2.6.2 集合的函数式 API

集合的函数式 API 有很多个，这里我并不打算带你涉猎所有函数式 API 的用法，而是重点学习函数式 API 的语法结构，也就是 Lambda 表达式的语法结构。

首先我们来思考一个需求，如何在一个水果集合里面找到单词最长的那个水果？当然这个需求很简单，也有很多种写法，你可能会很自然地写出如下代码：

```
val list = listOf("Apple", "Banana", "Orange", "Pear", "Grape", "Watermelon")
var maxLengthFruit = ""
for (fruit in list) {
    if (fruit.length > maxLengthFruit.length) {
        maxLengthFruit = fruit
    }
}
println("max length fruit is " + maxLengthFruit)
```

这段代码很简洁，思路也很清晰，可以说是一段相当不错的代码了。但是如果我们使用集合的函数式 API，就可以让这个功能变得更加容易：

```
val list = listOf("Apple", "Banana", "Orange", "Pear", "Grape", "Watermelon")
val maxLengthFruit = list.maxBy { it.length }
println("max length fruit is " + maxLengthFruit)
```

上述代码使用的就是函数式 API 的用法，只用一行代码就能找到集合中单词最长的那个水果。或许你现在理解这段代码还比较吃力，那是因为我们还没有开始学习 Lambda 表达式的语法结构，等学完之后再来重新看这段代码时，你就会觉得非常简单易懂了。

首先来看一下 Lambda 的定义，如果用最直白的语言来阐述的话，Lambda 就是一小段可以作为参数传递的代码。从定义上看，这个功能就很厉害了，因为正常情况下，我们向某个函数传参时只能传入变量，而借助 Lambda 却允许传入一小段代码。这里两次使用了“一小段代码”这种描述，那么到底多少代码才算一小段代码呢？Kotlin 对此并没有进行限制，但是通常不建议在 Lambda 表达式中编写太长的代码，否则可能会影响代码的可读性。

接着我们来看一下 Lambda 表达式的语法结构：

```
{参数名1: 参数类型, 参数名2: 参数类型 -> 函数体}
```

这是 Lambda 表达式最完整的语法结构定义。首先最外层是一对大括号，如果有参数传入到 Lambda 表达式中的话，我们还需要声明参数列表，参数列表的结尾使用一个->符号，表示参数列表的结束以及函数体的开始，函数体中可以编写任意行代码（虽然不建议编写太长的代码），并且最后一行代码会自动作为 Lambda 表达式的返回值。

当然，在很多情况下，我们并不需要使用 Lambda 表达式完整的语法结构，而是有很多种简化的写法。但是简化版的写法对于初学者而言更难理解，因此这里我准备使用一步步推导演化的方式，向你展示这些简化版的写法是从何而来的，这样你就能对 Lambda 表达式的语法结构理解