# 数据类型

数据类型包括：整形，浮点型，字符串  
  
整形：   
内置函数  
bin(): 将十进制的整数转化为二进制  
oct(): 将十进制整数转化为八进制的字符串  
hex(): 将十进制转化为十六进制的数  
  
整数的创建

int()  
print(int(118.12)) 118  
print(int('118')) 118 所以当传入一个参数的时候不管是都会变为整形的  
print(int('1110110'),2) 传递两个参数的时候第一个参数必须是字符串int（字符串），第二个参数是2，8，16  
  
浮点数类型（小数）  
print(0.31313)  
float()  
浮点数的创建  
print(float)  
print(float()) 当不传递任何数的时候返回0.0  
print(float(867))转化为浮点数  
print(float('31'))转化为浮点数  
  
用科学计数法来表示浮点数（很大或者很小的浮点数）

men表示：m乘以10的n次方  
print(2.1e2+1.2e1) 21.0表示2（1乘以10的一次方）  
print(2.3e-4) s0.00023  
  
浮点数存储的不精确性  
print(1.1+2.2-3.3) 4.440892098500626e-16 并不是0  
print(1.1+2.2) 3.3000000000000003  
总结：计算机采用的是二进制存储浮点数的时候是不精确的，可能会存在误差，因此，浮点数的运算要格外的小心  
  
解决方案  
导入模块:decemal或者fraction  
from decimal import Decimal  
Float = Decimal('1.1') +Decimal('2.2')+ Decimal('3.3')  
print("利用Decimal运算之后的结果" + str(Float))  
from fractions import Fraction Fraction类用来表示分数  
print(Fraction(22,10) - Fraction(22,1)) 结果：-99/5  
  
布尔类型

True Falst  
print(5<3) False  
print(True == 1) True  
print(False == 0) True  
print(False==1) False

运算符

运算符的概括：

运算符是一种特殊的符号  
 '+'整数进行运算  
 'and'布尔运算符，逻辑运算符与

常见的运算符  
 标准运算符  
 取余运算符  
 幂运算符  
 布尔运算符  
 比较运算符  
 赋值运算符  
  
  
标准运算符  
+.-.\*./  
  
整数除运算符  
//（表示两个数相除后的整数部分）  
print(9//8) 1 忽略掉小数部分，只保留整数部分。用于计算一个数值对另一个数值的多少倍  
print(9//-4)  
print(9//-4) 9=(-4)\*(-3)+(-3) 最后的余数是0--3之间的  
print(-9//4) -9 = 4 \* (-3) +3 最后的余数是0-4之间的  
  
  
取余运算符  
a%b   
  
  
幂运算符  
'\*\*'  
3\*\*2 = 9 (表示3的平方)  
内置函数  
pow(计算的数,幂运算) pow(3,2)  
  
  
布尔运算符

布尔运算符计算结果仍然是布尔值  
and，or，not  
and：当两个都为True的时候才为True  
or： 当一个为True的时候就为True  
not: 表示非 not True 就为False  
  
  
赋值运算符" = "  
 格式：变量 = 对象  
 变量就相当于一个标签 ：相当于给对象贴了一个标签，标签名字就是对象名  
 i = 18 (python会分配一块内存空间来存储18，相当于给18贴上了名字为i的标签，然后通过名字为i的标签来访问18)  
 j = i  
 在某一时刻上一个标签只能贴在一个对象上，但是一个对象可以  
 注意：变量是没有数据类型的  
 通常情况下一个变量只引用一种数据类型的对象  
 i = 18  
 print(i)  
 i = "dhaiod"  
 print(i) (可以但是不推荐)  
赋值运算符是支持链式赋值的  
a=b=c=18  
赋值运算符支持参数赋值  
+= —= /+ \*=  
a += b a = a + b  
  
  
多个变量同时赋值  
 a,b = [5,7]（相当于把5赋值给a，把7赋值给b）  
 print(a,b)  
 运行结果：5 7  
 a,b = 5,3  
 print(a, b)  
 运行结果：5 3  
总结：赋值运算符的多变可以是一个所有元素的元组或者列表，而且能一次给多个变量赋值，赋值运算符的左右两边的个数必须一样  
 a,\*b = 1,2,3,4,5,7  
 print(a, b)

运行结果：1 [2, 3, 4, 5, 7]

关键字和标识符

关键字  
获取关键字方法1：

help()→keywords  
关键字：  
False class from or  
None continue global pass  
True def if raise  
and del import return  
as elif in try  
assert else is while  
async except lambda with  
await finally nonlocal yield  
break for not  
方法2：  
import keyword  
print(keyword.kwlist)  
关键字时使用不同的颜色来表示的  
  
  
标识符  
标识符的命名规则：  
 1：区分大小写  
 2：不能使用关键字  
 3：不能以数字开头  
 4：不能以空格，制表符，数学符号，中划线  
命名规范：  
 1：见名之意：由一个或者有多个有意义的单词而组成的  
 计算机科学中最难的事情，命名和缓存失效  
 2：单词全部小写，单词之间使用下划线进行分割  
 例如：student\_namei = 3  
I = 5   
  
  
对象None  
 None表示数据值的不存在  
 但是None是占据内存空间的，并不意味着"空"或者"没有意义"  
 None是something而不是nothing  
 print(id(None)) 查看None的内存地址  
None的常见用法  
 a = None重置为None 经常用于变量的初始化，或者重置为None

# 序列类型range

range是一种序列类型，表示不可变的整数序列  
调用内置函数  
 range(stop)  
 range(start,stop)  
 range(start,stop,step)  
 stop表示结束  
 start表示开始  
 step表示步长  
整数序列起始值默认值是0，可以使用参数start指定  
步长的默认值是1  
例子：  
range(5)  
print(range(5))  
print(list(range(65)))  
range()返回值是一个迭代器对象，为了清楚的表示迭代器对象可以将其转化为列表  
print(list(range(1,5,1)))

# **列表**

为了使程序能够存储和操作数据  
提供了列表，元组，字典，他们是python的一本分  
列表的特点：  
 列表中的所有数据都是按照顺序有序排列的  
 每个数据都有两个整形数据的"索引"  
 通过指定的索引总能映射到数据  
 列表中可以存在重复的数据  
 列表可以保存任何数据类型的数据并且多种数据的数据可以混合存储在一块  
 系统根据需要动态的分配内存，不需要声明列表的容量  
  
  
列表的创建

[]和list()  
 1：使用['n',189,True]进行创建  
 当把列表赋值给变量的时候变量名字不要取名给list，l被误读为阿拉伯数字1  
 创建一个空列表[]  
 print([])  
 2: 调用内置函数list  
 print(list(range(0,4)))  
 print(list(['n',189,True]))  
 创建一个空列表：list()

列表的增操作

append(),extend(),insert(),为指定的切片赋予一个新值  
调用方法append(),  
 例子：  
 L = [1,2,3,4,6]  
 L.append(798)  
 print(L)  
 运行结果：[1, 2, 3, 4, 6, 798]  
 注意：使用append追加元素后面可以增加元素  
 例子：  
 L = [1,2,3,4,6]  
 L.append(['d','d','dadjaydgawijy'])  
 print(L)  
 运行结果：  
 [1, 2, 3, 4, 6, ['d', 'd', 'dadjaydgawijy']]  
 总结：在列表的末尾一次只是添加一个元素  
调用方法extend()  
 L = [1,2,3,4,6]  
 L.extend([7,8])  
 print(L)  
 运行结果：[1, 2, 3, 4, 6, 7, 8]、  
 总结：extend是将元素一次添加到列表的末尾  
调用方法insert()

在列表的任意位置一次只添加一个元素  
 例子：  
 L = [1,2,3,4,6]  
 L.insert(3,8) 第一个参数是插入的位置，第二个参数是要插入的元素  
 print(L)  
 运行结果：[1, 2, 3, 8, 4, 6]  
为指定的切片赋予一个新值  
 L = [1,2,3,4]  
 L[1:1] = [7,8]  
 print(L)  
 运行结果：[1, 7, 8, 2, 3, 4]  
  
  
删除列表中的元素

renove(),del语句,给指定的切片赋一个新值  
renove()一次只是删除一个元素  
 L为列表  
 L.renove(元素名称)  
 如果列表中存在多个指定元素那么删除第一个指定元素  
 如果删除的原宿列表中不存在那将会报错  
调用方啊pop()一次只删除一个指定索引的元素，该方法返回被删除元素  
 被删除元素 = L.pop(指定索引)  
使用del语句，一次至少删除一个元素  
 del L[2] 表示删除索引为2的元素  
 del L[1,4]表示索引为123的元素删除  
给指定的切片赋一个新值  
 L[2,3] = []  
清空列表  
 L[:] = []  
调用方啊clear()清空列表  
 L.clear()

修改列表中的元素  
第一种：  
 L = [1,2,3,4,5,6,7,8]  
 L[1] = 999  
 print(L)  
 结果：[1, 999, 3, 4, 5, 6, 7, 8]  
 总结：一次只能修改一个元素  
第二种：  
 L = [1,2,3,4,5,6,7,8]  
 L[1:4]  
 L = [1, 2, 3, 4, 5, 6, 7, 8]  
 L[1:4] = [4,3,2]  
 print(L)  
 结果：[1, 4, 3, 2, 5, 6, 7, 8]  
 注意：再给切片赋值的时候左右两边的元素个数可以不相同

获得列表的指定元素的索引值  
 列表中每个元素都有两个整数类型的索引  
 第一个元素的索引是0，后面元素的索引依次递减1  
index()返回两个索引中大于0的哪一个  
index(传入参数)返回一个索引值  
print(list(['49274928742947238974kadhao',23,'31']).index('31'))  
如果列表中存在多个相同元素，那么index返回第一个函数的索引值  
如果列表中不存指定元素那么返回valueErrort  
方法index()还可以指定起始索引和结束索引  
index(7,3,6) index(查找7在索引中的位置但是是从第3个开始的到第6个（不包含6）)  
  
  
使用切片操作  
语法格式：[start:stop:step]  
得到的切片中结果仍然是列表，是原始列表片段的一个拷贝  
不包括stop  
如果不指定step那么step为1.代表的是步长  
如果不指定start。切片的第一个元素是列表的第一个元素  
如果不指定stop，切片的最后一个元素默认是列表的最后一个元素  
从索引start开始往后计算切片  
当step为负数的时候  
如果不指定start，切片的第一个元素默认是列表的最后一个元素  
如果不指定stop，切片的最后一个元素是列表的第一个元素  
从索引start开始往前计算切片  
L = [5,4,45,53,53,53]  
print(L[1:4:1])  
结果：[4, 45, 53]  
  
L = [1,2,3,4,5,6]  
print(L[0::6])不指定最后的结束值，默认就是到最后一个  
  
L = [1,2,3,4,5,6]  
print(L[-2::])  
结果：[5, 6]  
注意：切片的操作是允许下表越界的  
可以调用内置函数slice()  
slice(stop)  
slice(start,stop)  
slice(start,stop,step)  
start,stop,step默认值都是None  
slice(start,stop,step)和切片操作start：stop：step是等价的  
L = [1,2,3,4,5,6]  
print(L[slice(1,3)])  
  
  
使用in检查是否存在元素  
L = [4,5,6,7,8,9]  
leixing = 4 in L  
print(leixing)  
结果：True  
  
使用加法和乘法进行列表运算  
 L1 = [1,2,3]  
 L2 = [4,5,6]  
 L3 = L1 + L2  
 print(L3)  
 运行结果：[1, 2, 3, 4, 5, 6]  
   
 L1 = L2 = [1, 2]  
 L1 = L1 + [3, 4]  
 print(L1,L2)  
 运行结果：[1, 2, 3, 4] [1, 2]  
  
使用比较运算符对两个列表进行比较  
 >  
 >=  
 <  
 <=  
 ==  
 !=  
 L1 = [1,2,3,4]  
 L2 = [2,3,4]  
 print(L1 < L2)  
 运行结果：True  
还可以使用is对两个列表进行比较  
 is与==区别 == 是相等性测试， is 是同一性测试 is比较的内存地址  
  
  
对列表中的元素进行翻转  
调用方法reverse()  
 L.reverse()  
调用内置函数reversed() (返回的是一个迭代器对象)  
 iterator = L.reversed()  
 list(iterator)  
  
  
列表的排序  
调用方法sort()  
 L.sort() (从小到大进行排序)  
 指定参数reverse  
 L.sort(reverse = True) (反转之后从大到小进行排序)  
 print(L)  
调用sorted()  
 sorted(L)返回值是排序后的新列表，并且排序后的就列表不发生变化  
 例子：  
 L = [3,45,12,2]  
 L1 = sorted(L)  
 print(L, L1)  
 运行结果：[3, 45, 12, 2] [2, 3, 12, 45]  
  
  
多维列表  
 L = [[1,2,3,4],[5,6,7,8]]  
 print(L[1] [2])  
 运行结果：7  
 L = [[1,2,3,4],[5,6,7,8]]  
 L.append(4)  
 print(L)  
 运行结果：[[1, 2, 3, 4], [5, 6, 7, 8], 4]  
 注意：一维列表和使用的多为列表  
  
  
多维列表的初始化  
 print([[0]\*3]\*4)  
 运行结果：[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]  
 初始化：print([[ ]\*3]\*4)  
列表生成式  
 print([0 for in range(3)]) 0是列表中的元素  
 print([[1 for i in range(3)] for j in range(5)])  
 运行结果：[[1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1]]  
  
  
列表生成式  
 j = []  
 for i in range(100):  
 j.append(i)  
 print('j列表为：',j)  
 运行结果：j列表为： [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]  
更好的替代品  
 语法格式：[表示列表元素的表达式 for 自定义变量 in 可迭代对象]  
 j = [i\*i for i in range(18)]   
 print(j)  
 运行结果：[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289]  
 注意：凡是可以使用for in循环创建的列表都可以使用列表生成式来创建  
 L = [i \* i for i in range(19) if not i % 2] 在列表生成式中添加if判断  
 运行结果：[0, 4, 16, 36, 64, 100, 144, 196, 256, 324]  
 L = [(i, j) for i in range(1) for j in range(7)]  
 print(L)  
 运行结果：[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6)]  
 L = [(i, j) for i in range(1) for j in range(7) if i != j]  
 print(L)  
 运行结果：[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6)]  
 注意：列表生成式是可以嵌套的  
 L = [[i for i in range(6)] for j in range(7)]  
 print(L)  
 运行结果：[[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]  
 matrix = [[1,2,3,4],[5,6,7,8],[9,0,1,2]]  
 L = [[row[i] for row in matrix] for i in range(4)]  
 print(L)  
 运行结果：[[1, 5, 9], [2, 6, 0], [3, 7, 1], [4, 8, 2]]  
  
matrix = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 0, 1, 2]]  
L = [[row[i] for row in matrix] for i in range(4)]  
print(L)

# **元组**

也是python内置的数据结构之一  
列表和元组的区别  
 在形式上元组用小括号表示（）（元组的小括号是可以省略的）（列表用小括号）  
 元组一旦创建是不可改变的  
  
  
空元组  
 print(())  
 print(tuple())（调用元组的类无参构造）  
  
T = (1,[4,2],4)  
T[1][0] = 5  
print(T)  
(1, [5, 2], 4)  
注意：对于元组中可变类型的数据，存储的引用是不能被改变的，不能在引用其他对象，引用指向的可改变的数据是可以被改变的  
  
只包含一个元素的元组  
 t = (18)  
 print(t)  
 print(type(t)) INT   
 注意：元组中至少包含一个逗号，才能别称为元组。例如(18,)

# 字符串

'Hellow word'  
字符串与列表和元组一样，都是有序的  
把字符串看作为字符的列表  
没有单独的字符类型  
  
Hello  
'H''e''l''l''o'  
  
字符串的创建  
 使用单引号  
 s = 'abcd'  
 注意:不能取名给str因为str是类名  
 空字符串  
 ''或者""  
 可以在单引号中使用双引号，也可以在双引号中使用单引号  
 但是不能在单引号中使用单引号，在双引号中使用双引号  
 '你好'd''这种语法格式是不允许的  
 调用str()  
 str('a,b,c,d')  
 可以将其他类型的字符串来转化为字符串  
  
  
  
字符串的查操作（和列表的查操作是类似的）  
 区别：  
 当获取字符串指定索引的时候不仅可以食用index(返回第一个字符串的索引)还可以调用find（返回第一个字符串的索引）  
 rfind（返回最后一个字符串的索引），rindex（返回最后一个字符串的索引）  
 当字符串中不存在指定的字符串时，方法index和rindex会抛出ValueError错误，而rfind和rindex会返回-1  
  
  
字符串是不可变类型  
 s = '1234'  
 s[0] = '2'  
 print(s)  
 运行结果：TypeError: 'str' object does not support item assignment  
 如果想使用可以使用加法运算符进行切片和拼接  
 s = '123467'  
 S = s[:4] + '5' + s[4:]  
 print(S)  
 运行结果：1234567  
  
  
使用比较运算符对两个字符串进行比较  
 >  
 >=  
 <  
 <=  
 s = '1234'  
 S = '123'  
 print(s == S)  
 运行结果：False  
 ord（查看字符的ordinal和value）  
 print(ord('a')) 97  
 chr(查看数值对应的字符)  
 print(chr(97)) a  
也可以使用is进行比较  
  
  
字符串反转  
 与列表不同的是，字符串是不可改变类型  
 不能使用reverse只能使用reversed   
 reversed 返回值是一个reversed对象  
 例子：  
 s = 'da'  
 print(list(reversed(s)))  
 运行结果：['a', 'd']  
  
字符串的排序（不可变类型）  
调用内置函数  
sorted(字符串)  
 例子  
 s = 'ljasgduyagdouay'  
 print(str(sorted(s)))  
 运行结果：['a', 'a', 'a', 'd', 'd', 'g', 'g', 'j', 'l', 'o', 's', 'u', 'u', 'y', 'y']  
 当然也能在sorted(s)中写入参数sored(s,reverse=True)这样就能使排序后的字符串进行翻转  
 还可以指定一个函数key，key= 函数名，或者key = 类名.方法  
 这样会对字符串中的所有字符，调用方法，然后按照方法的返回值进行排序，从而自定义规则  
 sorted(s, key = str.lower)  
 sorted(s, key = len) 按照字符串长度进行排序  
  
  
格式化字符串  
使用百分号作为占位符  
 例子：  
 print('今天是%s-%s-%s' % (2019,12,4))  
 运行结果：今天是2019-12-4  
 例子：  
 from datetime import datetime  
 time = datetime(2019,12,4,13,57,20).strftime('%Y-%m-%d %H:%M:%S')  
 print(time)  
 运行结果：2019-12-04 13:57:20  
 拓展：获取当前时间  
 from datetime import datetime  
 time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  
 print('当前时间为%s' % time)  
 运行结果：当前时间为2019-12-04 14:01:08  
 %s：用来表示字符串  
 %d:用来表示整数  
 %f：表示浮点数  
 注意：当字符串中出现两个或者两个以上的时候必须要将他封装在元组中  
 如果定义的格式化字符串是一个普通字符，那就需要使用两个字符串进行转义（%%通用两个百分号来表示一个百分号）  
 占位符%是可以指定宽度  
 print('%18d' % 67)  
 占位符%也可以指定精度  
 print('%.3f' % 3.1415926)(保留三位小数)（如果不加f那就是一共是三位）  
使用花括号作为占位符  
 format应使用花括号使用占位符  
 book = '数据库'  
 '买了一本书：{}'.format(book)  
 如果传入多个占位符拿在format方法中会按照顺序依次匹配所有的  
 花括号中可以指定位置参数  
 '花了{0}，买了一本{1},还剩余{0}'.format(price,book)  
 在方法fotmat中指定关键字的名称  
 '花了{p}，买了一本{b}，花了{p}钱'.format(p = price, b = book)  
 使用':'来指定整数  
 '{:d}'.format(66) 十进制  
 '{:b}'.format(58) 二进制  
 '{:x}'.format(58) 十六进制  
 '{:X}'.format(58) 十六进制(大写)  
 '{:f}'.format(58) 浮点数  
 使用逗号作为千位的分隔符  
 '{:,}'.format(1234567)   
 '{0:b}'.format(58) 二进制 位置参数'0'，都是放在冒号前面的  
 print('{:10}'.format(58)) 58  
 print('{:10}'.format('58'))58   
 注意：数字是右对齐的，字符串是左对齐的  
 print('{0:10}'.format(58)) 指定位置参数  
 print('{num:10}'.format(num = 58))指定名称参数  
 设置精度  
 print('{:.3}'.format(3.1415926)) 3.14(总共三位)  
 print('{:.3f}'.format(3.1415926)) 3.142（小数点后三位）  
 使用format进行格式化时间  
 from datetime import datetime  
 print('{:%Y-%m-%d %H:%M:%S}'.format(datetime(2019,12,4,14,28,20)))  
 运行结果：2019-12-04 14:28:20  
 格式：{（指定位置数字或者是位置名称）：（指定宽度）.（指定精度）（指定类型）}  
 也可以调用内置函数format得到格式化字符串，他与字符串format是等价的  
 format(3.1415926,'.3')  
 运行结果：3.14  
使用美元符号作为占位符  
 导入string类中的Template  
 from string import Template  
 teml = Template('买了一本$b书花了$p').substitute(p = 20.5,b = '《数据》')  
 teml = Template('买了一本$b书花了$p').substitute(p：20.5,b：'《数据》')  
 print(teml)  
 运行结果：买了一本《数据》书花了20.5  
 注意：当占位符没有匹配到实际值的时候就会抛出错误：KeyError  
 调用方法safe\_substitute,如果没有匹配到值的时候不会抛出错误，而是使用占位符本身作为值  
 from string import Template  
 teml = Template('买了一本$b书花了$p').safe\_substitute(p = 20.5)  
 print(teml)  
 运行结果：买了一本$b书花了20.5  
  
  
字符串大小写转换  
 upper：把所有字符转化为大写  
 lower:把所有字符转化为小写  
 swapcase：大写转小写，小写转大写  
 capitalize:把第一个字符转化为大写，把其余的转化为小写  
 title:把每个单词的第一个字符转化为大写，把剩余的字符转化为小写  
 因为过于简单不再进行测试  
判断字符串的大小写  
 isupper:判断所有字符全为大写  
 islower：是否全部为小写  
 istitle:判断每个单词第一个字符为大写，并且其他的为小写  
  
  
字符串的对齐  
 center()中心对其  
 ljust()左对齐  
 rjust()右对齐  
 三个方法都可传入两个参数  
 第一个参数：指定字符串的宽度  
 第二个参数：（可选），执行填充字符，默认为空格  
 例子：  
 H = 'Heelow'  
 print(H.center(18,'\*'))  
 运行结果：\*\*\*\*\*\*Heelow\*\*\*\*\*\*  
 zfill右对齐，左边用0表示  
 print('567'.zfill(6))  
 print('-3454'.zfill(9)) -00003454  
  
  
字符串中的子串替换  
 replace()  
 第一个参数：指定被替换的子串。第二个参数指定替换子串的字符串,第三个参数指定最大的替换次数  
 注意：该方\*\*\*9法返回的是替换后得到的字符串，以前的字符串不发生变化  
 s = 'Hellow - Hellow'  
 print(s.replace('Hellow','Hi',2)) Hi - Hi  
  
  
字符串的某些字符进行转换  
 maketrans和translate  
 首先调用maketrans创建一个转换表，然后把创建的转换表作为参数的方法传给方法translate  
 table = str.maketrans('bcd','234')  
 print(table) {98: 50, 99: 51, 100: 52}\  
 print('dahkghi'.translate(table))  
 运行结果：4ahkghi  
 方法：maketrans("要转换的字符","转换后的字符","要删除的字符")  
   
  
  
字符串的劈分和合并  
 split（）从字符串的左边开始劈分，  
 rsplit（）从字符串右边进行劈分  
 使用sep参数进行指定劈分的劈分符  
 使用maxsplit指定最大劈分次数（这时从左边劈分和从右边劈分就不一样了）  
 注意：他们的劈分后返回的都是一个列表  
 s = 'java MySQL Oracle html python'  
 S = s.split(sep = 'o')  
 print(S)  
 运行结果：  
 ['java MySQL Oracle html pyth', 'n']  
调用方法partition和rpartition进行劈分（注意：都必须指定劈分符，这两个方法返回值都是一个元组，而且只有三个返  
回值。三个返回值分别是，分割前，分割的，分割后面的）  
 例子：  
 s = 'java MySQL oracle html python'  
 S = s.partition('o')  
 print(list(S))  
 运行结果：['java MySQL ', 'o', 'racle html python']  
 例子：  
 S = s.rpartition('o')  
 print(list(S))  
 运行结果：['java MySQL oracle html pyth', 'o', 'n']  
 如果字符串中不存在指定的劈分符号，依然返回三个元组，只是变为了空字符串  
  
合并字符串  
调用方法join  
 例子：  
 s = ['java','c++','pthon']  
 s = ('java', 'c++', 'pthon')  
 S = '|'.join(s)  
 print(S)  
 运行结果：java|c++|pthon  
 注意：元组也是可以的  
 S = ''.join(s)（字符串进行合并）  
 s = '|'.join('python')  
 例子：  
 s = '|'.join('python')  
 print(s)  
 运行结果：p|y|t|h|o|n  
 注意：可以把字符串看作字符的列表  
  
以is开头的方法  
isidentifier()用来判断是否是合法的标识符 ''.isisidentifier()  
iskeyword()判断是否是关键字 注意：要先导入模块  
isspace()判断是否是由空白或者是水平制表符组成  
isalpha()判断字符串是否是由字母组成  
isdecimal(）是否是由十进制组成  
isnumeric()判断是否全部由数字组成 罗马数字或者是汉字数字都是返回True  
isalnum()判断是否由数字和字符串组成  
  
去除字符串的前导字符串和后续字符串  
 lstrip()去除字符串的前导字符串  
 rstrip()去除字符串的后续字符串  
 strip()去除前导和后续字符串  
 注意：这些指的是空白字符串  
 指定字符串：从左边第一个字符串依次往后，只得到指定的字符不在字符串中（后续是往前）  
 例子：  
 s = 'dagdjavdj'  
 S = s.lstrip('adgd')  
 print(S)  
 运行结果：javdj  
 注意：指的就是latrip中的参数依次和前后进行比较，直到没有的时候才进行截取  
  
借助字典进行格式化字符串  
 phoneBook = {'张三':'123456','李四':'13165'}  
 print('张三的电话%s,李四的电话%s' % (phoneBook['张三'],phoneBook['李四']))  
 运行结果：张三的电话123456,李四的电话13165  
 例子：  
 phoneBook = {'张三': '123456', '李四': '13165'}  
 print('张三的电话%(张三)s,李四的电话%(李四)s' % phoneBook)  
 运行结果：张三的电话123456,李四的电话13165  
 例子：  
 phoneBook = {'张三': '123456', '李四': '13165'}  
 print('张三的电话{},李四的电话{}'.format(phoneBook['张三'],phoneBook['李四']))  
 print('张三的电话{张三},李四的电话{李四}'.format\_map(phoneBook))  
 运行结果：  
 张三的电话123456,李四的电话13165  
 张三的电话123456,李四的电话13165

转义字符  
 表示复发直接表示的特殊字符  
 \n换行  
 \r回车  
 \t水平制表符  
 \b退格  
 例子：  
 print("dada\ndadaad")  
 运行结果：  
 dada  
 dadaad  
 \\用来表示反斜杠  
 \'用来表示单引号  
 \"用来表示双引号  
  
原始字符串  
 print(r'\tC:\\Program Files') 不让转义字符生效 raw表示原始字符串r或者R添加到前面  
 注意：原始字符串的最后一个字符不能为\(反斜杠)，最后两个字符都是反斜杠除外  
  
跨越多行的字符串  
使用三个引号  
 print('''  
 这是一个跨越多行的  
 ''')  
 就可以实现原始格式打印，使用双引号也是同样的效果  
 也可以在三个双引号中嵌套三个单引号，反则亦成立  
在每行的结尾添加一个反斜杠  
 print("我是一个\  
 nihao1\  
 hkaj")

# 字典

为什么需要字典  
字典就相当于Java中的键值对，他们都是一一对应的  
 例子：  
 names = ['张三','李四','王二','麻子']  
 numbers = [136,135,141,456]  
 print(numbers[names.index('张三')])  
 运行结果：136  
 格式{'':'',  
 '':'',  
 '':'',  
 }  
 例子：  
 Map = {'张三':'136','李四':'123'}  
 print(Map['张三'])  
 运行结果：136  
 注意：这种方法只能通过第一个值找到第二个值不能通过第二个值查找第一个值  
 例子：  
 Map = {'张三':'136','李四':'123','张三':'139'}  
 print(Map['张三'])  
 运行结果：139  
 结论：如果存在相同的键值的时候时返回最后一个键值的值  
 例子：  
 Map = {1:136,'李四':12.1,'张三':139}  
 print(Map[1])  
 运行结果：136  
 结论：整个键值存储的时候，数据类型是没有限制的  
  
  
 字典也是python内置语言之一  
 字典的字数越多。查询的效率越低  
 也可以直接查找对应的页码。然后直接翻到这一页。这样无论查找到那个字，查找速度都非常快  
 key值是不可改变类型  
 不需要声明字典容量  
 字典浪费大量的内存。因为他是用空间换取的时间  
  
  
字典的创建  
使用花括号  
 花括号中间使用逗号','进行分割  
调用内置函数dict()  
 例子：  
 dict('':'','':'')  
 dict(age = 18, name = '张三')  
 dict([('name','张三'),('age',19)])(把元组放在列表里面实现的效果是一样的)  
 运行结果:{'name': '张三', 'age': 19}  
 例子：  
 print(dict(zip(range(9),'012345678')))  
 运行结果：{0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8'}  
 注意：后面的‘012345678’只能是带''的  
调用dict的方法formkeys，通过参数指定所有的key，所有的value默认值都为None  
 例子：  
 print(dict.fromkeys(['name','age']))  
 运行结果：{'name': None, 'age': None}  
 在formkeys中写入第二个参数设置默认值  
 例子：  
 print(dict.fromkeys(['name','age'],'N/A'))  
 运行结果：{'name': 'N/A', 'age': 'N/A'}  
  
  
在字典的查操作  
使用中括号[]  
 例子：  
 d = {'name':'张三','age':18}  
 print(d['name'])  
 运行结果：张三  
使用get()方法  
 d.get('name')  
 注意：如果查找的元素不存在，使用中括号会抛出错误，而是用get()方法则会返回None  
 d.get('name','男') 第二个参数就是如果指定的参数不存在，就是返回的是None的时候默认值改为'男'  
使用运算符in检查是否存在key  
 例子：  
 d = {'name':'张三','age':18}  
 doo = 'name' in d  
 print(doo)  
 运行结果：True  
  
  
字典的改操作  
为已经存在的key赋一个新的值  
 一次只能修改一个  
调用方法update()  
 一次至少修改一个  
 d.update({'key的值':'修改的值','....':'....'})  
 d.update([('key的值','修改的值'),('....','....')])  
  
  
字典的曾操作  
为不存在的key赋予一个value的值  
 d = {'name':'张三','age':18}  
 d['name1'] = '李四'  
 print(d)  
 运行结果：{'name': '张三', 'age': 18, 'name1': '李四'}  
调用方法update()  
 d.update({'':'','':''})  
 传入的参数可以是元组和列表  
 还可以是等于关键字形式  
 一次至少添加一个key-value  
  
  
删除字典中的key-value  
调用方法pop(),该方法返回key对应的value  
 返回被删除的的value值 = d.pop('key值')  
 注意：如果指定的key值不存在那就会抛出错误，解决办法，在第二个参数中写入默认值  
 d.pop('被删除的key',''默认返回的值)  
使用del语句  
 del d['要删除的key的名字']  
调用方法popitem()返回被删除的元素  
 d.popitem()（如果没有指定那她删除的是任意的值）  
  
  
清空字典  
 d.clear()  
  
为字典设置默认的value值  
调用方法setdefault()  
如果字典中存在指定的key，该方法返回指定的key对应的value，字典不发生变化  
如果字典中不存在指定的key，该方法返回指定的默认值，字典中就会添加一个key - value对  
"指定的key-：指定的value值。"  
 例子：  
 d = {'name':'jack'}  
 d.setdefault('age','默认值')  
 print(d)  
 运行结果：{'name': 'jack', 'age': '默认值'}  
  
字典的视图  
 keys：返回字典所有key的视图  
 values：返回所有字典value的视图  
 items:返回字典所有key - value的视图  
 视图会随着字典的变化而变化  
  
字典生成式  
 L = {i for i in range(7)}  
 L1 = {s for s in range(7)}  
 Map = dict()  
 Map.update(zip(L,L1))  
 print(Map)  
 运行结果：{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6}  
语法格式：  
 L = {i for i in range(7)}  
 L1 = {s for s in range(7,14)}  
 MAP = {i:j for i,j in zip(L,L1)}  
 print(MAP)  
 运行结果：{0: 7, 1: 8, 2: 9, 3: 10, 4: 11, 5: 12, 6: 13}  
 MAP = {i:j for i,j in zip({i for i in range(7)},{s for s in range(7,14)})}  
 print(MAP)  
 同上一样的：{0: 7, 1: 8, 2: 9, 3: 10, 4: 11, 5: 12, 6: 13}  
同样的在列表生成式中能用的在这里都能用

# **集合**

集合的创建  
使用花括号{}  
 {3,4,5,6,7,8,9}  
 例子：  
 s = {3,9,5,6,7,8,9}  
 print(s)  
 运行结果：{3, 5, 6, 7, 8, 9}  
 注意：集合中重复的数据会被去除掉  
 不能使用花括号来表示空集合（它用来表示空字典）  
使用set()内置函数创建集合  
 例子：  
 s = set(range(8))  
 print(s)  
 运行结果：{0, 1, 2, 3, 4, 5, 6, 7}  
 传入的参数可以是：字符串，元组，列表，花括号  
 创建一个空集合set()不传入任何元素  
  
集合的查操作  
使用in 使用not in检查是否不存在  
  
  
集合的曾操作  
调用方法add() 注意：集合已经存在的元素不会被添加，一次只能添加一个元素  
调用update() 注意：传入的参数可以是集合，列表，元组。一次至少添加一个元素  
  
  
集合删操作  
调用方法remove('要删除的元素')

注意：如果删除的元素不存在那将会抛出错误，KeyError，一次只删除一个元素

调用方啊discard('要删除的元素')

注意：如果指定元素不存在不会抛出错误，一次也只是删除一个元素

调用方法pop()

注意：不传入参数将会删除任意一个元素。该方法返回被删除元素

调用方法clear()

清空集合  
  
  
不可变集合（被冻结的）  
frozenset()  
存在哈希值  
可以作为字典中的key  
可以作为set中的元素  
对象的创建  
 调用内置函数frozenset()  
 print(frozenset()) 传入参数是不限制数据类型的，可以传入元组，列表，字符串，set对象  
 例子：  
 print(frozenset('dhakdhbakjwjg'))  
 运行结果：frozenset({'d', 'b', 'k', 'w', 'h', 'g', 'j', 'a'})  
  
  
集合生成式  
 s = {i for i in range(6)}  
 print(s)  
 运行结果：{0, 1, 2, 3, 4, 5}

集合的关系  
 使用运算符'='或者是'!='  
判断一个集合是另外一个集合的子集  
 issubset()方法  
判断一个集合是一个集合的超集  
 issuperset()进行判断  
判断有没有交集  
 isdisjoint()进行判断  
  
  
集合的数学操作  
 交集，并集，差集，对称差集。   
交集：intersection() 运算符& 这两种方法是等价的 注意：左交集操作后创建一个新的集合  
 intersection\_update()原理是先使用intersection()的返回值然后更新s1  
并集：union()  
差集：difference()方法和使用'-'是等价的 difference\_update()  
对称差集：symmetric\_difference()和使用'^'是一样的 symmetric\_difference\_update()

总结：  
把集合看作没有value的key值  
集合中不能存在重复的数据  
集合中的数据是无序的  
集合中的数据是不可改变的类型  
并且多种数据可以混合存储  
集合可以根据需要动态的伸缩，所以不需要声明容量  
集合会浪费较大的内存，与列表相比使用空间换取的时间

# 流程控制语句

if语句  
 格式  
if 判断条件  
 条件执行体  
elif 判断条件2  
 条件执行体  
else 判断条件3  
例子：  
if 1<2:  
 print('条件执行体')  
elif 2<1:  
 print('条件执行体2')  
else:  
 print()  
与Java中的区别就是python有elif  
  
  
对象的布尔值  
所有的对象都有布尔值  
 以下对象的布尔值为false:False,数值0，None,空字符串，空元组，空字典，空集合   
 print(bool(False)) 使用bool方法进行判断  
 注意：这些对象会直接调用布尔值，就是可以直接使用  
 例子：  
 if 16:  
 print("成功")  
 运行结果：成功  
  
  
条件表达(相当于Java中的三目运算)  
x if 判断条件 else y  
 如果判断条件为True那将会返回x否则返回y  
 例子：  
 print('x' if 687 else 'y')  
 运行结果：x  
 注意：可以嵌套  
 x if 判断条件 else （x if 判断条件 else y）  
  
  
while 1:  
 print('真好玩')  
s = (3,4,5,5,67)  
for i in s:  
 print('第' + str(i + 1) + '次')  
注意for循环如果要删除元素的时候循环控制语句一定要进行拷贝操作，不然就会报错  
 works = ['java','python','mysql','orcale','html','spring','c++']  
 print('删除前的wroks为：', works)  
 for word in works[:]:  
 if(len(word)>5):  
 works.remove(word)  
 print('删除后为：', works)  
  
  
循环中的break- else 结构  
 在while或者for - in语句没有正常结束的时候，就执行else语句（注意必须是while里面有break才行）  
while 判断条件  
 代码块  
else:  
 代码块  
  
  
使用continue()语句  
  
  
并行遍历  
调用内置函数zip()可以将多个对象打包压缩成zip对象  
zip(names,ages)  
for name,age in list(zip(names,ages))：  
 print(naem,'年龄是：',age)

## 目录操作

import os

os相关函数

os.name

获取当前操作系统的类型,nt代表windows,posix代表Linux,UNIX或者MaxOS

os.linesep

获取当前操作系统的换行符

os.sep

获取当前系统下的路径分隔符

os.getcwd(): 获取当前工作目录

os.listdir(path): 指定目录下的文件和目录信息

os.mkdir(name,model): 创建目录 (model为权限0o777)

mkdir只能创建一级目录,如果创建的目录过多将会报错

os.makedirs(path/path): 创建多及目录

os.rmdir(path): 删除目录

os.makedirs(path/path): 删除多级目录

删除只能删除空目录,如果想要删除非空目录可以使用以下方法

import shutil

shutil.rmtree(path): 删除当前路径下的所有文件

os.chdir(path): 把path作为当前工作目录

os.walk(name): 遍历目录树返回一个数组,包含所有路径名,所有目录列表和文件列表三个元素.

注意walk只在windows和UNIX中有效

os.path相关函数

os.path.absoath(path): 获取文件目录绝对路径

os.path.exists(path): 判断文件或目录是否存在,存在返回true不存在返回false

os.path.join(path[name], path[name]): 将目录或者文件名连接起来

os.path.spliext(): 分离文件名和扩展名

os.path.basename(path): 从一个目录中提取文件名

os.path.dirname(path): 从一个路径中提取文件路径,不包含文件名

os.path.isdir(path): 用于判断是否为有效路径

高级文件操作

os.access(path, accessmode): 获取对文件的访问权限,(读,写,执行)

accessmode的值

R\_OK: 读取

W\_OK: 写入

X\_OK: 执行

F\_OK: 存在

如果返回1代表有制定权限,返回0代表没有制定权限

os.chmod(path, mode): 修改path执行文件的访问权限

os.remove(path): 删除path指定文件路径

os.rename(src, dst): 将文件目录src重命名为dst

os.stat(path): 返回path指定文件信息

os.statfile(path): 使用关联的应用程序打开path指定文件

# 函数

在这里主要描述python和Java的不同  
 def 函数名(形式参数1，形式参数2.......)：  
 函数体  
 def init(i):  
 print('完成了初始化操作:',i)  
 init(1)  
 运行结果：完成了初始化操作: 1  
 注意：i不需要单独定义，但是在调用init函数的时候必须给init赋值否则就会报错！  
 def init(i):  
 print('完成了初始化操作:',i)  
 return i+1  
 j = init(7)  
 print('函数init()函数的返回值为:', j)  
 注意：在这里返回值是靠return来返回值的，如果不给返回值那就返回None  
 注意：关键字实参在所有关键字只要指定就可以，不按照顺序进行填写  
  
  
使用\*将序列中的每个元素都转化为形参  
 L = [1,2,3]  
 def f(i,j,k):  
 print('i=',i, 'j=',j,'k=', k)  
 f(\*L)  
 运行结果：i= 1 j= 2 k= 3  
 def fun(\*args):  
 print(args)  
 fun(\*[1,2,3])  
 运行结果：(1, 2, 3)  
  
使用\*\*定义个数可变的位置形参  
接收0个或者多个位置形参,这些关键字实参。会将个数可变的关键字形参初始化为一个字典  
 def f(\*\*www):  
 print(www)  
 f()  
 f(a=1)  
 f(a=1,b=2,c=3,d=4,e=4)  
 运行结果：  
 {}  
 {'a': 1}  
 {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 4}  
 注意：定义函数时，最多只能定义一个可变的位置形参  
 L = ['Python','java','Swift']  
 print(sorted(L))  
 print(sorted(L,key = len))  
 print(sorted(L,key=len, reverse=True))  
 运行结果：  
 ['Python', 'Swift', 'java']  
 ['java', 'Swift', 'Python']  
 ['Python', 'Swift', 'java']  
 因为效用函数时位置实参必须位于关键字形参之前  
  
  
class Student (Object): 编写一个类  
 pass  
Student() 用来调用一个类  
  
class Student\_Zhang (object):  
 def \_\_init\_\_(self,data1,data2): 进行初始化的特殊方法 self表示实例方法的实例对象同产生过hi用self来表示  
 self.data1 = data1 等号的右边是参数，等号的左边是给实例对象绑定的值  
 self.data2 = data2  
 print('初始化！！！')  
 def one\_fangfa(self):  
 self.data = 56  
 def one\_fangfa1(self):  
 print(self.data)  
st = Student\_Zhang(1,1)  
st.one\_fangfa()  
st.one\_fangfa1()  
st.data = 88  
st.one\_fangfa1()  
  
类方法  
类方法指的是使用@classmethod进行装饰的方法  
，第一个形参表示类对象，其对应的实参由系统自动传入，第一个形参通常是cls，也可以是其他名称  
类方法可以理解为Java中的静态方法  
代码如下：  
class Student(object):  
 def \_\_init\_\_(self):  
 print("已经初始化！！！")  
 @classmethod  
 def toString(sel):  
 print('该方法是使用@classmethod来修饰的方法')  
Student.toString()  
运行结果：  
该方法是使用@classmethod来修饰的方法  
代码如下：  
class Student(object):  
 def \_\_init\_\_(self):  
 print("已经初始化！！！")  
 @classmethod  
 def toString(sel):  
 print('该方法是使用@classmethod来修饰的方法')  
Student().toString()  
运行结果：  
已经初始化！！！  
该方法是使用@classmethod来修饰的方法  
  
  
静态方法  
class Student(object):  
 @staticmethod  
 def satic():  
 print('这里是静态方法！！！')  
 def s():  
 print('这是嵌套在·static中的方法')  
 s()  
 @classmethod  
 def lei(slf):  
 print('这里是类方法')  
 slf.satic()  
 print('在类方法中调用静态方法')  
Student.satic()  
s = Student()  
s.satic()  
s.lei()  
Student.lei()  
运行结果：  
这里是静态方法！！！  
这是嵌套在·static中的方法  
这里是静态方法！！！  
这是嵌套在·static中的方法  
这里是类方法  
这里是静态方法！！！  
这是嵌套在·static中的方法  
在类方法中调用静态方法  
这里是类方法  
这里是静态方法！！！  
这是嵌套在·static中的方法  
在类方法中调用静态方法  
代码：  
class Student(object):  
 @staticmethod  
 def satic():  
 print('这里是静态方法！！！')  
 def s():  
 print('这是嵌套在·static中的方法')  
 s()  
 @classmethod  
 def lei(slf):  
 print('这里是类方法')  
 slf.satic()  
 print('在类方法中调用静态方法')  
s = Student()  
s1 = Student()  
print(s) <\_\_main\_\_.Student object at 0x010CAF58>  
print(s1) <\_\_main\_\_.Student object at 0x0114E118>  
print(Student) <class '\_\_main\_\_.Student'>  
print(Student()) <\_\_main\_\_.Student object at 0x0114E0E8>  
运行结果：  
<\_\_main\_\_.Student object at 0x0173AF58>  
<\_\_main\_\_.Student object at 0x0336E118>  
<class '\_\_main\_\_.Student'>  
<\_\_main\_\_.Student object at 0x0336E0E8>  
  
  
访问控制  
控制类对象的  
如果在某个属性或者方法前添加两个\_\_，那么在类对象的外部就不能直接访问该属性和方法了，但是在类的内部时可以使用的  
  
print(s.\_\_age) name 'Stundet' is not defined  
  
class Student(object):  
 def \_\_init\_\_(self):  
 print('执行了初始化操作！')  
 self.\_\_age = 18  
 def \_\_konghzi(self):  
 print('执行了\_\_konghzi（）方法')  
 print(self.\_\_age)  
Student.\_\_init\_\_(Student())  
s = Stundet()  
  
  
python把这些属性和方法改成了另外的名字！！！  
在这里不建议这样访问，因为不同版本的python解析器讳改为不同的名字  
class Student(object):  
 \_\_age = 18  
 def \_\_init\_\_(self):  
 print(self.\_\_age)  
 def \_\_konghzi(self):  
 print('执行了\_\_konghzi（）方法')  
 print(self.\_\_age)  
print(Student.\_Student\_\_age)  
  
  
class Student(object):  
 \_\_age = 18  
 def \_\_init\_\_(self):  
 self.\_\_name = '张三'  
 def \_\_konghzi(self):  
 print('执行了\_\_konghzi（）方法')  
s = Student()  
print(dir(s))  
运行结果： 可以使用dir()来查看所有方法和属性  
['\_Student\_\_age', '\_Student\_\_konghzi', '\_Student\_\_name', '\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_',  
 '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_',   
 '\_\_init\_subclass\_\_', '\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_',  
 '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', '\_\_weakref\_\_']  
所以仍然可以在类对象的外部动态绑定属性和方法！！！  
  
除了添加两个下划线还可以添加单下划线  
虽然可以在类对象的外部访问，但是最好不好访问

# 一些经典案例

舍韩王赏麦  
L = [sum(2 \*\* i for i in range(0,65))]  
print(L)  
  
不重复的三位数  
描述：不重复的三位数的个数  
count = 0  
for i in range(1,10):  
 for j in range(10):  
 for k in range(10):  
 if i != j and j != k and k != i:  
 count +=1  
 print(i,j,k,'\t')  
print('共有',count,'个不重复的三位数')  
  
角谷猜想  
对于任意的自然数  
若为奇数则乘三加一  
若为偶数，除以二  
n = int(input('请输入要验证角谷猜想n的值：'))  
while 1:  
 n = n / 2 if n % 2 == 0 else n \* 3 + 1  
 print('当前n的值为：',n)  
 if n == 1:  
 print('n的值符合角谷猜想')  
 break  
  
鸡兔同笼  
共有h个头和k只脚  
设鸡的数目为x则x的取值范围为x=h-1,则兔的数目为y = h - x  
x + y = h  
2x + 4y = k  
h = int(input('请输入头的数量：'))  
k = int(input('请输入脚的数量：'))  
算出鸡的数量  
代码如下：  
for x in range(h):  
 算出兔子的数量  
 for y in range(h - x + 1):  
 符合表达式，则证明可以实现  
 if x + y == h and 2\*x + 4\*y == k:  
 print('鸡的数量为：%d,兔子的数量为:%d' % (x,y))  
  
  
用一百文钱买一百只鸡

其中公鸡5文钱一个，母鸡三文钱一个，小鸡1文钱三个求麦克几只公鸡，母鸡和小鸡  
设买公鸡数量为x  
母鸡的数量为y  
小鸡的数量为z  
代码如下：  
x + y + z = 100  
5x + 3y + z/3 = 100  
for x in range(100+1):  
 for y in range(100-x+1):  
 for z in range(100-x-y+1):  
 if x + y + z == 100 and 5 \* x + 3 \* y + z/3 == 100:  
 print('公鸡%d只,母鸡%d只,小鸡%d只' % (x, y, z))  
  
  
三个孩子谁跑得慢  
三家九个孩子在比赛跑步  
规定：  
 第一名9分，第二名8分，第三名7分。。。。。。第九名1分  
比赛结果如下：  
 各家三个孩子总分相同  
 第一名是李家的孩子，第二名是王家的孩子  
 所有孩子名词是没有并列的  
 各家三个孩子名次是没有相链的  
求：最后一名是谁家的孩子  
设王家三个孩子为W1,W2,W3  
李家三个孩子为：L1,L2,L3  
张家三个孩子为：Z1,Z2,Z3  
可得W1+W2+W3 = L1+L2+L3 = Z1+Z2+Z3 = 15  
排名：L1,W1......  
设李家孩子的另外一个分数为x则李家三个孩子的分数为：9，x,15-9-x  
设王家孩子另外一个分数为y则王家三个孩子的分数为：8，y,15-8-y  
代码如下：  
for li in [[9,x,6-x] for x in range(1,6) if x-6+x>1]: 穷举出王家孩子的成绩  
 sum\_results = list(range(1,10)) 生成所有的成绩存入列表中  
 for results in li: 吧穷举出的王家孩子的成绩从所有的成绩中删除  
 sum\_results.remove(results)  
 for wang in [[8, y, 7 - y] for y in sum\_results if 7 - y in sum\_results and y - 7 + y > 1]: 穷举出王家孩子的分数  
 for results1 in wang: 把王家孩子从列表中删除  
 sum\_results.remove(results1)  
 zhang = sum\_results 最后把剩余的张家孩子的成绩复制到zhang列表中  
 if zhang[2] - zhang[1] > 1 and zhang[2] - zhang[0] > 1: 最后因为所有孩子的分数都不想链接，穷举出张家孩子的分数  
 print('李家的孩子分数为：', li)  
 print('王家的孩子分数为：', wang)  
 print('张家的孩子分数为：', zhang)  
  
  
杨辉三角  
 1  
 1 1  
 1 2 1  
 1 3 3 1  
 1 4 6 4 1  
代码如下：  
L = [1]  
L1 = [1]  
line = int(input('请输入打印的行数：'))  
konggeshu = line  
for i in range(line):  
 print(' ' \* konggeshu \*2, ' '.join([str(x) for x in L])) 用来打印L和空个数  
 konggeshu -= 1 用来表示空个数3  
 L1 = [1 for i in range(len(L))] 用来给L1初始化  
 if len(L) >= 2:  
 for j in range(0,len(L)-1):  
 L1[j+1] = L[j]+L[j+1]  
 L = L1  
 L.append(1)  
代码如下：  
Zdef YangHui (num):  
 LL = [[1]]  
 for i in range(1,num):  
 LL.append([(0 if j== 0 else LL[i-1][j-1])+ (0 if j ==len(LL[i-1]) else LL[i-1][j]) for j in range(i+1)])  
 return LL  
print(YangHui(10))