Cheat Sheet

# 数据类型

一、列表

1. 列表排序：list.sort(reverse=True or False)
2. 单行多数据：map（type, input().split()）
3. isinstance(a,type)：判断变量类型
4. 建立一个n项都是0的列表，可以通过为每项重新赋值来记录运算结果；用列表索引替代多个if
5. 判断列表是否相同：is或等号
6. 排序后返回新列表：sorted（list）
7. 推导式：【新列表加入元素 for 原列表元素 in原列表】，新列表加入的元素可以是原列表元素的函数返回值，也可以和原元素无关，直接同位替换
8. 列表对应跟随排序：

# 以鞋子品牌和价格为例，创建两个列表

shoes = ["huili","lining","anta","tebu"]

price = [20,10,50,40]

# 将品牌跟价格合在一起

shoes\_price = lists（zip(shoes,price)），排序后再用推导式拆开

1. append函数可以直接添加一段列表

10、切片：就是从前面的索引切到后面的索引减一

11、创建二阶空列表：尽量用推导式，用[[0]]相加可能会出现拷贝问题

# 使用list()转为列表形式

print(list(shoes\_price))

1. 字符串
2. f可以把字符串中的变量保留
3. “ ”可以在连接字符串时添加空格
4. Ord（）可以比较字母顺序
5. 数据类型判断
6. 判断类型：isinstance（a,type）
7. 数字
8. range默认起点为0，定义起点后就从该数字开始
9. ‘/’的除法一定返回浮点数，‘//’的除法返回分子同类型的数

高效质数表：pl = [1]\*(10\*\*6+1)  
for i in range(2,10\*\*3+1,1):  
 if pl[i] == 1:  
 for t in range(2\*i,10\*\*6+1,i):  
 pl[t] = 0

3、保留a的n位小数：'%.2f'%a

1. 集合
2. set函数可以将字符串的每个字符转成集合的元素
3. set（lists）可以将列表转成集合；增减元素是add和remove
4. set的元素不能是可变类型
5. 迭代器和生成器
6. 字典
7. defaultdict（list）可以直接导入变量作为键，避免键不存在时报错；a[x].append(y)，x为键，y为值
8. 遍历键值对：

for k,v in dict\_1.items():

print("字典中的键值对为:(%s:%s)"%(k,v))

3、快速创建字典：dict.fromkeys(seq[, value])

# 功能

1. 查找
2. 二分查找细节：若while lo<hi，则hi应每次更新成mid；若while lo<=hi，则hi应每次更新成mid-1；除以二时应用整除
3. Bisect\_left(lists,target)
4. 函数
5. 直接运行：不加def，填上需要的参数
6. 中断运行：遇到return时就中断
7. 全局变量：# pylint: skip-file加在第一行
8. 堆

1、可以维护一个最小值在首位的列表

十一、相邻位置遍历

1. arounds =

十二、排序

1. 倒序不用加reverse，直接在key里加负号就行

十三、字符串运算

1、eval函数：将运算符（str类型）和数字（str类型）用+连接，可以直接算出结果

输入

1、对于空行，应当加上一个没有赋值的input（）

输出

1. 不换行：end=‘分隔符’
2. 列表转字符串：'分隔符'.join(list)

# 算法

一、循环

1. 简单顺序循环不用把结果加入列表，直接把一个变量重置为结果
2. For+else循环为避免输出多个结果，需在for循环中加入break
3. 根据每次迭代的结果选择下一循环的起点：打标记，遇到被标记的数据就continue
4. 在循环中对计数器迭代，就可以直接用while循环，省去循环内的条件判断

二、递归

1、优化：

import sys

sys.setrecursionlimit(1 << 30) # 将递归深度限制设置为 2^30

from functools import lru\_cache

@lru\_cache(maxsize=None)

2、延迟读取：一个函数调用完后才会开始调用下一个

三、DP

1、每个子问题可能不只由上一个子问题决定——可能与上一个子问题有相互干涉（上一个子问题取最优时，这个不一定取到最优），应当找到最近的不相干子问题

2、dp数组中可以包含一些实际不存在的情况，以保证遍历的连贯性

3、dp有时是对“可能性”的遍历——把每个选项的True和False都考虑一遍，自然能生成最优解——常见的情况是一种可能性对应到唯一确定的子问题，将子问题相加即可。

4、最优解的另一面：没有任何优化空间，需要时应整个舍弃

5、公共子序列：如果两项相同，则用左上角的值（这两个相同项都不含）加上

6、若各个物品某个指标的权值一样（比如个数），则可以将原本“可选这一物品”的坐标轴变为“实际选了几件”，这样就可以保证该轴坐标最大的即为（这一指标的）最优解。

注意此情况下必须从后往前遍历以防止重复计算

7、当坐标轴为“可以选以上物品”时，最优解一定位于底部（最全的可能性）；若为“真正选了以上物品”，则最优解位置不定。

四、DFS

1、全局变量：# pylint: skip-file防止报错

五、最短路径

1、堆顶是什么：到该节点的最短路径（只有第一次到时能保证）（证明：堆中其他点本来就远，到这个点一定还要经过中转，不可能更短）；

2、堆里有什么：未知真实最短距离的所有节点的已知最短距离

3、有向图中的应用：可以和bfs结合，将原本步数最少的在队头的规则变成自定义的指标（证明：堆中包含了已有步数能到达的全部节点，未进堆的节点距离不可能比堆中的小）

4、标记方法：列表中记录已知最短路径（默认inf），若新路径比表中的长，则不用考虑；

当标记的信息不只坐标时（小游戏），应当用defaultdict