Python中的短路逻辑

布尔运算符有个特性：只有在需求值时才进行求值。表达式x and y需要两个变量都为真时才为真，所以如果x为假，表达式就会立刻返回false，而不管y的值（事实上各个语言都有这个特性）。实际上，如果x为假，表达式会返回x得值----否则它就返回y的值。这种行为被称为短路逻辑（short-circuit logic）或惰性求值（lazy evaluaion）：布尔运算符通常被称为逻辑运算符，就像你看到的那样第2个值有时“被短路了”。这种行为对于or来说也同样适用。在表达式x or y中，x为真时，它直接返回x的值，否则返回y值。注意，这意味着在布尔运算符之后的所有代码都不会执行。



为什么会出现这种情况呢？

这段程序可以这样来分解：因为优先级（not>and>or）

代码=（not 1）or （0 and 1）or （3 and 4）or （5 and 6）or （7 and 8 and 9）

=0 or 0 or 4 or 6 or 9

电脑从左到右判断时

(0 or 0=0) (0 or 4=4 ) (4 or.....)（此时电脑已经判断了‘4’为真时，直接跳过后面的‘4’，直接输出‘4’），这种就属于短路逻辑的一种。