python 程序设计——列表

1、 求一个正整数（大于零的整数）除了 1 和它本身以外的升序排列的因数列表。

2、 求 a 和 b 两个正整数（大于零的整数）除了 1 以外的升序排列的公因数列表。

测试用例：

输入：3, 61　　=>输出：[]
输入：40, 120　　=>输出：[2, 4, 5, 8, 10, 20, 40]
输入：12, 18　　=>输出：[2, 3, 6]

3、 给定正整数 n，求[2, n)内的所有回文质数（既是质数又是回文数）构成的降序列表

测试用例：

输入：1　　=>输出：　[]
输入：11　　=>输出：[7, 5, 3, 2]
输入：200　　=>输出：[191, 181, 151, 131, 101, 11, 7, 5, 3, 2]

4、 求列表的中位数。中位数是按升序或降序排列的一组数据中居于中间位置的数。如有偶数个数据，则取中间两个数的平均值的取整值为中位数。

5、 求列表中有重复值的元素个数。

6、 给定一个整数列表，对列表中重复出现的整数，只保留一份，删除重复的整数，请将结果以列表形式返回。

7、 给定一个整数列表，请将奇数统一放在前面，偶数统一放在后面，并且奇数之间的相对顺序不变，同样偶数之间的相对位置也不变，将重新排序的结果以列表形式返回。输入一定是满足要求的整数列表，无需做边界条件判断。

8、 给定一个正整数列表 lst，请将列表中元素重新排序。奇数集中存放在列表首部，偶数集中存放在列表尾部，奇数增序排列，偶数降序排列。返回一个重新排序列表。

9、 给定一个整数列表，求其中出现次数最多的元素。

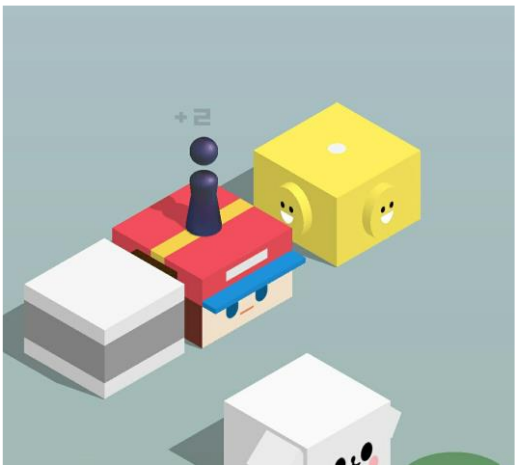
10、　已知一个等差数列缺失了一个值，求出这个缺失值。

测试用例：

输入：[1, 2, 3, 4, 5, 8, 7, 9, 10]　=> 输出：6
输入：[3, 4, 1, 2, 5, 6, 7, 9, 10]　=> 输出：8
输入：[2, 4, 6, 8, 12]　　=> 输出：10

11、　已知一个正整数集合，如果将其中的数按从小到大的顺序排列，是缺失一个数的等比数列。请求出这个缺失的值。

12、　有一个称为跳一跳的游戏，曾经非常风靡。



现在有一个简化后的跳一跳，玩家顺序向前跳跃，得分规则如下：

- 如果没有跳到下一个方块上则游戏结束；
- 如果跳到了方块上，但没有跳到方块的中心则获得 1 分；
- 如果跳到方块且在中心时，若上一次的得分为 1 分或这是本局游戏的第一次跳跃则此次得分为 2 分，否则此次得分比上一次得分多 2 分（即连续跳到方块中心时，总得分将 +2，+4，+6，+8...）。

现在有一个列表 lst 给出一个玩家跳一跳的过程，请你求出他游戏的得分。lst 的元素的取值一定是整数 0、1、2 之一，具体意义如下：

- 0 表示此次跳跃没有跳到方块上（此时游戏结束）；
- 1 表示此次跳跃跳到了方块上但是没有跳到中心；
- 2 表示此次跳跃跳到了方块上并且跳到了方块中心；

测试用例：

输入：[1,1,2,2,2,1,1,2,2,0] =>输出：22
输入：[0] =>输出：0
输入：[2,2,2,2,2,2,2] =>输出：56

13、 给定一个元素全部是整数的列表 lst，如果有一对数字（i，j），满足 lst[i]等于 lst[j] 并且 i 小于 j，那么 i 和 j 可以称为一个好数对，请返回 lst 的好数对的个数。

测试样例：

输入:[1,2,3,1,1,3] =>输出：4 说明：分别是(0, 3)、(0, 4)、 (3, 4)、 (2, 5)
输入: [1,1,1,1] =>输出：6 说明：每组数字都是好数对
输入: [1,2,3] =>输出：0 说明：没有好数对

14、 给定一个列表，包含整数、字符串、浮点数三种类型的元素。不同类型之间比较的规则：字符串>浮点数>整数；同类型元素之间则正常比较。请按照大小规则对列表从大到小排序，返回排序后的列表。

15、 一个正整数的头部和尾部分别是其第一位数字和最后一位数字。比如 123 的头部是 1，尾部是 3。5 的头部和尾部都是 5。给定一个正整数列表，将其中每个元素用它的头部和尾部进行替换，从而得到一个包含若干数字的列表 T，将 T 中的质数保持不变并看成分隔符，可以把 T 分割成若干个子序列，对于每个子序列，将其中的数字进行合并得到一个新的数，返回合并之后的列表。比如，对于列表[1, 234, 5, 6, 70, 890]，替换之

后的列表是[1, 1, 2, 4, 5, 5, 6, 6, 7, 0, 8, 0]，其中包含的子序列有<1,1>, <4>, <6,6>和<0,8,0>，合并之后的列表是[11, 2, 4, 5, 5, 66, 7, 80]。

测试用例：

输入：[1, 234, 5, 6, 7, 890] =>输出：[11, 2, 4, 5, 5, 66, 7, 7, 80]

输入：[12, 34, 56, 78, 90] =>输出：[1, 2, 3, 4, 5, 6, 7, 890]

输入：[123] =>输出：[1, 3]

16、 将一个 m 行 m 列的矩阵 A 按照对角线转置成一个 2m-1 行 m 列的矩阵 B, 即 A 的第 i 条对角线上的元素成为 B 的第 i 行上的元素（为保证 B 的每行都有 m 列，空缺位置用 0 填充）。下面是一个 3 行 3 列矩阵 A 按照对角线转置的结果。

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 2 & 0 \\ 7 & 5 & 3 \\ 8 & 6 & 0 \\ 9 & 0 & 0 \end{bmatrix}$$

在上面的例子中，A 的第 0 条对角线有 1 个元素（1），第 1 条对角线有 2 个元素（4，2），第 2 条对角线有 3 个元素（7，5，3），第 3 条对角线有 2 个元素（8，6），第 4 条对角线有 1 个元素（9）。

测试用例：

输入:[[1,2,3], [4,5,6], [7,8,9]] =>输出：[[1,0,0], [4,2,0], [7,5,3], [8,6,0], [9,0,0]]

输入:[[1,2], [3,4]] =>输出：[[1,0], [3,2], [4,0]]

17、 有一个二维列表 lst 存储了 n 个不同学号学生的两门课考试成绩。二维列表的每一个元素是一个列表，其中依次存储学号、Python 成绩、英语成绩。学号应该为 3 位正整数，成绩应该为 0-100 之间的整数。编写程序删除其中的非法数据，然后按照总分从高到低排序；当总分相同时 Python 成绩高的排序在前；如果总分单课分都相同，学号小的排序在前。返回最终得到的二维列表。

测试用例：

输入:[[100,3,4],[101,4,3]] =>输出： [[101, 4, 3], [100, 3, 4]]

输入: [[2,3,5],[100,3,4],[101,4,3]] =>输出：[[101, 4, 3], [100, 3, 4]]

输入: [[2,3,5],[100,173,4],[101,4,3]] =>输出： [[101, 4, 3]]

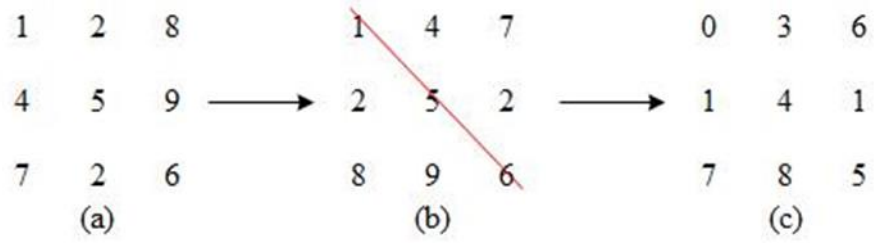
18、 现有一个列表 lst，其中有若干个元素，每个元素是一个元组，每个元组包含一个字符串和一个整数，字符串存了学号（合法学号是 9 位数字符号），整数存储了志愿者活动的时间(单位是小时，每次志愿者活动的合法时间是 1-3 小时之间)。请编写程序，去除学号或者时间不合法的记录，然后计算出每位同学的志愿者活动总时间， 按照总时间从大到小排序，时间相同的时候按照学号升序排序。最后以元组的形式返回排名第一的同学学号和志愿者活动的总时间。

测试用例:

输入	返回
[('192740506',3),	('192740101', 4)

('192740101',2) ('192740101',2)]	解释： '192740101'的总时间最大。
[('192740506',3), ('192740A01',2), ('192740101',3)]	('192740101', 3) 解释： 第二个元组中学号不合法，'192740101'的时间和'192740506'时间相同，但是学号更小
[('19274056',3), ('192740A01',2), ('192740101',3)]	('192740101', 3) 解释： 有两个不合法学号'19274056'和'192740A01'

19、 给定一个二维列表 lst，该列表存储了如下图(a)所示的矩阵，以矩阵的对角线(下图红色线条)为轴进行翻转得到下图(b)，将图(b)中的所有元素减去对角线上的最小值即可得到图(c)所示的矩阵。最后，以二维列表的形式返回最终的矩阵。本例中图(a)与(b)对应的二维列表分别是[[1,2,8],[4,5,9],[7,2,6]]和[[1,4,7],[2,5,2],[8,9,6]]，对角线上的最小值是 1，返回结果对应的二维列表是[[0,3,6],[1,4,1],[7,8,5]]。



测试样例：

输入：[[1,2,8],[4,5,9],[7,2,6]] =>输出： [[0,3,6],[1,4,1],[7,8,5]]

输入：[[4, 5], [6, 7]] =>输出： [[0, 2], [1, 3]]

20、