5-8 不只是数字才能做比较运算

str 也能比较 就是比较ASCII码

eg:

>>> 'a'>'b'

False

两个字母的比较：

>>> 'be'>'cd'

False

首先会比较 b 与 c， b<c ====> false , 就是第一位的两个字母先进行比较

然后就是 false 得出了结论 是不是蒙了啊 总结一下

(b<c 这个是经过比较得出来的结论 但是题目说b>c 那就不对了啊 就false了)

总结：先去第一位的字母相比较 若相同则对比下一个字母 得出结论后再与条件进行对比

列表对比

[1,2,3]<[2,3,4]

列表比较规则类似于字符串比较

首先比较 两个列表的第一个元素 1 与 2 1<2 与题目相符 true

元祖对比

(1,2,3)<(2,3,4)

规则同上 首先比较两个远足的第一个元素 1=1 相等 然后比较两个元组的第二个元素 2<3 与题目相符 所以 ===> true

5-9 逻辑运算符

主要操作bool类型 返回结果也是bool类型

逻辑运算符有三个: and(且/与) or(或) not(非)

======================================================

>>> True and True

True

>>> True and False

False

>>> False and True

False

>>> False and False

False

and 总结: 但凡有假 那就会是假的

======================================================

>>> True or False

True

>>> True or True

True

>>> False or True

True

>>> False or False

False

or 总结: 只要有True存在 那就是为真 只有两个假的情况下才会为假

======================================================

>>> not True

False

>>> not False

True

not 总结: 不是真的就是假的了 不是假的就是真的了

>>> not not False

False

======================================================

>>> 1 and 1

1

>>> not 1

False

>>> not 0

True

>>> not 2

False

>>> not 3.14

False

>>> not 0.1

False

对于 int 和 float 类型， 0 被认定为 False 非0 被认定是 True

========================================================

>>> 'a' and 'b'

'b'

>>> 'a' or 'b'

'a'

>>> not 'a'

False

>>> not 'b'

False

>>> not ' ' # 这不是一个空字符串 因为这里有一个空格

False

>>> not '' #这才是真正的空字符串

True

>>> not 'a'

False

对于 str 字符串 若是空字符串 被认为 False 否则 认定为 True

========================================================

>>> not []

True

>>> not [1,2,3]

False

对于 列表 list 空列表被认为 False 否则为 True

========================================================

tuple 元组, dict 字典 与列表 list 相同。

>>> [1] or []

[1]

>>> [] or [1]

[1]

>>> not []

True

>>> not [1]

False

>>> not {}

True

>>> not {1:2}

False

对于 tuple 元组, dict 字典 空的元组(tuple) 空的字典(dict) 被认定为 False

非空的元组(tuple) 和 非空的字典(dict) 被认定为 True

========================================================

当我们知道 int 1 和 2 都代表 True 的时候 我们运行如下代码

>>> 1 and 2

2

我们可以看到返回的结果为 2 而不是 1 为什么呢？

因为计算机先读取了 1 在读取了 2 进行对比 都是True 所以按照就近原则 返回了 2

同理

>>> 2 and 1

1

这证明了返回结果中有按照就近原则

>>> 1 or 0

1

计算机 读取了 1 (1 为True) 读取了 or (只要有一个True就返回True) 就不会再继续往下读取了 并且返回 1

>>> 1 or 2

1

这个理由同上