1. 列表的逗号间隔输出

```
a=[1,2,3,4,5,5,5,7,8,9]

print(",".join(map(str,a)))

#join 时要求数据形式为字符串
```

2. 字典自动创建默认值

```
from collections import defaultdict

a=defaultdict(int)
```

3. enumerate(fruit,start=x)代表设定第一个的索引为 x，同时返回索引和元素

4. itertools 遍历

```
fruits=['a','b','c','d']
import itertools
for x in itertools.permutations(fruits,2):
    print(''.join(x))#生成所有长度为 2 的排列,12 种

for x in itertools.product(fruits,repeat=2):#或者写 fruits,fruits

    print(x)#生成二重笛卡尔积，16 种
```

5. deque 双端队列（BFS 常用）

```
from collections import deque

d=deque(maxlen=10)

d.append([1,2])

d.appendleft([3,4])

x,y=d.pop()

z,w=d.popleft()
```

6. heapq 模块（dijkstra 找最短路径常用）

heapq 维护一个最小堆，即每个父节点的值都小于等于其子节点的值，

heap[k]<=min(heap[2*k+1],heap[2*k+2])

```
import heapq

q=[('HO',1,1),('HH',1,1)]

heapq.heapify(q)#转化为最小堆

heapq.heappush(q,('OO',1,1))

i,x,y=heapq.heappop(q)#弹出最小元素，保持堆的性质
```

7. bisect 模块 二分法

```
import bisect

a=[1,2,4,4,8]

print(bisect.bisect_left(a,4))#输出 2，最左可以插入的位置

print(bisect.bisect_right(a,4))#输出 4，最右可以插入的位置
```

8. 日期与时间

```
import calendar,datetime

print(calendar.isleap(2020))#返回 true，是闰年

print(datetime.datetime(2024,12,25).weekday())#星期三，输出 2
```

9. 有序字典（按照插入的顺序排序）

```
import collections

od=collections.OrderedDict()

od['b']=3

od['a']=2

print(od)#输出 OrderedDict({'b': 3, 'a': 2})
```

10.排序

```
lis=[[0,1],[1,1],[2,1],[1,2]]

lis.sort(reverse=True, key=lambda x:(x[0],x[1]))
```

11.深拷贝

lis1=lis 会造成浅拷贝的问题，lis1=lis[:]就不是浅拷贝

```
import copy
original=[1,2,[3,4]]
copied=copy.deepcopy(original)
```

12.素数筛法

欧拉筛（筛 a 以内的素数）

```
    integer=[True]*a
prime=[]
integer[0]=False
for i in range(1,a):
    if integer[i]:
        prime.append(i+1)
        s=0
        while s<=len(prime)-1 and (i+1)*prime[s]<=a:
            integer[(i+1)*prime[s]-1]=False
            s+=1
    else:
        t=0
        while (t==0 or (i+1)%prime[t-1]!=0) and t<=len(prime)-1
and (i+1)*prime[t]<=a:
            integer[(i+1)*prime[t]-1]=False
            t+=1
    i+=1
```

直接判断一个数是否为素数：用“5 及以上的素数都是 6k 加减 1 型”优化

13.ASC II 码

ord('a')=97　ord('z')=122

ord('A')=65　ord('Z')=90

14.数据组数未知时的输入格式

```
while True:
    try:


    except EOFError:
        break
```

15.全局变量导致 compile error 时

```
#pylint:skip-file
```

16.需要整段输入时的输入格式

```
lines=list(sys.stdin.read().split())
```

17.用 lru_cache 缓存递归结果:

```
from functools import lru_cache
@lru_cache(maxsize=None)
def ……
```

18.手动提高递归深度

```
import sys
sys.setrecursionlimit(200000)
```

19.多重背包二进制优化

```
def binary_optimized_multi_knapsack(weights, values,
quantities, capacity):
```

```
    n = len(weights)
    items = []
    # 将每个物品拆分成若干子物品
    for i in range(n):
        w, v, q = weights[i], values[i], quantities[i]
        k = 1
        while k < q:
            items.append((k * w, k * v))
            q -= k
            k <<= 1#二进制左移一位，右边填充 0
        if q > 0:
            items.append((q * w, q * v))
    # 动态规划求解 01 背包问题
    dp = [0] * (capacity + 1)
    for w, v in items:
        for j in range(capacity, w - 1, -1):#逆序防止重复使用
            dp[j] = max(dp[j], dp[j - w] + v)
    return dp[capacity]
```

20.下一个全排列

```
def next_permutation(nums):
    i = len(nums) - 2
    while i >= 0 and nums[i] >= nums[i + 1]:
        i -= 1
    if i >= 0:
```

```
        j = len(nums) - 1
        while nums[j] <= nums[i]:
            j -= 1
        nums[i], nums[j] = nums[j], nums[i]
    nums[i + 1:] = reversed(nums[i + 1:])
    return nums
nums = [1, 2, 3]
print(next_permutation(nums))  # 输出: [1, 3, 2]
```

21.Potion(greedy)

```
import heapq
def drink():
    global potion,n
    num=0
    p=[]
    for k in potion:
        heapq.heappush(p,k)
        num+=k
        if num<0:
            num-=p[0]
            heapq.heappop(p)
    return len(p)
```

```
n=int(input())
potion=list(map(int,input().split()))
print(drink())
```

22. 田忌赛马

```
while True:
    n=int(input())
    if n==0:
        break
    Tian=list(map(int,input().split()))
    King=list(map(int,input().split()))
    Tian=sorted(Tian)
    King=sorted(King)
    win=0
    head=headk=0
    tail=tailk=n-1
    while head<=tail and tailk>=0:
        if Tian[tail]>King[tailk]:
            tail-=1
            tailk-=1
            win+=200
        elif Tian[head]>King[headk]:
            head+=1
            headk+=1
            win+=200
```

```
        elif Tian[head]==Tian[tail]==King[tailk]==King[headk]:
            break
        else:
            head+=1
            tailk-=1
            win-=200
    print(win)
```

23.中国剩余定理

$$(S):\ \begin{cases} x\equiv a_1 \pmod{m_1} \\ x\equiv a_2 \pmod{m_2} \\ \vdots \\ x\equiv a_n \pmod{m_n} \end{cases}$$

有解的判定条件，并用构造法给出了在有解情况下解的具体形式。

中国剩余定理说明：假设整数$m_1,m_2,\dots,m_n$两两互质，则对任意的整数：$a_1,a_2,\dots,a_n$，方程组(S)有解，并且通解可以用如下方式构造得到：

设$M=m_1\times m_2\times\cdots\times m_n=\prod_{i=1}^{n}m_i$是整数$m_1,m_2,\dots,m_n$的乘积，并设$M_i=M/m_i,\ \forall i\in\{1,2,\cdots,n\}$是除了$m_i$以外的$n-1$个整数的乘积。

设$t_i=M_i^{-1}$为M_i模m_i的数论倒数(t_i为M_i模m_i意义下的逆元)$M_it_i\equiv 1 \pmod{m_i},\ \forall i\in\{1,2,\cdots,n\}.$

方程组(S)的通解形式为$x=a_1t_1M_1+a_2t_2M_2+\cdots+a_nt_nM_n+kM=kM+\sum_{i=1}^{n}a_it_iM_i,\quad k\in\mathbb{Z}.$

在模M的意义下，方程组(S)只有一个解：$x=\left(\sum_{i=1}^{n}a_it_iM_i\right)modM$

24.排队　(身高差<=d 即可交换，求最小字典序)

```
n,d=map(int,input().split())
line=[0]*n
for i in range(n):
    line[i]=int(input())
check=[False]*n
```

```
line_new=[]
while len(line_new)<n:
    buffer=[]
    i=0
    while i<n:
        if check[i]:
            i+=1
            continue
        if len(buffer)==0:
            buffer.append(line[i])
            maxh=line[i]
            minh=line[i]
            check[i]=True
            continue
        maxh=max(maxh,line[i])
        minh=min(minh,line[i])
        if maxh<=line[i]+d and minh>=line[i]-d:
            buffer.append(line[i])
            check[i]=True
        i+=1
    buffer.sort()
    line_new.extend(buffer)
for i in range(n):
    print(line_new[i])
```

25.二分查找

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid  # 返回目标元素的索引
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1  # 如果未找到目标元素，返回 -1
```