可能你会觉得，函数只有一行代码的情况并不多嘛，这个语法糖也不会很常用吧？其实并不是这样的，因为它还可以结合 Kotlin 的其他语言特性一起使用，对简化代码方面的帮助很大，后面我们会慢慢学习它更多的使用场景。

2.4 程序的逻辑控制

程序的执行语句主要分为 3 种：顺序语句、条件语句和循环语句。顺序语句很好理解，就是代码一行一行地往下执行就可以了，但是这种“愣头青”的执行方式在很多情况下并不能满足我们的编程需求，这时就需要引入条件语句和循环语句了，下面我们逐个进行介绍。

2.4.1 if 条件语句

Kotlin 中的条件语句主要有两种实现方式：if 和 when。

首先学习 if，Kotlin 中的 if 语句和 Java 中的 if 语句几乎没有任何区别，因此这里我就简单举个例子带你快速了解一下。

还是以上一节中的 largerNumber()函数为例，之前我们借助了 Kotlin 内置的 max()函数来实现返回两个参数中的较大值，但其实这是没有必要的，因为使用 if 判断同样可以轻松地实现这个功能。将 largerNumber()函数的实现改成如下写法：

```
fun largerNumber(num1: Int, num2: Int): Int {
    var value = 0
    if (num1 > num2) {
        value = num1
    } else {
        value = num2
    }
    return value
}
```

这段代码相信不需要我多做解释，任何有编程基础的人都应该能看得懂。但是有一点我还是得说明一下，这里使用了 var 关键字来声明 value 这个变量，这是因为初始化的时候我们先将 value 赋值为 0，然后再将它赋值为两个参数中更大的那个数，这就涉及了重新赋值，因此必须用 var 关键字才行。

到目前为止，Kotlin 中的 if 用法和 Java 中是完全一样的。但注意我前面说的是“几乎没有任何区别”。也就是说，它们还是存在不同之处的，那么接下来我们就着重看一下不同的地方。

Kotlin 中的 if 语句相比于 Java 有一个额外的功能，它是可以有返回值的，返回值就是 if 语句每一个条件中最后一行代码的返回值。因此，上述代码就可以简化成如下形式：

```
fun largerNumber(num1: Int, num2: Int): Int {
    val value = if (num1 > num2) {
        num1
```