得更加深刻了。那么接下来我们就由繁入简开始吧。

还是回到刚才找出最长单词水果的需求，前面使用的函数式 API 的语法结构看上去好像很特殊，但其实 maxBy 就是一个普通的函数而已，只不过它接收的是一个 Lambda 类型的参数，并且会在遍历集合时将每次遍历的值作为参数传递给 Lambda 表达式。maxBy 函数的工作原理是根据我们传入的条件来遍历集合，从而找到该条件下的最大值，比如说想要找到单词最长的水果，那么条件自然就应该是单词的长度了。

理解了 maxBy 函数的工作原理之后，我们就可以开始套用刚才学习的 Lambda 表达式的语法结构，并将它传入到 maxBy 函数中了，如下所示：

```
val list = listOf("Apple", "Banana", "Orange", "Pear", "Grape", "Watermelon")
val lambda = { fruit: String -> fruit.length }
val maxLengthFruit = list.maxBy(lambda)
```

可以看到，maxBy 函数实质上就是接收了一个 Lambda 参数而已，并且这个 Lambda 参数是完全按照刚才学习的表达式的语法结构来定义的，因此这段代码应该算是比较好懂的。

这种写法虽然可以正常工作，但是比较啰嗦，可简化的点也非常多，下面我们就开始对这段代码一步步进行简化。

首先，我们不需要专门定义一个 lambda 变量，而是可以直接将 lambda 表达式传入 maxBy 函数当中，因此第一步简化如下所示：

```
val maxLengthFruit = list.maxBy({ fruit: String -> fruit.length })
```

然后 Kotlin 规定，当 Lambda 参数是函数的最后一个参数时，可以将 Lambda 表达式移到函数括号的外面，如下所示：

```
val maxLengthFruit = list.maxBy() { fruit: String -> fruit.length }
```

接下来，如果 Lambda 参数是函数的唯一一个参数的话，还可以将函数的括号省略：

```
val maxLengthFruit = list.maxBy { fruit: String -> fruit.length }
```

这样代码看起来就变得清爽多了吧？但是我们还可以继续进行简化。由于 Kotlin 拥有出色的类型推导机制，Lambda 表达式中的参数列表其实在大多数情况下不必声明参数类型，因此代码可以进一步简化成：

```
val maxLengthFruit = list.maxBy { fruit -> fruit.length }
```

最后，当 Lambda 表达式的参数列表中只有一个参数时，也不必声明参数名，而是可以使用 it 关键字来代替，那么代码就变成了：

```
val maxLengthFruit = list.maxBy { it.length }
```

怎么样？通过一步步推导的方式，我们就得到了和一开始那段函数式 API 一模一样的写法。是不是现在理解起来就非常轻松了呢？