1. Dictionary 是 Python 的内置数据类型之一, 它定义了键和值之间一对一的关系。
2. 每一个元素都是一个 key-value 对, 整个元素集合用大括号括起来
3. 您可以通过 key 来引用其值, 但是不能通过值获取 key
4. 在一个 dictionary 中不能有重复的 key。给一个存在的 key 赋值会覆盖原有的值。  在任何时候都可以加入新的 key-value 对。这种语法同修改存在的值是一样的。
5. 当使用 dictionary 时, 您需要知道: dictionary 的 key 是大小写敏感的
6. Dictionary 不只是用于存储字符串。Dictionary 的值可以是任意数据类型, 包括字符串, 整数, 对象, 甚至其它的 dictionary。在单个 dictionary 里, dictionary 的值并不需要全都是同一数据类型, 可以根据需要混用和匹配。  Dictionary 的 key 要严格多了, 但是它们可以是字符串, 整数和几种其它的类型 (后面还会谈到这一点) 。也可以在一个 dictionary 中混用和配匹 key 的数据类型
7. del 允许您使用 key 从一个 dictionary 中删除独立的元素。
8. clear 从一个 dictionary 中清除所有元素。注意空的大括号集合表示一个没有元素的 dictionary。

1. list是一个使用方括号括起来的有序元素集合。
2. List 可以作为以 0 下标开始的数组。任何一个非空 list 的第一个元素总是 li[0]
3. 负数索引从 list 的尾部开始向前计数来存取元素。任何一个非空的 list 最后一个元素总是 li[-1]。 如果负数索引使您感到糊涂, 可以这样理解: li[-n] == li[len(li) - n]。 所以在这个 list 里, li[-3] == li[5 - 3] == li[2]。
4. 您可以通过指定 2 个索引得到 list 的子集, 叫做一个 “slice” 。返回值是一个新的 list, 它包含了 list 中按顺序从第一个 slice 索引 (这里为 li[1]) 开始, 直到但是不包括第二个 slice 索引 (这里为li[3]) 的所有元素。
5. 如果将两个分片索引全部省略, 这将包括 list 的所有元素。但是与原始的名为 li 的 list 不同, 它是一个新 list, 恰好拥有与 li 一样的全部元素。li[:] 是生成一个 list 完全拷贝的一个简写。
6. append 向 list 的末尾追加单个元素。
7. insert 将单个元素插入到 list 中。数值参数是插入点的索引。请注意, list 中的元素不必唯一, 有有两个独立的元素具有 'new' 这个相同的值。
8. extend 用来连接 list。请注意不要使用多个参数来调用 extend, 要使用一个 list 参数进行调用。
9. Lists 的两个方法 extend 和 append 看起来类似, 但实际上完全不同。 extend 接受一个参数, 这个参数总是一个 list, 并且添加这个 list 中的每个元素到原 list 中
10. 另一方面, append 接受一个参数, 这个参数可以是任何数据类型, 并且简单地追加到 list 的尾部。 在这里使用一个含有 3 个元素的 list 参数调用 append 方法。
11. index 在 list 中查找一个值的首次出现并返回索引值。
12. 要测试一个值是否在 list 内, 使用 in, 如果值存在, 它返回 True, 否则返为 False 。
13. remove 从 list 中删除一个值的首次出现。
14. pop 是一个有趣的东西。它会做两件事: 删除 list 的最后一个元素, 然后返回删除元素的值。请注意, 这与 li[-1] 不同, 后者返回一个值但不改变 list 本身。也不同于 li.remove(*value*), 后者改变 list 但并不返回值。
15. Lists 也可以用 + 运算符连接起来。 *list* = *list* + *otherlist* 相当于 *list*.extend(*otherlist*)。 但 +运算符把一个新 (连接后) 的 list 作为值返回, 而 extend 只修改存在的 list。 也就是说, 对于大型 list 来说, extend 的执行速度要快一些。
16. Python 支持 += 运算符。 li += ['two'] 等同于 li.extend(['two'])。 += 运算符可用于 list, 字符串和整数, 并且它也可以被重载用于用户自定义的类中。
17. \* 运算符可以作为一个重复器作用于 list。 li = [1, 2] \* 3 等同于 li = [1, 2] + [1, 2] + [1, 2], 即将三个 list 连接成一个。
18. Tuple是不可变的list.一是创建了一个tuple就不能以任何方式改变它.
19. 定义tuple与定义list的方式相同,除了整个元素集是用小括号包围的而不是方括号.
20. Tuple的元素与list一样按定义的次序进行排序.Tuples的索引与list一样从0开始,所以一个非空的tuple的第一个元素总是t[0].
21. 负数索引与 list 一样从 tuple 的尾部开始计数。
22. 与 list 一样分片 (slice) 也可以使用。注意当分割一个 list 时, 会得到一个新的 list ；当分割一个 tuple 时, 会得到一个新的 tuple。
23. Tuple 没有方法：没有 append 或 extend 方法、没有 remove 或 pop 方法、没有 index 方法、可以使用 in 来查看一个元素是否存在于 tuple 中。

 　　其中list()函数与tuple()函数接受可抚今迭代的对象(比如一个序列)作为参数,并通过浅拷贝数据来创建一个新的列表与元组.虽然字符串也是序列类型,但一般它们不用于list()与tuple().更多的情况下,它们用于在两种类型之间进行轩换,比如你需要把一个已有的元组转换成列表类型(然后你可以修改它的元素),反之也是.

     alist=['123','456'];

     atuple=tuple(alist);

     print atuple

     >>>('123', '456')

    alist==atuple

    >>> False

    alist2=list(atuple)

   alist2==alist

   >>>True

   alist is alist2

   >>>False

   再说id()确认一下, [id(x)  for x in alist,atuple,alist2]

    >>>[10903800,12003900,11730280]

    所以无论是list()还是tuple()都不可能做到完全转换,也就是说传递一个元组到list不会变成真正的列表,同时传递一个列表到tuple()也不会变成一个真正的元组.虽然前后两个对象有相同的数据集合,但是变量指向的不是同一个对象.需要注意的是:它们的所有值相同,一个列表也不可能"等于"一个元组的.