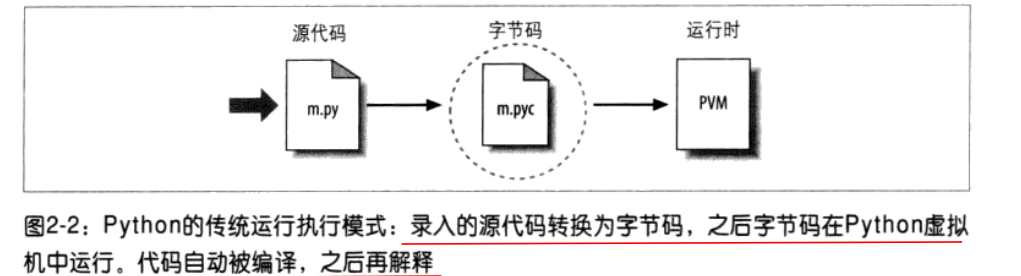
1、python运行脚本之前，会先将代码编译成字节码（和平台无关），之后将其转发到所谓的“虚拟机”当中。

2、如果Python进程拥有机器上的写权限，会将程序的字节码保存为以.pcy为扩展名的文件，只要源程序文件没有被修改，之后的执行都会直接加载.pcy文件而跳过编译步骤。若是源程序文件修改（观察源程序的时间戳可以知道是否修改），会重新编译创建.pcy文件。

3、假如没有机器上的写权限，字节码会在内存中生成并在程序结束时简单的丢弃，对于大型的程序，最好保证拥有机器上的写权限，让其.pcy文件存在磁盘当中。

4、PVM（python虚拟机）实际上是python的运行引擎，时常表现为python系统的一部分，其迭代运行字节码（程序编译成的或者从.pyc读入的）指令的一个大循环，一个接一个的完成操作。

5、python的工作并没有“build”和“make”，代码在写好之后立即运行。python字节码也不是机器的二进制代码，而是和平台无关的特定于python的一种表现形式。

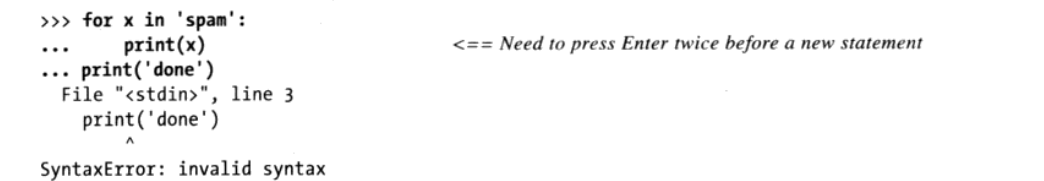


6、python在程序开始执行前不需要预编译和连接，其代码可以动态的修改，用户可以改进系统内部的python部分而不需要拥有或编译整个系统的代码。

7、牢记python中真正拥有的只有运行时：完全不需要初始的编译阶段，所有的事情（包括建立函数和类以及连接的模块）都是在程序的运行时发生的。

8、当你对一段python代码的运行有任何疑问的时候，马上打开交互命令并实验代码，看看会发生什么。（交互命令只适合短小的代码）

9、交互提示模式每次只运行一条语句：必须两次按下Enter键来运行一个循环或其它多行语句，然后才能输入下一条语句：



10、为了能够永久的保存程序，需要在文件中写入代码，这样的文件通常叫做模块。模块就是包含了Python语句的简单文本文件。模块文件也叫做脚本

11、在系统提示模式下使用文件扩展名，但是在导入时别使用文件扩展名。

12、UNIX可执行脚本（#！）

（1）要想向csh或ksh程序一样在UNIX系统可直接执行，第一行必须以字符#！开始，其后紧跟着机器Python解释器的路径

（2）脚本文件需要有可执行的权限（chmod +x file.py）

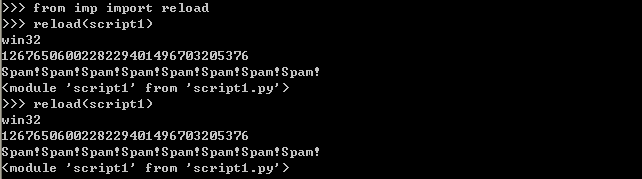
13、在直接点击python源程序文件的时候，需要在源程序文件最后添加input()语句等待标准输入，以达到让脚本暂停，否则点击图表只是一闪而过。

14、模块导入和重载：导入同时也是一种启动程序的方法。每一个以扩展名py结尾的Python源码文件都是一个模块。其它的文件可以通过导入一个模块读取这个模块的内容。

15、大型程序往往有多个模块文件，并且文件当中还可能导入了其它模块文件的工具。其中一个模块文件设计成主文件，或叫做顶层文件（也就是能运行整个程序的文件）

16、导入必须找到文件。将其编译成字节码，并且运行代码，若不采取特殊的措施，在第一次导入后，其它的导入都不会再工作，即使改变了源程序代码。除非调用imp标准库中可用的reload函数，再次导入会生效（无论源程序是否改变）。也就是重载。

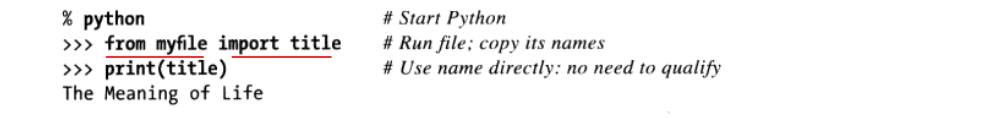
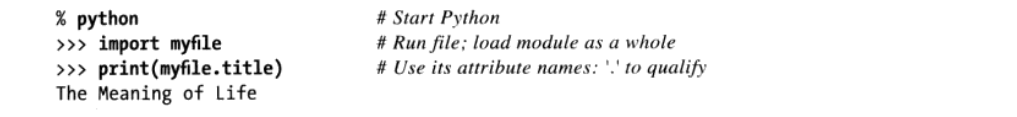


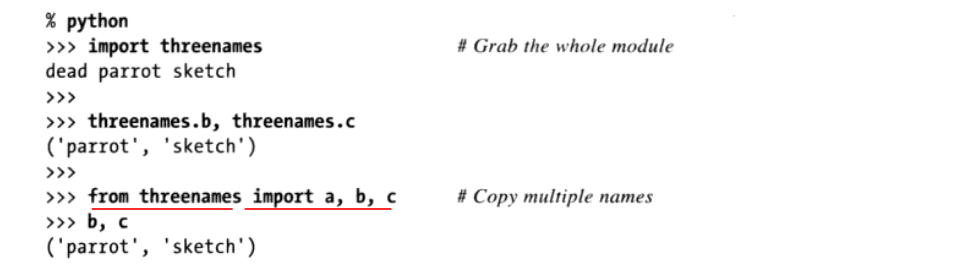


改变了源程序代码的2 \*\* 16后



17、模块往往可以看作是变量名的的封装，被认作是命名空间。变量名就是所谓的属性。

****

****

**可见from…import可以导入一个模块的多个变量名，from…import相当于复制模块的属性，以便属性直接成为接收者的直接变量.**

18、dir函数可以得到模块内部可用的变量名的列表。

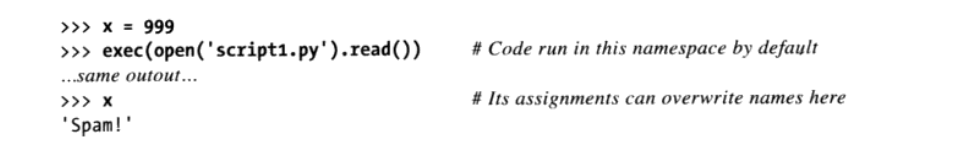


19、模块和命名空间：买个模块文件都是一个命名空间。除非显示的导入其它模块文件，否则一个模块文件不能看到其它文件的变量名。这样可以使文件中的变量名的命名冲突达到最小化（因为每个文件都是独立完备的命名空间，即使两个文件中的变量名一样也不会发生冲突）。

20、模块是一个不需要重复输入而可以反复运行代码的方法

21、reload是不可传递的，重载一个模块只会重载该模块，而不会重载该模块所导入的任何模块，因此，有时候必须reload多个文件。因此，import和reload是Python类中的一种常用的测试技术。

22、exec(open(‘module.py’).read())不导入模块，它就好像将文件里的内容插入到交互模式，进而执行程序。每次exec都是运行文件的最新版，不需要随后的重载。缺点是对当前正在使用的变量有默认覆盖的可能。



而基本的import语句每个进程只运行文件一次，并且将其生成到一个单独的名称空间当中，以便影响作用域中的变量。所付出的代码时，在修改之后需要重载

23、各种方式启动保存在文件中的代码：在系统命令行中运行、import和exec、使用IDLE这样的GUI等。

24、python程序被嵌入在其它程序中运行。Python代码可以保存到一个文本文件中、存储在数据库中、从一个HTML页面获取。另一个系统会告诉Python去运行创建的代码

25、一个游戏程序，也许允许用户进行游戏定制（及时的在策略点存取Python代码）。用户可以提供或者修改Python代码来定制这种系统（由于其是解释型的，不必重新编译整个系统以融入修改）。

26、冻结二进制的可执行性是集成了程序的字节码以及Python解释器为一个单个的可执行程序的包。

27、调试Python代码

（1）阅读出错信息，并修改标记的行和文件。适合于短小的文件

（2）插入print语句

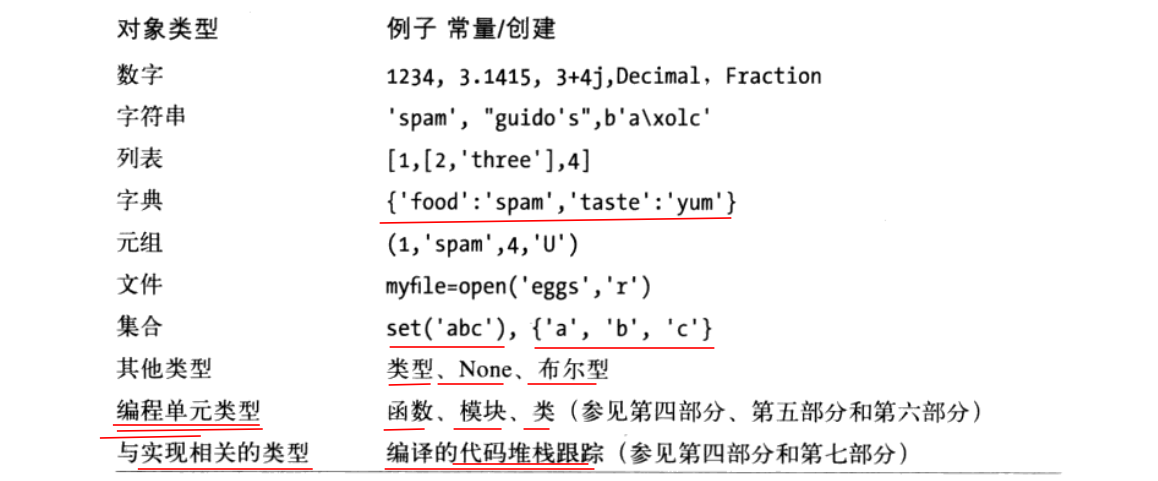
（3）使用pdb命令行调试器，python附带了一个名为pdb的源代码调试器，可以作为Python的标准库中的一个模块使用。在pdb中，可以输入命令一行一行的步进执行，显示变量、设置和清断点、继续执行到下一个断点。

28、命名空间就是变量的封装。它在Python中以一个带有属性的对象的形式出现。每个模块文件自动成为一个命名空间。

29、程序由模块构成，模块包含语句，语句包含表达式，表达式建立并处理对象。

30、C或C++需要用数据结构去表现应用领域的组件。若不使用第三方库，需要部署内存结构、管理内存分配、实现搜索和读取例程等。而Python提供强大的对象类型作为语言的组成部分，它的内置类型往往能够表现问题领域的所有结构（集合和搜索表（字典）），而人工实现的对象也往往是在内置类型的基础之上建立。内置类型作为语言的一部分，性能优于自己所实现的类似的数据类型。

31、内置对象：数字、字符串、列表、字典、元祖、文件、集合、其它类型（类型、NONE、布尔型）、编程单元类型（函数、模块、类）、与实现相关的类型（编译的代码堆栈跟踪）



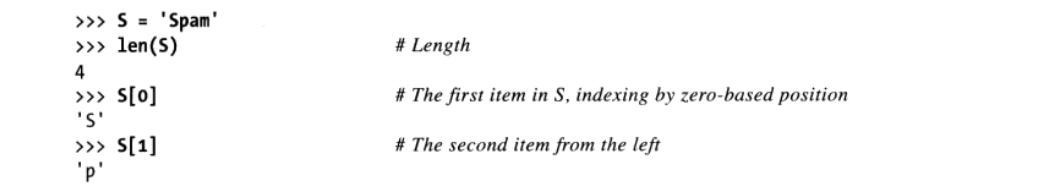
32、在python中，常量指的是其语法会生成对象的表达式，有时也叫做常数。这里的常数并不是像C++中的const一样指的是不可改变的对象或变量。

33、Python中没有类型申明，运行的表达式的语法决定了创建和使用的对象的类型。并且注意，一旦创建了一个对象，它就和操作集合绑定了—只可以对字符串进行字符串相关的操作，对列表进行列表相关的操作。

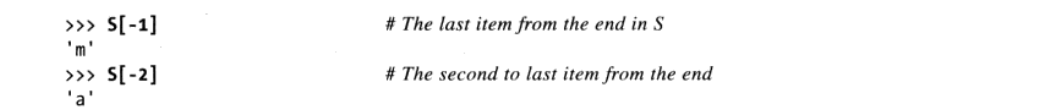
34、数字



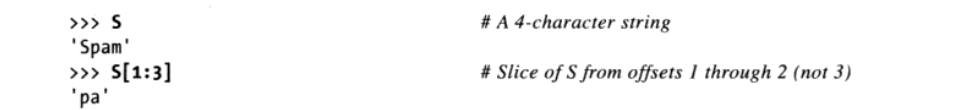
35、字符串

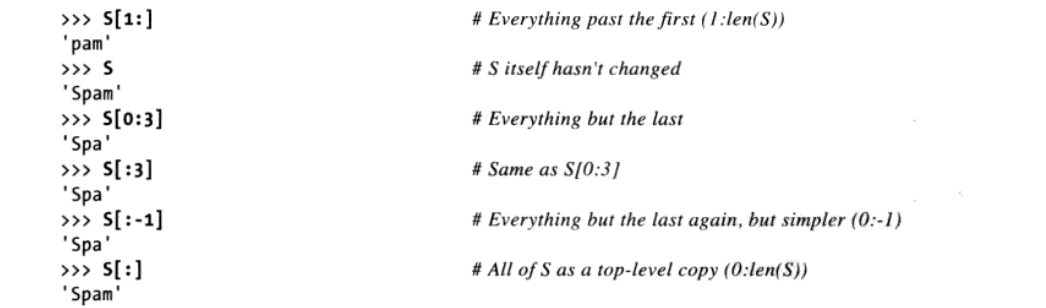


Python支持反向索引，从最后一个开始（正向索引是从左边开始计算，反向索引是从右边开始计算），S[-1]相当于S[len(S)-1]。

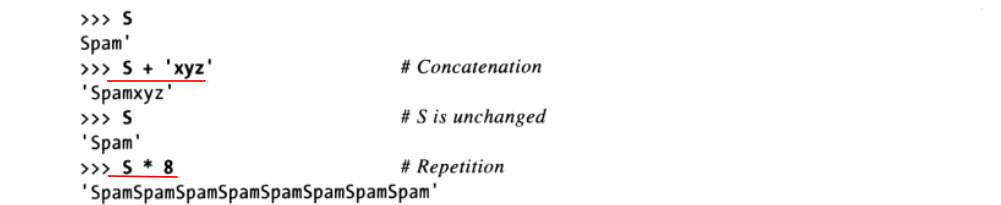


支持分片操作，类似于范围索引。

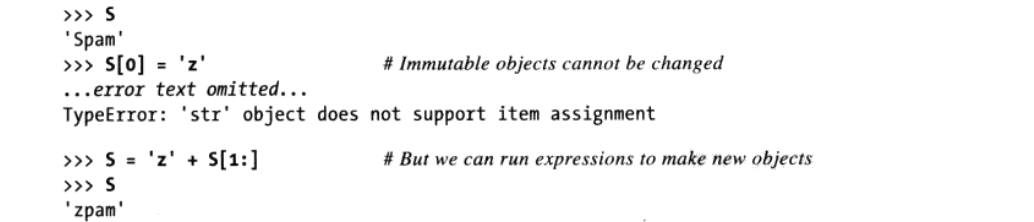




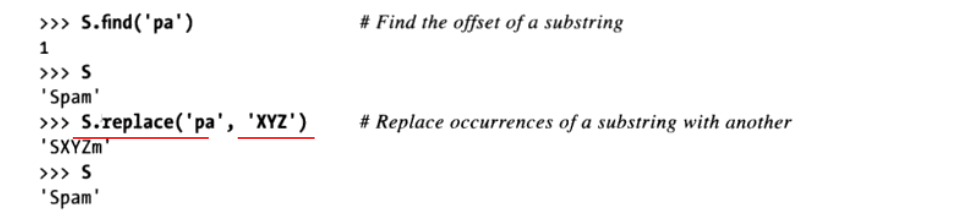
支持“+”操作和“\*”操作。



36、字符串具有不可变性：也就是不能通过对某一位置进行赋值而改变字符串，但是你可以通过建立一个新的字符串并以同一个变量名对其进行赋值。

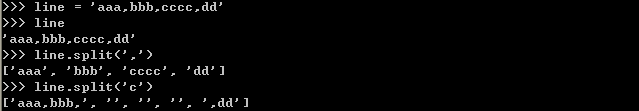


37、在Python中的每一个对象都可以分为可不变性或者可变性。在核心类型当中，数字、字符串和元祖是不可变的；列表和字典是可变的。

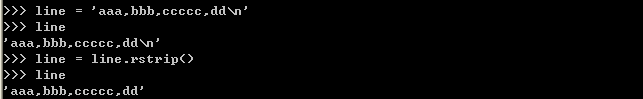
38、上述操作都是通用序列的操作，字符串也有属于自己的方法存在

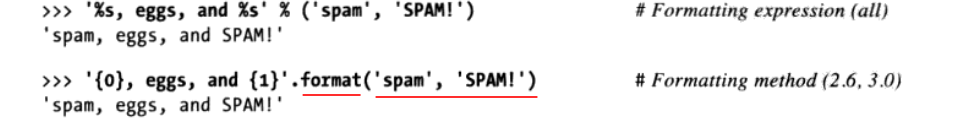
（1）find返回查找到子字符串的位置，没有找到返回-1。

（2）而字符串的replace方法将会对全局进行搜索和替换。可以看到，replace操作并不会改变原字符串，而是创建一个新字符串作为结果



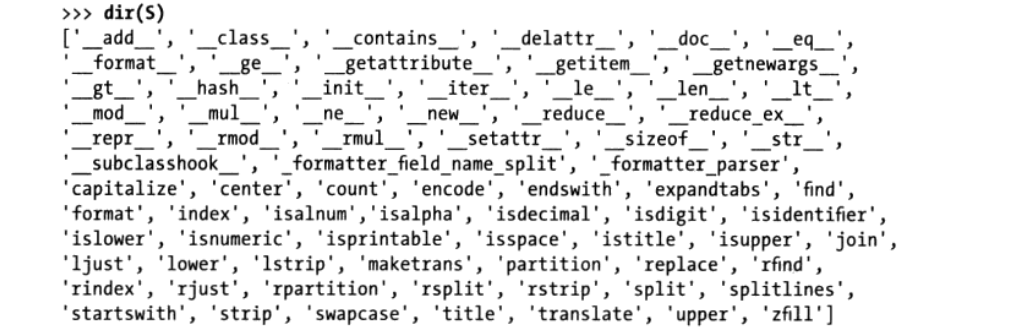
可以看到split是以方法里的参数为分隔符，将字符串分割为了多个子字符串的列表。



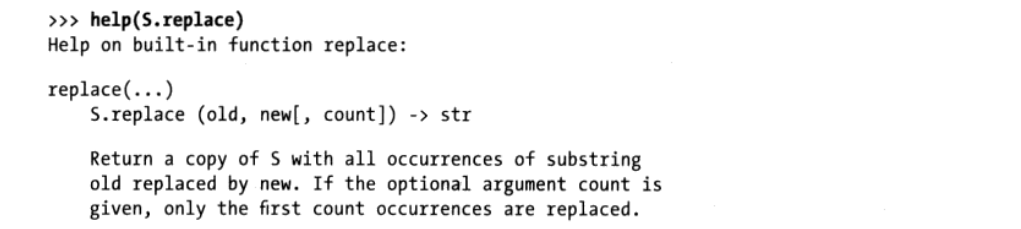


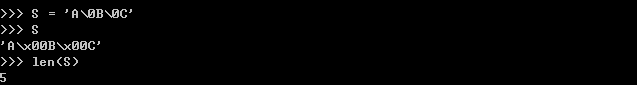
39、一条简明的法则是这样的：可做用于多种类型的通用型操作都是以内置函数或表达式的形式出现的[例如，len(X),X[0]],但是类型特定的操作是以方法调用的形式出现的[例如，aString.upper()]。

40、若想观察对象的所有特定的操作，可以用dir函数，dir函数返回一个列表，其中包含了对象的所有属性。由于方法是函数属性，它们也会在这个列表中出现。



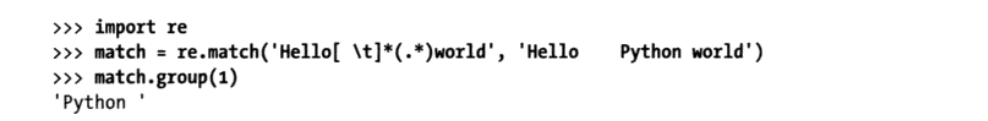
41、若想查询这些特定方法是做什么的，怎么使用的，可以将其传递给help函数。





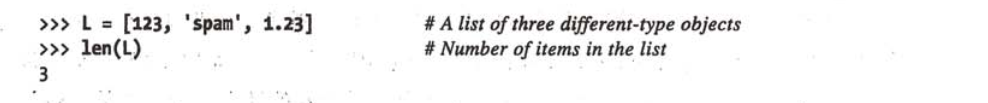
可见’\0’并不是代表字符终结，而是表示二进制的0。

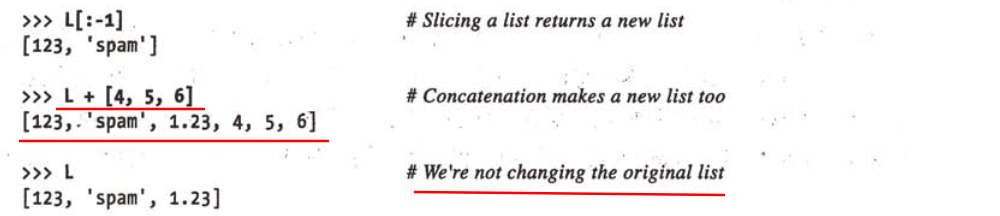
42、模式匹配，需要导入re模块，这个模块包含了类似搜索、分割和替换等调用。

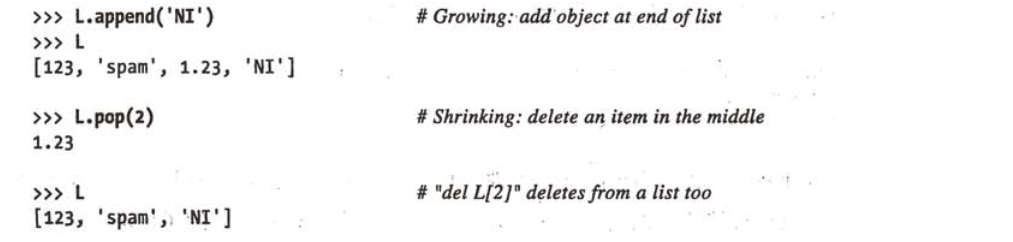


（1）这个例子的目的是搜索子字符串，这个子字符串以”Hello,”开始，后面跟着零个或几个制表符或空格，接着有任意字符并将其保存至匹配的group中，最后以“world”结尾

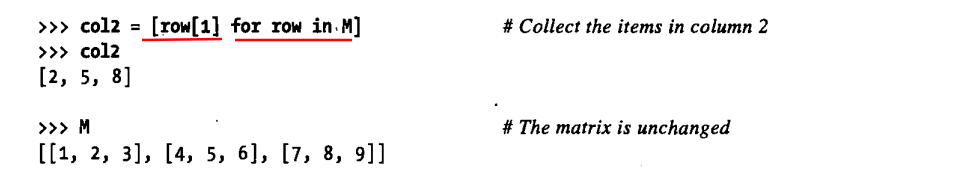
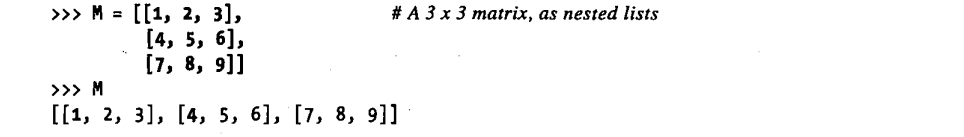
43、列表：列表是一个任意类型的对象的位置相关的有序集合，它没有固定的大小。并且它不像字符串，列表是可变的（运用append）



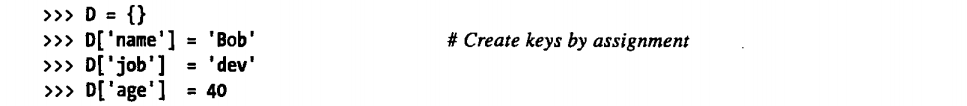


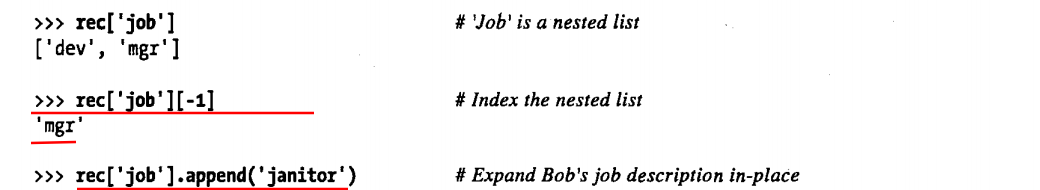
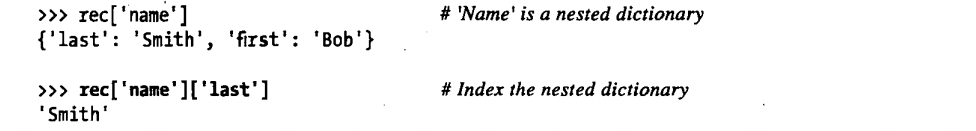
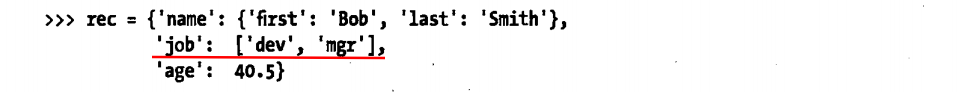


44、嵌套：Python核心数据类型的一个优秀的特性就是它们支持任意的嵌套。能够以任意的组合对其进行嵌套，甚至可以进行深层次的嵌套（让一个列表包含一个字典，让字典包含另一个列表）

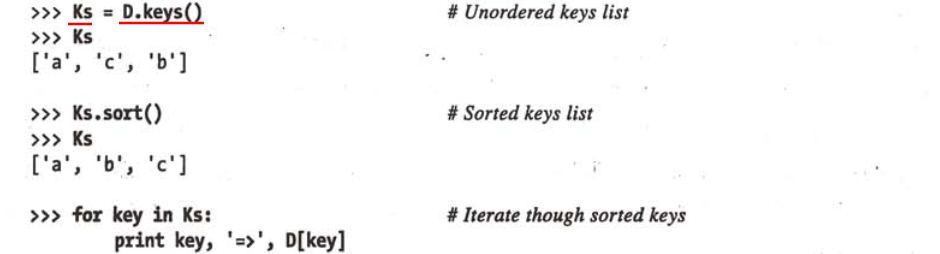


45、字典：（映射关系）

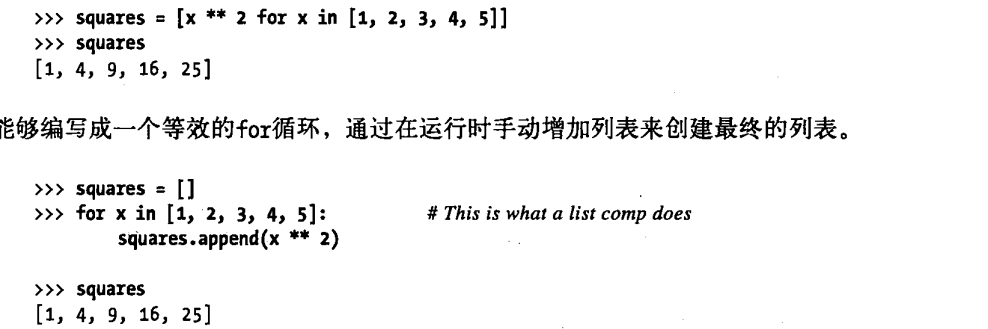




46、键的排序：for循环，也就是建立一个字典的keys的列表，排序然后对其进行for循环逐个显示结果。或者通过内置函数sorted。





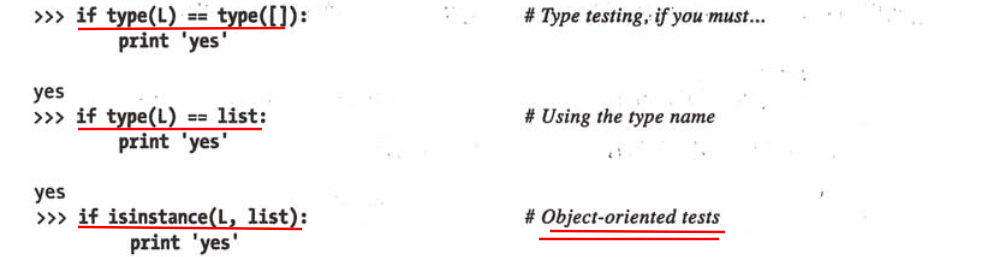


验证key的存在性（has\_key方法）

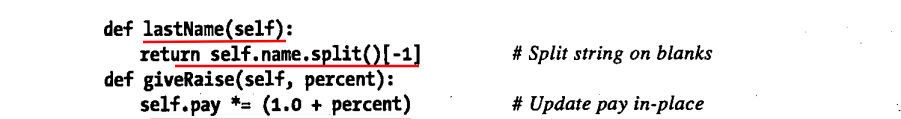
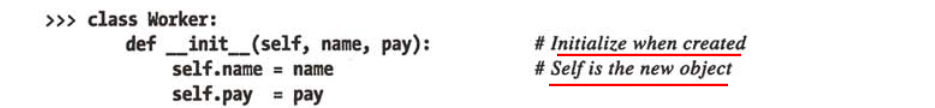


47、元祖：就是一个不可改变的列表，它也是序列，但是像字符串一样具有不可变性。

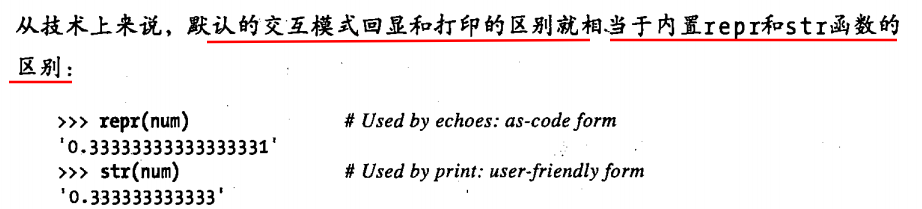
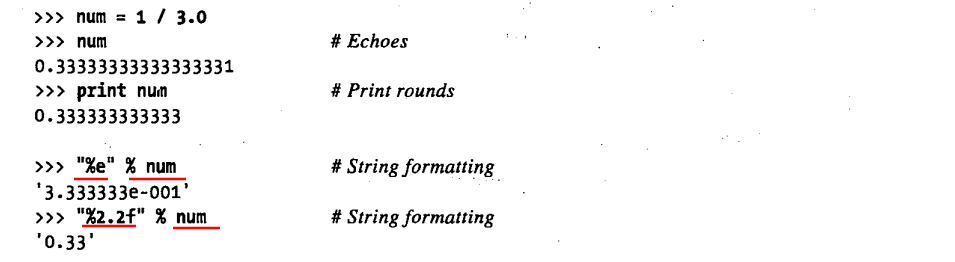
48、让代码检验所使用的对象类型



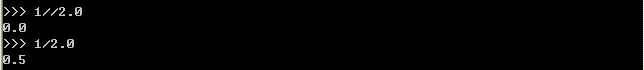
49、用户定义的类：



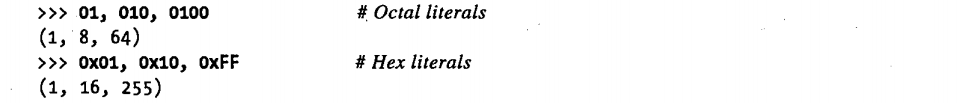
50、数字



X//Y Floor除法，总会省略结果的小数部分，剩下最小的能整除的整数部分



51、8进制和16进制



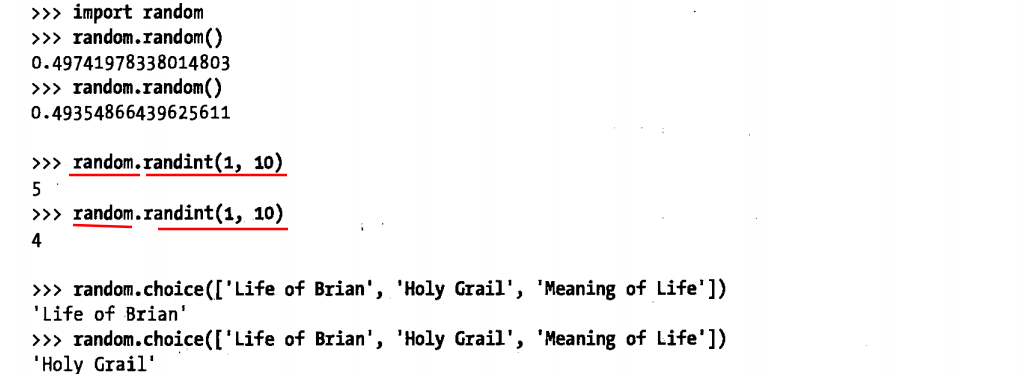
52、内置的int函数会将一个数字的字符串变换为一个整数，并可以通过定义的第二个参数来确定变换后的数字的进制。



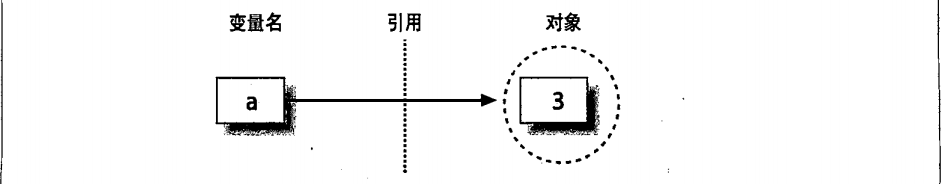
还能将一个整数通过字符串格式表达式转换成八进制数和十六进制数的字符串



53、随机数（random模块）

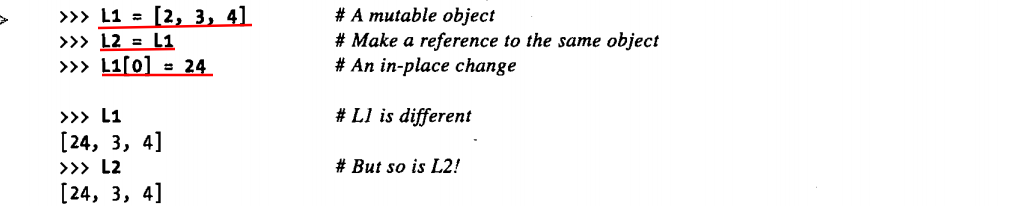


54、变量和对象保存在内存中的不同部分，并通过连接相关联。变量总是连接到对象，并且绝不会连接到其它变量上（连接到另一个变量所连接的对象上），但是更大的对象可能连接到其它的对象（一个列表对象能够连接到它所包含的对象）。

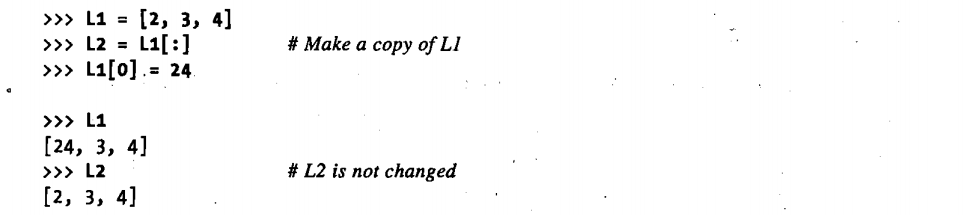


55、每一个对象都有两个标准的头部信息：一个类型标志符去标示这个对象的类型，以及一个引用的计数器，用来决定是不是可以回收这个对象。

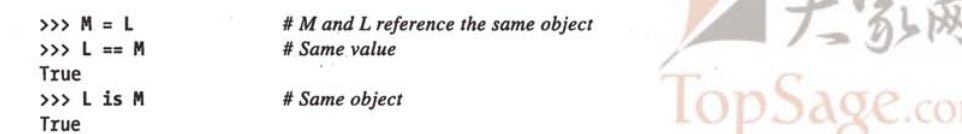
56、类型属于对象，变量没有类型，当变量被赋值时，只是指向了某个特定类型的对象。



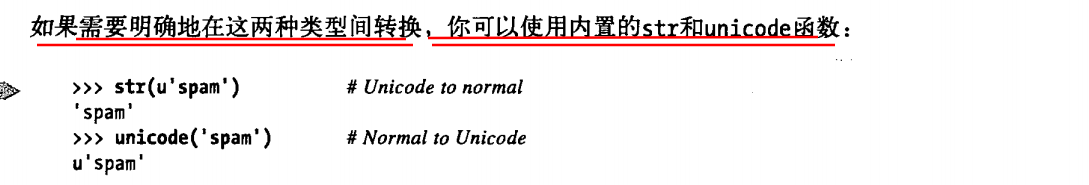
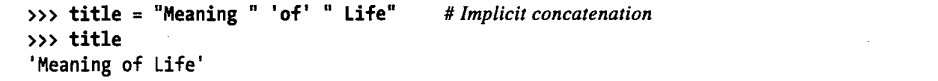
为防止受影响，可以使用copy模块，也可以从头到尾的分片。



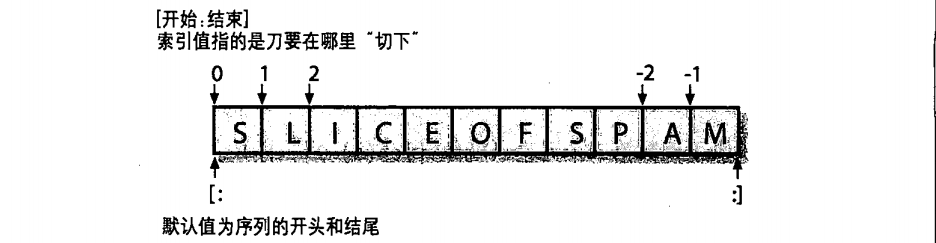




56、字符串：Python自动在任意的表达式中合并相邻的字符串常量，当然也可以使用+。如果使用逗号，创建的是一个元祖而不是字符串。



57、偏移和负偏移



S[:]获取了从偏移0到末尾之间的元素，这有效的实现了顶层S的拷贝，也就是S[:]获取的对象和S不在同一个内存片区。

S[I:J:K]表示以K为间距，索引I到J-1的内容。S[::2]以2为间距，从头到尾取出序列。

通过将间距设置为负数，可以实现反转，例如S=’hello’ S=[::-1]

得到’olleh’。负数表示从右到左步进。

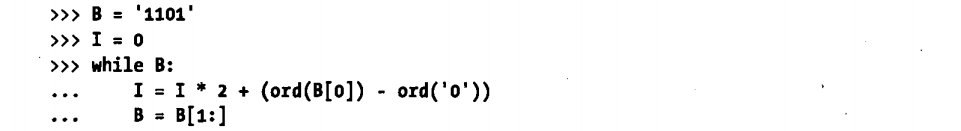
通过一个负数步进，两边的边界也进行了反转。也就是S[5:1:-1]以反转的顺序获取从2到5的元素。（5，4，3，2）不包括1。

58、为了去掉换行字符常常使用line.rstrip方法，这个调用会留下没有换行字符那行的最后一个字符。

59、将字符串变为数字用int，变为浮点数用float，将数字变为字符串str。将单个的字符转换为相应的ASCLL用ord。将ASCLL换为单个的字符用chr。通常这样使用（将单个字符转换为ASCLL后加1再换回单个字符）。

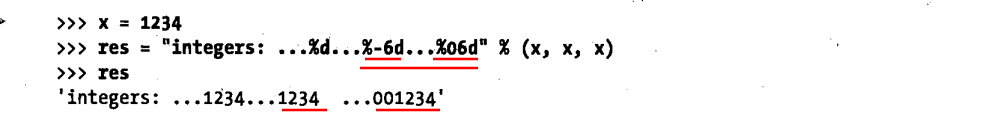


60、将2进制的字符串转换为10进制



精髓是这里的\*2相当于左移一位。那么多次乘2就是多次移位。

61、字符串的方法有replace方法。可将长度不想等的字符串替换

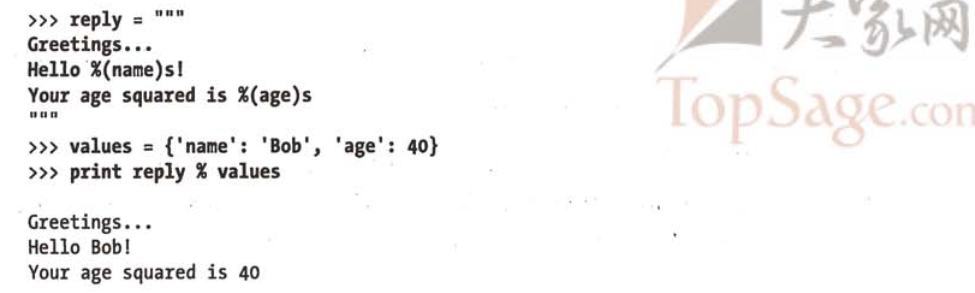
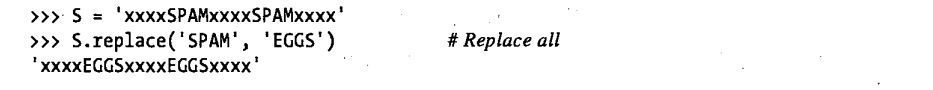


62、基于字典的字符串格式化

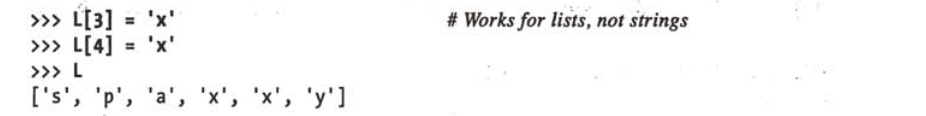
字符串的格式化同时也允许左边的转换目标来引用右边字典中的键来提取对应的值。



n和x就是引用右边字典的键来提取对应的值。



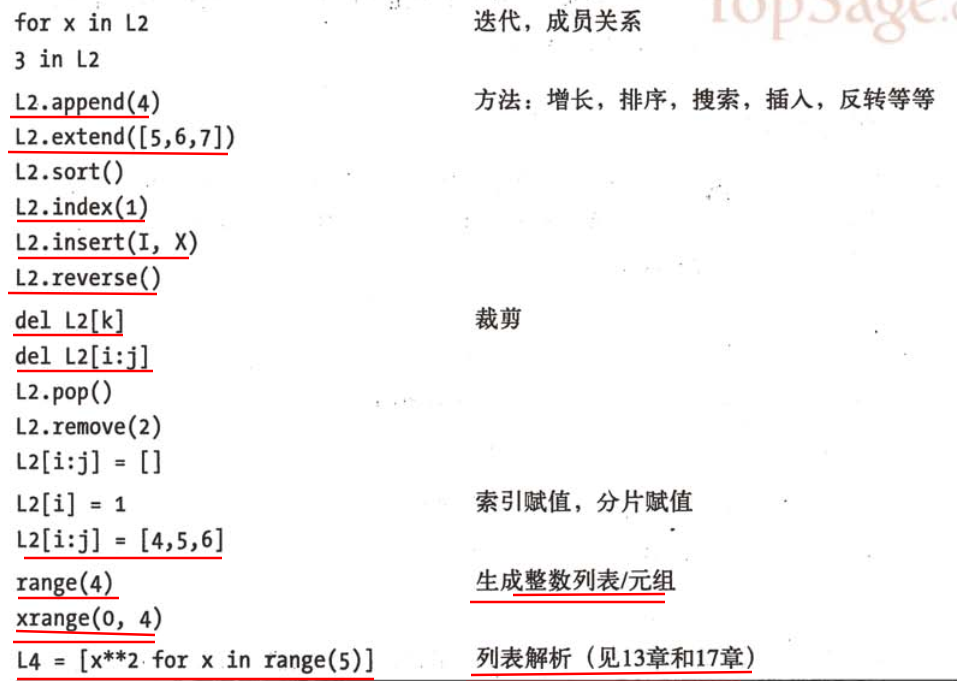
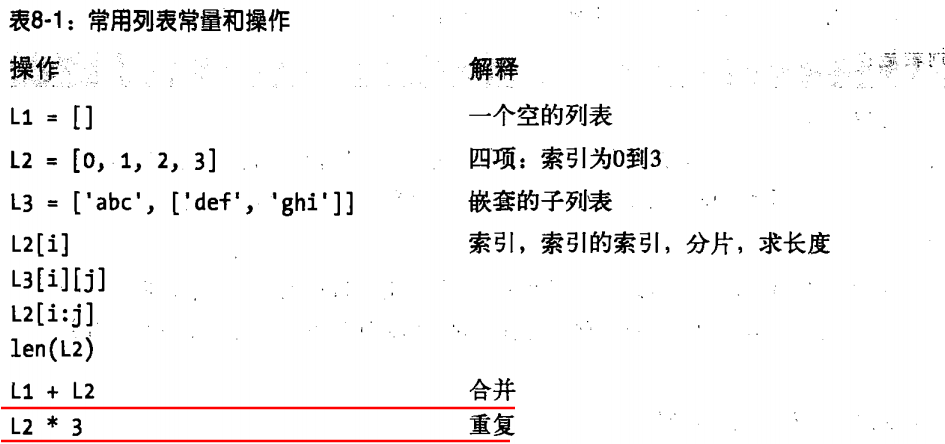
63、可以对字符串调用内置的list函数以序列中的每一个元素的元素创立一个新的列表。这样就可以修改列表当中的字符。然后再使用join函数将其合为一个字符串。主要是通过设定的分隔符来调用，假如为’’，则就是将列表当中的字符合并。



64、可以将一个字符串以某个字符作为分隔符分隔为字符串的列表。例如以空格为分隔符以及以逗号为分隔符或以某个字符串为分隔符



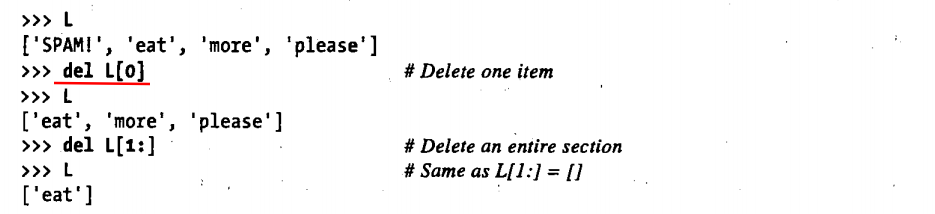
65、列表所包含的每一项都保持了从左到右的位置顺序。当把一个对象赋给一个数据结构元素或变量名时，Python总是会存储对象的引用，而不是对象的一个拷贝。（除非明确要求保存拷贝）



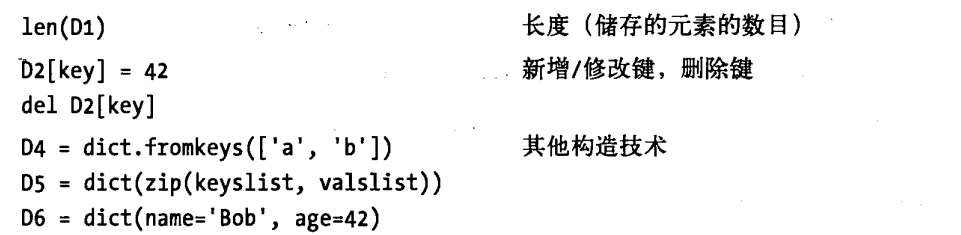
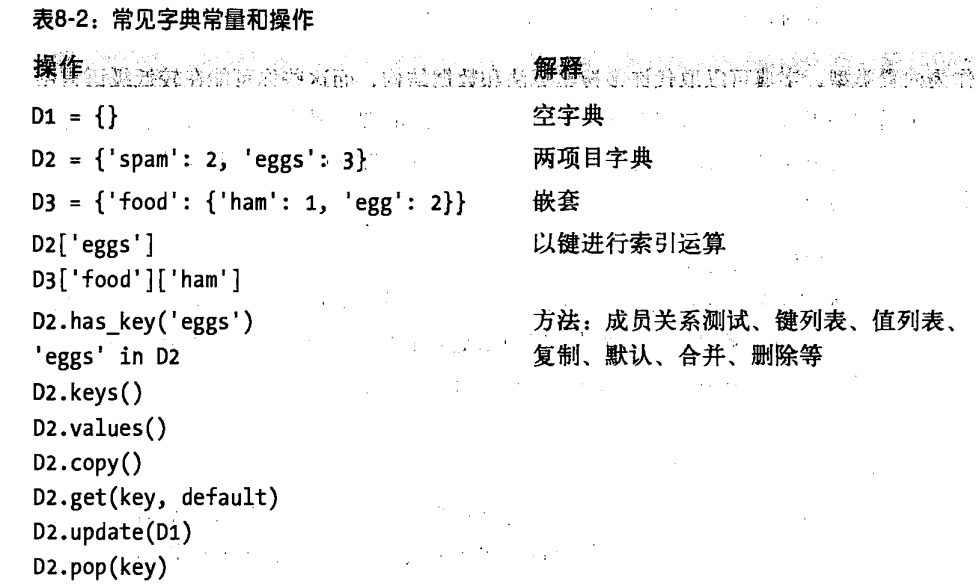
66、列表的分片不同于字符串的分片，字符串的分片是重新生成新的字符串，而列表的分片加赋值是在原来的列表基础上修改元素。

67、列表的分片赋值当中分片的长度和赋值的长度可以不一样。列表的append不产生新对象，也是在原来的基础上修改。

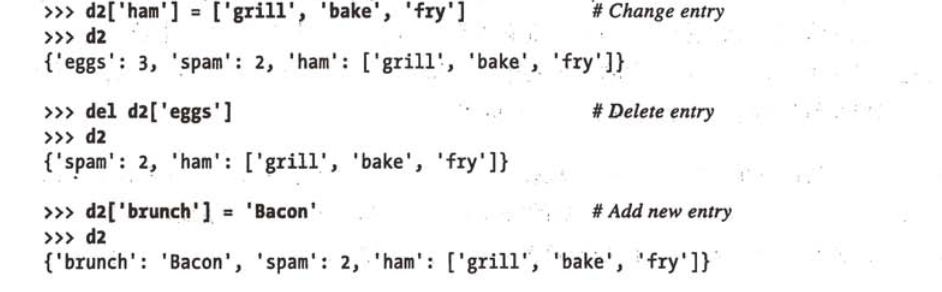
68、删除某项或某分片运用del



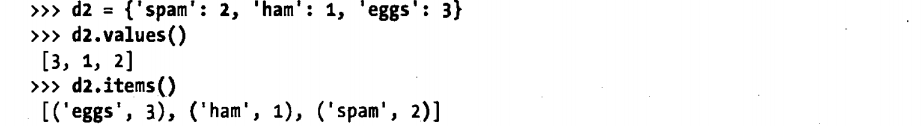
69、列表是有序对象集合，字典是无序的集合。



我们可以把键/值元祖或关键词函数自变量的列表打包传给新的dict调用。



字典values和items方法分别返回字典的值列表和(key,value)对元祖



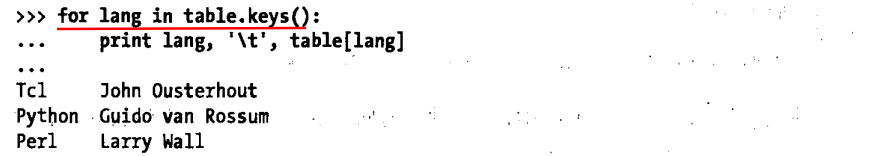
字典的update方法优点类似于合并，它把一个字典的键和值合并到另一个，盲目地覆盖相同键的值。



字典的pop用于删除一个键并返回相应的值

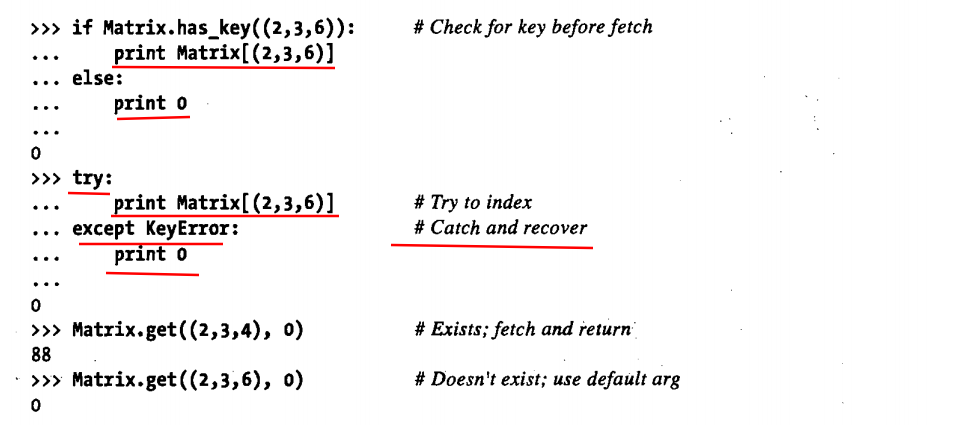


字典的keys方法返回键列表，这对于遍历字典很有用。



70、要注意，字典无序，因此有序合并和分片这样的运算是不能使用的。对新索引赋值会添加项，键不一定总是字符串

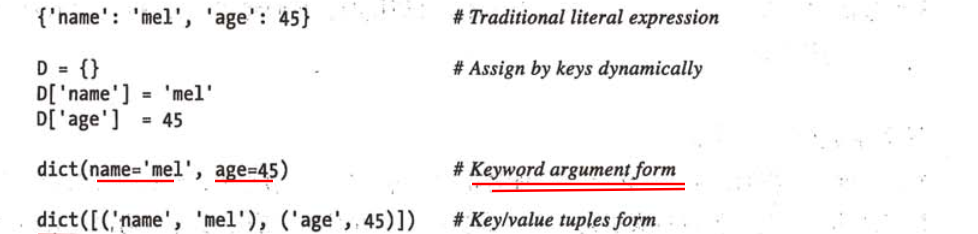
71、字典适合用于稀疏数据结构，可以用if..has\_key 或菏泽try或者get为不存在的键提供一个默认值。



使用字典作为‘记录’



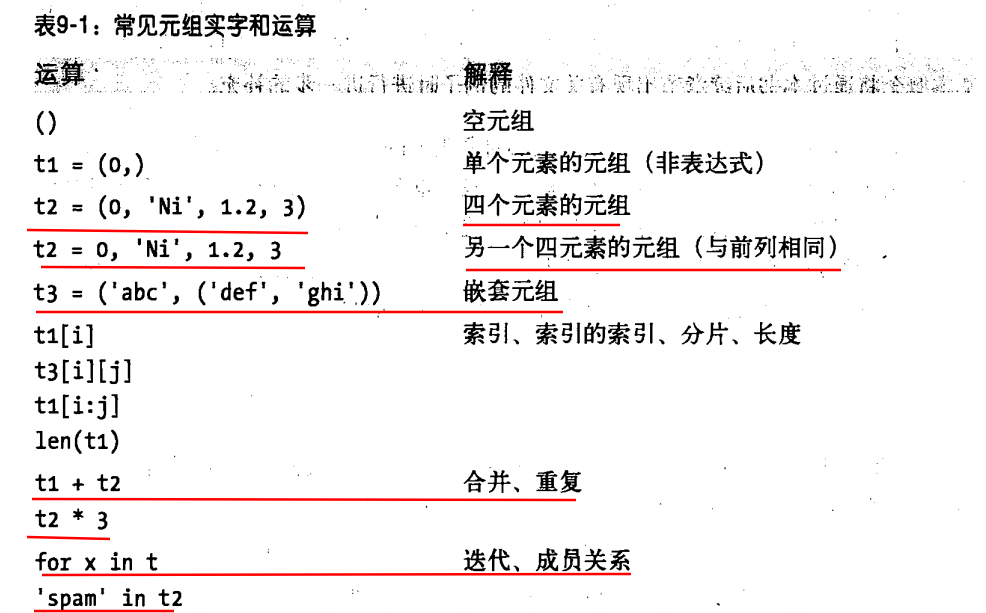
创建字典的其它方法



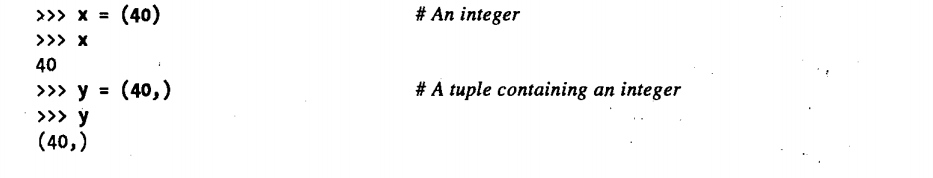
简单的传入一个键列表并进行初始化



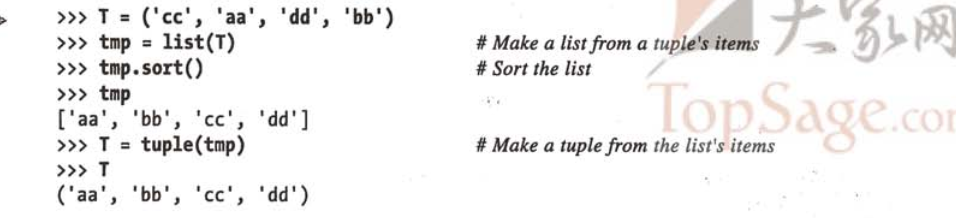
72、元祖：不能再原处修改（不可变性），不支持任何方法调用，它是任意对象的有序集合（类似于字符串和列表），与列表相同，可以嵌入到任何类别的对象中。可以通过偏移来访问（索引和分片）。元祖存储的也是指向对象的存取点（引用），并且对元祖进行索引操作的速度相对较快。（谨记，元祖没有方法）



73、如果想得到只有一个对象的元祖，就需要在这一单个元素的之后，关闭圆括号之前加一个逗号就可以。

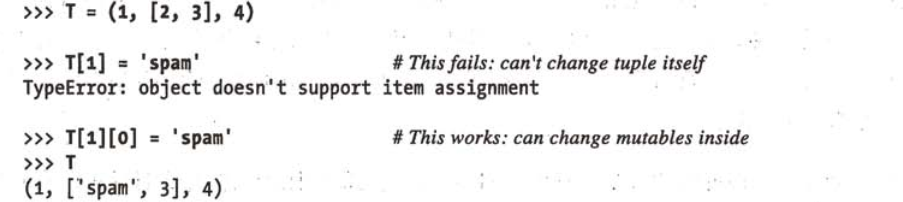


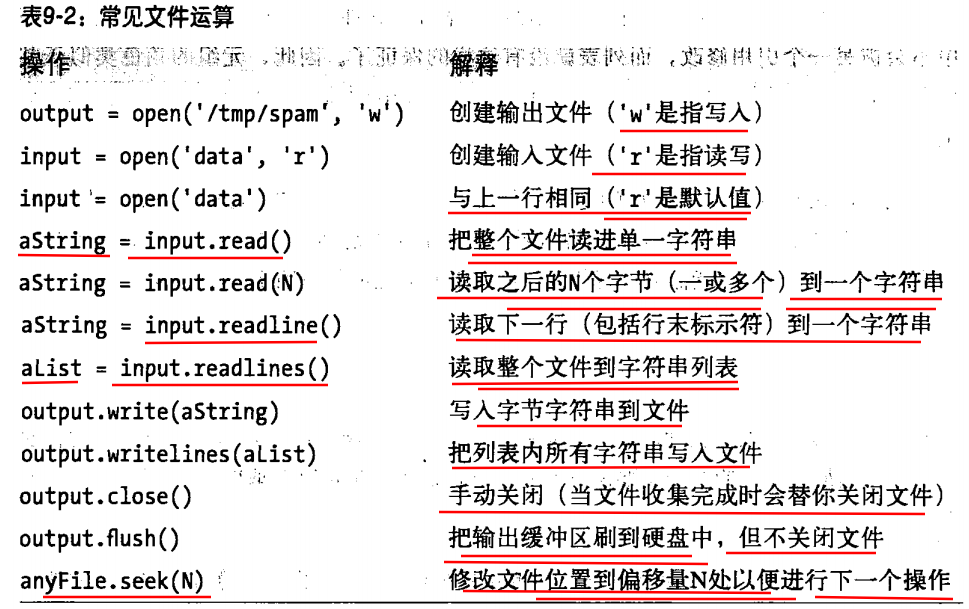
74、如果想对元祖进行排序，通常先得将它转换为列表（使用list或列表解析）才能获得使用排序方法调用的权限，并将它变为一个可变对象。然后再将其转变回元祖（使用tuple）





75、需要注意的是元祖的不变性只适用于元祖本身顶层而非其内容。例如，元祖内的列表当中的内容是可变的

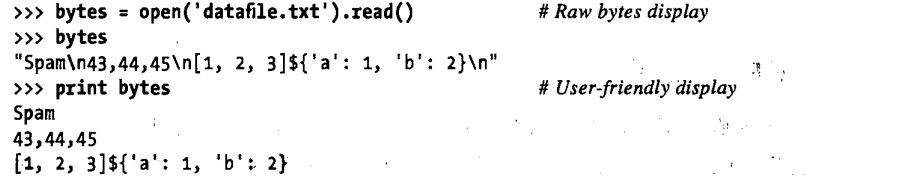
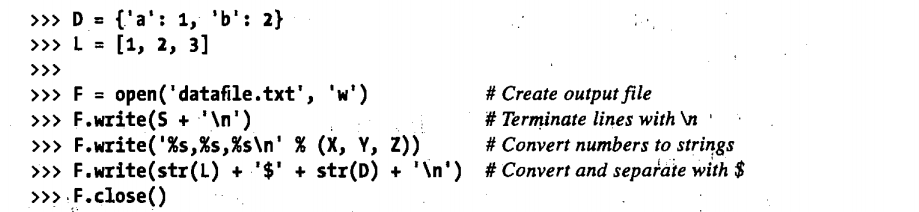


76、文件：多数文件方法都与执行外部文件相关的文件对象的输入和输出有关，其它文件方法可以查找文件中的新位置、刷新输出缓存等。

77、从文件读取的数据回到脚本时是一个字符串。如果字符串不是你所需的，需要将其转换成其它类型的Python对象。当把数据写入文件时，Python也不会自动把对象转换为字符串—因此你必须传递一个已经格式化的字符串（int、float、str函数以及字符串格式化表达式派上用场了）。

78、readline调用返回空字符串表示Python文件方法已经到达文件底部



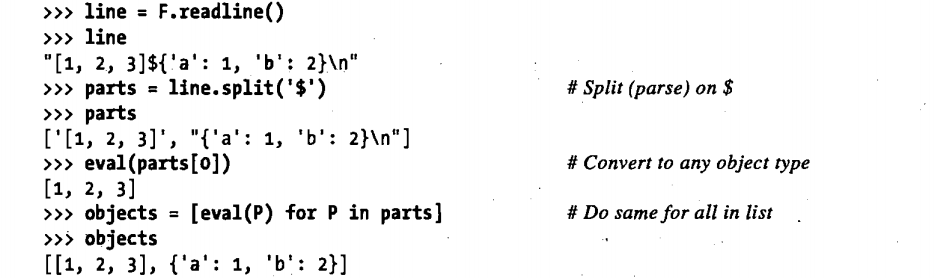




使用spilit函数将数字字符串分割,并且调用int将其数字化，会发现int也能够对于带’\n’的数字进行有效转换。



对于列表和字典可以调用eval函数把字符串转换成对象。

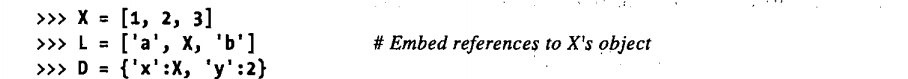


79、更通用的是使用pickle模块让我们直接在文件中存储几乎任何Python对象的高级工具，并不要求我们将字符串转来转去。以下为利用pickle在文件中存储字典。通过调用pickle的dump和load

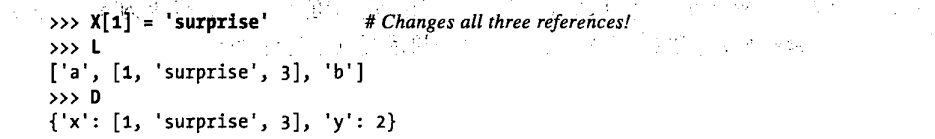


80、再次谨记，赋值操作总是存储对象的引用，而不是这些对象的拷贝。

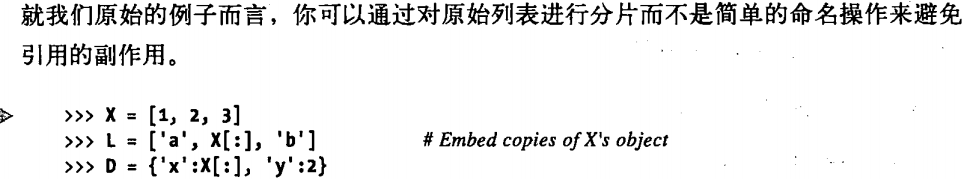
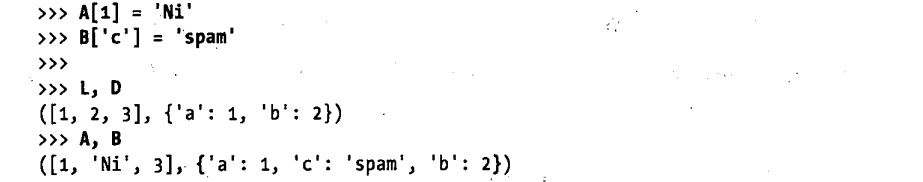
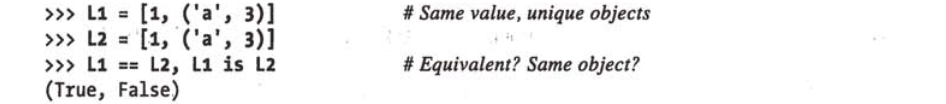
81、引用VS拷贝：

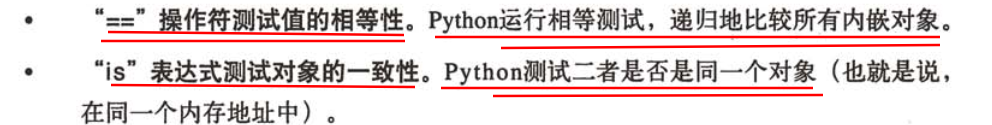


修改任意一个共享的引用，都会改变另外两个引用的对象

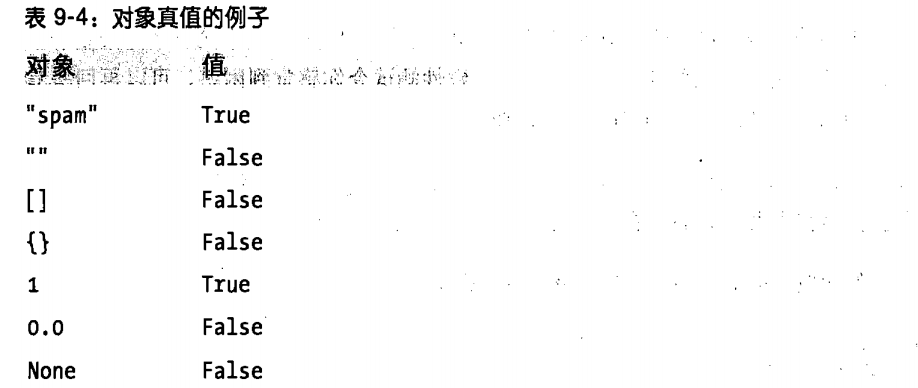
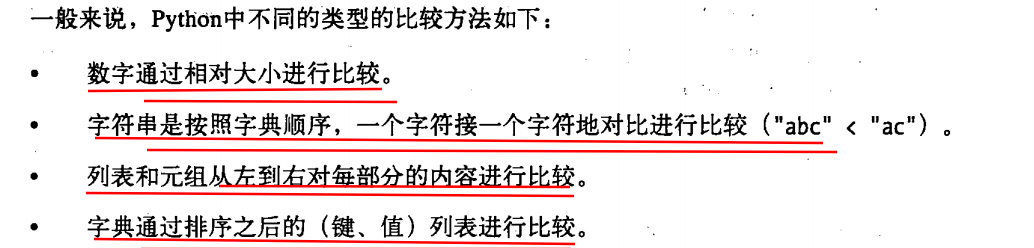
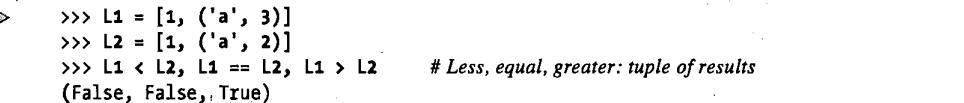


可以使用相应的拷贝方法，假如确实需要拷贝：没有限制条件的分片表达式（L[:]）能够复制序列。字典copy方法（D.copy()）能够复制字典。有些内置函数能够生成拷贝（list(L)）。copy标准库模块能够生成完整的拷贝。

要注意：无条件的分片以及字典的copy方法只能做顶层复制，也就是说不能复制嵌套的数据结构。如果需要一个深层嵌套的数据结构的完整的、完全独立的拷贝，那么应当使用标准的copy模块—import copy，并编写x = copy.deepcopy(Y)对任意嵌套对象Y做完整的复制。



82、比较大小



83、预先分配一个100项的列表，这样可以在100个偏移的任何一个加上None对象。

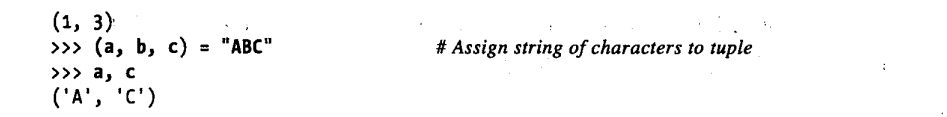
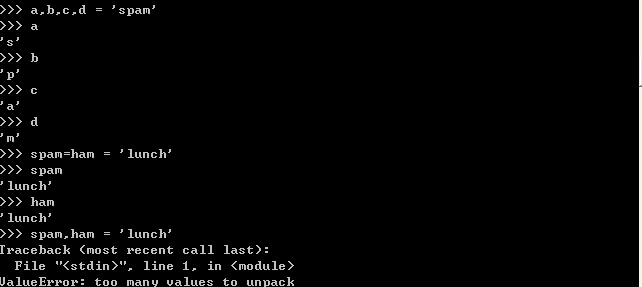
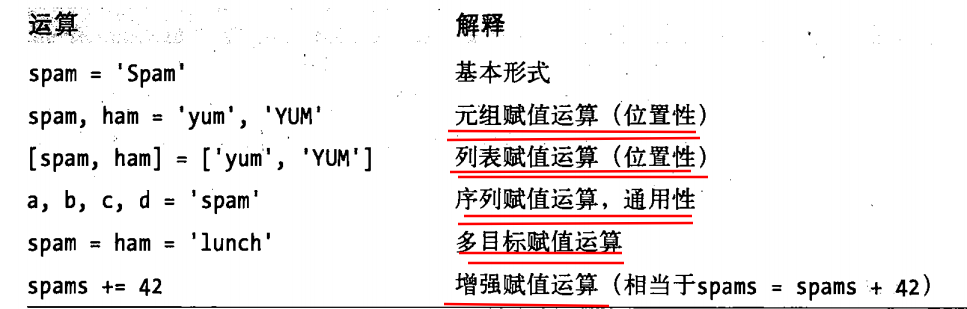
84、实现swap()函数

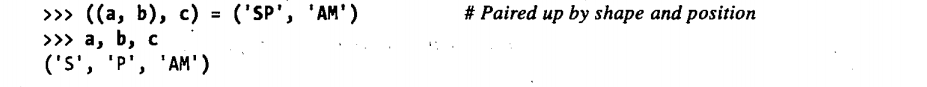
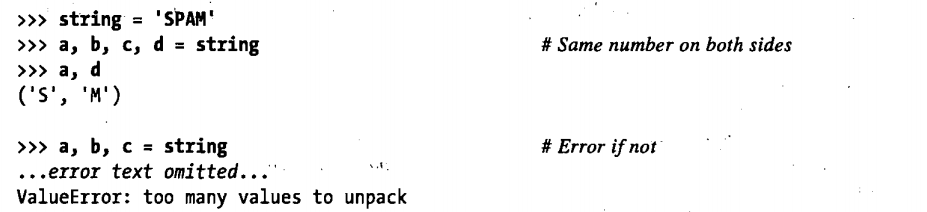


85、若想知道字符串是否为数字，可以用isdigit方法检查字符串的内容



85、赋值语句：

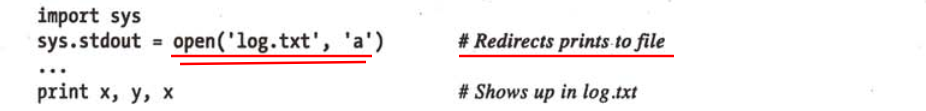




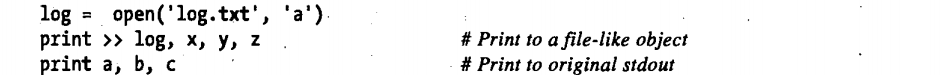
增强赋值：假如对象具有可变性，那么“+=”对列表做原处修改，并不像“+”合并，总是生成新对象。

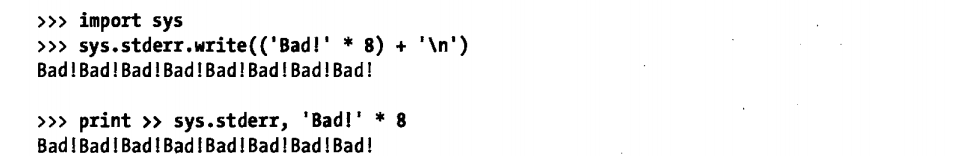
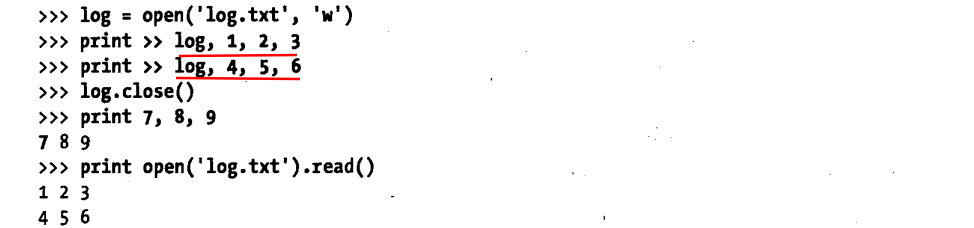
86、表达式语句和在原处的修改：对列表调用append、sort或reverse这类在原处修改的运算，一定是对列表做原处的修改，这些方法在列表修改后并不会把列表返回。

87、重定向输出流：载入sys模块并调用sys.stdout可以使得print函数重定向输出。



也可以调用print>>进行重定向

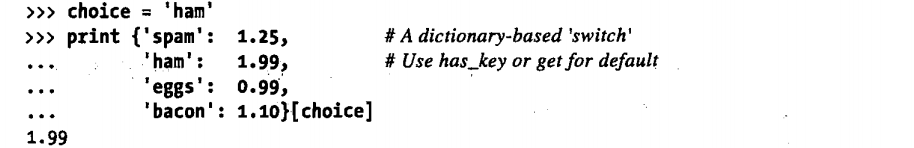


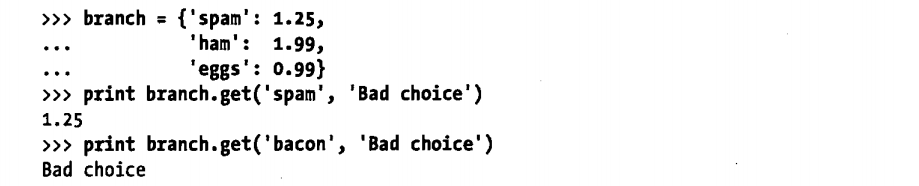


88、print和stdout：print语句和sys.stdout之间的等效是很重要的。因为print语句是传送文本给sys.stdout.write方法，可以把sys.stdout赋值给一个对象，而由该对象的write方法通过任意方式处理文字。

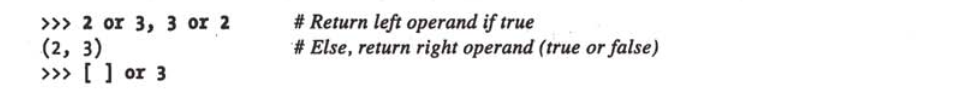
89、Python中的多路分支可以写成一系列的if/elif测试，或者对字典进行索引运算或搜索列表（因为字典和列表在运行时创建）。

以下就是模拟C语言的swich..case，根据变量选择特定的动作





90、注意Python中会返回真或假对象，而不是值True或False。



就像上述例子，or会返回第一个为真的操作对象，否则返回最后一个操作对象。

而and会停止在第一个为假的操作对象，否则返回最后一个操作对象



等价于



等价于

