1.切片操作三个参数的意义：

切片是Python序列的重要操作之一，适用于列表、元组、字符串、range对象等类型。

切片使用2个冒号分隔的3个数字来完成***，第一个数字表示切片开始位置（默认为0），第二个数字表示切片截止（但不包含）位置（默认为列表长度），第三个数字表示切片的步长（默认为1），当步长省略时可以顺便省略最后一个冒号。可以使用切片来截取列表中的任何部分，得到一个新列表，也可以通过切片来修改和删除列表中部分元素，甚至可以通过切片操作为列表对象增加元素。***

与使用下标访问列表元素的方法不同，切片操作不会因为下标越界而抛出异常，而是简单地在列表尾部截断或者返回一个空列表，代码具有更强的健壮性。

2.注释方式：

一个好的、可读性强的程序一般包含30%以上的注释。常用的注释方式主要有两种：

(1) 以#开始，表示本行#之后的内容为注释

(2) 包含在一对三引号'''...'''或"""..."""之间且不属于任何语句的内容将被解释器认为是注释

3.zip函数：

>>> aList = [1, 2, 3]

>>> bList = [4, 5, 6]

>>> cList = zip(a, b)

>>> cList

>>> list(cList)

[(1, 4), (2, 5), (3, 6)]

4.不同进制数的表示：

十进制整数如，0、-1、9、123

十六进制整数，需要16个数字0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f来表示整数，必须以0x开头，如0x10、0xfa、0xabcdef

八进制整数，只需要8个数字0、1、2、3、4、5、6、7来表示整数，必须以0o开头，如0o35、0o11

二进制整数、只需要2个数字0、1来表示整数，必须以0b开头如，0b101、0b100

5.append函数：

使用列表对象的append()方法，原地修改列表，是真正意义上的在列表尾部添加元素，速度较快，也是推荐使用的方法。

>>> aList.append(9)

>>> aList

[3, 4, 5, 7, 9]

6.列表的乘法：

使用乘法来扩展列表对象，将列表与整数相乘，生成一个新列表，新列表是原列表中元素的重复。

>>> aList = [3,5,7]

>>> aList = aList\*3

>>> aList

[3, 5, 7, 3, 5, 7, 3, 5, 7]

7.元组的特性：

(1)元组与列表的区别:

元组中的数据一旦定义就不允许更改。

元组没有append()、extend()和insert()等方法，无法向元组中添加元素；

元组没有remove()或pop()方法，也无法对元组元素进行del操作，不能从元组中删除元素。

内建的tuple( )函数接受一个列表参数，并返回一个包含同样元素的元组，而list( )函数接受一个元组参数并返回一个列表。从效果上看，tuple( )冻结列表，而list( )融化元组。

(2)元组的优点:

元组的速度比列表更快。如果定义了一系列常量值，而所需做的仅是对它进行遍历，那么一般使用元组而不用列表。

元组对不需要改变的数据进行“写保护”将使得代码更加安全。

一些元组可用作字典键（特别是包含字符串、数值和其它元组这样的不可变数据的元组）。***列表永远不能当做字典键使用，因为列表不是不可变的。***

8.字典的键：

字典是键值对的无序可变集合。

定义字典时，每个元素的键和值用冒号分隔，元素之间用逗号分隔，所有的元素放在一对大括号“｛”和“｝”中。

字典中的每个元素包含两部分：键和值，向字典添加一个键的同时，必须为该键增添一个值。

字典中的键可以为任意不可变数据，比如整数、实数、复数、字符串、元组等等。

字典中的键不允许重复。

9.python语言的特点：

 python是一门跨平台、开源、免费的解释型高级动态编程语言，同时也支持伪编译将Python源程序转换为字节码来优化程序和提高运行速度，并且支持使用py2exe工具将Python程序转换为扩展名为“.exe”的可执行程序，可以在没有安装Python解释器和相关依赖包的Windows平台上运行。

 python支持命令式编程、函数式编程，完全支持面向对象程序设计，语法简洁清晰，并且拥有大量的几乎支持所有领域应用开发的成熟扩展库。

 python可以把多种不同语言编写的程序融合到一起实现无缝拼接，更好地发挥不同语言和工具的优势，满足不同应用领域的需求。

10.python命名规则：

变量名必须以字母或下划线开头，但以下划线开头的变量在Python中有特殊含义，本书后面第6章会详细讲解；

变量名中不能有空格以及标点符号（括号、引号、逗号、斜线、反斜线、冒号、句号、问号等等）；

不能使用关键字作变量名，可以导入keyword模块后使用print(keyword.kwlist)查看所有Python关键字；

不建议使用系统内置的模块名、类型名或函数名以及已导入的模块名及其成员名作变量名，这将会改变其类型和含义，可以通过dir(\_\_builtins\_\_)查看所有内置模块、类型和函数；

变量名对英文字母的大小写敏感，例如student和Student是不同的变量。

1. python中的空类型：

空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值。

1. python中的运算符：



1. python可变序列与不可变序列：

Python在heap中分配的对象分成两类：可变对象与不可变对象。所谓可变对象是指，对象的内容可变，而不可变对象是指对象内容不可变。列表、集合、字典等可变序列类型，元组，字符串，int是不可变序列。

1. sort函数默认情况下是按升序排列：
2. lambda函数：

lambda表达式可以用来声明匿名函数，即没有函数名字的临时使用的小函数，只可以包含一个表达式，且该表达式的计算结果为函数的返回值，不允许包含其他复杂的语句，但在表达式中可以调用其他函数。

>>> f=lambda x,y,z:x+y+z

>>> f(1,2,3)

6

>>> g=lambda x,y=2,z=3:x+y+z

>>> g(1)

6

>>> g(2,z=4,y=5)

11

>>> L=[(lambda x:x\*\*2),(lambda x:x\*\*3),(lambda x:x\*\*4)]

>>> print(L[0](2),L[1](2),L[2](2))

4 8 16

>>> D={'f1':(lambda:2+3),'f2':(lambda:2\*3),'f3':(lambda:2\*\*3)}

>>> print(D['f1'](),D['f2'](),D['f3']())

5 6 8

>>> L=[1,2,3,4,5]

>>> print(list(map((lambda x:x+10),L)))

[11, 12, 13, 14, 15]

>>> L

[1, 2, 3, 4, 5]

>>> def demo(n):

 return n\*n

>>> demo(5)

25

>>> a\_list=[1,2,3,4,5]

>>> list(map(lambda x:demo(x),a\_list))

[1, 4, 9, 16, 25]

>>> data = list(range(20))

>>> data

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

>>> import random

>>> random.shuffle(data)

>>> data

[4, 3, 11, 13, 12, 15, 9, 2, 10, 6, 19, 18, 14, 8, 0, 7, 5, 17, 1, 16]

>>> data.sort(key=lambda x:x)

>>> data

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

>>> data.sort(key=lambda x:len(str(x)))

>>> data

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

>>> data.sort(key=lambda x:len(str(x)),reverse=True)

>>> data

[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

简答题

1. 为什么尽量从列表尾部进行列表的增加和删除？

应尽量从列表尾部进行元素的增加与删除操作。列表的insert()可以在列表的任意位置插入元素，但由于列表的自动内存管理功能，insert()方法会涉及到插入位置之后所有元素的移动，这会影响处理速度，类似的还有后面介绍的remove()方法以及使用pop()函数弹出列表非尾部元素和使用del命令删除列表非尾部元素的情况。

1. 异常与错误什么区别：