```python
#Euclidean algorithm!!!!!!!!!!!!!!!!!
def func2():
    num1 = int(input("Input the first number"))
    num2 = int(input("Input the second number"))
    m = max(num1, num2)
    n = min(num1, num2)
    r = m % n
    while r != 0:
        m = n
        n = r
        r = m % n
    print("%s,%s The greatest common divisor is :%s" %(num1,num2,n))
```

最大公约数

File 文件操作： open(文件路径, mode='r') ，mode 为文件打开模式：r 只读，w 只用于写入 ***e.g.*** with open(outfile, 'w') as b:

- **file.write(str)** 将字符串写入文件，返回的是写入的字符长度。

OOP 面向对象 init 和 单继承：

```python
#类定义
class people:
    #定义基本属性
    name = ''
    age = 0
    #定义私有属性,私有属性在类外部无法直接进行访问
    __weight = 0
    #定义构造方法
    def __init__(self,n,a,w):
        self.name = n
        self.age = a
        self.__weight = w
    def speak(self):
        print("%s 说: 我 %d 岁。" %(self.name,self.age))

# 实例化类
p = people('runoob',10,30)
p.speak()
```

Super(): 子类用函数直接承接到的函数可以继承父类构造的函数！！

isinstance()：判断一个对象是否为该类的一个实例。
issubclass()：判断一个类是否为另一个类的子类。

```python
#单继承示例
class student(people):
    grade = ''
    def __init__(self,n,a,w,g):
        #调用父类的构造函
        people.__init__(self,n,a,w)
        self.grade = g
    #覆写父类的方法
    def speak(self):
        print("%s 说: 我 %d 岁了，我在读 %d 年级"%(self.name,self.age,self.grade))
```

MRO – 多继承时回溯找父类函数的追踪顺序

～异常 assert 断言：表达式条件为 false 的时候触发异常
～异常捕捉可以使用 try： except： 语句。
Try: 执行代码 except: 发生异常时执行的代码
Else: 没有异常时执行的代码
Finally: 不管有没有异常都会输出的代码
chr(int)：转换数字为相应 ASCI 码字符。 **chr(97) >>> 'a'**
ord(str)：转换 ASCII 字符为相应的数字。 **ord('A') >>> 65**
Bin() 十进制转 2 进制
Oct() 十进制转 8 进制
Hex() 十进制转 16 进制 【单词表⬇️】
Maximum Convention 最大公约
Least common multiple 最小公倍数
Division 除法 odd number 奇数 even number 偶数
Dividend, Divisor, and Quotient 被除数 除数 商
Multiplier 乘数 multiplication 乘法 addition 加法 subtraction 减法 operator 运算 times 减掉

prime number 素数 composite number 合数 argument(s)参数
Convert 转换 hint 提示 total 总数 align 对齐
precedence 优先级 indicator 标识符 prefix 前缀 postfix 后缀
Time complexity 时间复杂度 Space complexity 空间复杂度
non- decreasing 非递减 non- increasing 非递增
ascending order 升序排序 descending order 降序排序
element (of list) 元素 no longer 不再是 solution 方案
index out of range 超出索引范围 Stack overflow 堆栈溢出
yield an error 产生一个错误 infinite loop 无限循环
expression 表达式 lazy 惰性
P.s. 一些问是不是的题可以直接举反例!!!

负数	negative number	有效数字	significant digit
正数	positive number	方程	equation
整数	integer	素数/质数	prime numbers
分数	fraction	参数	arguments
有理数	rational number	平行	parallel
角	angle	算术平方根	arithmetic square root
原点	origin	被开方数	radicand
相反数	opposite number	实数	real number
绝对值	absolute value	x 轴	x-axis
乘方、幂	power	无理数	irrational number
底数	base number	坐标	coordinate
指数	exponent	中点	center
多边形	polygon	开平方	extraction of square root
对角线	diagonal	立方根/三次方根	cube root
正多边形	regular polygon	不等式	inequality

数学相关

abs(a)：求取绝对值。abs(-1)
max(list)：求取 list 最大值。max([1,2,3])
min(list)：求取 list 最小值。min([1,2,3])
sum(list)：求取 list 元素的和。 sum([1,2,3]) >>> 6
sorted(list)：排序，返回排序后的 list。
divmod(a,b): 获取商和余数。 divmod(5,2) >>> (2,1)
pow(a,b)：获取乘方数。pow(2,3) >>> 8
round(a,b)：获取指定位数的小数。a 代表浮点数，b 代表要保留的位数。round(3.1415926,2) >>> 3.14
math.ceil(x) ：返回数字的上入整数 如 math.ceil(4.1) 返 5
random.randint(0,9) 生 0～9 之间包括 0 和 9 的随机整数
random() ：随机生成下一个实数，它在[0,1)范围内。

类型转换

float(int/str)：将 int 型或字符型转换为浮点型。float('1') >>> 1.0
str(int)：转换为字符型。str(1) >>> '1'
bool(int)：转换为布尔类型。 str(0) >>> False str(None) >>> False
tuple(iterable)：转换为 tuple。 tuple([1,2,3]) >>>(1,2,3)
set(iterable) ：转换为 set。 set([1,4,2,4,3,5]) >>> {1,2,3,4,5}
set({1:'a',2:'b',3:'c'}) >>> {1,2,3}
hex(int)：转换为 16 进制。hex(1024) >>> '0x400'
oct(int)：转换为 8 进制。 oct(1024) >>> '0o2000'
bin(int)：转换为 2 进制。 bin(1024) >>> '0b10000000000'

（一些常用的函数）

filter(func, iterable)：通过判断函数 fun，筛选符合条件的元素。filter(lambda x: x>3, [1,2,3,4,5,6])
reduce(func, iterable) : **from functools import reduce 别忘了!!**
然后，reduce(lambda x, y: x+y, [1,2,3,4,5]) # 计算 1 加到 5
all(iterable): 如果可迭代对象中有「0, 空串, False」, 就返回 False !!
any(iterable): 只要 iterable 中有一个 element 的 bool(element)是 True，就返回 True!!!!
map(func, *iterable) ：将 func 用于每个 iterable 对象。map(lambda a,b: a+b, [1,2,3,4], [5,6,7]) >>> [6,8,10]
zip(*iterable) ：将 iterable 分组合并。返回一个 zip 对象。list(zip([1,2,3],[4,5,6])) >>> [(1, 4), (2, 5), (3, 6)]
dict(zip([],[])) #字典初始化
reversed(sequence)：生成一个反转序列的迭代器。list(reversed('abc'))>>> ['c','b','a']
upper() & lower() 转大写、转小写

运算符说明	Python运算符	优先级	结合性	优先
小括号	()	19	无	高
索引运算符	x[i] 或 x[i1: i2 [:i3]]	18	左	
属性访问	x.attribute	17	左	
乘方	**	16	右	
按位取反	~	15	右	
符号运算符	+ (正号) 、- (负号)	14	右	
乘除	*、/、//、%	13	左	
加减	+、-	12	左	
位移	>>、<<	11	左	
按位与	&	10	右	
按位异或	^	9	左	
按位或	|	8	左	
比较运算符	==、!=、>、>=、<、<=	7	左	
is 运算符	is、is not	6	左	
in 运算符	in、not in	5	左	
逻辑非	not	4	右	
逻辑与	and	3	左	
逻辑或	or	2	左	
逗号运算符	exp1, exp2	1	左	低

字符串 相关函数：
String.XXXX():
count(str, beg=0,end=len(string)) 返 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数
isupper()/islower() 如果字符串中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是大/小写，则返回 True，否则返回 False
join(seq) 以指定字符串作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串
lower()/ upper() 转换字符串中所有 大写字符为小写/小写字母为大写
max(str) 返回字符串 str 中最大的字母。
min(str) 返回字符串 str 中最小的字母。
replace(old, new [, max]) 将字符串中的 old 替换成 new,如果 max 指定，则替换不超过 max 次。

lstrip() 截掉字符串左边的空格或指定字符。
rstrip() 删除字符串末尾的空格或指定字符。
strip([chars]) 在字符串上执行 lstrip()和 rstrip()

split(str="", num=string.count(str)) 以 str 为分隔符截取字符串，如果 num 有指定值，则仅截取 num+1 个子字符串
swapcase() 将字符串中大写转换为小写，小写转换为大写

title() 返回"标题化"的字符串,就是说所有单词都是以大写开始

List 列表

List:

1 list.append(obj)
在列表末尾添加新的对象

2 list.count(obj)
统计某个元素在列表中出现的次数

3 list.extend(seq)
在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）

4 list.index(obj)
从列表中找出某个值第一个匹配项的索引位置

5 list.insert(index, obj)
将对象插入列表

6 list.pop([index=-1])
移除列表中的一个元素（默认最后一个元素），并且返回该元素的值

7 list.remove(obj)
移除列表中某个值的第一个匹配项

8 list.reverse()
反向列表中元素

9 list.sort(key=None, reverse=False)
对原列表进行排序

10 list.clear()
清空列表

11 list.copy()
复制列表

```
def permutations(ls):
    if not ls:
        return ((),)
    return tuple((ls[i],) + j
            for i in range(len(ls))
            for j in permutations(ls[:i] + ls[i + 1:]))
```

get()中元素字典没有返回 None
pop()的元素字典中没有的话会报错

字典相关函数

dict.copy() 返回一个字典的浅复制
dict.get(key, default=None) 返回指定键的值，如果键不在字典中返回 default 设置的默认值
key in dict 如果键在字典 dict 里返回 true，否则返回 false
dict.items() 以列表返回一个视图对象 eg. [(k1, v1), (k2, v2), (k3, v3), ...]
dict.keys() 返回一个键列表 eg. [k1, k2, k3, ...]
dict.setdefault(key, default=None) 和 get()类似，但如果键不存在于字典中，将会添加键并将值设为 default
dict.update(dict2) 把字典 dict2 的键/值对更新到 dict 里
dict.values() 返回一个值列表 eg. [v1, v2, v3, ...]
pop(key[,default]) 删除字典 key（键）所对应的值，返回被删除的值。
popitem() 返回并删除字典中的最后一对键和值

merge sort 合并排序

```
def merge(left, right):
    results = []
    while left and right:
        if left[0] < right[0]:
            results.append(left.pop(0))
        else:
            results.append(right.pop(0))
    results.extend(left)
    results.extend(right)
    return results

def merge_sort(lst):
    if len(lst) < 2: # Base case!
        return lst
    mid = len(lst) // 2
    left = merge_sort(lst[:mid]) #sort left
    right = merge_sort(lst[mid:]) #sort right
    return merge(left, right)
```

二分查找

```
def binary_search(key, seq):
    if seq == []:
        return False
    mid = len(seq) // 2
    if key == seq[mid]:
        return True
    elif key < seq[mid]:
        return binary_search(key, seq[:mid])
    else:
        return binary_search(key, seq[mid+1:])
```

贪心

「每一步都找（子问题 - 最优子结构）局部最优解」 合成

动态规划

先找状态转移方程 dp[i][j] 递推式 （ = min{ } + xxxx）
!! 找更小的子问题和大问题的关系 !!

Itertools.**permutation()** **组合生成器**：排列的所有可能组合 例：

```
perms = map(list, set(permutations(p1moves)))
```

（组合生成器的手动代码在 list 那块）

【深层拷贝】（浅层拷贝时，深层的地址与原 var 指向同一处!）
Import copy f = copy.deepcopy(a)
 # 浅拷贝不用调包 f = a.copy() 或者 f = a[:]
~ 手动实现 deepcopy 版本：

```
def deepMap(func,seq):
    if seq == []:
        return seq
    elif type(seq) != list:
        return func(seq)
    else:
        return [deepMap(func,seq[0])]+ deepMap(func,seq[1:])
```

l2 = deepMap(lambda x: x.copy() if type(x)==list else x, l)

集合内置方法完整列表

方法	描述
add()	为集合添加元素
clear()	移除集合中的所有元素
copy()	拷贝一个集合
difference()	返回多个集合的差集
difference_update()	移除集合中的元素，该元素在指定的集合也存在。
discard()	删除集合中指定的元素
intersection()	返回集合的交集
intersection_update()	返回集合的交集。
isdisjoint()	判断两个集合是否包含相同的元素，如果没有返回 True，否则返回 False。
issubset()	判断指定集合是否为该方法参数集合的子集。
issuperset()	判断该方法的参数集合是否为指定集合的子集
pop()	随机移除元素
remove()	移除指定元素
symmetric_difference()	返回两个集合中不重复的元素集合。
symmetric_difference_update()	移除当前集合中在另外一个指定集合相同的元素，并将另外一个指定集合中
union()	返回两个集合的并集
update()	给集合添加元素

add() 添加一个元素
update() 添加多个元素
remove(): 将元素 x 从集合 s 中移除，如果元素不存在，则会发生错误
discard():元素不存在，不会发生错误

【一些常考的】

round(1.5) == round(2.5) == 2 #其他时候正常「四舍五入」
print(5**&**6) 按位与（二进制） "**|**"按位或（二进制） *负数补码*
5**^**6 # equal to 3 按位异或（二进制） **~**8 # equal to -9 按位取反

12、list中间添加list或tuple

```
a = [1,2,3,4,5]
a[3:3] = (6,7,8)
print(a)
```

```
b = [ [0] ] * 3
b[0][0] = 1
print(b)                # [ [1], [1], [1] ]
```

```
b = [ [0] for i in range(3) ]  (改进)
b[0][0] = 1
print(b)                # [ [1], [0], [0] ]
```

 # format()

```
{<参数序号>:<格式控制标记>}
b、d、o、x 分别是二进制、十进制、八进制、十六进制。
'{:b}'.format(11)   1011
'{:d}'.format(11)   11
'{:o}'.format(11)   13
'{:x}'.format(11)   b
'{:#x}'.format(11)  0xb
'{:#X}'.format(11)  0XB
```

```
Tempstr = eval(input())
print("0x{0:x},0o{0:o},0b{0:b}".format(Tempstr))
#十六进制、八进制、二进制
100
0x64, 0o144, 0b1100100
```

不可变数据：Number（数字）、String（字符串）、Tuple（元组）；
可变数据：List、Dictionary、Set

is 与 == 区别：is 用于判断两个变量引用对象是否为同一个(同一块内存空间)， == 用于判断引用变量的值是否相等。

'abc'[3]报错 'abc'[3:]>>>' '
andom() ：随机生成下一个实数，它在[0,1)范围内

快速排序 !!

```
a =[7,1,3,2,6,54,4,4,5,8,12,34]
def sort(a,low,high):
    while low < high:
        temp = a[low]
        while low < high and a[high]>=temp:
            high = high-1
        while low<high and a[high]<temp:
            a[low]=a[high]
            low =low+1
            a[high]=a[low]
        a[low]=temp
    return low
def quicksort(a,low,high):
    if low<high:
        middle = sort(a,low,high)
        quicksort(a,low,middle)
        quicksort(a,middle+1,high)
        print(a)
sort(a,0,len(a)-1)
quicksort(a,0,len(a)-1)
print(a)
```

二维数组转置：
[[1,4,7],[2,5,8],[3,6,9]]; 被 **zip(*array)** 之后的结果恰好是 [(1, 2, 3), (4, 5, 6), (7, 8, 9)] 输出的是存放元组的数组

```
print(6//-4)        # -2
print(6//-5, 6%-5)  # -2 -4
print(-6//5, -6%5)  # -2 4
```

```
>>> 3 + 2 == 4.0 + 1
True
>>> 4.0+1
5.0
>>> 1 and True and 3
3
>>> 1 and True and 100 and 0
0
>>> True or 3 or 0
True
```

```
>>> a = 'abcdefz'
>>> min(a)
'a'
>>> max(a)
'z'
>>> '\'abc\'' in 'zabcdef'
False
```

0**0=1

DFS

```
def DFS(graph, s):
    stack = []
    stack.append(s)
    seen = set()
    seen.add(s)
    while (len(stack) > 0):
        vertex = stack.pop()
        nodes = graph[vertex]
        for w in nodes:
            if w not in seen:
                stack.append(w)
                seen.add(w)
        print(vertex)
```

BFS

```
def BFS(graph, s):
    queue = []
    queue.append(s)
    seen = set()
    seen.add(s)
    while (len(queue) > 0):
        vertex = queue.pop(0)
        nodes = graph[vertex]
        for w in nodes:
            if w not in seen:
                queue.append(w)
                seen.add(w)
        print(vertex)
```

not 返回 bool

```
> '' in 'abc'
True
> ' ' in 'abc'
False
> not ''
True
> not 'abc'
False
> not not True
True
> not 0
True
> not 9999
False
```