### 3.4 加密与解密

本节内容按2课时设计。

**第一课时**

**教学重难点**

**【**教学重点**】**会分析生活中由密码引起的数据安全问题，理解对数据进行保护的意义。

**【**教学难点**】**用穷举法破解密码。

**【教学过程】**

**一、引入**

1.学生预习，阅读第77页“任务一 解开网站账号被盗之谜”之“活动1 防范‘盗窃’行为”，填写第78页的表3.4.1。

2.教师检查，并评讲填写情况，引出数据安全、密码的话题。

表3.4.1 密码盗窃与防盗分析表

|  |  |
| --- | --- |
| 盗窃方式 | 防盗窃措施 |
| 肩窥方式 | 我们可以在输入密码时遮挡自己的操作过程，防止别人偷看密码信息；或者确定环境安全后再进行操作 |
| 字典破解 | 密码不要用生日、姓名及拼音缩写、手机号、简单数字组合、单词、身份证号、用户Id |
| 暴力破解 | 同时包含大小写字母、数字、特殊符号，不少于8个字符 |

**二、密码**

密码是指用来核对用户ID以验证用户就是本人的一组字符。

活动2设置安全密码

打开教科书配套资源中的“数据安全性测试.py”，运行并测试穷举搜索一个5位、7位和9位的数字密码，分别需要多长时间，填写表3.4.2。**（详见配套资源“数据安全性测试(有界面).fld”和“数据安全性测试(有界面).py”）**

表3.4.2 穷举搜索密码需要的时间

|  |  |  |
| --- | --- | --- |
| 密码长度 | 密码值 | 破解时间 |
| 5位 | 12567 | 1.0毫秒 |
| 98302 | 15.6毫秒 |
| 7位 |  |  |
|  |  |
| 9位 |  |  |
|  |  |

实现穷举搜索的自定义函数jiemi()的程序代码如下：

def jiemi():

d1=datetime.datetime.now() #获取当前系统时间d1

p=int(varin.get()) #获取输入文本框的数字密码

for i in range(0,p+1): #从0循环到正确密码数值

if i==p: #如果密码相同

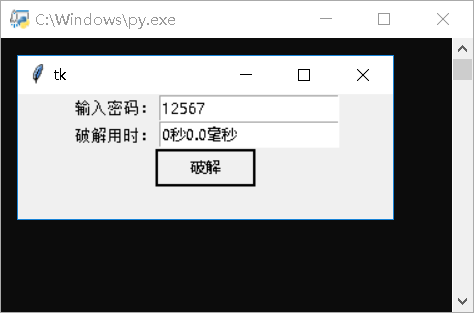
d2=datetime.datetime.now() #获取当前系统时间d2

d=d2-d1 #取得时间差

#在输出文本框中显示解密用时

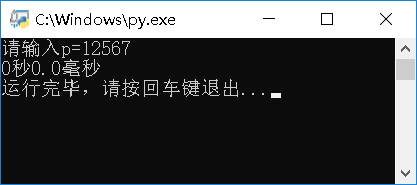
varout.set(str(d.seconds)+"秒"+str(d.microseconds/1000)+"毫秒")

程序运行效果如下：



【比一比】当然，我们也可以编写无可视化界面的程序，程序更简单。**（详见配套资源“数据安全性测试(无界面).py”）**

程序运行效果如下：



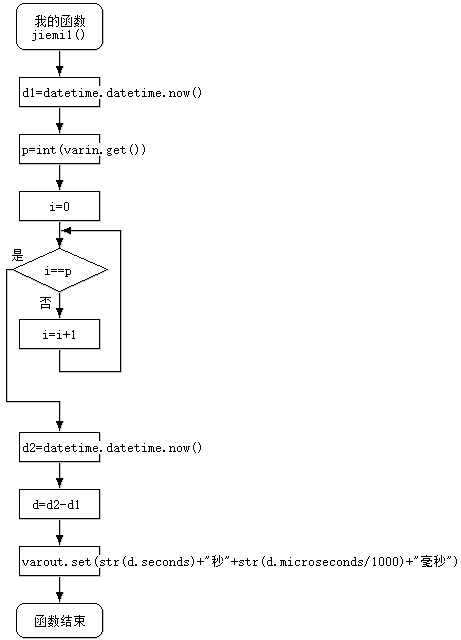
请比较有可视化界面与无可视化界面两个程序运算的快慢，将最快的结果填入表3.4.2。

【算一算】

你的电脑1秒最多能试探大约多少个整数密码？

【做一做】

1.有可视化界面程序中的jiemi()函数中的for循环改为while条件循环：

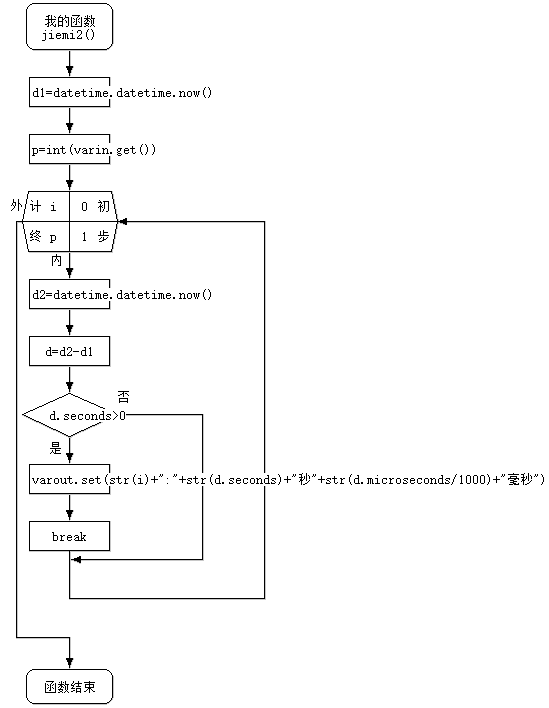


运算速度是提高了，还是降低了？

再将i=i+1改为i+=1，运算速度有无变化？

你能得出什么结论？

2.为计算1秒最多能试探多少个整数密码，某同学认为用原来的jiemi()函数要试探性多次输入p的值，结果不稳定不精确，于是他将jiemi()函数修改为jiemi2()：



结果会怎么样？为什么？

【想一想】

1.如果密码包含数字和大小写英文字母，那么5位、7位、9位密码的最大循环次数分别是多少次？估算你的电脑分别需要多少时间试探？

2.如果密码包含数字、大小写英文字母和键面特殊字符（不含空格共有32个键面特殊字符），那么5位、7位、9位密码的最大循环次数分别是多少次？估算你的电脑分别需要多少时间试探？

**三、小结**

（防字典破解）设置安全密码，不用生日、姓名及拼音缩写、手机号、简单数字组合、单词、身份证号、用户Id等等。

（防暴力破解）设置安全密码，要同时包含大小写字母、数字、特殊符号，不少于8个字符

为提高运算速度，尽可能采用计数循环for语句代替条件循环while语句，复合赋值语句（如i+=1）代替简单赋值语句(i=i+1)

**四、练习**

试编写程序，破解超级密码——可能包含数字、大小写英文字母和键面特殊字符（不含空格共有32个键面特殊字符）在内的4位字符组成的密码，每个字符的ASCII码值在33至126之间。将运行用时记入下表：

|  |  |  |
| --- | --- | --- |
| 密码值 | 破解用时 | |
| **超级密码(字符串).py** | **超级密码(列表).py** |
| !876 |  |  |
| AbY5 |  |  |
| Z$\*m |  |  |
| z)0< |  |  |
| ~?x= |  |  |

**详见配套资源“超级密码(字符串).py”和“超级密码(列表).py”，比较两种算法的快慢。**

**第二课时**

**教学重难点**

**【**教学重点**】**恺撒密码的加密与解密算法。

**【**教学难点**】**恺撒密码的加密与解密算法。

**【教学过程】**

**一、引入**

1.学生预习，阅读第79页“任务二 让数据 ‘隐身’”之“活动1 古老的‘隐身术’”，填写表3.4.3，引入“加密”话题。

2.教师检查，并评讲填写情况，引出数据安全、数据加密话题。

表3.4.3 数据加密技术的发展

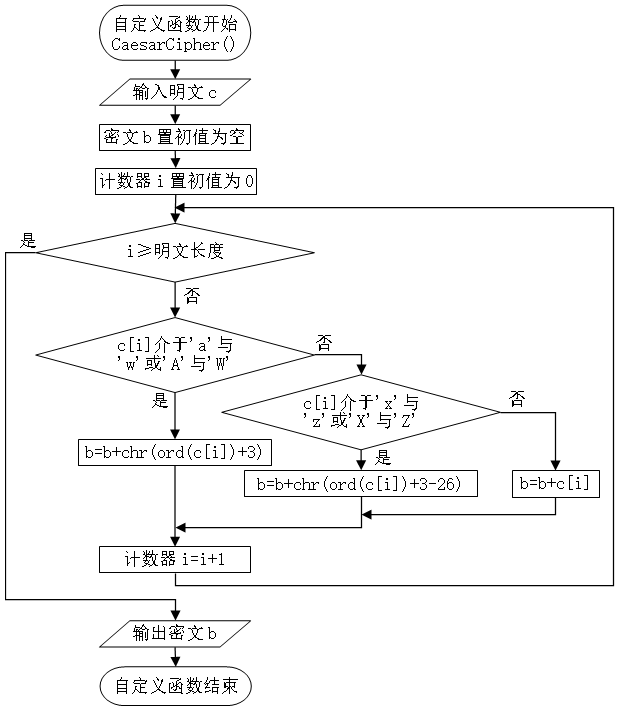
|  |  |
| --- | --- |
| 时间 | 加密方式 |
| 683年 | 拆字法。将明文中的文字进行组合生成新的字，即为密文，比如“十二月”合起来为“青” |
| 北宋 | 代码法。北宋进士曾公亮曾搜集了40个常用军事短语，然后对其进行顺序编码：一、请弓;二、请箭;三、请刀;四、请甲;五、请枪旗;六、请锅幕;七、请马;八、请衣赐;九、请粮料……四十、战小胜。军队出征前，指挥机关将用上述短语编码的密码本发给将领，并约定用一首不含重复文字的40字五言律诗与密码相对应 |
| 公元前五世纪 | 移位法。希罗多德（Herodotus）的《历史》中记载了公元前五世纪，希腊城邦和波斯帝国发生多次冲突和战争。这些战争中希腊城邦中广泛使用了移位法进行加密处理战争通讯信息，使波斯帝国难以获得希腊城邦的军事情报，也就无法提前做军事部署。希腊城邦用来传输军事信息、命令的每段文字都有固定的字数，解密者手中会有一份文字移位说明。解密者拿到密文后，根据文字移位说明进行解密，从而破解其中的军事命令或消息 |
| 古罗马时期 | 《高卢战记》有描述恺撒曾经使用密码来传递信息，即所谓的“恺撒密码”，它是一种替代密码，通过将字母按顺序推后起3位起到加密作用，如将字母A换作字母D，将字母B换作字母E。因据说恺撒是率先使用加密函的古代将领之一，因此这种加密方法被称为恺撒密码 |
| 第二次世界大战 | 密码机。在第二次世界大战期间，德国军方启用“恩尼格玛”密码机，密码学在战争中起着非常重要的作用 |

**二、加法密码**

加密就是将原始信息（数据）隐匿起来，使之在缺少特殊信息（数据）时不可读。原始信息（数据）称为明文，加密后的信息（数据）称为密文。将密文还原成明文的过程称为解密（或解码）。

恺撒密码只是简单地将明文中的每一个字母用字母表中该字母后的第3个字母替换。例如，将明文中的a用d替换，b用e替换，……，z用c替换。像恺撒密码这样，明文中的所有字母都在字母表上向后（或向前）按照一个固定数目进行偏移后被替换成密文，这种密码称为加法密码，又称为移位密码。

**（详见配套资源“**恺**撒密码加密算法流程图.doc”、“**恺**撒密码(加密有界面).py”）**



对应代码如下：

def CaesarCipher(): #Button按钮激发函数

c=mingwen.get("0.0", "end") #获取Text组件mingwen的明文内容

b=""

miwen.delete("0.0", "end") #清空Text组件miwen的内容

for i in range(len(c)): #获取明文内容的每一个字母，并加密

if 'a'<=c[i]<='w' or 'A'<=c[i]<='W': #判断a～w或A～W间的字母

b=b+chr(ord(c[i])+3) #生成密文

elif 'x'<=c[i]<='z' or 'X'<=c[i]<='Z': #判断x～z或X～Z间的字母

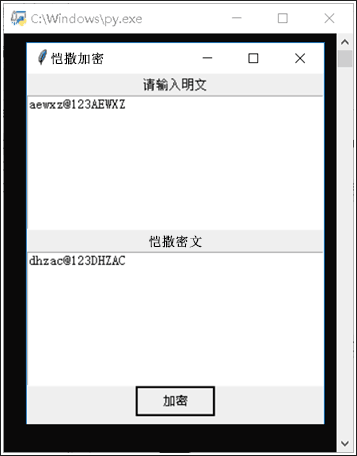
b=b+chr(ord(c[i])-23) #生成密文

else: #字母以外的明文内容不变

b=b+c[i]

miwen.insert("0.0",b) #在Text组件miwen中显示结果

程序运行效果如下：



【试一试】请写出恺撒密码的解密算法。

**（详见配套资源“恺撒密码(解密无界面).py”）**

c=input("请输入密文:")

b=""

for i in range(0,len(c)): #获取密文内容的每一个字母，并破解

if 'd'<=c[i]<='z' or 'D'<=c[i]<='Z': #判断d-z或D-Z间的字母

b=b+chr(ord(c[i])-3) #破解密文

elif 'a'<=c[i]<='c' or 'A'<=c[i]<='C': #判断a-c或A-C间的字母

b=b+chr(ord(c[i])+23) #破解密文

else:

b=b+c[i] #字母以外的密文不变

print("你的明文为:"+b)

【想一想】恺撒密码的安全性能如何？

可以说，保密性能极差。将明文字符前移或后移一个固定的长度d（称为密钥），即使改变d的值，也最多只需25次尝试d的值，就能破解。

**三、小结**

加密和解密也是属于字符编码范畴。密码学是研究编制密码和破译密码的技术科学。研究密码变化的客观规律，应用于编制密码以保守通信秘密的，称为编码学；应用于破译密码以获取通信情报的，称为破译学。二者总称密码学。

**四、练习**

编程：考虑到恺撒密码的安全性极差，应对算法进行改进。一种可行的改进方式是，建立一个明文字符与密文字符之间的一一映射表，即“密表”，如：

明文：ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文：QWERTYUIOPASDFGHJKLZXCVBNM

加密时，A→Q,B→W，…,解密时Q→A,W→B，…，小写字母对应法则也相同，即a→q,b→w,…。原先恺撒密码只有一个“密钥”，现在好比有25个“密钥”，这样保密性能大大提升，破解难度大大增加。另外，这个“密表”也是可以按需要改变的。

你能写出这种改进的恺撒密码的加密与解密算法吗？试一试。

**（详见配套资源“恺撒密码改进.py”）**