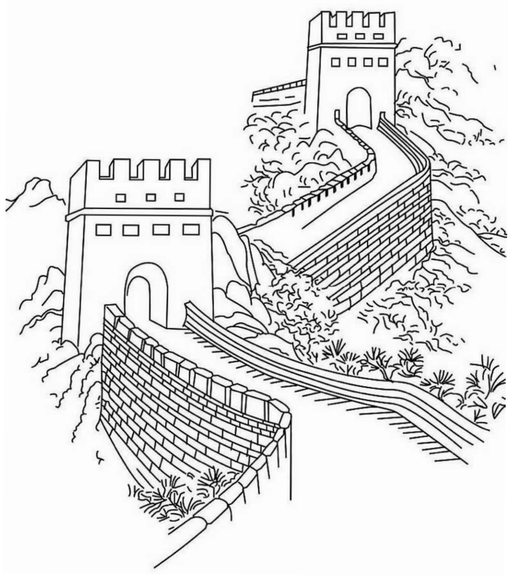
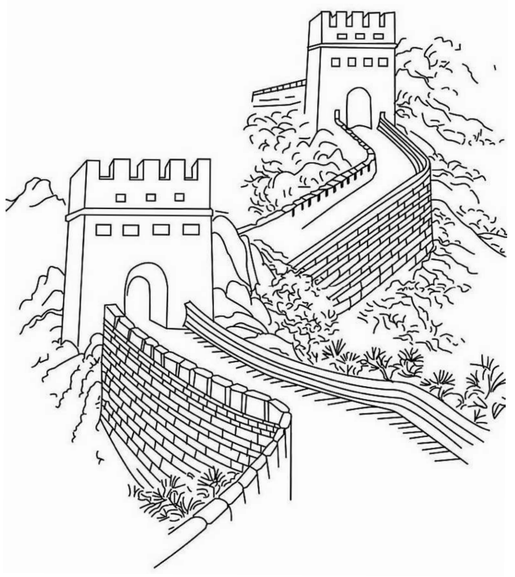
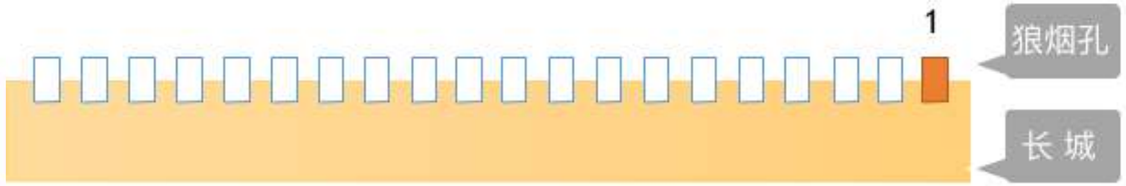
第2章
二进制运算、字符编码、数据类型


古时，人们如何通信....



如何准确的告诉友军，来了多少敌人？





二进制与十进制转换

> 刚才我们已经发现，二进制的第n位代表的十进制值都刚好遵循着2的n次方这个规律

十进制如何转二进制呢?

填位大法:

先把每个2进制位代表的值依次写出来，然后再根据10进制的值把数填到相应位置，就好了～～～

	128	64	32	16	8	4	2	1
20				1	0	1	0	0
200	1	1	0	0	1	0	0	0

ASCII（American Standard Code for Information Interchange，美国信息交换标准代码）

十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符
0		16	▶	32		48	0	64	@	80	P	96	`	112	p	128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
1	☺	17	◀	33	!	49	1	65	A	81	Q	97	a	113	q	129	ü	145	æ	161	í	177	▓	193	┴	209	╤	225	ß	241	±
2	☻	18	↕	34	"	50	2	66	B	82	R	98	b	114	r	130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
3	♥	19	‼	35	#	51	3	67	C	83	S	99	c	115	s	131	â	147	ô	163	ú	179	▓	195	├	211	╙	227	π	243	≤
4	♦	20	¶	36	$	52	4	68	D	84	T	100	d	116	t	132	ä	148	ö	164	ñ	180	▓	196	─	212	╘	228	Σ	244	⌠
5	♣	21	§	37	%	53	5	69	E	85	U	101	e	117	u	133	à	149	ò	165	Ñ	181	▓	197	┼	213	╒	229	σ	245	⌡
6	♠	22	▬	38	&	54	6	70	F	86	V	102	f	118	v	134	å	150	û	166	ª	182	▓	198	╞	214	╓	230	µ	246	÷
7	•	23	↨	39	'	55	7	71	G	87	W	103	g	119	w	135	ç	151	ù	167	º	183	▓	199	╟	215	╫	231	τ	247	≈
8	◘	24	↑	40	(	56	8	72	H	88	X	104	h	120	x	136	ê	152	ÿ	168	¿	184	▓	200	╚	216	╪	232	Φ	248	°
9	○	25	↓	41	)	57	9	73	I	89	Y	105	i	121	y	137	ë	153	Ö	169	⌐	185	▓	201	╔	217	┘	233	Θ	249	•
10	◙	26	→	42	*	58	:	74	J	90	Z	106	j	122	z	138	è	154	Ü	170	¬	186	▓	202	╩	218	┌	234	Ω	250	·
11	♂	27	←	43	+	59	;	75	K	91	[	107	k	123	{	139	ï	155	¢	171	½	187	▓	203	╦	219	▓	235	δ	251	√
12	♀	28	∟	44	,	60	<	76	L	92	\	108	l	124	|	140	î	156	£	172	¼	188	▓	204	╠	220	▄	236	∞	252	n
13	♪	29	↔	45	-	61	=	77	M	93	]	109	m	125	}	141	ì	157	¥	173	¡	189	▓	205	═	221	▌	237	φ	253	²
14	♫	30	▲	46	.	62	>	78	N	94	^	110	n	126	~	142	Ä	158	Pts	174	«	190	▓	206	╬	222	▐	238	∈	254	■
15	☼	31	▼	47	/	63	?	79	O	95	_	111	o	127	⌂	143	Å	159	ƒ	175	»	191	▓	207	╧	223	▀	239	∩	255	ÿ

文字转2进制

请把 #Alex　按ASCII表转成二制式形式

	128	64	32	16	8	4	2	1
# 51	0	0	1	1	0	0	1	1
A 65	0	1	0	0	0	0	0	1
l 108	0	1	1	0	1	1	0	0
e 101	0	1	1	0	0	1	0	1
x 120	0	1	1	1	1	0	0	0

计算机如何分清哪段是代表#，哪段是代码A呢?

```
#Alex  110011  1000001  1101100  1100101  1111000
```

论断句的重要性与必要性：

上次在网上看到个新闻，讲是个小偷在上海被捕时高喊道：“我一定要当上海贼王！”

计算机容量单位横空出世

正是由于这些字符串长的长，短的短，写在一起让我们难以分清每一个字符的起止位置，所以聪明的人类就想出了一个解决办法，既然一共就这255个字符，那最长的也不过是11111111八位，不如我们就把所有的二进制都转换成8位的，不足的用0来替换。

```
#Alex  110011  1000001  1101100  1100101  1111000
```

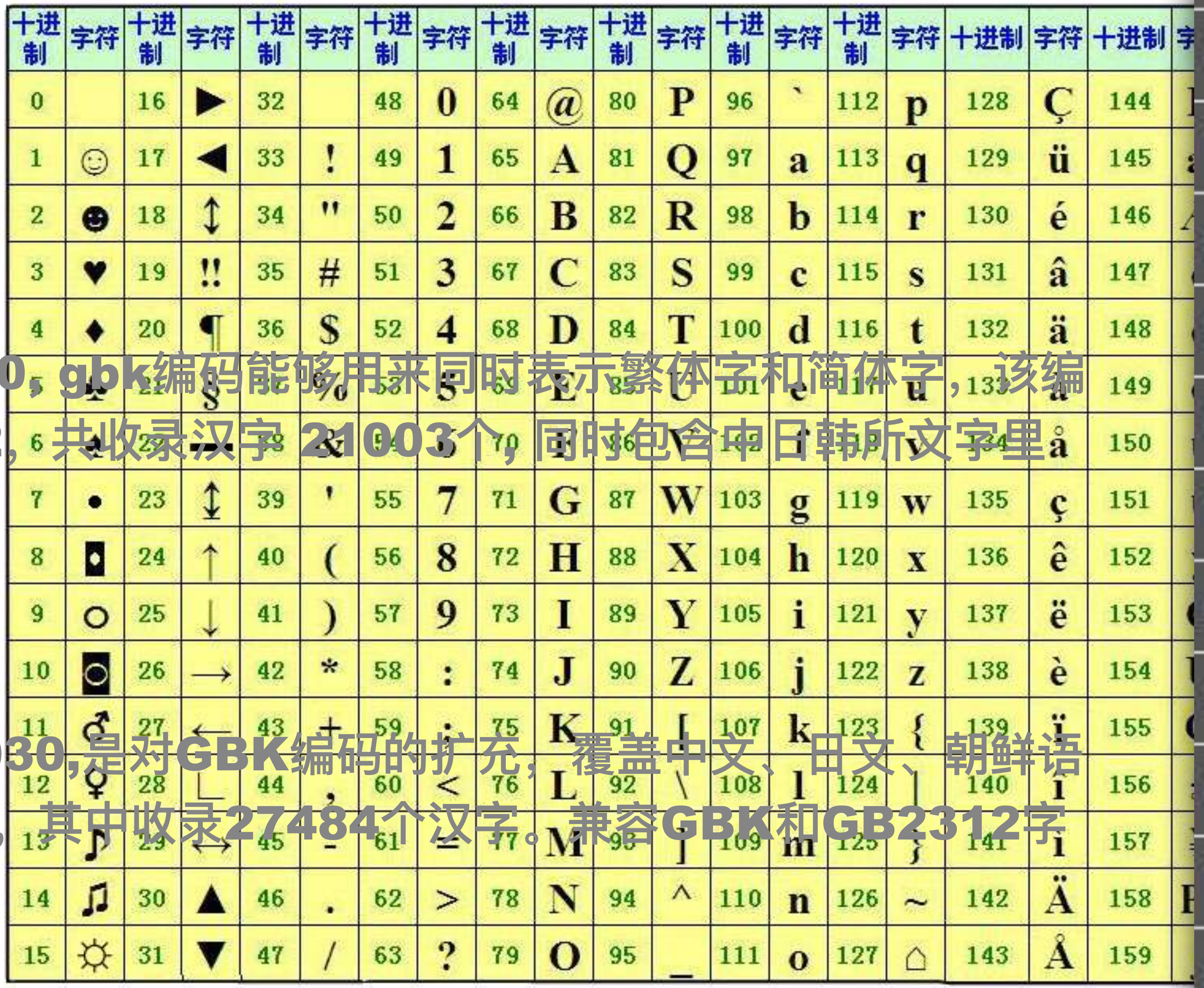
```
#Alex  00110011  01000001  01101100  01100101  01111000
```

每一位0或者1所占的空间单位为bit(比特)，这是计算机中最小的表示单位

8bit = 1bytes 字节，最小的存储单位，1bytes缩写为1B
1KB=1024B
1MB=1024KB
1GB=1024MB
1TB=1024GB
1PB=1024TB
1EB=1024PB
1ZB=1024EB
1YB=1024ZB
1BB=1024YB

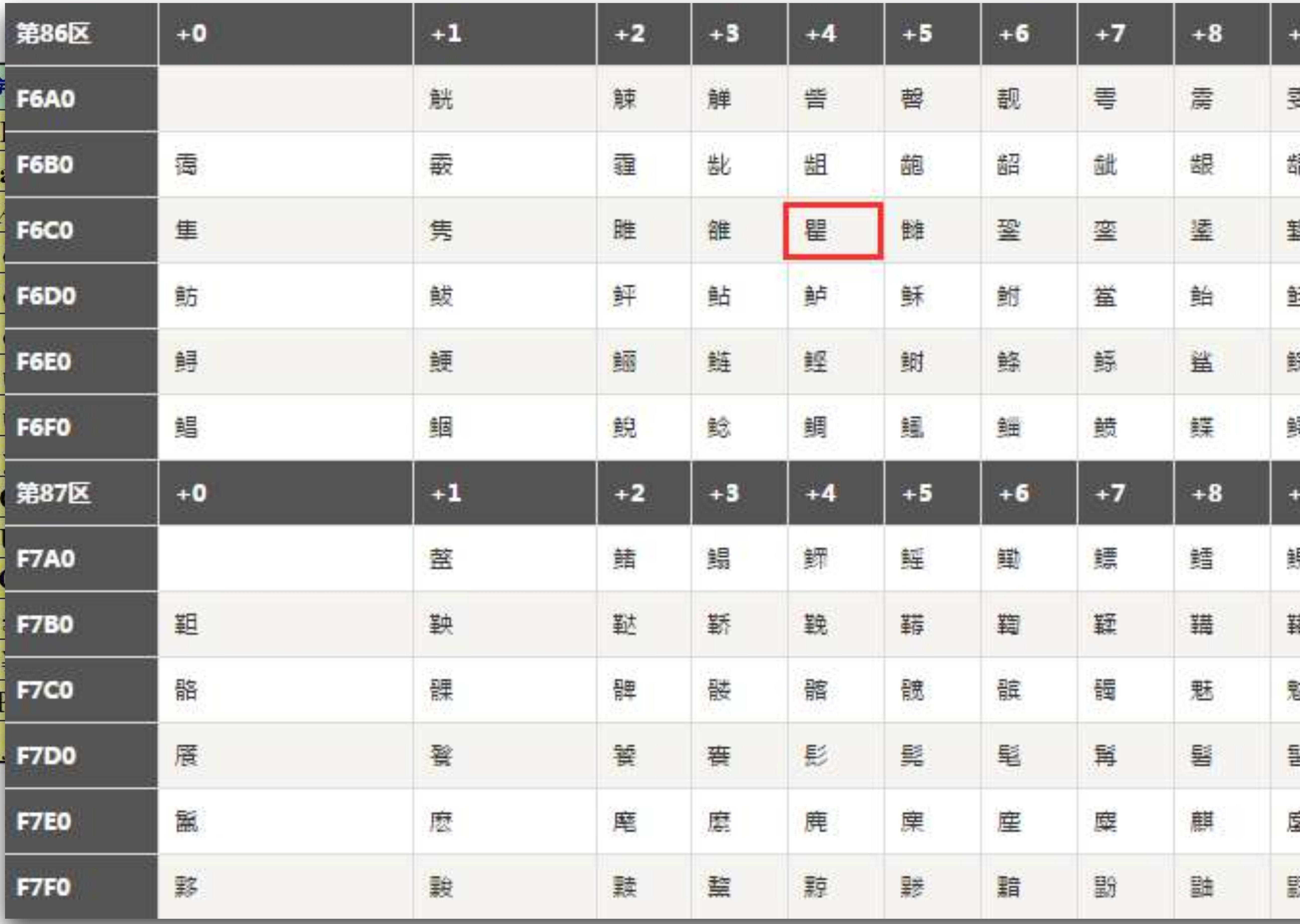
字符编码

GB2312又称国标码，由国家标准总局发布，1981年5月1日实施，通行于大陆。新加坡等地·······6763个。



1995年发布GBK1.0，gbk编码能够用来同时表示繁体字和简体字，该编码标准兼容GB2312，共收录汉字 21003个，同时包含中日韩所有文字里所有汉字

2000年发布GB18030，是对GBK编码的扩充，覆盖中文、日文、朝鲜语和中国少数民族文字，其中收录27484个汉字，兼容GBK和GB2312字符集



BIG5编码：台湾地区繁体中文标准字符集，采用双字节编码，共收录13053个中文字，1984年实施。

字符编码

为解决每个国家不同编码间不互通的问题，ISO标准组织出马了！

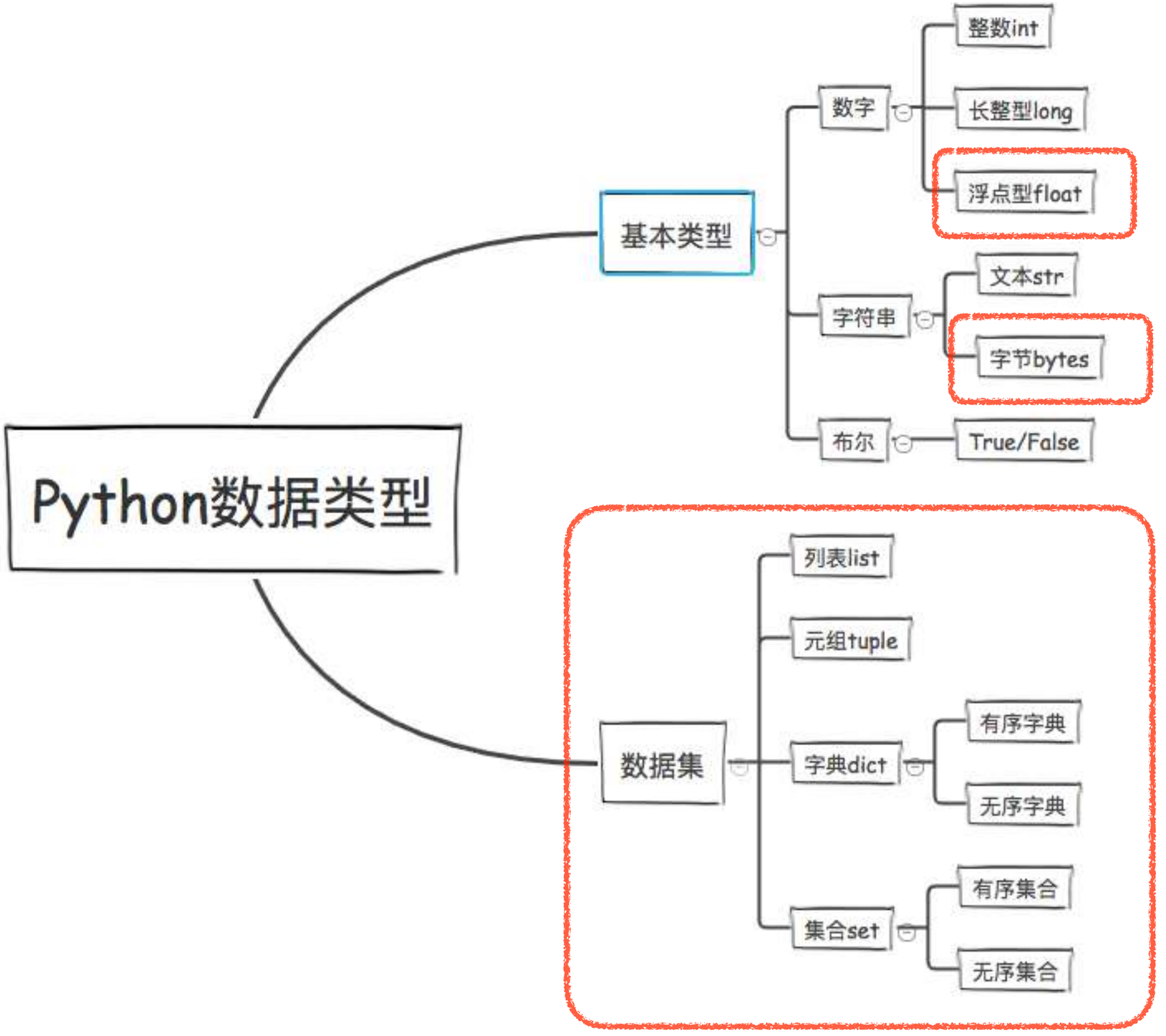
Unicode编码：国际标准字符集，它将世界各种语言的每个字符定义一个唯一的编码，以满足跨语言、跨平台的文本信息转换。**Unicode**（统一码、万国码）规定所有的字符和符号最少由 **16** 位来表示（**2**个字节），即：**2 **16 = 65536**，

UTF-8，是对**Unicode**编码的压缩和优化，他不再使用最少使用**2**个字节，而是将所有的字符和符号进行分类：**ascii**码中的内容用**1**个字节保存、欧洲的字符用**2**个字节保存，东亚的字符用**3**个字节保存**...**

windows系统中文版默认编码是**GBK**
Mac OS \ Linux 系统默认编码是 **UTF-8**

Python 2.x 默认编码是ASCII

Python 3.x 默认编码是UTF-8


Python数据类型
基本类型
数字
整数int
长整型long
浮点型float
字符串
文本str
字节bytes
布尔
True/False
数据集
列表list
元组tuple
字典dict
有序字典
无序字典
集合set
有序集合
无序集合

浮点数

浮点数是属于**有理数**中某特定子集的数的数字表示，在计算机中用以近似表示任意某个**实数**。具体的说，这个实数由一个整数或定点数（即尾数）乘以某个基数的整数次幂得到(10**4, 10为基数)，这种表示方法类似于基数为10的**科学计数法**。

有理数

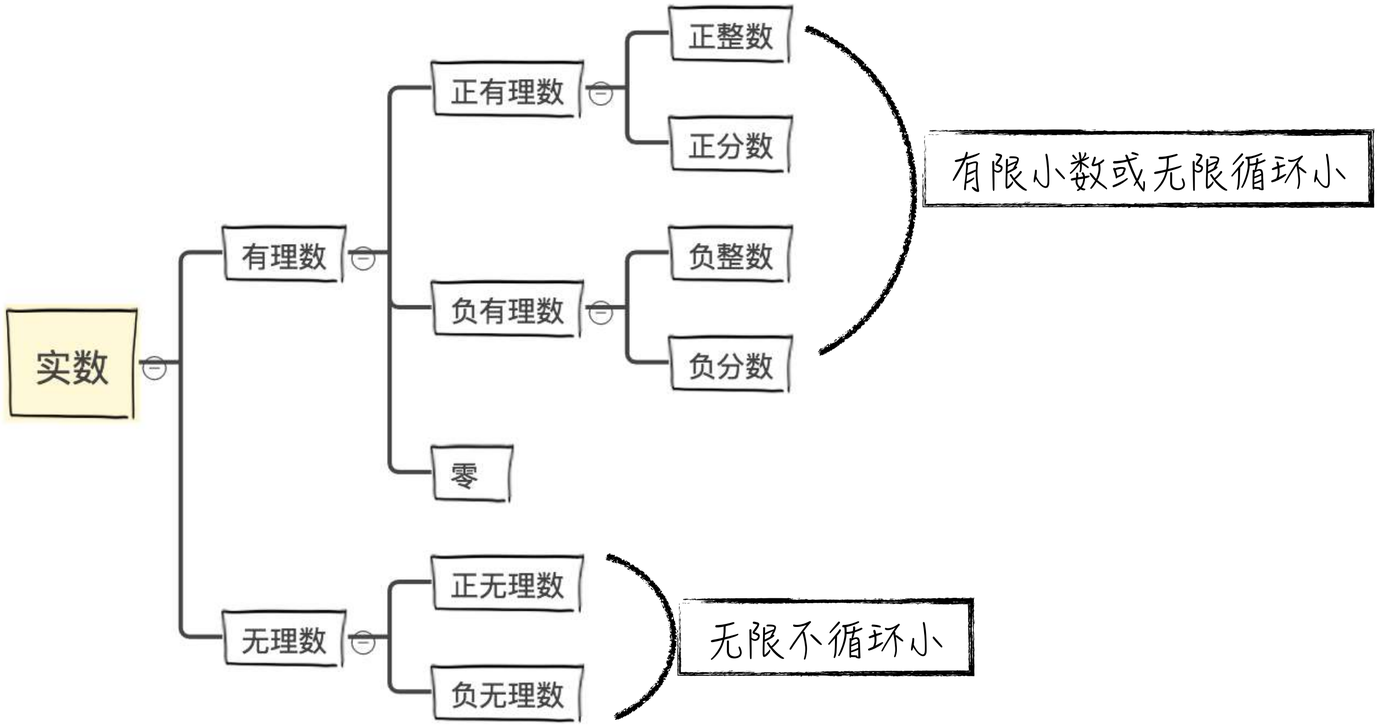
数学上，有理数是一个整数a和一个非零整数b的比，例如3/8，通则为a/b，又称作分数,0也是有理数,有理数是整数和分数的集合，整数也可看做是分母为一的分数。
有理数的小数部分是有限或为无限循环的数。

无理数

无理数，也称为无限不循环小数，不能写作两整数之比。若将它写成小数形式，小数点之后的数字有无限多个，并且不会循环。 常见的无理数有非完全平方数的平方根、圆周率（Pi）和e

实数

实数，是有理数和无理数的总称



浮点数

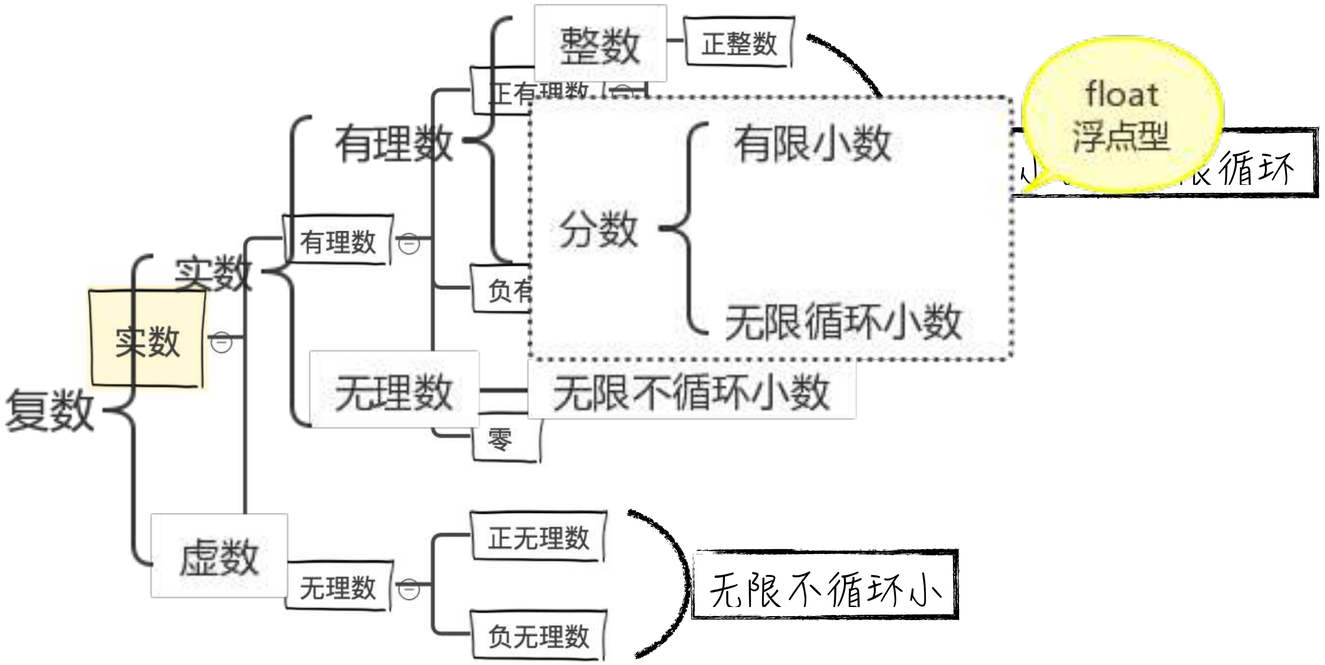
浮点数是属于**有理数**中某特定子集的数的数字表示，在计算机中用以近似表示任意某个**实数**。具体的说，这个实数由一个整数或定点数（即尾数）乘以某个基数的整数次幂得到(10**4, 10为基数)，这种表示方法类似于基数为10的**科学计数法**。

科学记数法(scientific notation)

科学记数法是指把一个数表示成a×10的n次幂的形式（1≤a<10，n 为正整数。）。例如199714000000000=1.99714x10^13。计算器或电脑表达10的的幂是一般是用E或e，也就是1.99714E13=199714000000000。 用幂的形式，有时可以方便的表示日常生活中遇到的一些较大的数。

复数

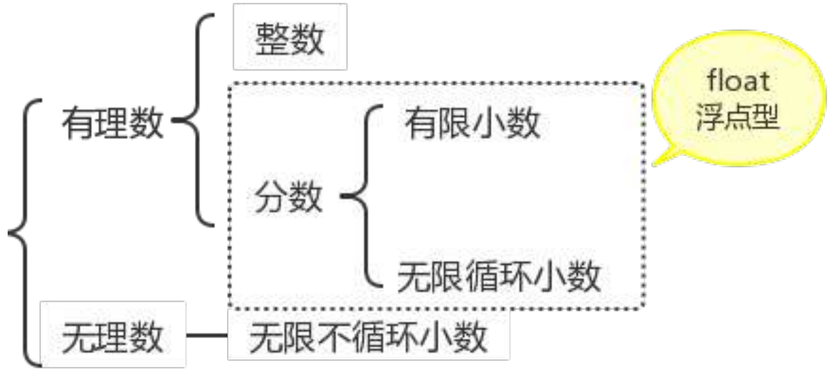
复数是指能写成如下形式的数a+bi,这里a和b是实数，i是虚数单位(即-1开根)。在复数a+bi中，a称为复数的实部，b称为复数的虚部，i称为虚数单位。当虚部等于零时，这个复数就是实数；当虚部不等于零时，这个复数称为虚数
(-5+4j)和(2.3-4.6j)是复数的例子，其中-5,4为实数，j为虚数，数学中表示复数是什么？。



浮点精确度问题

整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的而浮点数运算则可能会有四舍五入的误差。

Python默认的是17位精度,也就是小数点后16位，尽管有16位，但是这个精确度却是越往后越不准的。这个问题不是只存在在python中，其他语言也有同样的问题。原因与浮点数存储结构有关。



计算高精确度的浮点数方法

```
#借助decimal模块的"getcontext"和"Decimal" 方法
>>> a = 3.14159265351365105460831782833 2
>>> a
3.141592653513651
>>> from decimal import *
>>> getcontext()
Context(prec=50, rounding=ROUND_HALF_EVEN, Emin=-999999, Emax=999999, capitals=1, clamp=0, flags=[FloatOperati
>>> getcontext().prec = 50
>>> a = Decimal(1) / Decimal(3)#注，在分数计算中结果正确，如果直接定义超长精度小数会不准确
>>> a
Decimal('0.33333333333333333333333333333333333333333333333333')

>>> a = '3.14159265351365105460831782833 2'
>>> Decimal(a)
Decimal('3.14159265351365105460831782833 2')
```

```
#不推荐：字符串格式化方式，可以显示，但是计算和直接定义都不准确，后面的数字没有意义。
>>> a = ("%.30f" % (1.0/3))
>>> a
'0.333333333333333314829616256247'
```

数据类型-列表

如何通过一个变量存储公司所有员工的名字?

列表是一个数据的集合，集合内可以放任何数据类型，可对集合进行方便的增删改查操作

```
L1 = [] #定义空列表
L2 = ['a','b','c','d'] #存4个值，索引为0-3
L3 = ['abc',['def','ghi']]  #嵌套列表
```

列表的功能:

- 创建
- 查询
- 切片
- 增加
- 修改
- 删除
- 循环
- 排序

数据类型-列表

列表的功能：

- **创建**
- 查询
- 切片
- 增加
- 修改
- 删除
- 循环
- 排序

```
#方法一
L1 = [] #定义空列表

L2 = ['a','b','c','d'] #存4个值，索引为0-3

L3 = ['abc',['def','ghi']]  #嵌套列表


#方法二

L4 = list()
print(L4)
```

```
列表
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6
[]

Process finished with exit code 0
```

数据类型-列表

列表的功能：

- 创建
- **查询**
- 切片
- 增加
- 修改
- 删除
- 循环
- 排序

```
>>> L2 = ['a','b','c','d','a','e',1,2]
>>>
>>> L2[2] #通过索引取值
'c'
>>> L2[-1] #通过索引从列表右边开始取值
2
>>> L2[-2]
1
>>> L2.index('a')  #返回指定元素的索引值，从左右查找，找到第一个匹配值 则返回
0
>>> L2.count('a')  #统计指定元素的个数
2
>>>
```

数据类型-列表

列表的功能：

- 创建
- 查询
- **切片**
- 增加
- 修改
- 删除
- 循环
- 排序

```
>>> L2
['a', 'b', 'c', 'd', 'a', 'e', 1, 2]
>>>
>>> L2[0:3] #返回从索引0至3的元素，不包括3，顾头不顾尾
['a', 'b', 'c']
>>> L2[0:-1] #返回从索引0至最后一个值 ，不包括最后一个值
['a', 'b', 'c', 'd', 'a', 'e', 1]
>>> L2[3:6] #返回从索引3至6的元素
['d', 'a', 'e']
>>> L2[3:] #返回从索引3至最后所有的值
['d', 'a', 'e', 1, 2]
>>> L2[:3] #返回从索引0至3的值
['a', 'b', 'c']
>>> L2[1:6:2] #返回索引1至6的值，但是步长为2(每隔一个元素，取一个值)
['b', 'd', 'e']
>>>
>>> L2[:] #返回所有的值
['a', 'b', 'c', 'd', 'a', 'e', 1, 2]
>>> L2[::2] #按步长为2，返回所有的值
['a', 'c', 'a', 1]
```

数据类型-列表

列表的功能：

- 创建
- 查询
- 切片
- **增加**
- **修改**
- 删除
- 循环
- 排序

```
>>> L2
['a', 'b', 'c', 'd', 'a', 'e', 1, 2]
>>> L2.append('A') #列表最后面追加A
>>> L2
['a', 'b', 'c', 'd', 'a', 'e', 1, 2, 'A']
>>> L2.insert(3,'B') #在列表的索引为3的位置 插入 一个值 B
>>> L2
['a', 'b', 'c', 'B', 'd', 'a', 'e', 1, 2, 'A']
>>>
```

```
>>> L2
['a', 'b', 'c', 'B', 'd', 'a', 'e', 1, 2, 'A']
>>> L2[3] = 'Boy' #把索引3的元素修改为Boy
>>> L2
['a', 'b', 'c', 'Boy', 'd', 'a', 'e', 1, 2, 'A']
>>>
>>> L2[4:6] = 'ALEX LI' #把索引4-6的元素改为ALEX LI, 不够的元素自动增加
>>> L2
['a', 'b', 'c', 'Boy', 'A', 'L', 'E', 'X', ' ', 'L', 'I', 'e', 1, 2, 'A']
```

数据类型-列表

列表的功能：

- 创建
- 查询
- 切片
- 增加
- 修改
- **删除**
- 循环
- 排序

```
>>> L2
['a', 'b', 'c', 'Boy', 'A', 'L', 'E', 'X', ' ', 'L', 'I', 'e', 1, 2, 'A']
>>>
>>> L2.pop() #删除最后一个元素
'A'
>>> L2
['a', 'b', 'c', 'Boy', 'A', 'L', 'E', 'X', ' ', 'L', 'I', 'e', 1, 2]
>>> L2.remove('L') #删除从左找到的第一个指定元素
>>> L2
['a', 'b', 'c', 'Boy', 'A', 'E', 'X', ' ', 'L', 'I', 'e', 1, 2]
>>> del L2[4] #用python全局的删除方法删除 指定元素
>>> L2
['a', 'b', 'c', 'Boy', 'E', 'X', ' ', 'L', 'I', 'e', 1, 2]
>>> del L2[3:7] #删除多个元素
>>> L2
['a', 'b', 'c', 'L', 'I', 'e', 1, 2]
>>>
```

数据类型-列表

列表的功能：

- 创建
- 查询
- 切片
- 增加
- 修改
- 删除
- **循环**
- 排序

```
L2 = ['a', 'b', 'c', 'L', 'I', 'e', 1, 2]

for i in L2:
    print(i)
```

表

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/alex/Docum
a
b
c
L
I
e
1
2

Process finished with exit code 0
```

数据类型-列表

列表的功能：

- 创建
- 查询
- 切片
- 增加
- 修改
- 删除
- 循环
- **排序**

```
>>> L2
['a', 'b', 'c', 'L', 'I', 'e', 1, 2]

>>> L2.sort()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module> #不能对包含了str和int的列表进行排序
TypeError: '<' not supported between instances of 'int' and 'str'
>>>
>>> L2.pop()
2
>>> L2.pop()
1
>>> L2.sort()
>>> L2
['I', 'L', 'a', 'b', 'c', 'e'] #按什么规则排序的呢?
>>> L2.insert(4,'#')
>>> L2.insert(1,'!')
>>> L2
['I', '!', 'L', 'a', 'b', '#', 'c', 'e']
>>> L2.sort()
>>> L2
['!', '#', 'I', 'L', 'a', 'b', 'c', 'e']
>>>
>>> L2.reverse() #反转
>>> L2
['e', 'c', 'b', 'a', 'L', 'I', '#', '!']
>>>
```

数据类型-列表

列表的功能：

- 创建
- 查询
- 切片
- 增加
- 修改
- 删除
- 循环
- 排序
- **其它用法**

```
>>> L2
['e', 'c', 'b', 'a', 'L', 'I', '#', '!']
>>> L2.extend([1,2,3,4]) #把一个列表，扩展到L2列表
>>> L2
['e', 'c', 'b', 'a', 'L', 'I', '#', '!', 1, 2, 3, 4]
>>>
>>> L2[2] = ['Alex','Jack','Rain']
>>> L2
['e', 'c', ['Alex', 'Jack', 'Rain'], 'a', 'L', 'I', '#', '!', 1, 2, 3, 4]
>>> L2[2][2] #嵌套列表取值
'Rain'
>>>
>>> L2.clear() #清空列表
>>> L2
[]
>>>
>>> L2.copy() #copy 方法，单讲
```

列表-练习题

1.创建一个空列表，命名为names,往里面添加old_driver,rain,jack,shanshan,peiqi,black_girl 元素
2.往names列表里black_girl前面插入一个alex
3.把shanshan的名字改成中文，姗姗
4.往names列表里rain的后面插入一个子列表，[oldboy, oldgirl]
5.返回peiqi的索引值
6.创建新列表[1,2,3,4,2,5,6,2],合并入names列表
7.取出names列表中索引4-7的元素
8.取出names列表中索引2-10的元素，步长为2
9.取出names列表中最后3个元素
10.循环names列表，打印每个元素的索引值，和元素
11.循环names列表，打印每个元素的索引值，和元素，当索引值 为偶数时，把对应的元素改成-1
12.names里有3个2，请返回第2个2的索引值。不要人肉数，要动态找(提示，找到第一个2的位置，在此基础上再找第2个)

13.现有商品列表如下:
products = [['Iphone8',6888],['MacPro',14800], ['小米6',2499],['Coffee',31],['Book',80],['Nike Shoes',799]]
需打印出这样的格式：

```
----------商品列表----------
0. Iphone8     6888
1. MacPro     14800
2. 小米6     2499
3. Coffee     31
4. Book     80
5. Nike Shoes     799
```

14. 写一个循环，不断的问用户想买什么，用户选择一个商品编号，就把对应的商品添加到购物车里， 最终用户输入q退出时，打印购物车里的商品列表

数据类型-字符串

字符串是一个有序的字符的集合，用于存储和表示基本的文本信息，一对单、双、或 三引号中间包含的内容称之为字符串

创建：

```
s = 'Hello,beauty! How are you?'
```

特性：

- 有序
- 不可变

数据类型-元组

元组其实跟列表差不多，也是存一组数，只不是它一旦创建，便不能再修改，所以又叫只读列表

特性：
- 不可变
- 元组本身不可变，如果元组中还包含其他可变元素，这些可变元素可以改变

```
names = ("alex","jack","eric")
```

功能：
- index
- count
- 切片

使用场景：
- 显示的告知别人，此处数据不可修改
- 数据库连接配置信息等

hash

Hash，一般翻译做“散列”，也有直接音译为“哈希”的，就是把任意长度的输入，通过散列算法，变换成固定长度的输出，该输出就是散列值。这种转换是一种压缩映射，也就是，散列值的空间通常远小于输入的空间，不同的输入可能会散列成相同的输出，所以不可能从散列值来唯一的确定输入值。简单的说就是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数。

不可变类型	可变类型
数字	列表
字符串	
元组	

特征：
hash值的计算过程是依据这个值的一些特征计算的，这就要求被hash的值必须固定，因此被hash的值必须是不可变的

用途：
文件签名
md5加密
密码验证

```
>>> hash('abc')
5632034241942406109
>>> hash('alex')
-2488059217263545430
>>> hash('abc')
5632034241942406109
```

如何在一个变量里存储公司每个员工的个人信息?

```
names = [
   ['Alex',26,'技术部','工程师', 13651054608],
   ['Shanshan',25,'公关部','野模', 13374245235],
   ['龙婷',24,'设计部','UI', 13824234452],
   ...
]
```

数据类型-字典

如何在一个变量里存储公司每个员工的个人信息?

字典一种key – value 的数据类型，使用就像我们上学用的字典，通过笔划、字母来查对应页的详细内容。

特性:

- key–value结构
- key必须可hash、且必须为不可变数据类型、必须唯一
- 可存放任意多个值、可修改、可以不唯一
- 无序
- 查找速度快

语法:

```
info = {
    'stu1101': "TengLan Wu",
    'stu1102': "LongZe Luola",
    'stu1103': "XiaoZe Maliya",
}
```

增加

```
>>> info["stu1104"] = "苍井空"
>>> info
{'stu1102': 'LongZe Luola', 'stu1104': '苍井空', 'stu1103': 'XiaoZe Maliya', 'stu1101': 'TengLan Wu'}
```

修改

```
>>> info['stu1101'] = "武藤兰"
>>> info
{'stu1102': 'LongZe Luola', 'stu1103': 'XiaoZe Maliya', 'stu1101': '武藤兰'}
```

数据类型-字典

查找

```
>>> info = {'stu1102': 'LongZe Luola', 'stu1103': 'XiaoZe Maliya'}
>>>
>>> "stu1102" in info #标准用法
True
>>> info.get("stu1102")  #获取
'LongZe Luola'
>>> info["stu1102"] #同上，但是看下面
'LongZe Luola'
>>> info["stu1105"]  #如果一个key不存在，就报错，get不会，不存在只返回None
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'stu1105'
```

数据类型-字典

多级字典嵌套

```
av_catalog = {
    "欧美":{
        "www.youporn.com": ["很多免费的,世界最大的","质量一般"],
        "www.pornhub.com": ["很多免费的,也很大","质量比yourporn高点"],
        "letmedothistoyou.com": ["多是自拍,高质量图片很多","资源不多,更新慢"],
        "x-art.com":["质量很高,真的很高","全部收费,屌比请绕过"]
    },
    "日韩":{
        "tokyo-hot":["质量怎样不清楚,个人已经不喜欢日韩范了","听说是收费的"]
    },
    "大陆":{
        "1024":["全部免费,真好,好人一生平安","服务器在国外,慢"]
    }
}

av_catalog["大陆"]["1024"][1] += ",可以用爬虫爬下来"   #修改

print(av_catalog["大陆"]["1024"])
#ouput
['全部免费,真好,好人一生平安', '服务器在国外,慢,可以用爬虫爬下来']
```

数据类型-字典

其它方法

```
#values
>>> info.values()
dict_values(['LongZe Luola', 'XiaoZe Maliya'])

#keys
>>> info.keys()
dict_keys(['stu1102', 'stu1103'])

#setdefault
>>> info.setdefault("stu1106","Alex")
'Alex'
>>> info
{'stu1102': 'LongZe Luola', 'stu1103': 'XiaoZe Maliya', 'stu1106': 'Alex'}
>>> info.setdefault("stu1102","龙泽萝拉")
'LongZe Luola'
>>> info
{'stu1102': 'LongZe Luola', 'stu1103': 'XiaoZe Maliya', 'stu1106': 'Alex'}

#update
>>> info
{'stu1102': 'LongZe Luola', 'stu1103': 'XiaoZe Maliya', 'stu1106': 'Alex'}
>>> b = {1:2,3:4, "stu1102":"龙泽萝拉"}
>>> info.update(b)
>>> info
{'stu1102': '龙泽萝拉', 1: 2, 3: 4, 'stu1103': 'XiaoZe Maliya', 'stu1106': 'Alex'}

#items
info.items()
dict_items([('stu1102', '龙泽萝拉'), (1, 2), (3, 4), ('stu1103', 'XiaoZe Maliya'), ('stu1106', 'Alex')])


#通过一个列表生成默认dict,有个没办法解释的坑，少用吧这个
>>> dict.fromkeys([1,2,3],'testd')
{1: 'testd', 2: 'testd', 3: 'testd'}
```

数据类型-字典

循环

```
#方法1
for key in info:
    print(key,info[key])

#方法2
for k,v in info.items(): #会先把dict转成list,数据里大时莫用
    print(k,v)
```

字典-练习题

写代码,有如下字典,按照要求实现每一个功能dic = {'k1':'v1','k2':'v2','k3':'v3'}

1.请循环遍历出所有的 key

2.请循环遍历出所有的 value

3.请循环遍历出所有的 key 和 value

4.请在字典中添加一个键值对,'k4':'v4',输出添加后的字典

5.请删除字典中键值对'k1','v1',并输出删除后的字典

6.请删除字典中的键'k5'对应的键值对,如果字典中不存在键'k5',则不报错,并且让其 返回 None。

7.请获取字典中'k2'对应的值。

8.请获取字典中'k6'对应的值,如果键'k6'不存在,则不报错,并且让其返回 None。

9.现有 dic2 ={'k1':'v111','a':'b'}通过一行操作使 dic2 = {'k1':'v1','k2':'v2','k3':'v3','a':'b'}

10.组合嵌套题。写代码,有如下列表,按照要求实现每一个功能

```
lis = [['k',['qwe',20,{'k1':['tt',3,'1']},89],'ab']]
```

1.将列表 lis 中的'tt'变成大写(用两种方式)。

2.将列表中的数字 3 变成字符串'100'(用两种方式)。

3.将列表中的字符串'1'变成数字 101(用两种方式)。

11.按照要求实现以下功能:

现有一个列表 li = [1,2,3,'a','b',4,'c'],有一个字典(此字典是动态生成的,你并不知道他里面 有多少键值对,所以用 dic = {}模拟此字典);现在需要完成这样的操作: 如果该字典没有'k1'这个键,那就创建这个'k1'键和其对应的值(该键对应的值设置为空列表),并将列表 li 中的索引位为奇数对应的元素,添加到'k1'这个键对应的空列表中。 如果该字典中有'k1'这个键,且k1对应的value是列表类型, 那就将列表 li 中的索引位为奇数对应的元素,添加到'k1' 这个键对应的值中。

数据类型-集合

如何找出同时买了Iphone7和8的人?

集合是一个无序的，不重复的数据组合，它的主要作用如下：

- 去重，把一个列表变成集合，就自动去重了
- 关系测试，测试两组数据之前的交集、差集、并集等关系

```
iphone7 = ['alex','rain','jack','old_driver']

iphone8 = ['alex','shanshan','jack','old_boy']
```

```
iphone7 = ['alex', 'rain', 'jack', 'old_driver']
iphone8 = ['alex', 'shanshan', 'jack', 'old_boy']

both_list = []

for name in iphone8:
    if name in iphone7:
        both_list.append(name)

print(both_list)
```

数据类型-集合

集合中的元素有三个特征：

1.确定性（元素必须可hash）

2.互异性（去重）

3.无序性（集合中的元素没有先后之分），如集合{3,4,5}和{3,5,4}算作同一个集合。

```
>>> s = {1,2,3,2,4,5} #创建集合
>>> s
{1, 2, 3, 4, 5} #只有一个2
>>> s.add(2) #添加重复的2也加不进去
>>> s.add(44)
>>> s
{1, 2, 3, 4, 5, 44}
>>> s.update([2,3,4,5,5,99]) #把多个值加入到集合
>>> s
{1, 2, 3, 4, 5, 99, 44}

>>> s.discard(1) #删除元素，没有也不报错
>>> s.pop() #随便删除一个元素，集合为空的话会报错
2
>>> s.pop()
3
>>> s
{4, 5, 99, 44}
>>> s.clear() #清空
>>> s
set()
>>> s.add(1)
>>> s.copy()
{1}
>>> s.add([1,2,3]) #只能添加不可变数据
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

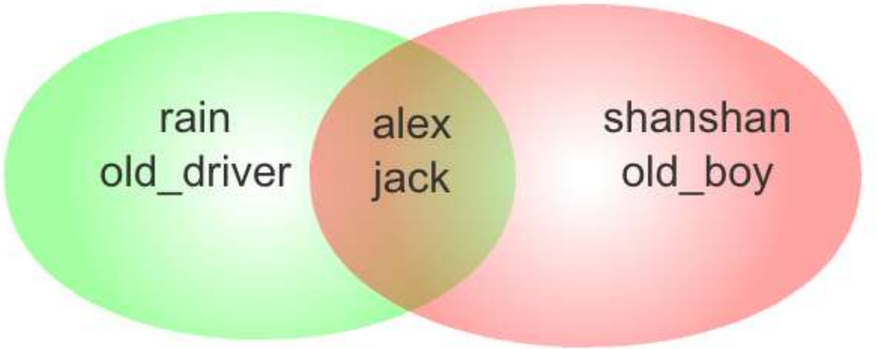
数据类型-集合

```
iphone7 = {'alex','rain','jack','old_driver'}
iphone9 = {'alex','jack','shanshan','old_boy'}
```

集合关系测试
交集
差集
并集

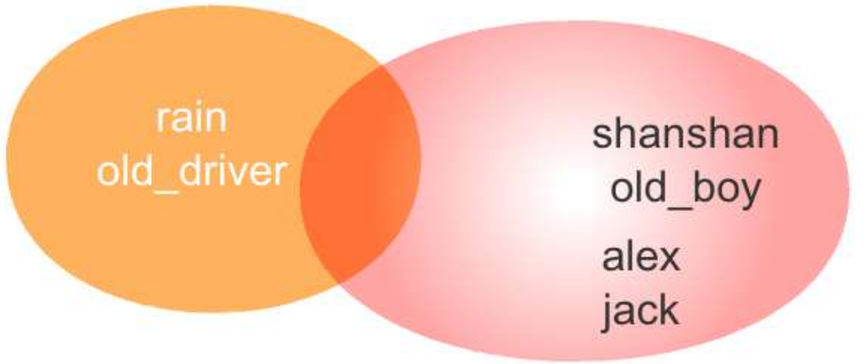
```
>>> iphone7.intersection(iphone8)
{'alex', 'jack'}
>>> iphone7 & iphone8
{'alex', 'jack'}
>>>
```

交集




```
>>> iphone7.difference(iphone8)
{'rain', 'old_driver'}
>>> iphone7 - iphone8
{'rain', 'old_driver'}
>>>
>>>
>>> iphone8.difference(iphone7)
{'old_boy', 'shanshan'}
>>> iphone8 - iphone7
{'old_boy', 'shanshan'}
```

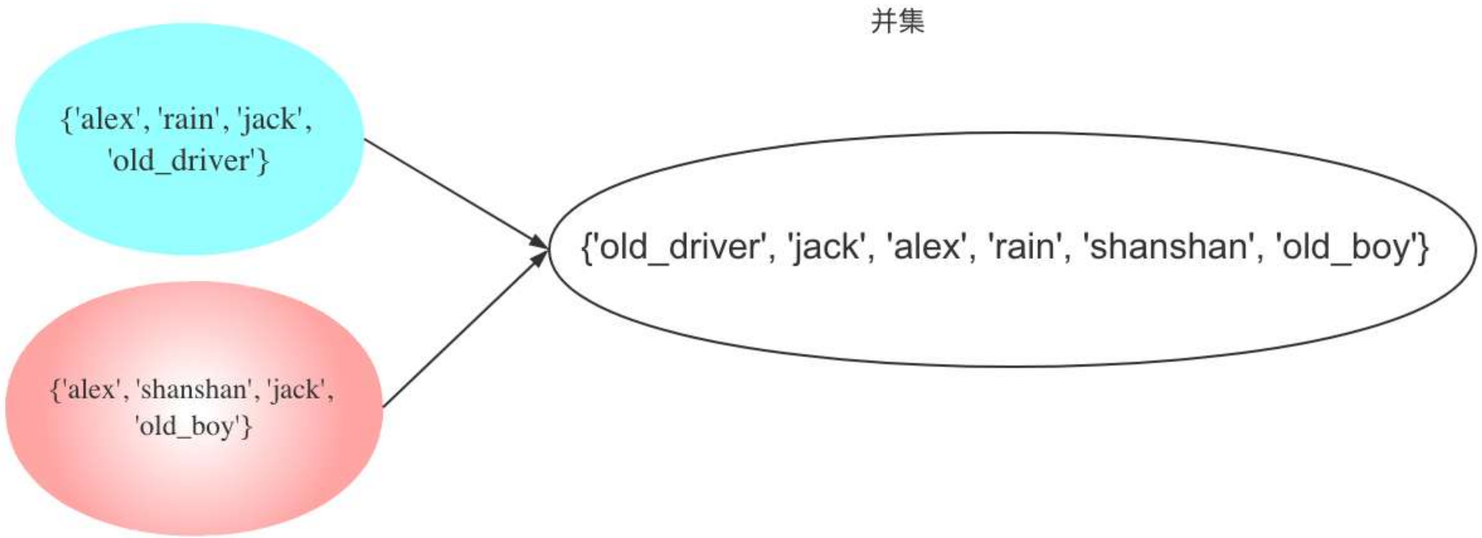
差集



数据类型-集合

```
iphone7 = {'alex','rain','jack','old_driver'}
iphone9 = {'alex','jack','shanshan','old_boy'}
```

集合关系测试
交集
差集
并集




```
>>> iphone8.union(iphone7)
{'old_driver', 'jack', 'alex', 'rain', 'shanshan', 'old_boy'}
>>> iphone8 | iphone7
{'old_driver', 'jack', 'alex', 'rain', 'shanshan', 'old_boy'}
>>>
```

数据类型-集合

集合关系测试
交集
差集
并集
对称差集
只买了iphon7 or iphon8的人

```
iphone7 = {'alex','rain','jack','old_driver'}
iphone9 = {'alex','jack','shanshan','old_boy'}


>>> iphone8.symmetric_difference(iphone7)
{'old_driver', 'old_boy', 'rain', 'shanshan'}
>>>
>>> iphone7 ^ iphone8
{'old_driver', 'old_boy', 'rain', 'shanshan'}
>>>
```

包含关系

in,not in：判断某元素是否在集合内 ==,!=:判断两个集合是否相等

两个集合之间一般有三种关系，相交、包含、不相交。在Python中分别用下面的方法判断：

set.isdisjoint(s)：判断两个集合是不是不相交
set.issuperset(s)：判断集合是不是包含其他集合，等同于a>=b
set.issubset(s)：判断集合是不是被其他集合包含，等同于a<=b

进制拾遗

二进制，01
八进制，01234567
十进制，0123456789
十六进制，0123456789ABCDEF

十进制转换8、16进制语法
oct() 8进制
hex() 16进制

字符编码

16进制

为什么用16进制

1、计算机硬件是0101二进制的，16进制刚好是2的倍数，更容易表达一个命令或者数据。十六进制更简短，因为换算的时候一位16进制数可以顶4位2进制数，也就是一个字节（8位进制可以用两个16进制表示）

2、最早规定ASCII字符集采用的就是8bit(后期扩展了,但是基础单位还是8bit)，8bit用2个16进制直接就能表达出来，不管阅读还是存储都比其他进制要方便

3、计算机中CPU运算也是遵照ASCII字符集，以16、32、64的这样的方式在发展，因此数据交换的时候16进制也显得更好

4、为了统一规范，CPU、内存、硬盘我们看到都是采用的16进制计算

16进制用在哪里

1、网络编程，数据交换的时候需要对字节进行解析都是一个byte一个byte的处理，1个byte可以用0xFF两个16进制来表达。通过网络抓包，可以看到数据是通过16进制传输的。

2、数据存储，存储到硬件中是0101的方式，存储到系统中的表达方式都是byte方式

3、一些常用值的定义，比如：我们经常用到的html中color表达，就是用的16进制方式，4个16进制位可以表达好几百万的颜色信息。

二进制数与十六进制数 http://jingyan.baidu.com/album/47a29f24292608c0142399cb.html?picindex=1

字符编码转换总结

日本编码 —解码(decode)—> unicode --编码(encode)--> gbk

Python2
内存中字符默认编码是ASCII,默认文件编码亦是ASCII
当声明了文件头的编码后，字符串的编码就按文件编码来，总之，文件编码是什么，那py2的str就是什么编码
python 2中的unicode是一个单独的类型，用u’路飞’ 来表示
python 2 str == bytes, bytes直接是按字符编码存成的2进制格式在内存里

Python3
字符串都是unicode
文件编码默认是utf-8,读到内存会被py解释器自动转成unicode
bytes与str做了明确区分
所有的unicode字符编码后都会变成bytes格式

本章作业

作业1

写程序：购物车程序

需求:

- 启动程序后，让用户输入工资，然后打印商品列表
- 允许用户根据商品编号购买商品
- 用户选择商品后，检测余额是否够，够就直接扣款，不够就提醒
- 用户可一直购买商品，也可随时退出，退出时，打印已购买商品和余额

作业2

写程序：多级菜单

需求:

- 现有省、市、县3级结构，要求程序启动后，允许用户可依次选择进入各子菜单
- 可在任意一级菜单返回上一级
- 可以在任意一级菜单退出程序

所需新知识点：列表、字典