      wordTwo = []
51      for i in wordOne:
52          wordTwo.extend(i.split())
53      return wordTwo
54
55  def ListOfNeighbors(str, L):
56      res = []
57      for i in range(len(L)):
58          if offByExtra(str, list(L[i])) or offByOne(str, list(L[i])) or offBySwap(str, list(L[i])):
59              res.append(L[i])
60      for j in range(len(res)):
61          print(res[j], end=' ')
62
63  str = input()
64  s = list(str)
65  filepath = 'EnglishWords.txt'
66  L = getWords(filepath)
67  ListOfNeighbors(s,L)
```

处理文本文档，获取单词列表

处理文本文档，获取单词列表

强制转换单词为列表，以便于函数中的操作

存放结果列表

4. MasterMind：

1) 首先是管理员输入数字，将输入默认的字符串转换为列表，设置函数对传入的列表进行判断。若输入的列表长度不为 5 或有数字重复，提示错误，并重新输入。这里对数字重复的判断方法如下：根据列表长度设置双重循环，若在第 i 个位置之后有元素与第 i 个元素相同，则返回 False。
2) 管理员正确输入数字后，列出游戏规则并允许玩家进行输入。对玩家输入的数字同样进行合法性判断，除了长度是否为 5 以及是否有元素重复外，还需判断输入是否只含数字：根据长度设置循环，依次检查每个位置，如果出现非数字，返回 False，并提示玩家重新输入。
3) 玩家正确输入数字后，开始判断是否猜对了数字，并开始统计玩家猜测的次数。首先是正确位数的统计，设置双重循环，从管理员数字的第一位开始，查找玩家输入的数字中是否有与第一位数字相同的数字，如果有，正确位数加 1，直到循环结束；接着是正确位置位数的统计，根据数字长度设置循环，依次检查管理员数字和玩家数字相同位置的元素是否相同，若相同，正确位置位数加 1，直到循环结束。
4) 若上述求得正确位数和正确位置位数皆为数字长度即 5，则说明猜对了数字，输出猜测次数等信息；若不为数字长度，则说明仍需要继续进行猜数，输出当前猜测状态信息，提示玩家再次输入数字。

5) 设置提前退出游戏标志，若用户输入 EXIT，则退出游戏，并输出当前猜测次数。

6) 在进行判断时，将输入转换为列表，因此在进行当前猜测状态信息的输出时，需要将列表重新转换为字符串。

代码及细节解释如下：

```
def Judgeinput1(str):                                   对管理员输入数字的合法性进行判断
    if len(str) != 5:
        print("错误!数字长度必须是5!", end=' ')
        return False
    for i in range(len(str)-1):
        for j in range(i+1, len(str)):
            if str[i] == str[j]:
                print("错误!不能重复数字!", end=' ')
                return False
    return True

def Judgeinput2(str):                                   对玩家输入数字的合法性的进一步判断
    for i in range(len(str)):
        if str[i] < '0' or str[i] > '9':
            print("错误!只允许使用数字!", end=' ')
            return False
    return True

def Guess(str1, str2):                                  判断玩家猜数是否正确
    wei = 0
    pos = 0
    for i in range(len(str1)):                          设置双重循环统计正确位数
        for j in range(len(str2)):
            if str1[i] == str2[j]:
                wei += 1
    for i in range(len(str1)):                          设置循环统计正确位置位数
        if str1[i] == str2[i]:
            pos += 1
    if wei == 5 and pos == 5:
        print("成功!!!你一共猜了", end='')
        return True
    print(f'正确位数{wei}, 正确位置位数{pos}')
```

```
print("请输入原始数字:", end=' ')
str1 = list(input())
while Judgeinput1(str1) != True:
    print("重新输入:", end=' ')
    str1 = list(input())
Maxcount = 5
print(f"游戏规则:\n(1)输入数字格式错误不计次数!\n"
      f"(2)输入EXIT可以结束游戏,视为游戏失败!\n"
      f"(3)你只有{Maxcount}次机会!\n"
      f"游戏开始!")
count = 1
while count <= Maxcount:
    print("请输入你猜的数: ", end=' ')
    str2 = list(input())
    temp = ''.join(str2)
    if temp == 'EXIT':
        print(f'你已提前退出游戏,游戏失败,你一共猜了{count-1}次!')
        break
    while Judgeinput2(str2) != True or Judgeinput1(str2) != True:
        print("再猜一次:", end=' ')
        str2 = list(input())
        temp = ''.join(str2)
    print(f'猜测次数{count}, 猜测数字{temp},', end=' ')
    if Guess(str1, str2) == True:
        print(f'{count}次')
        break
    count += 1
if count > Maxcount:
    print("达到猜测上限!失败!!!")
```

管理员输入合法之前，重复输入

输出数字前将数字列表转换为字符串

提前退出游戏

玩家输入合法之前，重复输入

5. 抓狐狸：

1) 该游戏的实现比较简单，用基础知识并借助 random 库即可实现。
2) 首先是输入狐狸可以藏匿的总洞口数，并以此为范围，借助 random 中的 randint 函数随机生成狐狸的第一个位置，将玩家已尝试次数初始化为 1，并提示玩家输入猜测狐狸可能藏匿的位置。
3) 判断玩家输入的位置与狐狸当前藏匿的位置是否相同，如果相同，结束游戏；如果不同，进行接下来的操作。
4) 若狐狸当天在第一个洞口，则第二天只能到第二个洞口；如果狐狸当天在最后一个洞口，则第二天只能到前一个洞口；如果狐狸当天位置在中间，则借助 choice 函数，随机得到狐狸第二天的位置在前一个洞口或下一个洞口；得到狐狸新位置后，允许玩家重新猜测。
5) 若玩家已达尝试次数的上限，结束游戏。

代码及细节解释如下：

```python
from random import choice, randint
print("请输入狐狸可以藏匿的洞口数:", end=' ')
N = int(input())
Maxchoice = 5
print("游戏规则:\n"
      f'(1)狐狸可以藏匿的洞口一共有{N}个,尝试时请勿超出此范围\n'
      f'(2)狐狸会在第二天随机进入今天所在洞口的隔壁洞口,可能是前一个，也可能是后一个\n'
      f'(3)你一共有{Maxchoice}次尝试机会\n'
      f'游戏开始!')
count = 1
fox = randint(1, N+1)
while count <= Maxchoice:
    print("请输入你认为狐狸可能在的洞口位置:", end=' ')
    pos = int(input())
    if pos == fox:
        print(f'恭喜你成功抓住狐狸,你一共用了{count}天')
        break
    if fox == 1:
        fox += 1
    elif fox == N:
        fox -= 1
    else:
        fox += choice((-1, 1))
    count += 1
    if count <= Maxchoice:
        print(f'很遗憾,请第{count}天再来抓')
if count > Maxchoice:
    print("你的尝试次数已用完!失败!")
```

随机生成狐狸第一天所在的位置

得到狐狸相对于前一天，第二天藏匿的位置

（二）运行结果

1. 信用卡号码：

8 组输入及对应的输出：

```
4003600000000014
Valid, Visa

6177292929
Invalid

378282246310005
Valid, American Express

371449635398431
Valid, American Express

5555555555554444
Valid, MasterCard

4111111111111111
Valid, Visa

4222222222222
Valid, Visa

4222222222222
Valid, Visa
```

2. 分析名著：

运行程序，得到两个作家最常用的 20 个单词：

Robert Louis Stevenson:

```
Robert Louis Stevenson's Top 20 Words:
Frequency: 14845 Word: the
Frequency: 9640 Word: and
Frequency: 6550 Word: i
Frequency: 5994 Word: of
Frequency: 5896 Word: a
Frequency: 5222 Word: to
Frequency: 3876 Word: was
Frequency: 3346 Word: in
Frequency: 3189 Word: he
Frequency: 3156 Word: you
Frequency: 3137 Word: that
Frequency: 2696 Word: it
Frequency: 2461 Word: had
Frequency: 2244 Word: his
Frequency: 2177 Word: my
Frequency: 2109 Word: with
Frequency: 2067 Word: as
Frequency: 1944 Word: s
Frequency: 1942 Word: for
Frequency: 1698 Word: on
```

Leo Tolstoy:

```
Leo Tolstoy's Top 20 Words:
Frequency: 34540 Word: the
Frequency: 22234 Word: and
Frequency: 16674 Word: to
Frequency: 14880 Word: of
Frequency: 10542 Word: a
Frequency: 9998 Word: he
Frequency: 8976 Word: in
Frequency: 8190 Word: that
Frequency: 7985 Word: his
Frequency: 7356 Word: was
Frequency: 5664 Word: with
Frequency: 5601 Word: it
Frequency: 5365 Word: had
Frequency: 4725 Word: her
Frequency: 4676 Word: not
Frequency: 4637 Word: him
Frequency: 4530 Word: at
Frequency: 4522 Word: i
Frequency: 4403 Word: s
Frequency: 4049 Word: but
```

3. 字符串“邻居”：

offByOne(str1, str2)函数：

4 组输入及对应输出

```
read                    read
rexd                    xexd
True                    False

a                       a
x                       a
True                    False
```

offBySwap(str1, str2)函数：

4 组输入及对应输出：

```
read                    reaxd
raed                    read
True                    False

x                       aaa
Y                       aaa
False                   False
```

offByExtra(str1, str2)函数：

4 组输入及对应输出：

```
abcd                    abxcd
abxcd                   abcd
True                    True

abcd                    abcd
bcda                    abcdef
False                   False
```

ListOfNeighbors(str, L)函数：

4 组输入及对应输出：

```
blue
blued bluer blues bluet bluey blur clue flue glue slue
python
pythons
update
updated updater updates
user
muser suer use used usee users uses usher ussr
```

4. MasterMind：

为了显示方便，这里将最大猜测次数设置为 5，下面展示 3 种不同情况（猜数正确、超过上限、提前退出）对应的结果：

猜数正确:

请输入原始数字: *1234*
错误!数字长度必须是5! 重新输入: *12343*
错误!不能重复数字! 重新输入: *12345*
游戏规则:
(1)输入数字错误不计次数!
(2)输入EXIT可以结束游戏,视为游戏失败!
(3)你只有5次机会!
游戏开始!
请输入你猜的数: *12346*
猜测次数1, 猜测数字12346, 正确位数4, 正确位置位数4
请输入你猜的数: *12347*
猜测次数2, 猜测数字12347, 正确位数4, 正确位置位数4
请输入你猜的数: *12348*
猜测次数3, 猜测数字12348, 正确位数4, 正确位置位数4
请输入你猜的数: *12349*
猜测次数4, 猜测数字12349, 正确位数4, 正确位置位数4
请输入你猜的数: *12345*
猜测次数5, 猜测数字12345, 成功!!!你一共猜了5次

超过上限:

请输入原始数字: *12345*
游戏规则:
(1)输入数字错误不计次数!
(2)输入EXIT可以结束游戏,视为游戏失败!
(3)你只有5次机会!
游戏开始!
请输入你猜的数: *12344*
错误!不能重复数字! 再猜一次: *12346*
猜测次数1, 猜测数字12346, 正确位数4, 正确位置位数4
请输入你猜的数: *12347*
猜测次数2, 猜测数字12347, 正确位数4, 正确位置位数4
请输入你猜的数: *1234*
错误!数字长度必须是5! 再猜一次: *12348*
猜测次数3, 猜测数字12348, 正确位数4, 正确位置位数4
请输入你猜的数: *12349*
猜测次数4, 猜测数字12349, 正确位数4, 正确位置位数4
请输入你猜的数: *12340*
猜测次数5, 猜测数字12340, 正确位数4, 正确位置位数4
达到猜测上限!失败!!!

提前退出：

请输入原始数字： *12345*
游戏规则：
(1)输入数字格式错误不计次数！
(2)输入EXIT可以结束游戏，视为游戏失败！
(3)你只有5次机会！
游戏开始！
请输入你猜的数： *12346*
猜测次数1， 猜测数字12346， 正确位数4， 正确位置位数4
请输入你猜的数： *12347*
猜测次数2， 猜测数字12347， 正确位数4， 正确位置位数4
请输入你猜的数： *EXIT*
你已提前退出游戏，游戏失败，你一共猜了2次！

5. 抓狐狸：

为了显示方便，这里输入允许狐狸藏匿的洞口数为 10，将最大猜测次数设置为 5，下面展示 2 种不同情况（猜数正确、超过上限）对应的结果：

猜数正确：

请输入狐狸可以藏匿的洞口数： *10*
游戏规则：
(1)狐狸可以藏匿的洞口一共有10个，尝试时请勿超出此范围
(2)狐狸会在第二天随机进入今天所在洞口的隔壁洞口，可能是前一个，也可能是后一个
(3)你一共有5次尝试机会
游戏开始！
请输入你认为狐狸可能在的洞口位置： *5*
很遗憾，请第2天再来抓
请输入你认为狐狸可能在的洞口位置： *3*
很遗憾，请第3天再来抓
请输入你认为狐狸可能在的洞口位置： *6*
恭喜你成功抓住狐狸，你一共用了3天

超过上限（即游戏失败）：

请输入狐狸可以藏匿的洞口数：*10*
游戏规则：
(1)狐狸可以藏匿的洞口一共有10个,尝试时请勿超出此范围
(2)狐狸会在第二天随机进入今天所在洞口的隔壁洞口,可能是前一个，也可能是后一个
(3)你一共有5次尝试机会
游戏开始!
请输入你认为狐狸可能在的洞口位置：*5*
很遗憾,请第2天再来抓
请输入你认为狐狸可能在的洞口位置：*4*
很遗憾,请第3天再来抓
请输入你认为狐狸可能在的洞口位置：*3*
很遗憾,请第4天再来抓
请输入你认为狐狸可能在的洞口位置：*2*
很遗憾,请第5天再来抓
请输入你认为狐狸可能在的洞口位置：*1*
你的尝试次数已用完!失败!

四、实验总结：

1、该实验是利用 python 中列表、元组、字典等数据结构解决实际问题

2、在进行应用时，应注意不同数据结构不同的定义方式以及它们具备的不同功能

指导教师批阅意见：

成绩评定：

指导教师签字：

年　　月　　日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。