现在 largerNumber()函数已经编写好了，接下来我们可以尝试在 main()函数中调用这个函数，并且实现在两个数中找到较大的那个数这样一个简单的功能，代码如下所示：

```
fun main() {
    val a = 37
    val b = 40
    val value = largerNumber(a, b)
    println("larger number is " + value)
}

fun largerNumber(num1: Int, num2: Int): Int {
    return max(num1, num2)
}
```

这段代码很简单，我们定义了 a、b 两个变量，a 的值是 37，b 的值是 40，然后调用 largerNumber()函数，并将 a、b 作为参数传入。largerNumber()函数会返回这两个变量中较大的那个数，最后将返回值打印出来。现在运行一下代码，结果如图 2.11 所示。程序正如我们预期的那样运行了。

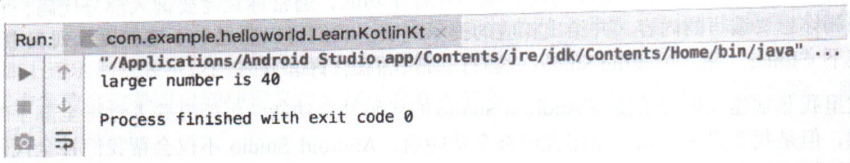

图 2.11　调用 largerNumber()函数的运行结果

这就是 Kotlin 中最基本也是最常用的函数用法，虽然这里我们实现的 largerNumber()函数很简单，但是掌握了函数的定义规则之后，你想实现多么复杂的函数都是可以的。

在本小节的最后，我们再来学习一个 Kotlin 函数的语法糖，这个语法糖在以后的开发中会起到相当重要的作用。

当一个函数中只有一行代码时，Kotlin 允许我们不必编写函数体，可以直接将唯一的一行代码写在函数定义的尾部，中间用等号连接即可。比如我们刚才编写的 largerNumber()函数就只有一行代码，于是可以将代码简化成如下形式：

```
fun largerNumber(num1: Int, num2: Int): Int = max(num1, num2)
```

使用这种语法，return 关键字也可以省略了，等号足以表达返回值的意思。另外，还记得 Kotlin 出色的类型推导机制吗？在这里它也可以发挥重要的作用。由于 max()函数返回的是一个 Int 值，而我们在 largerNumber()函数的尾部又使用等号连接了 max()函数，因此 Kotlin 可以推导出 largerNumber()函数返回的必然也是一个 Int 值，这样就不用再显式地声明返回值类型了，代码可以进一步简化成如下形式：

```
fun largerNumber(num1: Int, num2: Int) = max(num1, num2)
```