要不要使用复杂表达式

Perl语言的原作者*Larry Wall*曾经说过，伟大的程序员都有三个优点：懒惰、暴躁和自负。乍一看这三个词语没有一个是褒义词，但在程序员的世界里，这三个词有不同的意义。首先，懒惰会促使程序员去写一些省事儿的程序来辅助自己或别人更好的完成工作，这样我们就无需做那些重复和繁琐的劳动；同理能够用3行代码解决的事情，我们也绝不会写出10行代码来。其次，暴躁会让程序员主动的去完成一些你还没有提出的工作，去优化自己的代码让它更有效率，能够3秒钟完成的任务，我们绝不能容忍1分钟的等待。最后，自负会促使程序员写出可靠无误的代码，我们写代码不是为了接受批评和指责，而是为了让其他人来膜拜。

那么接下来就有一个很有意思的问题值得探讨一下，我们需要一个程序从输入的三个数中找出最大的那个数。这个程序对任何会编程的人来说都是小菜一碟，甚至不会编程的人经过10分钟的学习也能搞定。下面是用来解决这个问题的Python代码。

```
a = int(input('a = '))
b = int(input('b = '))
c = int(input('c = '))
if a > b:
    the_max = a
else:
    the_max = b
if c > the_max:
    the_max = c
print('The max is:', the_max)
```

但是我们刚才说了，程序员都是懒惰的，很多程序员都会使用三元条件运算符来改写上面的代码。

```
a = int(input('a = '))
b = int(input('b = '))
c = int(input('c = '))
the_max = a if a > b else b
the_max = c if c > the_max else the_max
print('The max is:', the_max)
```

需要说明的是，Python在2.5版本以前是没有上面代码第4行和第5行中使用的三元条件运算符的，究其原因是Guido van Rossum（Python之父）认为三元条件运算符并不能帮助 Python变得更加简洁，于是那些习惯了在C/C++或Java中使用三元条件运算符（在这些语言中，三元条件运算符也称为“Elvis运算符”，因为`?:`放在一起很像著名摇滚歌手猫王Elvis的大背头）的程序员试着用`and`和`or`运算符的短路特性来模拟出三元操作符，于是在那个年代，上面的代码是这样写的。

```
a = int(input('a = '))
b = int(input('b = '))
c = int(input('c = '))
the_max = a > b and a or b
the_max = c > the_max and c or the_max
print('The max is:', the_max)
```

但是这种做法在某些场景下是不能成立的，且看下面的代码。

```
a = 0
b = -100
# 下面的代码本来预期输出a的值，结果却得到了b的值
# 因为a的值0在进行逻辑运算时会被视为False来处理
print(True and a or b)
# print(a if True else b)
```

所以在Python 2.5以后引入了三元条件运算符来避免上面的风险（上面代码被注释掉的最后一句话）。那么，问题又来了，上面的代码还可以写得更简短吗？答案是肯定的。

```
a = int(input('a = '))
b = int(input('b = '))
c = int(input('c = '))
print('The max is:', (a if a > b else b) if (a if a > b else b) > c else c)
```

但是，这样做真的好吗？如此复杂的表达式是不是让代码变得晦涩了很多呢？我们发现，在实际开发中很多开发者都喜欢过度的使用某种语言的特性或语法糖，于是简单的多行代码变成了复杂的单行表达式，这样做真的好吗？这个问题我也不止一次的问过自己，现在我能给出的答案是下面的代码，使用辅助函数。

```
def the_max(x, y):
    return x if x > y else y


a = int(input('a = '))
b = int(input('b = '))
c = int(input('c = '))
print('The max is:', the_max(the_max(a, b), c))
```

上面的代码中，我定义了一个辅助函数`the_max`用来找出参数传入的两个值中较大的那一个，于是下面的输出语句可以通过两次调用`the_max`函数来找出三个数中的最大值，现在代码的可读性是不是好了很多。用辅助函数来替代复杂的表达式真的是一个不错的选择，关键是比较大小的逻辑转移到这个辅助函数后不仅可以反复调用它，而且还可以进行级联操作。

当然，很多语言中比较大小的函数根本没有必要自己来实现（通常都是内置函数），Python也是如此。Python内置的max函数利用了Python对可变参数的支持，允许一次性传入多个值或者一个迭代器并找出那个最大值，所以上面讨论的问题在Python中也就是一句话的事，但是从复杂表达式到使用辅助函数简化复杂表达式这个思想是非常值得玩味的，所以分享出来跟大家做一个交流。

```
a = int(input('a = '))
b = int(input('b = '))
c = int(input('c = '))
print('The max is:', max(a, b, c))
```