大体思路就是回溯搜索

但是单纯的回溯搜索比较慢，所以我们加两点改进：

1. 在搜索的时候每把一个数字写进一个格子里的时候同时把这个约束加到其所在的行，列，和方框，让之后搜索到这些位置的时候可以不用考虑取这个值的可能性从而缩减搜索空间。
2. 每次选择下一个要填的位置的时候总是选那个受到约束最强的，也就是可能的取值数目最小的那个位置来进行搜索，这很符合手工求解数独时的行为。（MRV启发式）

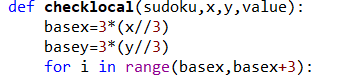
为了实现1，我们对每一行，每一列，每一方框都维护一个集合，里面装的是这一行/列/方框里还剩下的没填的数字（可能的取值集合）。然后在对某一个位置进行搜索的时候我们就可以通过对这个位置所在的行/列/方框的三个集合求交集，就是这个位置还可能填的值。

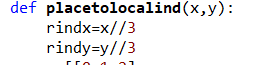
为了实现2，我们只需要遍历所有还没填的格子，选出取值集合大小最小的那个，取值集合和上面一样用三个对应集合求交来获得。

具体的情况看代码的详细注释

参考：《人工智能：一种现代的方法》6.3节

另：我用的py3





这几处//如果是py2的话改成/