d西安交通大学“大学计算机III”实验指导书

实验5 十进制转二进制

## 一、实验目的

1. 理解数的二进制表示；
2. 理解数和字符的区别；
3. 掌握数的转换方法；
4. 掌握Python有关编码的函数的使用。

## 二、实验环境

硬件环境：PC机。

软件：Windows操作系统，Python3.7IDLE

## 三、实验内容

1.输入一个整数，使用Python的内置函数将它转换为二进制、八进制和十六进制数，显示出来。例如：

输入：10

输出：

二进制：1010

八进制：12

十六进制：a

**提示**：本题可以使用Python的函数转换，如bin(),oct(),hex()，可以参考后面的指导。

2.显示ASCII值为33到126的ASCII字符表。结果为：

33 !

34 "

35 #

36 $

37 %

……

3.查看汉字的编码。使用Python字符串的方法encode()看看自己的名字，在计算机中是怎样表示的，利用网上的在线编码查看工具，观察Python给出的编码是否正确。。

>>> '李白'.encode('gb2312') #GB2312编码，2字节

b'\xc0\xee\xb0\xd7'

>>> '李白'.encode('gbk') #GBK编码

b'\xc0\xee\xb0\xd7'

>>> '李白'.encode('gb18030') #GB18030编码

b'\xc0\xee\xb0\xd7'

>>> '李白'.encode('utf-8') #UTF-8编码，3字节

b'\xe6\x9d\x8e\xe7\x99\xbd'

>>> '李白'.encode('utf-16be') #UTF-16 Big-Endian （字节的顺序，字节序）

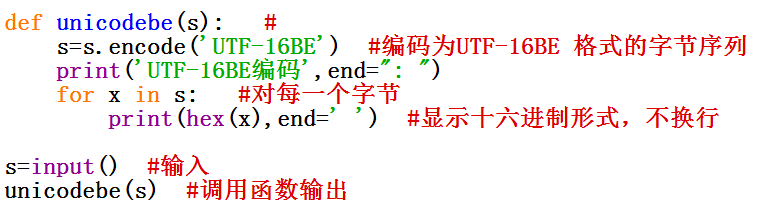
b'gNv}'

>>> '李白'.encode('utf-16le') # UTF-16 Little-Endian

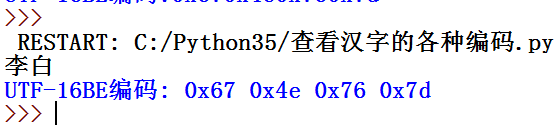
b'Ng}v'

>>>

显示以十六进制方式显示UTF-16编码



运行结果：



关于字节序的概念见实验指导部分。

4.输入一句话将其中的小写字母转换为大写字母。例如：

输入：hello world

输出：HELLO WORLD

**注意**：不能使用内置函数、字符串的方法直接转换。要通过ASCII值的变换。

1. 编写函数，将十进制整数，将其转换为二进制形式（用字符串表示），返回二进制形式的字符串。

编写主程序，输入整数，调用函数转换为二进制（字符串形式），显示二进制数。

例如，

输入：12

输出：1010

**注意，**本题不能用Python的转换函数直接转换，要自己通过除2取余运算转换。

6.编写函数，将十进制小数转换为二进制小数，返回二进制小数的字符串形式，最多保留小数点后8位。

编写主程序，输入小数，转换为二进制，显示。例如：

输入：0.625

输入：0.101

输入：.8125

输出：0.1101

注意，需要使用乘2取整的方法。

7.整合上述4、5题，写出十进制实数转二进制实数的程序。如

输入：12.625

输出：1010.101

## 四、实验要求

1.变量、函数命名要规范。

2.程序中要有适当的注释。

3.使用实验报告模板编写实验报告。

4.实验报告的文件名为:lab05\_+学号+姓名.doc, 如 lab05\_20190011001张乐乐.doc

5.要求提供源程序的文字，粘贴到实验报告中，对运行结果截图（同样只保留有用部分）。

## 五、实验指导

1. 第1题，输入一个整数，使用Python的内置函数将它转换为二进制、八进制和十六进制数，显示出来。

本题可以使用Python的函数转换

bin()转二进制， oct()转八进制， hex()转十六进制，返回字符串。如：

>>> bin(10)

'0b1010'

>>> oct(10)

'0o12'

>>> hex(10)

'0xa'

>>>

注意，结果不能显示数制符号，如二进制的0b,八进制的0o

2.第2题，显示ASCII值为33到126的ASCII字符表。

字符，直接显示；ASCII值使用ord()获得。再使用一个循环就能显示指定区间内的ASCII字符表。

3.第4题，输入一句话将其中的小写字母转换为大写字母。

使用循环逐个将小写字母转换为大写字母，利用字符串的连接运算，将结果连接成一个新的字符串。

注意：不能使用内置函数、字符串的方法直接转换。要通过ASCII值的变换。

4.第5题，编写函数，将十进制整数，将其转换为二进制形式（用字符串表示），返回二进制形式的字符串。

使用除2取余法转换。将每个二进制数字转换为字符型，然后连接在一起。

s=c+s

使得新加的字符在前面。

5.字节序

字节序是指多字节数据在计算机内存中存储或者网络传输时各字节的存储顺序。

PowerPC系列CPU采用big endian方式存储数据，而x86系列CPU则采用little endian方式存储数据。

1）Little-endian：将低序字节存储在起始地址（低位编址）（地址低位存储值的低位，地址高位存储值的高位）

2）Big-endian：将高序字节存储在起始地址（高位编址）（地址低位存储值的高位，地址高位存储值的低位）

一个数，0x1234abcd，右边的字节是低位字节，左边的数位为高位字节。例如：

如果我们将0x1234abcd写入到以0x0000开始的内存中，则结果为；

|  |  |  |  |
| --- | --- | --- | --- |
|  | address | big-endian  PowerPC系列 | little-endian  x86系列 |
| 低地址 | 0x0000 | 0x12 | 0xcd |
|  | 0x0001 | 0x34 | 0xab |
|  | 0x0002 | 0xab | 0x34 |
| 高地址 | 0x0003 | 0xcd | 0x12 |