第5章 信息表示与加解密

加密是以某种特殊的算法改变原有的信息数据，使得未授权的用户即使获得了已加密的信息，但因不知解密的方法，仍然无法了解信息的内容。数据加密是为了提高信息系统和数据的安全性和保密性，防止秘密数据被外部破译而采用的主要的技术手段之一，是计算机系统对信息进行保护的一种可靠方法。

原始或未加密的数据称为“明文”。数据加密的过程就是对原来为明文的文件或数据，按某种变换方法进行处理，使其成为不可读的一段代码，来达到保护数据不被非法窃取和阅读的目的。这段不可读的代码只能在输入相应的密钥之后才能显示出本来内容，通常称这段不可读的代码为“密文”。

信息的加密、解密过程如图 5-1 所示。

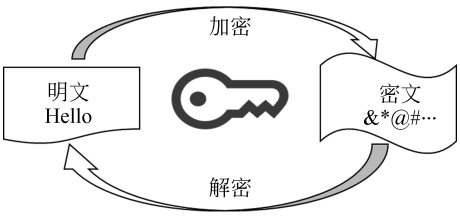

图 5-1　信息的加密、解密

简单地讲，数据加密就是将明文转换为密文的过程，加密所采用的变换方法称为加密算法。该过程的逆过程为解密，即将该加密信息转化为其原来数据的过程，解密采用的变换方法称为解密算法。其中，加密、解密的关键是依赖于密钥，通常密钥是由数字、字母或特殊符号组成的字符串。

5.1　加 密 原 理

经典加密算法有很多，按照国际上通行的惯例，可以根据双方收发的密钥标准是否相同，将加密算法划分为以下两大类。

一类是常规算法（又称私钥加密算法或对称加密算法），其特征是接收方和发送方使用相同的密钥，即加密密钥和解密密钥是相同的或等价的。常规算法主要有 DES 算法、3DES 算法、TDEA 算法、IDEA 算法以及以代换密码和转轮密码为代表的古典密码等。

另一类是公钥加密算法（又称非对称加密算法），其特征是接收方和发送方使用的密钥

互不相同，而且几乎不可能从加密密钥推导出解密密钥。公钥加密算法主要有 RSA 算法、背包算法、Elgamal、ECC(椭圆曲线加密算法)等。

本章以古典密码——凯撒密码(Caesar's code)为例，介绍一种信息的对称加密体制。

5.1.1 移位密码原理

凯撒密码(Caesar's code)也称为移位加密，作为一种最为古老的对称加密体制，在古罗马的时候就已经非常流行，是一种简单实用的对称加密方式。它的基本思想是：通过把字母移动一定的位数来实现加密和解密。加密是将明文中的所有字母都在字母表上向右(或向左)按照一个固定数目(n)进行偏移后被替换成密文。解密就是加密的逆过程。

位数 n 为移位加密和解密的密钥，n≥1，一般默认加密时向右移位 n 位，密钥为 n，向左移位 n 位，则密钥为－n。

例如，当密钥为 2 时，即对每一个字母向右时移位 2 位，即 A→C，B→D，C→E，以此类推，最后 Y→A，Z→B。简单来说就是当密钥为 n，其中一个待加密字符 ch 加密之后变为字符 ch＋n，当 ch＋n 超过 Z 时，回到 A 计数。那么，密钥为 2 的明文字母表及对应的密文字母表如下。

明文字母表：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

密文字母表：C D E F G H I J K L M N O P Q R S T U V W X Y Z A B

例如，明文为 COMPUTER，则经过密钥为 2 的加密运算后，得到的密文为 EQORWVGT；密钥为 3 时，得到的密文为 FRPSXWHU，由此可以看出，输入不同的密钥就会产生不同的加密结果。

我们知道，在计算机中所有的数据在存储和运算时都要使用二进制数表示，而具体用哪些二进制数字表示哪个符号，对于任意一个字符对象集合，不同的人都可设计自己的编码体系，但是为了减少编码体系之间转换的复杂性，提高处理效率，那么大家就必须使用相同的编码规则。例如，英文字符的 ASCII 编码、中国国家标准汉字编码和 Unicode 编码等，这些编码标准统一规定了常用的字母或符号用哪些二进制数来表示，是大家共同使用的标准编码规则。

因为移位密码适用于英文字母和一些符号的加密和解密，必须先了解如何将这些字符转换成二进制数，所以接下来介绍 ASCII 码。

5.1.2 ASCII 码

ASCII 码(美国标准信息交换代码)是由美国国家标准学会(American National Standard Institute，ANSI)制定的标准的单字节字符编码方案。它已被国际标准化组织(International Organization for Standardization，ISO)定为国际标准，称为 ISO 646 标准，适用于所有拉丁文字字母。一个 ASCII 码一般由 8 位二进制组成，实际只使用低 7 位来表示，最高位设置成恒为 0，如图 5-2 所示。

说明：

(1) 8 位 ASCII 码实际只使用低 7 位表示，字符个数为 $2^7=128$ 个，剩余的一半(128

H / L	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENG	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	↑	n	~
1111	SI	US	/	?	O	←	o	DEL

图 5-2　ASCII 码编码表

个)编码空置留作他用。在 ASCII 码能够表示的字符中,0～31 及 127(共 33 个)是控制字符或通信专用字符,如控制符 LF(换行)、CR(回车)、DEL(删除)等;32～126(共 95 个)是字符,其中 48～57 为 0～9 这 10 个阿拉伯数字,65～90 为 26 个大写英文字母,97～122 为 26 个小写英文字母。

(2) 列标题给出编码中的低 4 位,行标题给出编码的高 4 位(因最高位恒为 0),一个字符所在行列的低 4 位编码和高 4 位编码组合起来,即为该字符的编码。例如,数字符号 0 的编码为 00110000,对应十进制为 48;大写字母 A 的编码为 01000001,对应十进制为 65;小写字母 a 的编码为 01100001,对应十进制为 97。ASCII 编码表中数字的 ASCII 码与它表示的数值是完全不同的两个概念。

(3) 字符的编码依据一定规则,即 0～9＜A～Z＜a～z。数字符 0 的编码比数字符 9 的编码小,并按 0～9 的顺序递增,字母 A 的编码比字母 Z 的编码小,并按 A～Z 的顺序递增,同一个英文字母,其大写形式的编码比小写形式的编码小 32。

通过前面几章的学习可以知道,字符型数据不能进行算术运算。也就是说,在进行英文字母或字符的加解密时,不能将字符与密钥(n)直接进行运算,所以在移位加密过程中,只需根据 ACSII 码表将要加密的字符找到其对应的 ACSII 码,然后再进行移位运算。

5.1.3　转换函数

Python 语言内置函数 chr()和 ord()提供了字符及其 ASCII 码之间的转换功能。

1. chr( )函数

格式:

```
chr(<数值表达式>)
```

说明:该函数的参数是一个整数型数据(int),其数值表达式的取值范围为 0～255。该

函数用于将一个整数转换为一个字符。返回值类型为字符串类型(string)。

例如,在 Shell 中输入 chr(97),结果显示为'a'。

```
>>>chr(97)
'a'
```

2. ord( )函数

格式:

```
ord('字符串')
```

说明:该函数的参数是一个字符型数据(string)。该函数用于将字符转化成它对应的整数值。返回值类型为整数型数据类型(int)。

例如,在 Shell 中输入 ord('a'),结果显示为 97。

```
>>>ord('a')
97
```

注意:ord()函数的参数必须为字符型数据,若采用 ord()函数将 ASCII 码中的阿拉伯数字转换为对应的整数值,数字符必须带单引号或双引号。

例如,数字符'2'的 ASCII 码是 00110010,对应的十进制整数是 50,那么函数 ord('2')的计算结果为 50。

```
>>>ord('2')
50
```

5.2 字符串加解密

在进行字符串加解密学习之前,首先介绍单个字符的加解密过程。

5.2.1 单个字符加解密

【例 5-1】 通过键盘输入一个字母,实现移位为 2 的加密,并输出加密结果。

创建文件 encoding.py,代码如下。

```
pla=input("请输入要加密的字母,以回车结束:")
n=2
cip=chr(ord(pla)+n)
print(cip)
```

说明:

(1) 使用 input()函数,获取要加密的字母,并保存在一个字符串变量中。

(2) pla 是 plaintext(明文)的缩写,cip 是 ciphertext(密文)的缩写。

(3) 使用 print()函数,将加密结果输出。

运行代码,输入任意字母,运行结果如图 5-3 所示。

```
========== RESTART: C:/Users/zq/Desktop/file_reader/encoding.py ==========
请输入要加密的字母, 以回车结束:a
c
>>>
========== RESTART: C:/Users/zq/Desktop/file_reader/encoding.py ==========
请输入要加密的字母, 以回车结束:B
D
>>>
========== RESTART: C:/Users/zq/Desktop/file_reader/encoding.py ==========
请输入要加密的字母, 以回车结束:b
d
>>>
========== RESTART: C:/Users/zq/Desktop/file_reader/encoding.py ==========
请输入要加密的字母, 以回车结束:Y
[
>>>
========== RESTART: C:/Users/zq/Desktop/file_reader/encoding.py ==========
请输入要加密的字母, 以回车结束:y
{
>>>
                                                                Ln: 37  Col: 4
```

图 5-3　单个字符加密的运行结果

从运行结果可以看出,当输入的字母为 a～x 或 A～X 的任意字母时,加密结果仍为 26 个字母,但如果输入的字母是 y(Y)或者 z(Z)时,输出结果就不再是 26 个英文字母。这是因为在 ASCII 码表,65～90 为 26 个大写英文字母;97～122 为 26 个小写英文字母,加上密钥 n 的值超过了字母的 ASCII 的表示范围,加密结果就会出现其他字符。

如何使加密结果固定在大写字母或小写字母的范围内呢? 这就需要将加密后的字符的 ASCII 码值限定在 65～90(大写英文字母)或 97～122(小写英文字母)内,利用求余运算来完成。

修改文件 encoding.py,代码如下。

```
pla=input("请输入要加密的字母,以回车结束: ")
n=2
cip=chr((ord(pla)-ord('a')+n)%26+ord('a'))
print(cip)
```

运行结果如图 5-4 所示。

```
========== RESTART: C:/Users/zq/Desktop/file_reader/encoding.py ==========
请输入要加密的字母, 以回车结束:y
a
>>>
                                                                Ln: 17  Col: 4
```

图 5-4　单个字符循环加密文件的运行结果

上述的加密过程中,密钥是由题目给定,但加密程序应该允许用户设置密钥 n,用一个整数变量表示和存放。

```
n=int(input("请输入密钥: "))
pla=input("请输入要加密的字母,以回车结束: ")
cip=chr((ord(pla)-ord("a")+n)%26+ord("a"))
print(cip)
```

说明：

(1) int()函数用于数据类型转换，用于将 input()函数返回的字符串类型转化为整数类型。

(2) 若大写字母的加密结果同样限定在大写字母中，只需将第三行代码中'a'改为'A'。

运行上述代码，结果如图 5-5 所示。

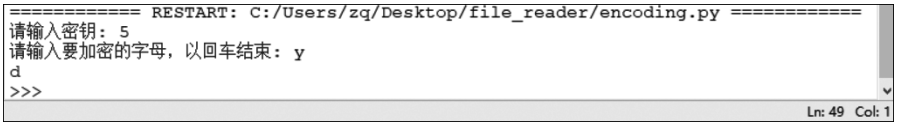

图 5-5　单个字符指定密钥循环解密的运行结果

对于移位加密来说，解密的过程就是按照相反的方向移动 n 位，请自行实现。

5.2.2　字符串加解密概述

【例 5-2】 从键盘中输入字符串 computer，实现密钥为 n 的加密，并输出加密结果。

分析：字符串是由一个个字符组成，在 5.2.1 节学习了如何将单个字符加密，那么字符串的加密需要解决以下几个问题。

1. 获取键盘输入的字符串(英文字母)

使用 input()函数，获取要加密的文字内容，并保存在一个字符串变量中。

代码示例如下。

```
pla=input("请输入要加密的文字,以回车结束:")
```

2. 实现字符串中的每个字符逐个加密

利用列表结构(list)将变量 pla 中的字母存储下来。

代码示例如下。

```
>>>n="abc"
>>>n
'abc'
>>>list=list(n)
>>>list
['a', 'b', 'c']
```

因此，只需将变量 pla 中的字符保存在列表中即可。

代码如下。

```
letter_list=list(pla.lower())
```

说明：

(1) 在对输入的字母进行加密和解密的过程，是针对每一个字符分别进行运算，所以需

要将字符串变量 pla 中保存的字符串存储到列表结构 list 中(list 是 Python 中内置的数据结构,可直接定义并使用)。

(2) pla.lower()的作用是将变量 pla 中的字符串全部转化为小写字母,方便计算。经过该语句的执行,pla 中的字符串转化为小写字母,并存储在 letter_list 这个列表中。

3. 定义对单个字符进行加密运算的函数

代码如下。

```
def move_letter (letter,n):
    if letter=='':
        return('')
    else:
        return chr((ord(letter)-ord('a')+n)%26+ord('a'))
```

说明:

(1) 自定义函数 move_letter(letter,n)有两个参数,其中 letter 参数表示要进行加密的一个字母,n 表示移位的个数,即密钥。

(2) 第二行和第三行代码是对空格进行处理,如果输入的字符串中包含空格,则不做任何处理,依然输出空格。

4. 定义一个列表,用于存储经过加密后的字符

代码如下。

```
e_list=[]
```

说明:因为尚未开始加密运算,所以一开始该列表为空。

利用 append()函数将加密后的字符加入 e_list 列表中。代码如下。

```
e_list.append()
```

5. 对列表中的每个字母进行加密并组合成新的字符串

使用循环结构,遍历 letter_list 中的每一个字母进行加密运算。利用 letter_list 存储需要加密的字符串,由于要对 letter_list 中的字符进行逐个移位运算,需要使用循环结构对 letter_list 进行遍历。

逐个加密后还需要将列表中的单个字符重新组合成字符串,代码如下。

```
e_letter=''.join(e_list)
```

至此,例 5-2 的问题都已解决,能够实现字符串加密并输出。

例 5-2 实现字符串加密并输出的完整代码如下。

```
def move_letter (letter,n):
    if letter=='':
        return('')
    else:
```

```
        return chr((ord(letter)-ord('a')+n)%26+ord('a'))
n=int(input('请输入密钥:'))
pla=input('请输入要加密的字符串,以回车结束:')
letter_list=list(pla.lower())
e_list=[]
for x in letter_list:
    c=move_letter(x,n)
    e_list.append(c)
e_letter=''.join(e_list)
print("加密后字符串:",e_letter)
```

说明：

(1) join()用于将序列中的元素以指定的字符连接生成一个新的字符串。其语法为

```
str.join(seq)
```

其中，str 是分隔符，可以为空；参数 seq 是要连接的元素序列。

(2) e_letter=''.join(e_list)代码是将 e_list 中的所有元素合并成一个新的字符串，元素之间无分隔符。

运行结果如图 5-6 所示。

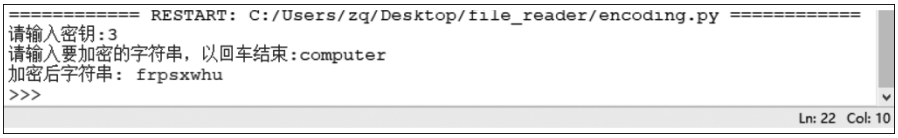

图 5-6　字符串加密输出结果

对于移位加密来说，解密的过程就是按照相反的方向移动 n 位，所以要调用 move_letter()函数，只要修改参数即可。

解密过程代码如下。

```
for x in e_list:
    c=move_letter(x,-n)
    d_list.append(c)
d_letter=''.join(d_list)
print("解密后字符串:" d_letter)
```

运行结果如图 5-7 所示。

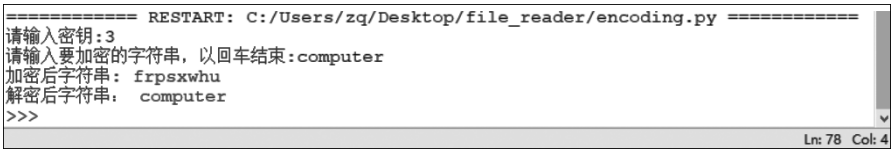

图 5-7　字符串解密输出结果

5.3 文件加解密

至此，已经学完了字符串的加密与解密。接下来，将如何加密保存在文件中的字符串数据并将加密结果保存到新的文件呢？本节将学习如何进行文件的加密与解密。

5.3.1 从文件中读取数据

要使用文本文件中的信息，首先需要将信息读取到内存中。为此，可以一次性地读取文件的全部内容，也可以用每次读取一行的方式逐行读取文件的内容。

1. 读取整个文件

要读取文件，需要一个包含几行文本的文件。下面首先来创建一个文件，创建记事本文件 file.txt 如图 5-8 所示。

图 5-8 文件 file.txt

下面的程序文件 file_reader.py 可以实现打开并读取 file.txt 文件，再将其内容显示在屏幕上，具体代码如下。

```
import os
f=open("file.txt")
contents=f.read()
print(contents)
f.close()
```

说明：

(1) os 模块：使用 import 指令，导入 os 模块。该模块中包含了对文件进行操作(如打开文件、关闭文件、读文件和写文件等)的一些函数，方便使用。

(2) open()函数：要以任何方式使用文件，哪怕是仅仅打印其内容，都要先打开文件。open()就是一个打开文件的函数，该函数接收一个参数即要打开的文件的名称。

这里需要注意，Python 在当前执行的程序所在的目录中查找指定的文件。

当前运行的文件是 file_reader.py，因此 Python 在 file_reader.py 所在的目录中查找文件 file.txt。

但有时可能要打开的文件不在程序文件所属的目录下。这时需要给 Python 提供文件在计算机中的准确位置(绝对文件路径)，这样就不用关心当前运行的程序存储在什么地

方了。

例如，若将文件 file.txt 保存在绝对路径 D:\qycache\file.txt 下，只需将 open()函数的参数改为该绝对路径即可。修改 file_reader.py，代码如下。

```
import os
f=open("D:\qycache\file.txt")
contents=f.read()
print(contents)
f.close()
```

运行程序，结果如图 5-9 所示。

```
========== RESTART: C:\Users\zq\Desktop\file_reader\file_reader.py ==========
Traceback (most recent call last):
  File "C:\Users\zq\Desktop\file_reader\file_reader.py", line 2, in <module>
    f=open("D:\qycache\file.txt")
OSError: [Errno 22] Invalid argument: 'D:\\qycache\x0cile.txt'
>>>
                                                                    Ln: 9  Col: 4
```

图 5-9　文件路径运行报错

说明：运行程序报错。这是因为 Windows 系统在文件路径使用反斜杠(\)而不是斜杠(/)，由于在 Python 中反斜杠(\)视为转义标记，为在 Windows 系统中确保万无一失，应采用原始字符串的方式指定路径，即在开头的单引号前加上 r，例如：

```
f=open(r"D:\qycache\file.txt")
```

(3) 在这个程序中，同时调用了 open()和 close()来打开和关闭文件，但这样做时，如果程序存在 bug，导致 close()语句未执行，那么文件将不会关闭，可能导致因未妥善关闭的文件造成数据丢失或文件受损。况且如果在程序中过早地调用 close()，可能导致需要使用文件时它已关闭(无法访问)，这会导致更多的错误。所以，为保证文件在恰当的时机关闭，可以使用关键字 with，修改 file_reader.py，代码如下。

```
import os
with open(r"D:\qycache\file.txt") as f:
    contents=f.read()
    print(contents)
```

关键字 with 在不再需要访问文件时将其关闭，即让 Python 去确定文件关闭的时机，也就是只需打开文件，并在需要时使用它，Python 会在合适的时候自动关闭文件。

(4) as f 是为 open()返回的 file.txt 文件对象取一个别名 f。

(5) 有了表示 file.txt 的文件对象后，使用方法 read()读取这个文件的全部内容，并将其作为一个长字符串存储在变量 contents 中。这样，通过打印 contents 的值，就可将这个文本文件的全部内容显示出来。

运行程序，结果如图 5-10 所示。

查看运行结果，发现多出一行空行，这是因为在 read()到达文件末尾时返回一个空字符串，而将这个空字符串显示出来时就是一个空行。要删除末尾的空行，可在 print()语句中使用 rstrip()，修改文件 file_reader.py，代码如下。